Politecnico di Bari

Repository Istituzionale dei Prodotti della Ricerca del Politecnico di Bari

Development of metaheuristic optimization algorithms and AI tools for advanced design/control of industrial components

This is a PhD Thesis

Original Citation:
Development of metaheuristic optimization algorithms and AI tools for advanced design/control of industrial components / Furio, Chiara. - ELETTRONICO. - (2026). [10.60576/poliba/iris/furio-chiara_phd2026]

Availability:
This version is available at http://hdl.handle.net/11589/295621 since: 2026-01-19

Published version
DOI:10.60576/poliba/iris/furio-chiara_phd2026

Publisher: Politecnico di Bari

Terms of use:

(Article begins on next page)

23 April 2026

Chiara FURIO

Development of metaheuristic optimization algorithms and AI tools for advanced design/control of industrial components

Thesis submitted for the degree of Philosophiae Doctor

Interuniversity Ph.D. Program in Industry 4.0
Polytechnic University of Bari

Tutors
Prof. Engr. Luciano LAMBERTI
Prof. Engr. Adrian KOLLER

2025

Dissertation submitted for the degree of *Philosophiae Doctor*
Interuniversity Ph.D. Program in Industry 4.0

Cycle:
38th (2022, November 1 – 2025, October 31)

Administrative Headquarters: Politecnico di Bari – Department of Electrical and Information Engineering

Hosting University:
Polytechnic University of Bari- Department of Mechanics Management and Mathematics

Title:
Development of metaheuristic optimization algorithms and AI tools for advanced design/control of industrial components

Ph.D. Candidate:
Chiara FURIO

Academic Discipline:
Mechanical Design (Italian code: IIND-03/A), Applied Mechanics (Italian code: IIND-02/A)

Tutors:
Prof. Luciano LAMBERTI, Polytechnic University of Bari (Bari, Italy)
Prof. Adrian KOLLER, HSLU Lucerne University of Applied Sciences and Arts (Lucerne, Switzerland)

Coordinator:
Prof. Engr. Salvatore Digiesi, Polytechnic University of Bari (Bari, Italy)

External Reviewers:
Prof./Dr. Filippo BERTO, University of Rome “La Sapienza” (Rome, Italy)

Prof./Dr. Katia GENOVESE, University of Basilicata (Potenza, Italy)

Last version:
December 20, 2025

All rights reserved. No part of this publication may be reproduced or transmitted, in any form or by any means, without permission.



Department of Electrical and Information Engineering

INDUSTRY 4.0

Ph.D. Program

SSD: ING-IND/14- Mechanical Design and Machine Construction

Final Dissertation

DEVELOPMENT OF METAHEURISTIC OPTIMIZATION ALGORITHMS AND AI TOOLS FOR ADVANCED DESIGN/CONTROL OF INDUSTRIAL COMPONENTS

---

by

Chiara FURIO

Supervisors:

Prof. Luciano LAMBERTI

Prof. Adrian KOLLER

Coordinator of Ph.D Program:

Prof. Salvatore Digiesi

Course n°38, 01/11/2022-31/10/2025

Politecnico di Bari

Department of Electrical and Information Engineering

INDUSTRY 4.0

Ph.D. Program

SSD: ING-IND/14- Mechanical Design and Machine Construction

Final Dissertation

DEVELOPMENT OF METAHEURISTIC OPTIMIZATION ALGORITHMS AND AI TOOLS FOR ADVANCED DESIGN/CONTROL OF INDUSTRIAL COMPONENTS

---

by

Chiara FURIO



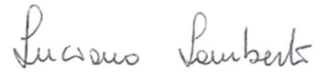
Referees:

Prof. Filippo BERTO

Prof. Katia GENOVESE

Supervisors:

Prof. Luciano LAMBERTI

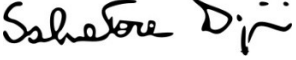


Prof. Adrian KOLLER



Coordinator of Ph.D. Program:

Prof. Salvatore Digiesi



Course n°38, 01/11/2022-31/10/2025

Dedication

This work is dedicated to the memory of my parents, my strong and guiding lights in life, who filled it with values and principles that continue to inspire me.

A dedication goes to my friend Carmen and her family, a constant and invaluable presence in my daily life.

A Special Dedication

To my sister Donata, my true guiding star: your unwavering support during difficult times has been my anchor and inspiration. This work would not have been possible without you.

Special Acknowledgement

I wish to express my deepest gratitude to my supervisor, Prof. Luciano Lamberti, whose tireless guidance and constant support have illuminated the path of my doctoral journey like a beacon in the fog. His invaluable advice, vision, and passion for research have shaped not only my academic growth but also my personal development, teaching me to look beyond difficulties, seek deeper connections, and cultivate curiosity and rigor at every step. His presence has been a milestone, a steadfast point of reference, capable of inspiring and guiding every decision along this journey.

Acknowledgements

I'm deeply grateful to my degree thesis supervisors in Mechanical Engineering, Prof. A.D. Ludovico and Prof. Casalino. With the latter, prof. Giuseppe Casalino, I had the privilege to continue collaborating after graduation and to cultivate my first genuine passion for research.

I would like to extend my personal thanks to Prof. Sabina Campanelli and Prof. Andrea Angelastro, for their support during a decisive and meaningful professional transition, which greatly influenced my development.

I would like to extend my heartfelt thanks to all the professors who have contributed to broadening my knowledge and inspiring me to pursue deeper understanding. In particular, I am deeply grateful to Professors Berardina De Carolis and Nicoletta Del Buono from the University "A. Moro" in Bari, who guided me in expanding my horizons in both theoretical and practical aspects of artificial intelligence applications, challenging me to think critically and creatively.

Special thanks also go to Prof. Caterina Ciminelli and Prof. Salvatore Digiesi, attentive coordinators of the doctoral program, for their availability, guidance, and continuous support, which made my research journey smoother and more productive.

My encounter with the research group led by Prof. Giulio Reina and Prof. Mario Massimo Foglia, Associate Professors in Applied Mechanics at the Department of Mechanics, Mathematics and Management, and Dr. Angelo Ugenti, Postdoctoral Research Fellow in the same department at the Polytechnic University of Bari, under the constant supervision of prof. Luciano Lamberti, was truly enlightening. Their expertise and collaborative spirit provided invaluable insights that significantly shaped the development of practical applications.

I would also like to thank the reviewers for their thoughtful and constructive feedback, which greatly contributed to improving the quality and clarity of this work.

I gratefully acknowledge the support of HSLU, Institut für Maschinen- und Energietechnik (IME) in Lucerne, particularly Prof. Adrian Koller, Professor at the Institute of Mechanical Engineering and Energy Technology, and Prof. Ralf Baumann, Head of the Competence Center for Mechanical Systems, for providing the financial and human resources, guidance, and stimulating environment essential for my research during my period abroad. I am also sincerely thankful to the Robotics Research Group at HSLU, whose collaboration, encouragement, and welcoming spirit made my stay in Switzerland both enjoyable and intellectually rewarding.

I would like to extend my deepest gratitude to the Italian Ministry of Education and Merit (MIM) for allowing me to work over all these years in an environment rich in human values, for giving me the opportunity to contribute through my daily work to the education of Italian adolescents in the fields of mechanics and mechatronics, and for supporting my personal and professional growth financially.

Dedica

Questo lavoro è dedicato alla memoria dei miei genitori, che hanno saputo arricchire la mia vita di valori e principi che continuano a ispirarmi ogni giorno. Grazie alla mia amica Carmen e alla sua famiglia, presenza costante e preziosa nella mia quotidianità.

Dedica speciale

Alla mia cara e insostituibile sorella Donata, la mia vera stella polare: il tuo sostegno incondizionato nei momenti difficili è stato il mio ancoraggio e la mia fonte di ispirazione. Senza di te questo lavoro non sarebbe stato possibile.

Un Ringraziamento speciale

Desidero esprimere la mia più profonda e sincera gratitudine al mio supervisore, Prof. Luciano Lamberti, la cui guida instancabile e il sostegno costante hanno illuminato il cammino del mio dottorato come un faro nella nebbia. I suoi preziosi consigli, la sua visione e la passione con cui affronta la ricerca hanno plasmato non solo la mia crescita accademica, ma anche il mio percorso personale, insegnandomi a guardare oltre le difficoltà, a cercare connessioni profonde e a coltivare curiosità e rigore in ogni passo. La sua presenza è stata una pietra miliare, un punto di riferimento saldo, capace di ispirare e orientare ogni scelta lungo questo viaggio.

Ringraziamenti

Desidero esprimere un sincero ringraziamento ai miei relatori di tesi di laurea in Ingegneria Meccanica, Proff. A.D. Ludovico e Casalino. Con quest'ultimo, il prof. Giuseppe Casalino, ho avuto il privilegio di collaborare dopo la laurea e di sviluppare il mio primo autentico interesse per la ricerca.

Un mio particolare ringraziamento va ai Proff. Sabina Campanelli e Andrea Angelastro, per avermi supportata in una transizione professionale decisiva e significativa che ha contribuito in modo determinante al mio sviluppo.

Ringrazio di cuore tutti i docenti che hanno arricchito le mie conoscenze e mi hanno ispirata a perseguire una comprensione sempre più profonda del mondo che mi circonda. In particolare, sono profondamente grata alle Prof.sse Berardina De Carolis e Nicoletta Del Buono dell'Università "A. Moro" di Bari, che mi hanno guidata nell'espandere i miei orizzonti, stimolandomi a confrontarmi con aspetti sia teorici che pratici delle applicazioni dell'intelligenza artificiale e mi hanno incoraggiata a sviluppare pensiero critico e creativo.

Un ringraziamento speciale va anche alla Prof.ssa Caterina Ciminelli e al Prof. Salvatore Digiesi, attenti coordinatori del dottorato, per la loro disponibilità, guida e supporto continuo. Il percorso di ricerca è risultato così più sereno e produttivo.

Il mio incontro con il gruppo di ricerca guidato dal Prof. Giulio Reina e dal Prof. Mario Massimo Foglia, professori associati di Meccanica Applicata presso il Dipartimento di Meccanica, Matematica e Management, e dal Dr. Angelo Ugenti, ricercatore post-dottorato nello stesso dipartimento del Politecnico di Bari, è stato illuminante. La loro competenza, la loro disponibilità e il loro spirito collaborativo, accompagnati sempre dalla supervisione del prof. Luciano Lamberti, hanno offerto intuizioni preziose e contribuito in modo significativo allo sviluppo di applicazioni pratiche.

Ringrazio inoltre i revisori per i loro commenti attenti e costruttivi, che hanno migliorato in modo significativo la chiarezza e la qualità del mio lavoro.

Esprimo la mia sincera gratitudine alla HSLU, Institut für Maschinen- und Energietechnik (IME) di Lucerna, in particolare al Prof. Adrian Koller, Professore presso l'Institute of Mechanical Engineering and Energy Technology, e al Prof. Ralf Baumann, Responsabile del Competence Center for Mechanical Systems, per aver messo a disposizione risorse finanziarie, umane e un ambiente stimolante, fondamentali per la mia ricerca durante il periodo trascorso all'estero.

Un ringraziamento speciale va anche al Robotics Research Group della HSLU, la cui collaborazione, accoglienza e incoraggiamento hanno reso il mio soggiorno in Svizzera piacevole e intellettualmente arricchente.

Infine, desidero esprimere la mia più sentita gratitudine al Ministero Italiano dell'Istruzione e del Merito (MIM) per avermi permesso di lavorare in tutti questi anni in un ambiente ricco di valori umani, per avermi dato l'opportunità di contribuire, con il mio lavoro quotidiano, alla formazione degli adolescenti italiani nei campi della meccanica e della meccatronica e per aver sostenuto finanziariamente la mia crescita personale e professionale.

SUMMARY

PREFACE *11*

CHAPTER I: NATURE-INSPIRED METAHEURISTICS AND AN IN-DEPTH STUDY OF THE JAYA AND GREY WOLF OPTIMIZER ALGORITHMS *12*

1.1 METAHEURISTICS INTRODUCTION **13**

1.2 INTRODUCTION TO EVOLUTIONARY ALGORITHMS **15**

1.3 JAYA ALGORITHM **19**

1.4 SELF-ADAPTIVE JAYA ALGORITHM **20**

1.5 QUASI-OPPOSITIONAL BASED JAYA (QO-JAYA) ALGORITHM **22**

1.6 SELF-ADAPTIVE MULTI-POPULATION (SAMP) JAYA ALGORITHM **23**

1.7 APPLICATION OF JAYA ALGORITHM TO A DAM DESIGN **25**

1.8 JAYA ALGORITHM: RESULTS AND DISCUSSION ON THE DAM EXAMPLE **28**

1.9 THE GREY WOLF ALGORITHM **30**

1.10 APPLICATION OF GWO ALGORITHM TO ACKLEY FUNCTION **37**

1.11 GWO ALGORITHMS: CODE RESULTS AND DISCUSSION ON ACKLEY FUNCTION EXAMPLE **37**

1.12 FIRST SUMMARY AND INSIGHTS ON JAYA AND GWO **39**

CHAPTER II: INTEGRATION OF METAHEURISTIC OPTIMIZATION AND AI-BASED CONTROL STRATEGIES FOR MULTI-AGENT CART-POLE SYSTEMS *42*

2.1 CONTROL SYSTEMS STRATEGIES **43**

2.2 DYNAMICS OF A SINGLE CART-POLE MECHANISM **44**

2.3 THE ADMM (ALTERNATING DIRECTION METHOD OF MULTIPLIERS) **50**

2.4. APPLICATION OF ADMM TO THE COORDINATION OF CART AND POLE SENSORS **54**

2.5 RESULTS OF ADMM FOR CART AND POLE SENSOR COORDINATION **57**

2.6 APPLICATION OF ADMM AND REINFORCEMENT LEARNING TO THE COORDINATION OF TWO CART-POLE SYSTEMS **62**

2.7 SUMMARY ON METAHEURISTIC OPTIMIZATION IN CONTROL SYSTEMS **65**

CHAPTER III: AI TOOLS FOR ADVANCED DESIGN AND PROCESSES *68*

3.1 INTRODUCTION **69**

3.2 INVESTIGATION ON METAHEURISTIC OPTIMIZATION IN HUMAN-ROBOT INTERACTION DESIGN FOR ALPHA MINI **70**

3.3 INVESTIGATION ON METAHEURISTIC OPTIMIZATION IN AUGMENTED AND VIRTUAL REALITY **77**

3.4 INVESTIGATION ON METAHEURISTIC OPTIMIZATION IN REVERSE ENGINEERING AND ADDICTIVE MANUFACTURING **86**

3.5 TOWARD INTEGRATED RESEARCH PATHWAYS: SYNERGIES IN ADVANCED DESIGN AND MANUFACTURING **95**

CHAPTER IV: SHGWJA: A SIMPLE HYBRID GREY WOLF-JAYA METAHEURISTIC ALGORITHM FOR OPTIMIZATION IN MECHANICAL AND CIVIL ENGINEERING *98*

4.1 DETAILED EXAMINATION OF METAHEURISTIC ALGORITHMS 99
4.2 HYBRID METAHEURISTIC DESIGN: FROM GWO AND JAYA TO SHGWJA 107
4.3 APPLICATION AND EVALUATION OF THE SHGWJA ALGORITHM 116
4.3.1 Mathematical optimization: Rosenbrock and Rastrigin 116
4.3.2 Engineering design optimization 118
4.3.2.1 Concrete Gravity Dam 118
4.3.2.2 Tension/Compression Spring 120
4.3.2.3 Welded Beam 122
4.3.2.4 Pressure Vessel 124
4.3.2.5 Industrial Refrigeration System 126
4.3.3 Robotics optimization 129
4.3.3.1 A six-variable path planning approach 129
4.3.4 Characterization of a flat composite panel subjected to axial compression 131
4.4 SHGWJA IMPLEMENTATION DETAILS 132
4.5 EXPERIMENTAL RESULTS WITH SHGWJA 134
4.5.1. Mathematical optimization: Rosenbrock and Rastrigin 134
4.5.2. Engineering design optimization 139
4.5.2.1 Concrete gravity dam 139
4.5.2.2. Tension/compression spring 143
4.5.2.3. Welded beam 146
4.5.2.4 Pressure vessel 150
4.5.2.5. Industrial refrigeration system 154
4.5.3 Robotics optimization 157
4.5.3.1 A six variable path planning 157
4.5.4. Characterization of a flat composite panel subjected to axial compression 160
4.6 STATISTICAL ANALYSIS OF THE RESULTS OF ENGINEERING OPTIMIZATION PROBLEMS 164
4.7 FINAL EVALUATION AND RESEARCH SIGNIFICANCE OF SHGWJA 169
CHAPTER V: FHGWJA: AN EFFICIENT FAST HYBRID GREY WOLF-JAYA METAHEURISTIC ALGORITHM FOR DESIGN OPTIMIZATION *..... 172*
5.1 INTRODUCTION TO THE FAST HYBRID GREY WOLF-JAYA ALGORITHM (FHGWJA) 173
5.2 DETAILED FORMULATION OF THE FAST HYBRID GREY WOLF-JAYA ALGORITHM 174
5.3 FHGWJA TEST PROBLEMS AND OPTIMIZATION RESULTS 182
5.3.1 2D path planning 182
5.3.2 Shape optimization of a concrete gravity dam under multiple earthquake loads. 188

5.3.3 Calibration of nonlinear hydrologic models 193

5.3.4 Optimal crashworthiness design of a vehicle subject to side impact 199

5.3.5 Weight minimization of a planar 200-bar truss structure 204

5.4 CONCLUSIONS ON FHGWJA 211

CHAPTER VI: HALSGWJA: AN HYBRID APPROXIMATE LINE SEARCH GREY WOLF JAYA METAHEURISTIC ALGORITHM FOR STRUCTURAL OPTIMIZATION 213

6.1 INTRODUCTION TO THE HYBRID APPROXIMATE LINE SEARCH GREY WOLF-JAYA ALGORITHM (HALGWJA) 214

6.2 The Hybrid Approximate Line Search Grey Wolf JAYA Algorithm 215

6.3 Test Problems and Optimization Results 224

6.3.1. Results of CEC 2020 problems 230

6.3.2. Comparative Performance Evaluation of HALSGWJA and Competing Algorithms on the CEC2020 Benchmark Problems 248

6.3.3. Results of the car side impact problem 253

6.4. Conclusions: Towards the Global Optimum from SHGWJA, FHGWJA to HALSGWJA 257

CHAPTER VII: SMART EXPLORER 1.0: INTELLIGENT ROBOTIC PATH PLANNING FOR INDUSTRY 4.0 APPLICATIONS 261

7.1 INTRODUCTION TO PATH PLANNING IN ROBOTICS: SMART EXPLORER 1.0 AND CLASSICAL ALGORITHMS 262

7.2 DESIGN AND PRINCIPLES OF THE SMARTEXPLORER1.0 (SE1.0) ALGORITHM 263

7.3 CHARACTERIZATION OF ENVIRONMENTAL FACTORS AND PARAMETERS FOR SE1.0 273

7.4 RESULTS OF SE1.0: NAVIGATION, PATH PLANNING, AND PREDICTIVE PERFORMANCE 277

7.5 EXPERIMENTAL VALIDATION: SE1.0 NAVIGATION PERFORMANCE OF DUNE ROVER ON UNSTABLE TERRAIN 290

7.6 CONCLUSIONS AND FUTURE PERSPECTIVES ON SE 1.0 294

CHAPTER VIII: TSPGWJO: TOWARDS HYBRID INTELLIGENCE FOR OPTIMAL NAVIGATION OF AUTONOMOUS ROBOTS 297

8.1 MOTIVATIONS AND BACKGROUND FOR HYBRID METAHEURISTIC APPROACHES TO THE TSP UNDER KINEMATIC CONSTRAINTS 298

8.2 CLASSICAL AND NOVEL METAHEURISTICS FOR THE TRAVELLING SALESMAN PROBLEM UNDER KINEMATIC CONSTRAINTS 299

8.3. TSPWJO: A HYBRID MULTI-STRATEGY ALGORITHM FOR TSP USING MIXED POPULATIONS AND INTERMEDIATE-POINT STRATEGIES 302

8.4. COST FUNCTION AND CONSTRAINTS IN TSPWJO 309

8.5 COMPARATIVE PERFORMANCE ANALYSIS: CLASSICAL APPROACH VS TSPWJO 312

8.6 RESULTS DECODED AND CONCLUSIONS.....321
FINAL CONCLUSIONS AND FUTURE DIRECTIONS *.....323*
APPENDIX 1-MATLAB JAYA CODE APPLIED TO A DAM'S DESIGN *.....351*
THE MAIN JAYA FILE 351
THE JAYA OBJECTIVE FUNCTION 356
THE JAYA CONSTRAINTS' FUNCTION 357
APPENDIX 2- MATLAB GWO CODE APPLIED TO A GENERIC FUNCTION *.....360*
THE "MAIN" GWO FILE 360
THE "GWO" FILE.....362
THE "INITIALIZATION" FILE.....365
THE "GET_FUNCTION_DETAILS" FILE.....366
THE FILE "FUNC_PLOT".....372
APPENDIX 3- PID+ADMM ON SINGLE CART-POLE. *.....374*
3.1 MATLAB CODE FOR LQR+ADMM ON DUAL CART-POLE 374
3.2 MATLAB CODE WITH LQR AND ADMM.....377
APPENDIX 4- LQR+ADMM CONTROL OF TWO CART-POLE SYSTEMS: MATLAB CODE *380*
APPENDIX 5- ALPHA MINI ROBOT INFORMATION *.....384*
APPENDIX 6- HOLOBOX FREECAD OPTIMIZATION *.....386*
APPENDIX 7- REVERSE ENGINEERING AND 3D PRINTING TOOLS *.....396*
APPENDIX 8- CATALOGUE OF STUDIED OPTIMIZATION CASES *.....412*
8.1 Applied Mathematics: 412
8.1.1 Rosenbrock 412
8.1.2 Rastrigin.....412
8.1.3 Himmelblau's Function 413
8.2 Mechanics:.....414
8.2.1 Tension/compression spring design (case 1: 3 variables and 3 constraints) 414
8.2.2 Car's Crashworthiness 417
8.2.3 Welded Beam.....421
8.2.4 Pressure Vessel 424
8.2.5 Topology: Characterization of a flat composite panel subjected to axial compression 428
8.3 Robotics: Path Planning 430
8.4 Fluid Machines: Industrial Refrigeration System 432
8.5 Hydraulics: 436
8.5.1 Muskingum 3 variables 436
8.5.2 Muskingum 25 variables 437

8.6 Hydraulic Engineering: Dam case 1- only vertical component for earthquake 437
8.7 Civil Engineering: Two hundred bar truss design 442

PREFACE

We are living at the dawn of a new industrial era, one where intelligence, matter, and energy intertwine. Industry 4.0 is no longer a distant vision but a living ecosystem, where machines think, systems evolve, and data becomes the language of creation. Factories are transforming into sentient environments, capable of sensing, learning, and adapting in real time.

This revolution, however, brings with it extraordinary complexity. Engineering design and control now unfold within vast spaces of uncertainty, shaped by nonlinear behaviours, interdependent dynamics, and ever-changing constraints. Traditional optimization methods, bound by rigid assumptions, often falter in such terrain.

Here, metaheuristic algorithms emerge as explorers of the unknown, creative, adaptive, and free from the confines of problem-specific logic. From Genetic Algorithms to Particle Swarm Optimization and the Grey Wolf Optimizer, they mimic the elegance of natural intelligence to discover balance in chaos, efficiency in diversity, and order in complexity.

In parallel, Artificial Intelligence and machine learning are redefining the concept of knowledge itself. When united with metaheuristics, they give rise to intelligent optimization, a form of computational intuition capable of learning, adapting, and self-improving. Hybrid Metaheuristics, blending the strengths of different paradigms, extend this vision even further, enabling systems that evolve as living organisms do.

Together, these technologies shape the soul of Industry 4.0, an ecosystem of self-optimizing, resilient, and health-aware systems that reflect both human creativity and machine precision.

This thesis stands at the intersection of Mechatronics, Artificial Intelligence Tools, and Computational Optimization, and seeks to contribute to this ongoing evolution: the pursuit of engineering systems that are not only efficient, but truly intelligent.

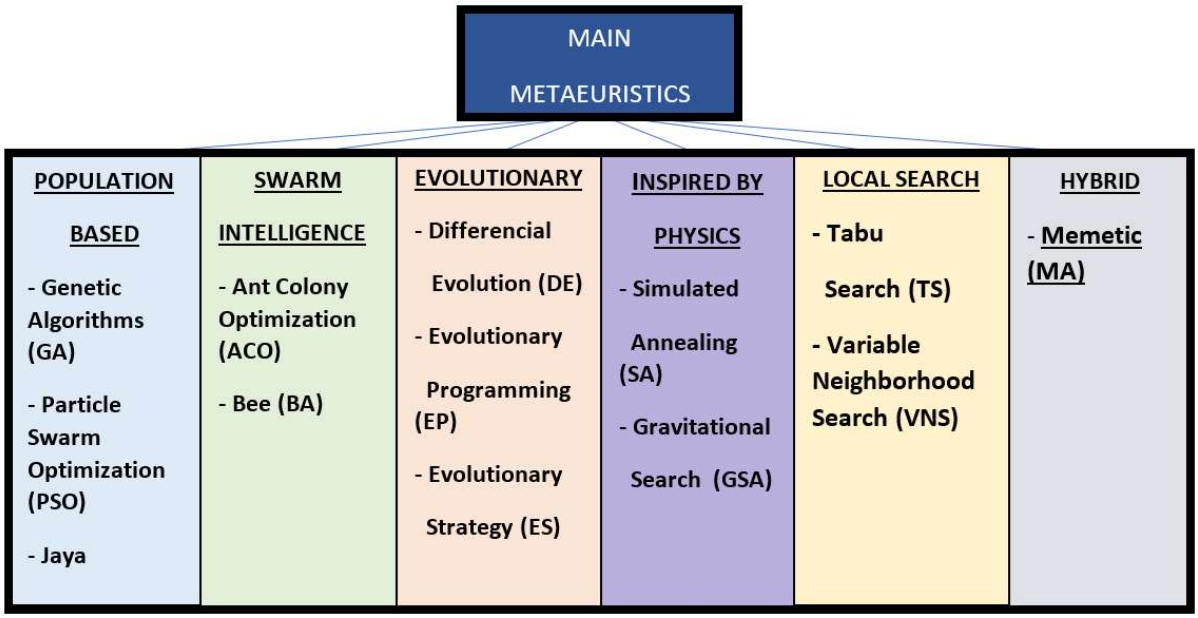
CHAPTER I: NATURE-INSPIRED METAHEURISTICS AND AN IN-DEPTH STUDY OF THE JAYA AND GREY WOLF OPTIMIZER ALGORITHMS

Wave after wave, the traversable sea reveals an endless order that repeats at every scale: what we discover outside, in the forests, mountains, and rivers, also lives within us.

1.1 METAHEURISTICS INTRODUCTION

Complex and nonlinear problem domains are commonly addressed through the application of metaheuristic algorithms due to their ability to efficiently explore solution spaces and find near-optimal solutions. The nomenclature "metaheuristic" denotes a higher-order strategy guiding the exploration of solution spaces. It encapsulates a diverse array of algorithms, united by common attributes such as iterative refinement, solution diversification, and an inherent capacity to evade local optima. The utility of metaheuristics is most pronounced in contexts where exhaustive search methodologies prove computationally prohibitive due to the inherent complexity of the problem at hand.

A preliminary classification proposed in this work outlines six main groups of metaheuristic algorithms, as follows:



Population-based algorithms are those that manipulate multiple candidate solutions simultaneously during the search process. Rather than focusing on a single trajectory, these algorithms explore the solution space collectively through the interaction or parallel evolution of several individuals. The Swarm Intelligence algorithms represent algorithms inspired by collective behaviours observed in nature, such as the foraging patterns of ants or the flocking of birds. These methods rely on the exchange of information among simple agents that cooperate to achieve global optimization. Another important type of metaheuristics is the group of Evolutionary Algorithms (EAs), which explicitly emulate the mechanisms of natural evolution and survival of the fittest. Through processes such as selection, crossover, and mutation, these algorithms iteratively refine populations of candidate solutions. Physics-inspired algorithms form another category in metaheuristic research. These methods are modelled on physical principles and processes, such as thermal annealing, gravitational attraction, or electromagnetic interactions. In contrast, Local Search algorithms operate on a single solution at a time, exploring its neighbourhood to iteratively produce improved solutions. They focus on intensification rather than diversification and are often used to refine solutions found by more global search methods. Finally, Hybrid or Memetic Algorithms integrate elements from different metaheuristic families to exploit the complementary strengths

of global exploration and local exploitation. Although evolutionary algorithms are a subset of population-based approaches, the two categories are not synonymous. All evolutionary algorithms maintain a population of solutions, but not all population-based algorithms are evolutionary. The distinction lies in their underlying metaphor and mechanism of adaptation: evolutionary algorithms are explicitly grounded in biological evolution, where reproduction, mutation, and selection drive progress; population-based algorithms more generally refer to any method in which multiple agents or solutions interact to collectively guide the search process.

Additionally, various subcategories of metaheuristic algorithms can be enumerated, including Water Waves Optimization (WWO) [1], Clonal Selection Algorithm (CLONALG) [2], Chemical Reaction Optimization (CRO) [3], Harmony Search (HS) [4], Sine Cosine Algorithm (SCA) [5], Teaching-Learning-Based Optimization (TLBO) [6], League Championship Algorithm (LCA) [7].

It is evident that metaheuristics algorithms represent a prominent category of optimization methodologies renowned for their efficacy in addressing intricate combinatorial and continuous optimization problems. In stark contrast to conventional optimization techniques reliant on specific problem structures or well-defined mathematical properties, metaheuristics provide adaptive and heuristic-driven strategies, furnishing flexible frameworks for achieving high-quality solutions within reasonable computational timeframes. This expanse of algorithms draws inspiration from an assortment of domains, including natural phenomena, social processes, and diverse problem-solving paradigms, thereby manifesting as versatile tools capable of surmounting a spectrum of optimization challenges.

After outlining the expansive domain of metaheuristic algorithms, the subsequent analysis focuses on a specific and influential subset: population-based metaheuristics, which include evolutionary algorithms [8] and artificial intelligence approaches. Within this broad category, well-established techniques such as Genetic Algorithms (GAs) [9] and Particle Swarm Optimization (PSO) [10] embody the principle of iterative refinement through successive generations. These algorithms utilize mechanisms that emulate adaptation and cooperation among candidate solutions to evolve populations toward optimal or near-optimal states. Through this refined lens, the following sections of this chapter explore two notable representatives of nature-inspired, population-based algorithms, Jaya [11] and the Grey Wolf Optimizer (GWO) [12], unveiling their mechanisms, applications, and unique contributions to the field of optimization. It is important to note that JAYA and GWO are not evolutionary algorithms in the strictest sense (they do not employ genetic operators such as crossover or mutation, nor do they rely on fitness-based selection), yet they are often included within the broader class of population-based metaheuristics because they share core principles such as population evolution, collective adaptation, and iterative search. This transition allows us to delve deeper into the intricacies of algorithms that specifically draw inspiration from the principles of biological evolution to tackle complex optimization challenges. Jaya and the GWO manifest unique methodologies and principles in their quest for optimal solutions to complex optimization problems.

Jaya, despite its simplicity, embodies an effective optimization algorithm inspired by the mechanisms of "improving" and "eliminating" solutions within the population. Unlike

conventional evolutionary algorithms, Jaya eschews explicit evolutionary operators such as crossover or mutation. Instead, it continuously refines existing solutions, improving the best and discarding the less promising. The algorithm adapts its population dynamically, fostering an environment where superior solutions persist, and less fit ones gradually fade away.

In contrast, the GWO draws inspiration from the intricate social hierarchy and cooperative hunting behaviour observed in grey wolves. This population-based algorithm introduces α, β and δ wolves to represent leaders within the pack. The positions of these key individuals are iteratively updated based on their fitness, reflecting the cooperative and competitive dynamics inherent in wolf packs. GWO thus leverages the principles of nature to guide the evolution of a population towards enhanced solutions over successive iterations.

Special emphasis is placed on sections 1.7, 1.8, 1.10, and 1.11, which provide a comprehensive analysis of two case studies, focusing particularly on the detailed implementation and evaluation of their MATLAB code. Specifically, the Jaya algorithm is employed for the design of a dam, while the Grey Wolf Optimization algorithm is systematically applied to the benchmark Ackley function [12], [20] for performance evaluation. The purpose of this analysis is to elucidate the practical implementation and performance outcomes of these algorithms across distinct domains, thereby contributing to a comprehensive understanding of their applicability and efficacy in varied real-world scenarios. For each algorithm an in-depth review will be proposed of the underlying principles, mathematical models, and key concepts that form the basis of these algorithms. By gaining a thorough understanding of the theoretical framework, we aim to provide a solid foundation for the subsequent practical analysis to bridge the gap between theoretical knowledge and practical application, so to consider their potential impact in practical scenarios.

1.2 INTRODUCTION TO EVOLUTIONARY ALGORITHMS

Evolutionary Algorithms (EAs) represent a class of optimization and search heuristics inspired by the principles of natural selection and Darwinian evolution. Rooted in the concept of survival of the fittest, these algorithms iteratively explore solution spaces to find optimal or near-optimal solutions. This introduction delves into the fundamental components of EAs, examining them through a mathematical lens. The main characteristics of the EAs could be summarized in the capabilities of mutation, recombination (crossover), reproduction, selection, and cooperation.

To facilitate the interpretation of the mathematical expressions presented in the following sections, the notation and definitions of all relevant parameters are provided in Table 1.1

Table 1.1-EAs nomenclature

Symbol	Description
$i = 1, \dots, N$	Index of a search agent (N=population size)
D	Number of decision variables (NDV)

Symbol	Description
Max_iter	Number of iterations
$x_i \in R^D$	Decision vector of agent i
$\Delta x_i \in R^D$	Mutation perturbation vector applied to x_i
$x_i^{(p)}, x_i^{(q)} \in R^D$	Decision vectors of two parent agents selected for recombination
$\alpha \in [0,1]$	Random mixing coefficient in recombination
$f(x_i) \in R$	Fitness value of agent i
$s = 1, \dots S$	Index of a cooperative subpopulation
P_s	Set of agents in subpopulation s
$x_i^{(s)} \in R^D$	Decision vector of agent i in subpopulation s
$y_i \in R^D$	Offspring decision vector of agent i

Let N denote the number of search agents (population members), and D the number of decision variables (NDV). Each candidate solution is represented by $x_i = \left[d, \dots, d_{i,D}\right] \in R^D, i = 1, \dots, N$ with associated fitness value $f(x_i) \in$.

Mutation introduces stochastic perturbations to the current population's solutions to promote exploration and can be mathematically represented as

$$y_i = x_i + \Delta x_i \tag{1.1}$$

where x_i is the individual solution and Δx_i denotes a mutation vector. The mutation process injects variability into the population, preventing premature convergence and allowing exploration of the solution space.

Recombination, also known as crossover, emulates genetic crossover in natural reproduction. It combines genetic material from two parent solutions to generate offspring. It can be represented by the following arithmetic equation:

$$y_i = \alpha x_i^{(p)} + (1 - \alpha) x_i^{(q)} \tag{1.2}$$

where $x_i^{(p)}$and $x_i^{(q)}$ are parent solutions and α is a random parameter controlling the mixing ratio. Recombination promotes the propagation of promising features and can lead to the convergence of the population towards optimal solutions.

Reproduction involves the creation of new individuals based on the existing population. The probability of selecting an individual for reproduction is often linked to its fitness, creating a bias towards fitter solutions.

Mathematically, the probability of selecting an individual x_i for reproduction can be defined as

$$P(x_i) = \frac{f(x_i)}{\sum_{k=1}^{N} f(x_k)} \tag{1.3}$$

where $f(x_i)$ denotes the fitness of x_i and N is the population size. This formulation introduces a probabilistic mechanism: individuals with higher fitness values have a greater likelihood of being selected, but selection is not deterministic. Even less-fit individuals retain a nonzero chance of being chosen, which helps maintain genetic diversity and prevents premature convergence.

Selection determines the survival of individuals in the population, akin to the natural selection process. Fitness-based selection methods ensure that individuals with higher fitness have a greater chance of surviving for the next generation. A common method is roulette wheel selection, also known as selection mechanism proportional to fitness, in which the probability of an individual x_i being selected is proportional to its fitness. The formula used for the selection process is the same as the reproduction phase.

Selection plays a pivotal role in driving the algorithm towards convergence by allowing fit individuals to contribute their genetic material to subsequent generations.

Cooperative Evolutionary Algorithms extend the classical EA framework by orchestrating the interplay between multiple subpopulations $S = \{P_1, \dots, P_S\}$. In a cooperative setting, each subpopulation operates independently, contributing to the overall exploration of the solution space. Collaboration among subpopulations allows for a more thorough exploration of diverse regions, leading to enhanced global optimization performance.

Cooperative mutation involves collective adjustments within subpopulations and can be expressed as:

$$y_i^{(s)} = x_i^{(s)} + \Delta x_i^{(s)} \tag{1.4}$$

where $y_i^{(s)}$ and $x_i^{(s)} \in P_s$ represent the $i-th$ mutated and original solutions in subpopulation s, respectively. Cooperative mutation fosters diversity within subpopulations, enabling a richer exploration of the solution space.

Cooperative Recombination goes beyond individual crossovers by facilitating information exchange among subpopulations. Mathematically, cooperative recombination involves the exchange of genetic material between two collaborating subpopulations:

$$y_i^{(s)} = \alpha x_i^{(s)} + (1-\alpha) x_i^{(t)} \tag{1.5}$$

where $x_i^{(t)} \in P_t$, $s \neq t$, and $\alpha \in [0,1]$. Take note that s identifies the subpopulation to which the offspring $y_i^{(s)}$ (and one parent $x_i^{(s)}$) belongs (i.e., where the child will be placed) and t

identifies a different subpopulation from which the second parent $x_i^{(t)}$ is drawn for recombination. This distinction is important because cooperative recombination is defined by cross-subpopulation genetic exchange: the offspring in subpopulation s inherits traits from a parent in s and a parent in another subpopulation $t \neq s$. This differs from within-subpopulation recombination, where both parents would belong to the same s. The index t emphasize that the second parent is drawn from a different subpopulation.

This mechanism ensures that beneficial genetic material from one subpopulation influences the evolution of another, fostering cooperative behaviour.

Cooperative reproduction extends beyond the confines of individual subpopulations. The probability of selecting a subpopulation for reproduction can be mathematically expressed as:

$$P(P_s) = \frac{\sum_{x_i \in P_s} f(x_i)}{\sum_{u=1}^{S} \sum_{x_i \in P_u} f(x_i)} \tag{1.6}$$

Cooperative selection strategies dictate the survival of subpopulations, emphasizing a balanced representation of diverse fitness-based solutions, ensuring that the subpopulations with higher cumulative fitness have a greater chance of enduring.

Table 1.2 contrasts selection and reproduction strategies employed in evolutionary algorithms (EAs), while Figure 1.1 details the operational structure of the genetic algorithm (GA), a specific EA. Figure 1.1.a denotes population evolution across three phases: initialization with a set of candidate solutions; fitness-based ranking incorporating elitism to preserve top-performing individuals while discarding low-fitness candidates; and offspring generation through recombination (crossover) between selected parents. Figure 1.1.b shows the GA process flow, comprising population initialization, fitness evaluation, and termination criterion assessment. If the stopping condition is not met, a new generation is produced via crossover and mutation operators, after which selection determines the individuals propagated to the next iteration.

Table 1.2- Selection and Reproduction in EAs

CONCEPT	SELECTION	REPRODUCTION
Goal	Choose rules for reproduction based on desired trats or natural skills	Creating new individuals (offspring) from selected parents
When	Happens before reproduction	Happens after selection
Focus	Determines who survives or who mates	Determines how offspring are created (via crossover, mutation)
Mechanism	Uses fitness to assign probability or rank	Use genetic operators (crossover and mutation)

CONCEPT	SELECTION	REPRODUCTION
Output	A subset or pool of selected parents	A new generation of individuals

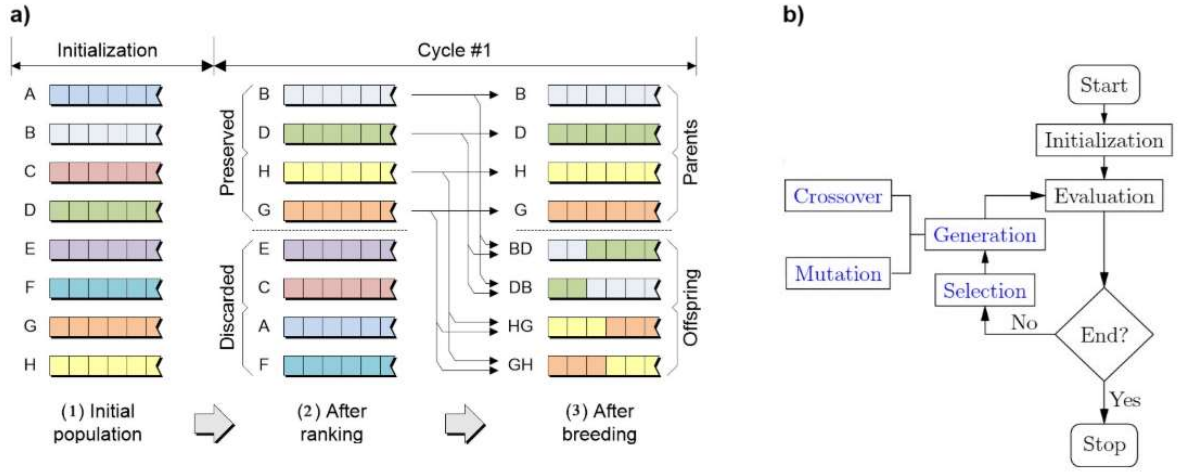


FIGURE 1.1- GENETIC ALGORITHM A) POPULATION EVOLUTION; B) FLOW CHART

1.3 JAYA ALGORITHM

Jaya is an algorithm designed for solving constrained and unconstrained optimization problems, originally proposed by Rao in 2016 [13]. This probabilistic algorithm shares characteristics with other optimization algorithms, such as Generic Algorithm (GA) [9], Particle Swarm Optimization (PSO) [10], Artificial Bee Colony (ABC) [14] and Harmony Search (HS) [15].

An interesting aspect of Jaya is that it belongs to the broader metaheuristic paradigm of collective intelligence. The name "Jaya," derived from the Sanskrit word meaning "Victory," symbolizes the algorithm's goal of moving toward success or improvement in each iteration. Importantly, Jaya can efficiently handle optimization problems without requiring any prior knowledge of the derivative of the objective function, which makes it particularly useful for non-differentiable or complex search spaces.

Like other probabilistic optimization algorithms, Jaya relies on a set of common control parameters, such as population size, number of generations, and elite size. These are often referred to as hyperparameters. In addition to these shared hyperparameters, many algorithms also include algorithm-specific control parameters that require careful tuning. For instance, GA uses mutation probability, crossover probability, and selection operator; PSO employs inertia weight, social and cognitive parameters; ABC involves parameters like the number of onlooker bees, employed bees, scout bees and limits; while the HS algorithm utilizes parameters such as harmony memory consideration rate, pitch adjusting rate and the number of improvisations. Similarly, other algorithms require tuning of their respective algorithm-specific parameters.

The proper adjustment of these parameters is a very crucial factor that affects the performance of the algorithms. Since the tuning of these parameters is problem-specific, each problem type or category requires its own adjustments. Improper tuning may lead to increased computational effort or result in optimal local solutions.

For Jaya, the essential parameters to define are the population size (N) and number of iterations (Max_iter).

Let us consider an optimization problem with D design variables collected in each candidate solution $x_i = [x_{i,1}, \dots, x_{i,D}] \in R^D$. Each variable $x_{i,j}$is bounded between a lower limit $l_j$and an upper limit u_j, such that $l \leq x_i \leq u$, where $l = [l_1, \dots, l_D]$and $u = [u_1, \dots, u_D]$.

A population of N candidate solutions (search agents) is denoted by x_i, with i=1, …, N, each $x_i \in R^D$. The objective function $f(x_i)$ evaluates the fitness of candidate i. The optimization problem is labelled as Equation 1.7 when it involves minimization, and as Equation 1.8 when it involves maximization, defined as follows:

$$\min f(x_i) \text{ s.t. } l \leq x_i \leq u, \text{ i=1, …, N} \tag{1.7}$$

or

$$\min f(x_i) \text{ s.t. } l \leq x_i \leq u, \text{ i=1, …, N} \tag{1.8}$$

At each iteration, the position of a candidate is updated according to:

$$x_i^{new} = x_i + r_1 \cdot (x_{best} - |x_i|) - r_2 \cdot (x_{worst} - |x_i|) \tag{1.9}$$

where x_{best} is the best solution in the population, x_{worst} denotes the worst solution in the population, $r_1, r_2 \in [0,1]$ indicates scalar random values.

The symbol $|\, x_i \,|$denotes the element-wise absolute value vector of the candidate $x_i = [x_{i,1}, \dots, x_{i,D}]$, and all operations are performed component-wise.

The first term moves the candidate toward the best solution, while the second term pushes it away from the worst.

At iteration $1 \leq t \leq Max_iter$, the iterative form is:

$$x_i^{(t+1)} = x_i^{(t)} + r_1 \cdot \left(x_{best}^{(t)} - \left|x_i^{(t)}\right|\right) - r_2 \cdot \left(x_{worst}^{(t)} - \left|x_i^{(t)}\right|\right) \tag{1.10}$$

After updating, for minimization, if $f(x_i^{new}) < f(x_i)$, then the solution is updated; otherwise, it is preserved. For maximization, inequality is reversed.

1.4 SELF-ADAPTIVE JAYA ALGORITHM

The self-adaptive Jaya algorithm [16], akin to other population-based algorithms, shares the fundamental requirement of common control parameters, specifically the population size and

the number of generations. However, it distinguishes itself by not necessitating algorithm-specific parameters, eliminating the need for parameter tuning. The selection of an optimal population size for specific case studies poses a challenging task within the basic Jaya algorithm.

To address this challenge, a self-adaptive Jaya algorithm can be introduced, wherein population sizes are self-adaptive, alleviating the burden of explicitly determining the population size, as follows: assuming a random initial population of size $10 \cdot D$, where D represents the number of design variables, the new population size (n_{new}) is formulated through the expression

$$n_{new} = round(n_{old} + r \cdot n_{old}). \quad (1.11)$$

Here, r is a random number sampled from the range [-0.5,0.5], functioning as a comparative population development rate.

The population size may either decrease or increase due to a negative or positive random value of r. For instance, considering $n_{old} = 20$ and $r = 0.23$, the resulting n_{new} is 25:

$$n_{new} = round(n_{old} + r \cdot n_{old}) = round(20 + 0.23 \cdot 20) = round(24.6) = 25. \quad (1.12)$$

In contrast, if $n_{old} = 20$ and $r = -0.16$, the new population size n_{new} becomes 17:

$$n_{new} = round(n_{old} + r \cdot n_{old}) = round(20 - 0.16 \cdot 20) = 17. \quad (1.13)$$

It is important to note that when $n_{new} > n_{old}$, the entire population advances to the next iteration, with the best optimal solution from the current iteration assigned to the additional $n_{new} - n_{old}$ solutions. Conversely, when $n_{new} < n_{old}$, only the best population from the current solution progresses to the next generation. When $n_{new} = n_{old}$ then no changes occur.

In situations where the population size decreases to a value less than the number of decision variable D, i.e. $n_{new} < D$, the algorithm safeguards against stagnation in local optima by setting $n_{new} = D$.

Table 1.3 introduces a comparison between the traditional Jaya and the Self-Adaptive variant, while Figure 1.2 shows the flow chart of the Self Adaptive Algorithm.

Table 1.3- Traditional vs. Self-Adaptive Jaya Algorithm

STANDARD JAYA	SELF-ADAPTIVE JAYA
Initialize population of solutions	Initialize population of solutions
Evaluate all solutions	Evaluate all solutions
Identify best solution $\bar{x}_{best}$	Identify best solution $\bar{x}_{best}$
Identify worst solution $\bar{x}_{worst}$	Identify worst solution $\bar{x}_{worst}$
For each solution $\bar{x}_i$ in the population,	For each solution $\bar{x}_i$ in the population,
for each dimension D:	for each dimension D:
- generate random numbers $r_1, r_2 \in [0,1]$	**- update adaptive parameters**
-update position: equation 1.9	- update position: equation 1.9 with r_1, r_2 dynamically adapted
If new solution is better:	If new solution is better:
replace the old solution	replace the old solution

STANDARD JAYA	SELF-ADAPTIVE JAYA
Repeat until stopping criterion is met	**Update adaptation parameters based on performance metrics** Repeat until stopping criterion is met

The flow chart of the Self-Adaptive Jaya Algorithm is shown in Figure 1.2- .

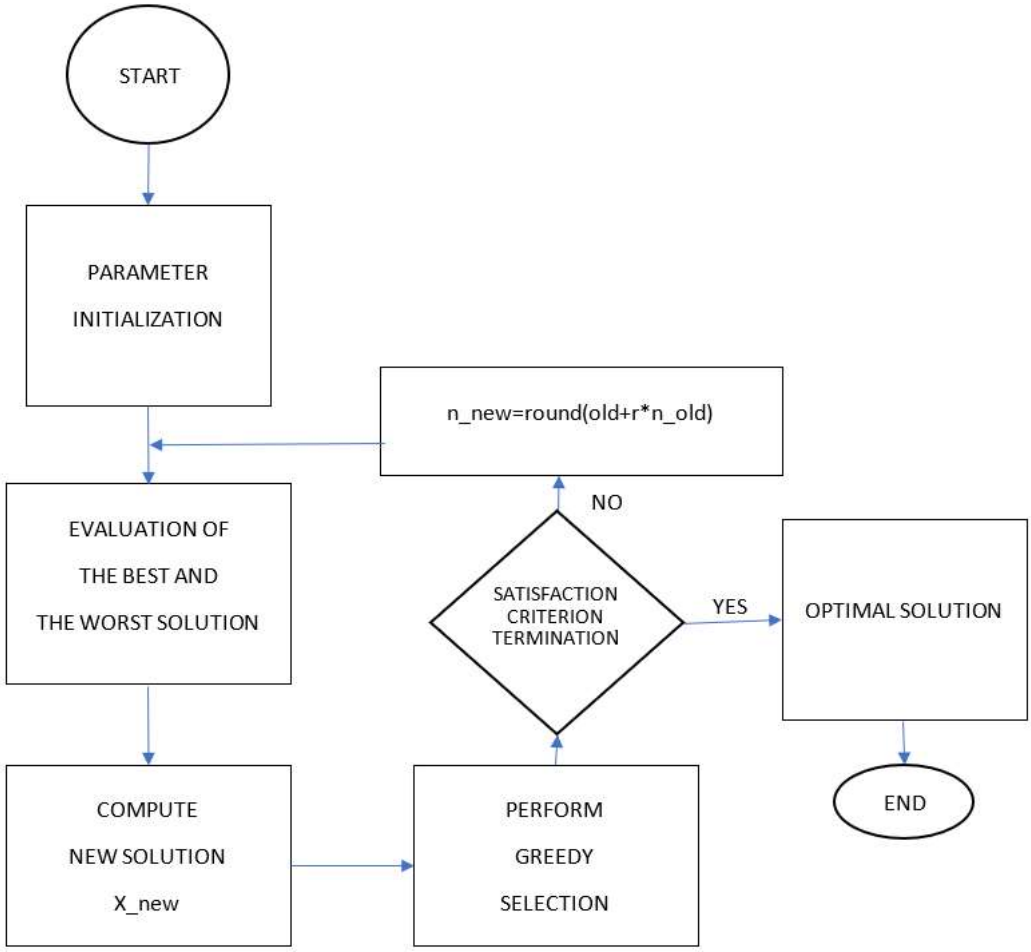


Figure 1.2- SELF-ADAPTIVE JAYA ALGORITHM FLOW CHART

1.5 QUASI-OPPOSITIONAL BASED JAYA (QO-JAYA) ALGORITHM

The quasi-oppositional based Jaya Algorithm [17] allows for an accelerated convergence rate. It involves the generation of an opposite population in tandem with the existing population, such that both are considered at the same time to facilitate improvement approximations.

To construct this opposite population, the mirror point of each variable within a candidate solution is first defined as $p'_{i,j} = L + U - p_{i,j}$. Here, L and U represent the lower and the upper bounds of the variable. The center of the search space is given by $c = \frac{(L+U)}{2}$. The quasi-opposite value of a variable is then randomly selected between the center c and its mirror point $p'_{i,j}$, as expressed by

$$p_{i,j}^{q} = rand\left(c, p'_{i,j}\right) \tag{1.14}$$

where $p_{i,j}^{q}$ represents the quasi-opposite value corresponding to $p_{i,j}$.

For instance, assuming $L = 10$ and $U = 60$, with the current iteration's variable value $p_{i,j} = 23$, the calculations yield $c = 35$ and $p'_{i,j} = 10 + 60 - 23 = 47$. Consequently, the quasi-opposite value $p_{i,j}^{q}$ is determined as:

$$p_{i,j}^{q} = rand(35{,}47) \tag{1.15}$$

This implies that any number within the range of 35 to 47 will be randomly generated as the quasi-opposite value. This formulation preserves the stochastic nature of the Jaya algorithm while introducing diversity through the quasi-opposite population $P^{q} = \{p_1^{q}, p_2^{q}, \dots, p_N^{q}\}$, thereby improving the exploration capability of the search process.

The flow chart of the Quasi-Opposite Jaya Algorithm is shown in Figure 1.3- .

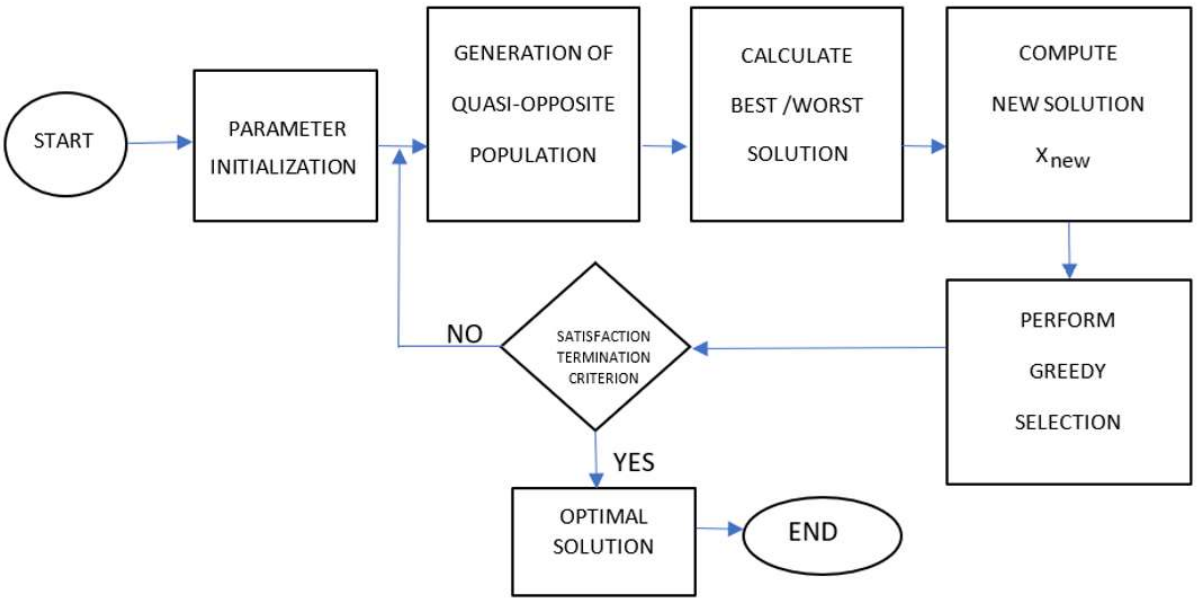


Figure 1.3- QUASI-OPPOSITE JAYA ALGORITHM FLOW CHART

1.6 SELF-ADAPTIVE MULTI-POPULATION (SAMP) JAYA ALGORITHM

In the realm of advanced optimization methods, the integration of multi-population frameworks stands as a crucial strategy aimed at augmenting search diversity. This involves partitioning the entire population into distinct groups or sub-populations and allocating them across the search space to effectively detect variations in the problem.

The employment of multi-population methodologies proves particularly advantageous for the preservation of population diversity. This basic concept aims to sustain the diversity of the search process by allocating different sub-populations to different regions.

One pivotal challenge lies in determining the optimal number of sub-populations, a parameter intricately linked with the complexity of the underlying problem. The dynamic nature of sub-population sizes throughout the search process poses another significant hurdle. Additionally, there is the concern of potential inadequacy in the diversity of solutions within these sub-populations.

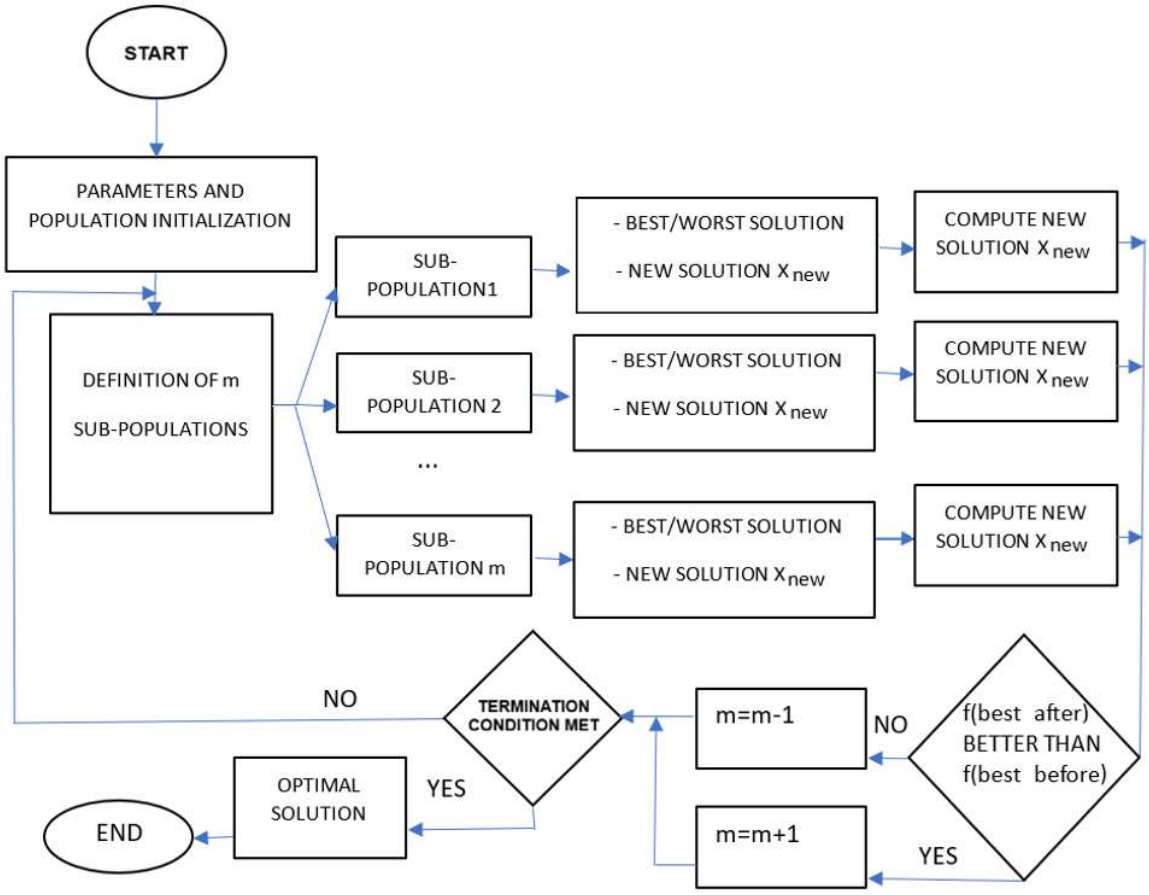


Figure 1.4- SAMP-JAYA ALGORITHM FLOW CHART

To address these challenges, a novel approach has been proposed: the Self-Adaptive Multi-Population (SAMP) Jaya algorithm [18]. The flow chart of the SAMP is shown in Figure 1.4-
.

This algorithm introduces a dynamic adaptation mechanism, aiming to overcome the identified challenges.

The fundamental steps of the SAMP-Jaya Algorithm encompass defining key parameters, such as the population size, design variables, and termination criterion.

It is subsequently computed the initial solutions and the fitness values, grouping the entire population into "m" distinct groups based on solution quality, employing the Jaya algorithm

independently within each sub-population for solution modification, consolidating all sub-populations, and performing Greedy selection.

The latter involves assessing whether the fitness of the best solution before modification surpasses that after modification.

Based on this assessment, the algorithm dynamically adjusts the number of sub-populations to strike a balance between exploration and exploitation.

For instance, check whether $f_{best\ before}$ is better than $f_{best\ after}$.

If yes, increase $\mathrm{m} = \mathrm{m} + 1$ with the aim of increasing the exploration feature of the search process.

Otherwise, $\mathrm{m} = \mathrm{m} - 1$, as the algorithm needs to be more exploitive than explorative.

The SAMP-Jaya finishes checking the stopping criterion.

1.7 APPLICATION OF JAYA ALGORITHM TO A DAM DESIGN

The most straightforward approach to implementing the JAYA algorithm and comprehending its functioning lies in its application to the design process of a static structure.

In this paragraph, we will delve into the scenario of a concrete gravity dam and the application of the JAYA technique to optimize its form.

This analysis is predicated upon an authentic case study derived from an academic undertaking concerning a dam situated in the Shafaroud region of Northern Iran. This case study was selected because it represents a real-world, data-driven optimization problem with practical engineering relevance.

The inaugural step to undertake involves defining the geometry of the structure to be optimized. The dam can be conceptualized as an object obtained through the extrusion of a specific cross-sectional profile. Therefore, the starting point is the delineation of the characteristic dimensions that will define the shape of the cross-sectional profile, as illustrated in Figure 1.5- :

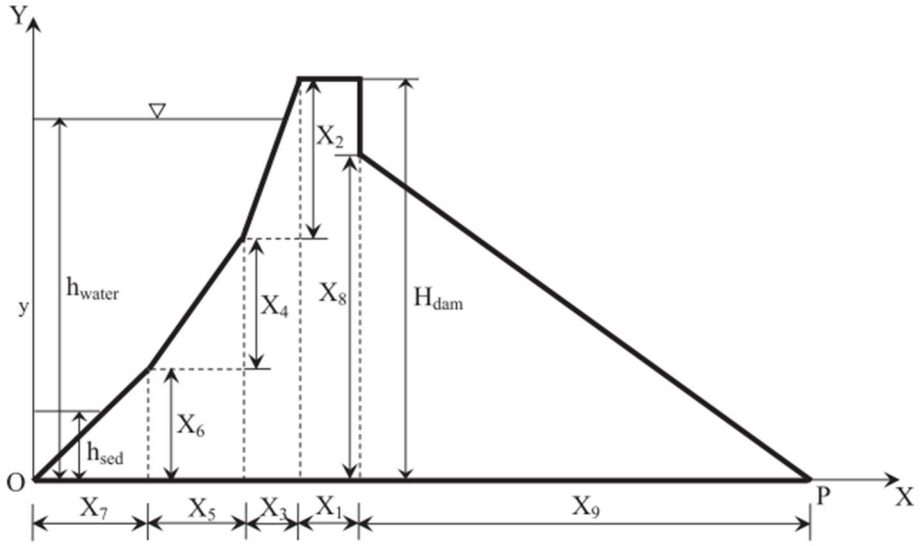


Figure 1.5- GEOMETRY OF THE DAM'S CROSS SECTION

Once the characteristic dimensions are established, it becomes imperative to identify an objective function that specifically pertains to these dimensions and encapsulates the desired outcomes.

What is typically sought in structural design is achieving maximum performance in terms of mechanical resistance while minimizing weight.

The weight M [kg], in turn, depends on the volume of the structure V [m^3] and the type of material, as defined by its specific weight ρ [kg/ m^3]:

$$M = \rho \cdot V \tag{1.16}$$

Hence, the objective function will be represented by the volume of the dam, which is contingent upon the geometric design parameters. The optimization type will be to minimize the weight, and consequently, the volume.

Therefore, the objective of the optimization problem is to minimize the volume per unit width of the dam, denoted as V_{dam}. The optimization process involves nine design variables, (X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9), as illustrated in Figure 1.5. The objective function is formulated as:

$$Min(V_{dam}) = H_{dam}X_1 + \frac{1}{2}X_2X_3 + (X_4 + X_6)X_3 + \frac{1}{2}X_4X_5 + X_5X_6 + \frac{1}{2}X_6X_7 + \frac{1}{2}X_8X_9 \tag{1.17}$$

This formulation establishes the mathematical basis for the optimization process, where the goal is to determine the set of design variables that minimize V_{dam}while satisfying the problem constraints.

The dimensions of the dam under consideration are characterized by a height H_{dam} measuring 150 meters. The water elevation in the normal operational state, denoted as h_{water}, located

upstream of the dam is established at 145 meters. Additionally, the sediment height h_{sed} is stipulated at 7 meters.

In addition to the objective function, the optimization must satisfy a set of constraints related to the dam's geometry and structural performance, expressed as follows:

$$X_2 + X_4 + X_6 \leq H_{dam} \tag{1.18}$$

$$X_4 + X_6 \leq X_8 \tag{1.19}$$

$$SFS \geq 4 \tag{1.20}$$

$$SF0 \geq 1.5 \tag{1.21}$$

$$0 \leq \sigma_u \leq \sigma_{max} \tag{1.22}$$

$$0 \leq \sigma_D \leq \sigma_{max} \tag{1.23}$$

where SFS represents the safety factor against dam sliding, SFO denotes the safety factor against dam overturning, the variables σ_u, σ_D, and σ_{max} correspond to the vertical fatigue specific loads in upstream and downstream, as well as the respective fatigue limit.

The safety factor is defined by the following equation

$$SFS = \frac{(f \cdot \Sigma F_V + \sigma \cdot b)}{\Sigma F_H} \tag{1.24}$$

where f is the static friction coefficient, $\sigma \left[\frac{N}{m^2}\right]$ is the allowable shear stress along the sliding surface, and b is the base width of the dam, defined as $b = X_7 + X_5 + X_3 + X_1 + X_9$. The term F_V represents the resultant of the vertical forces acting on the dam and is given by $F_V = -W - F_{h_V} - P_{SV} + F_{\text{UPLIFT}} + F_E$, where W is the self-weight of the dam, F_{h_V} is the vertical component of the hydrostatic pressure, P_{SV} is the vertical component of the sediment pressure, F_{UPLIFT} is the uplift force acting at the dam-foundation interface, and F_E is the vertical component of the earthquake force. Similarly, F_H denotes the resultant of the horizontal forces acting on the dam, defined as $F_H = F_{h_H} + P_{SH}$, where F_{h_H} is the horizontal component of the hydrostatic pressure, and P_{SH} is the horizontal component of the sediment pressure.

The SFO factor is defined by the following equation

$$SFO = \frac{\Sigma M_R}{\Sigma M_o} \tag{1.25}$$

where M_R is the torque of resisting forces (W , Fh_V, P_{SV}) and M_O is the torque of the driving forces (F_{UPLIFT},FH_H,P_{SH}) on the dam.

The vertical fatigue loads σ_u and σ_D are characterized by the subsequent equation:

$$\sigma_u = \frac{\Sigma F_V}{b} - 6\frac{\Sigma M_o}{b^2} \quad (1.26)$$

$$\sigma_D = \frac{\Sigma F_V}{b} + 6\frac{\Sigma M_o}{b^2} \quad (1.27)$$

Finally, the side constraints imposed on the shape variables are defined as follows: $1\,m \leq X_1, X_3 \leq 20\,m$, $1\,m \leq X_2 \leq 50\,m$, $5\,m \leq X_4, X_5, X_6, X_7 \leq 100\,m$, $50\,m \leq X_8, X_9 \leq 150\,m$.

In the developed simulation the total height of the dam was constrained to remain fixed at 150 meters, and the earthquake force was subtracted from the dam's gravity load, resulting in a reduction of the resultant resistance moment (ΣM_R) and rendering the compliance with the constraint on overturning (i.e., SFO ≤ 0) more challenging. Introducing an extra horizontal earthquake force in the reservoir could have potential to exacerbate overturning. Major details of the code are provided in Appendix 2.

1.8 JAYA ALGORITHM: RESULTS AND DISCUSSION ON THE DAM EXAMPLE

The outcomes derived from the optimization computations employing the Jaya algorithm are systematically documented in an output text file denominated as "ITERATIONS.txt."

Within the course of each algorithmic iteration, pertinent values pertaining to the optimized variables and the objective function are meticulously archived in this designated file.

Our design variables amount to 9 and encapsulate the characteristic design parameters of the cross-sectional profile of a dam, as illustrated in Figure 1.6.

In the code, predetermined ranges of values for variables have been established to "guide" the sizing process, employing the following values: $X_1 \in [1,20]$, $X_2 \in [20,50]$, $X_3 \in [1,20]$, $X_4 \in [40,80]$, $X_5 \in [10,30]$, $X_6 \in [30,80]$, $X_7 \in [20,40]$, $X_8 \in [50,150]$, $X_9 \in [80,150]$. Upon executing the code, the concluding section of the text file facilitates the visualization of the optimized variable values alongside the final value of the objective function. The physical significance of the optimized variables is discernible, representing quantities that dictate the dimensions of the dam, as delineated in the previously presented Figure 1.6. The objective function enables the assessment of the overall weight of the structure, given the specified material type. The material type is unequivocally defined by specific weight within the "CONSTRAINTS" function, and it is concrete with a specific weight of 23 kN/m^3). After examining the achieved outcomes, the identified values for the desired design parameters are as follows: $X_1 = 5\,m$, $X_2 = 49.999845\,m$, $X_3 = 1.005476\,m$, $X_4 = 67.494241\,m$, $X_5 = 18.237879\,m$, $X_6 = 32.488254\,m$, $X_7 = 39.614486\,m$, $X_8 = 99.960530\,m$, $X_9 = 80.012045\,m$. The final total volume of the dam for unit thickness of 1 m is about 6726.7 m^3. We can remark that this value is better than the one reported in literature (between 6830.2 and 6834.3 m^3) [158]. The total number of analyses made is 3027, after 650 iterations, as shown in Figure 1.6- .

```
648 6726.694223 6753.926127
648 6726.694223 6753.926127
649 6726.694223 6735.616157
649 6726.694223 6735.616157
650 6726.668130 6731.252135
650 6726.668130 6731.252135
XDES(1) = 5.000000
XDES(1) = 5.000000
XDES(2) = 49.999845
XDES(2) = 49.999845
XDES(3) = 1.005476
XDES(3) = 1.005476
XDES(4) = 67.494241
XDES(4) = 67.494241
XDES(5) = 18.237879
XDES(5) = 18.237879
XDES(6) = 32.488254
XDES(6) = 32.488254
XDES(7) = 39.614486
XDES(7) = 39.614486
XDES(8) = 99.960530
XDES(8) = 99.960530
XDES(9) = 80.012045
XDES(9) = 80.012045
DESIGN FINALE 6726.668130 kg
DESIGN FINALE 6726.668130 kg
NUMERO DI ANALISI 3027
6726.185527 0 0.000483
6726.185527 6726.668130
```

Figure 1.6- FINAL OUTPUT OF THE JAYA ALGORITHM FOR DAM OPTIMIZATION

1.9 THE GREY WOLF ALGORITHM

In the natural realm, the execution of tasks often demands collaborative efforts due to their inherent complexity. Particularly in the pursuit of ambitious goals, it becomes imperative to employ methodologies grounded in group cooperation and coordination. Within the context of group dynamics, the structuring of roles can be accomplished through diverse frameworks, with more straightforward models aligning with hierarchical and functional paradigms. In the hierarchical structure, individual roles are assigned within a group accompanied by established rules that define permissible actions and interaction protocols. This arrangement follows a pyramid-shaped hierarchy, wherein operational members at the base carry out routine and repetitive tasks as directed at higher-ranking levels. It's noteworthy that the foundational tier involves straightforward responsibilities, while the upper echelons require advanced strategic capabilities, encompassing analytical acumen. Adept information processing, optimal solution identification, authoritative leadership, and persuasive skills. This organizational structure aligns seamlessly with the principles of management. In particular, the adoption of a hierarchical approach in a design process becomes fundamental when the level of complexity and the requirements of actions based on a strict sequence of commands and controls are extremely high. This occurs especially when the project demands elevated levels of security.

Within this context the exploration of the novel metaheuristic algorithm named Grey Wolf Optimizer [12] appears to be intriguing.

The Grey Wolf Optimizer (GWO) algorithm emulates the hierarchical leadership structure and hunting strategies observed in grey wolves in the natural environment.

The simulation is based on distinguishing four distinct roles within wolf packs, denoted as α, β, δ, and ω, and three crucial phases: prey, encircling prey, and executing the attack (see Figure 1.7-).

The highest hierarchical level is held and enforced by the α wolf (dominant wolf), who has decision-making authority over all pack actions (identification of locations, wake-up and sleep patterns, hunting, etc.). The latest research has shown that the identification of the α wolf is based not on strength but rather on cooperation and familial structure, individual skills acknowledged by the pack, age, or generational hierarchical position.

β grey wolves exhibit submissive behaviours only towards α wolves and take over in the event of the death or decline in performance due to illness or old age of the alphas. Their role within the pack is to assist and support the α wolves in managing the group life.

The third hierarchical level is occupied by the δ wolf, which tends to be subordinate to the β wolf, has no direct interactions with the α and serves as an intermediary between the base and the hierarchically superior level. Predominantly their role is scouts (collection of information by observations, assessment, and preparation for informed decision-making), sentinels (detecting and alerting others to potential threats or dangers), elders (accumulate knowledge based on their experiences), hunters (guarantee sustenance or resources' acquisition), and caretakers (provide nurturance, support, and attention to the well-being and needs).

The base of the pack is marked by ω wolves, assigned to the role of scapegoats. Typically utilized as caregivers within the pack, they are assigned simple and repetitive tasks devoid of

complex logical elaboration. Scientific investigations substantiate that the loss of ω wolves heighten frustration levels in the higher pack hierarchies, precipitating instances of conflicts and aggressions.

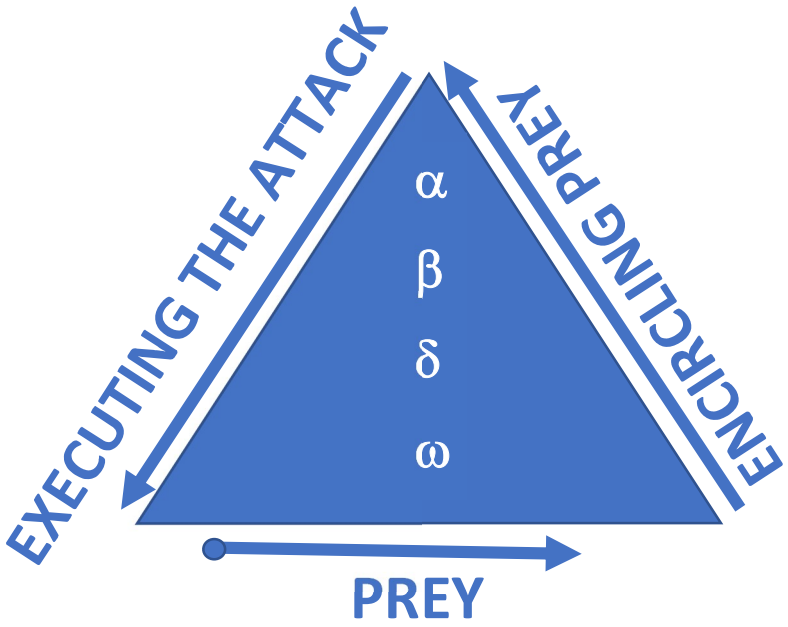


Figure 1.7- PHASES AND HIERARCHY OF THE GREY WOLF PACK

Muro et al. in 2011 [19] conducted an initial examination of the primary phases of social behaviour within a Grey Wolf pack, delineating the process as comprising:

- Tracking, chasing, and approaching prey
- Pursuing, encircling, and harassing the prey until it ceases movement
- Executing the attack on the prey

The hunting, guided by the α, β, δ wolves while the ω follows them, is the optimization process.

The mathematical model must encompass all these three phases and the hierarchical structure within the wolf pack, as follows:

- α represents the fittest solution
- β signifies the second-best solution
- δ denotes the third best solution
- ω embodies the rest of the candidate solutions.

The encircling prey phase (see Figure) is described by the following equations:

$$X(t+1) = X_p(t) - A \cdot D \tag{1.28}$$

$$A = 2\,a \cdot r_1 - a \tag{1.29}$$

$$C = 2\ r_2 \tag{1.30}$$

$$D = C \cdot X_p(t) - X(t) \tag{1.31}$$

where

X is the position vector of a grey wolf

X_p indicates the position vector of the prey

a is a coefficient vector linearly decreasing from 2 to 0 over the course of iterations

r_1 and r_2 are random vectors in [0.1].

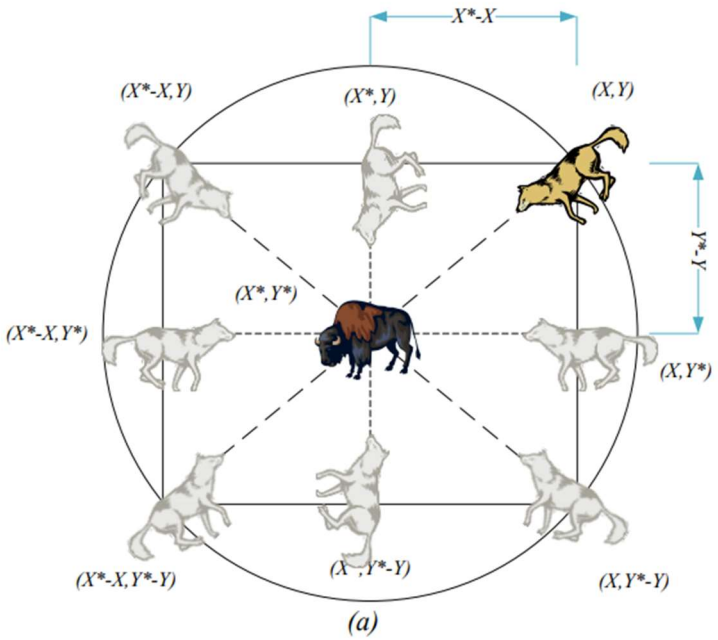


Figure 1.8- GEOMETRY OF THE ERCINGLING PREY PHASE IN 2D

For instance, in the Figure 1.8, the grey wolf is in the position (X, Y) and the prey is in the position (X*, Y*). The new position of the wolf (X*-X, Y*) can be arranged choosing a , r_1,r_2 such that A= (1,0) and C= (1,1).

The identical procedure holds true within a search space of n dimensions, in which the grey wolves move in hyper-cubes or a hyper-sphere, as illustrated in the Figure 1.9:

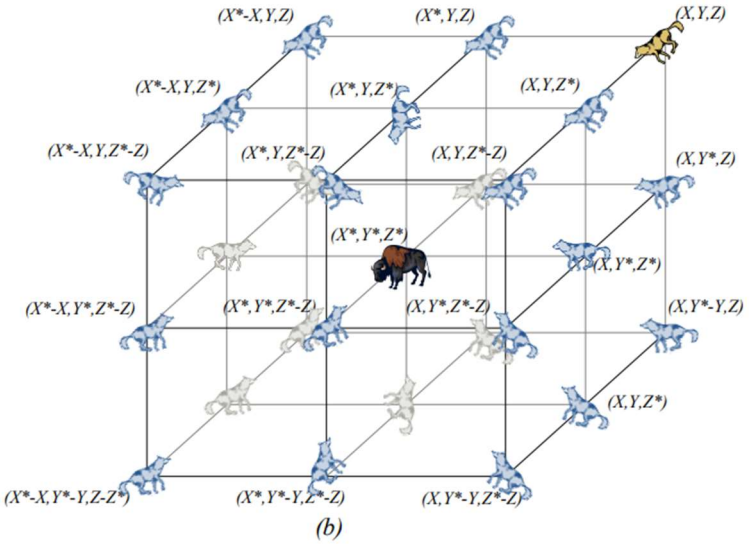


Figure 1.9- GEOMETRY OF THE ERCINGLING PREY PHASE IN 3D

The hunting phase is delineated by the subsequent equations:

$$X(t+1) = \frac{X_1 + X_2 + X_3}{3} \tag{1.32}$$

$$X_1 = X_\alpha - A_1 \cdot D_\alpha \tag{1.33}$$

$$X_2 = X_\beta - A_2 \cdot D_\beta \tag{1.34}$$

$$X_3 = X_\delta - A_3 \cdot D_\delta \tag{1.35}$$

$$D_\alpha = |C_1 \cdot X_\alpha - X| \tag{1.36}$$

$$D_\beta = \left|C_2 \cdot X_\beta - X\right| \tag{1.37}$$

$$D_\delta = |C_3 \cdot X_\delta - X| \tag{1.38}$$

where

X is the position vector of a grey wolf

X_1 indicates the fittest solution developed by the α wolf

X_2 specifies the second-best solution enhanced by the β wolf

X_3 denotes the third- best solution recognized by the δ wolf

A, C, D are coefficient vectors defined as described in the phase above and related respectively to the α, β, and δ wolf.

The logic associated with these equations is straightforward: the hunting phase is led firstly by the α wolf, who possesses a superior understanding of the prey's position. Additionally, both the β and δ wolves contribute to advancing toward prey, augmenting the collaborative hunting effort. ω wolf and the other search agents update their position according to the position of the best search agent, as shown in the Figure 1.10- .

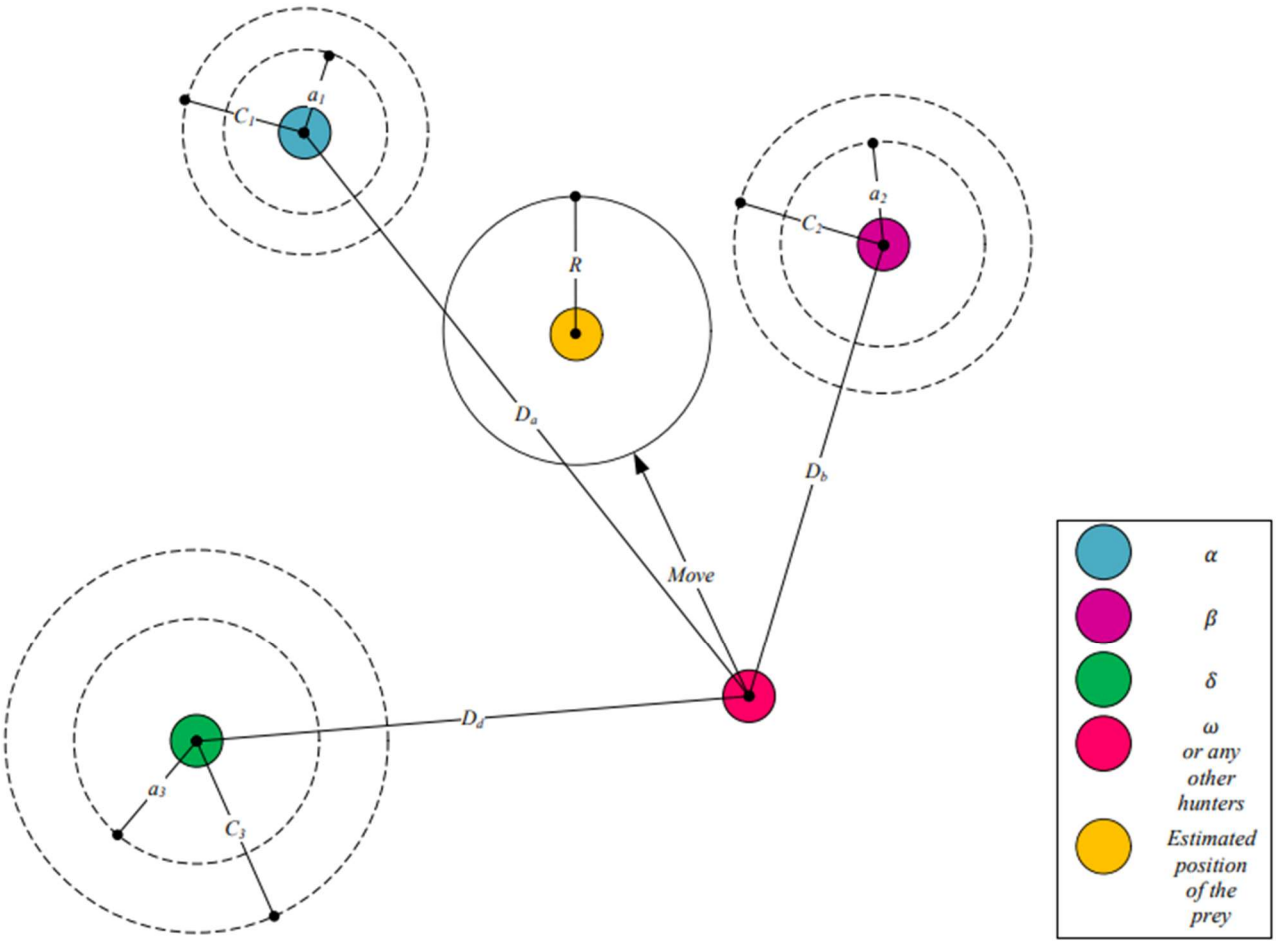


Figure 1.10- UPDATING POSITION PROCESS TOWARD THE PREY (YELLOW CIRCLE)

The attacking prey (exploitation) phase is characterized by the following logical sequence:

The coefficient vector a is decreased from 2 to 0 over the course of iterations, such that the random matrix A, defined in [-a, a], becomes

$$X|A| < 1 \qquad (1.39)$$

in a manner that allows the search agent to potentially occupy any position between its current location and the prey's location in the next step.

This means that the wolves are compelled to attack the prey, as intuitively understood from the following graphical representation of the attack (Figure 1.11-):

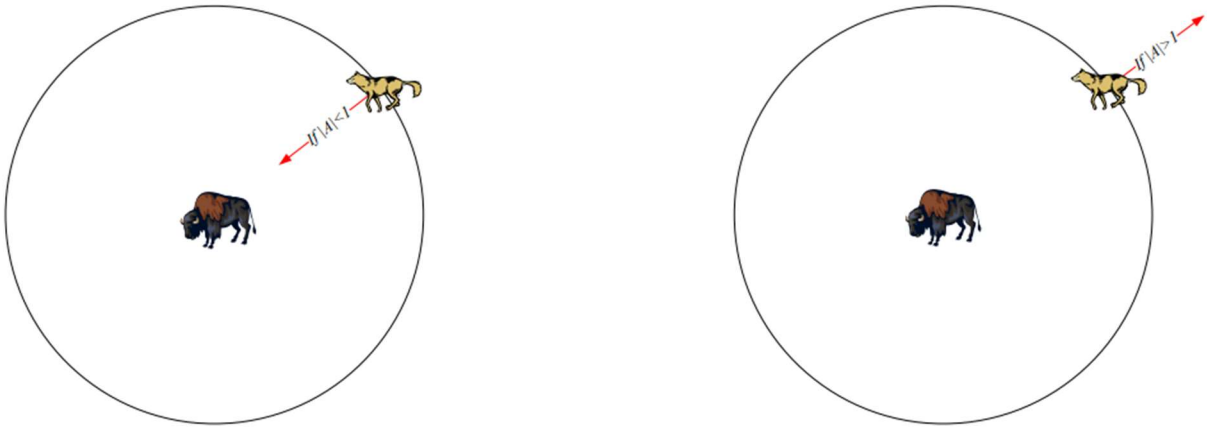


Figure 1.11- ATTACK PHASE REPRESENTATION

The suggested encircling mechanism exhibits a certain level of exploration, which can be further enhanced through the incorporation of supplementary operators.

For example, recognizing that the exploratory phase essentially involves searching for prey, it can be enhanced by manipulating the vector $C = 2 \cdot r_2$, where r is defined in the interval [0,2].

If C > 1, then there is a stochastic emphasis on the influence of the prey in determining the distance (increasing of exploration) and vice versa if C < 1. This suggests that C generates random values throughout all iterations, proving particularly advantageous in mitigating local optima stagnation, especially in the concluding iterations.

Briefly, the C vector mimics obstacles encountered by wolves in nature, hindering their efficient approach to prey. Depending on a wolf's position, the vector introduces random weights to the prey, either impeding or facilitating the approach.

Significant examinations can be conducted regarding the holistic character of the algorithm: the suggested encircling mechanism delineates a circular neighborhood around the solutions, extensible to higher dimensions in the form of hyper-sphere. Stochastic parameters, denoted s A and C, contribute to the generation of candidate solutions characterized by hyper-spheres with distinct random rays. This method empowers candidate solutions to determine the likely position of the prey. The balance between exploration and exploitation is assured through the adaptive value of A. The dynamic adjustment of parameters and A facilitates a smooth transition between exploitation and exploration phases. Specifically, as A decreases, half of the iterations are devoted to exploration ($|A| \geq 1$), while the remaining half is dedicated to exploitation ($|A| < 1$). It is noteworthy that the GWO involves only two primary parameters necessitating adjustment, namely "a" and "C".

The Grey Wolf Optimizer flow chart is presented as in Figure 1.12- :

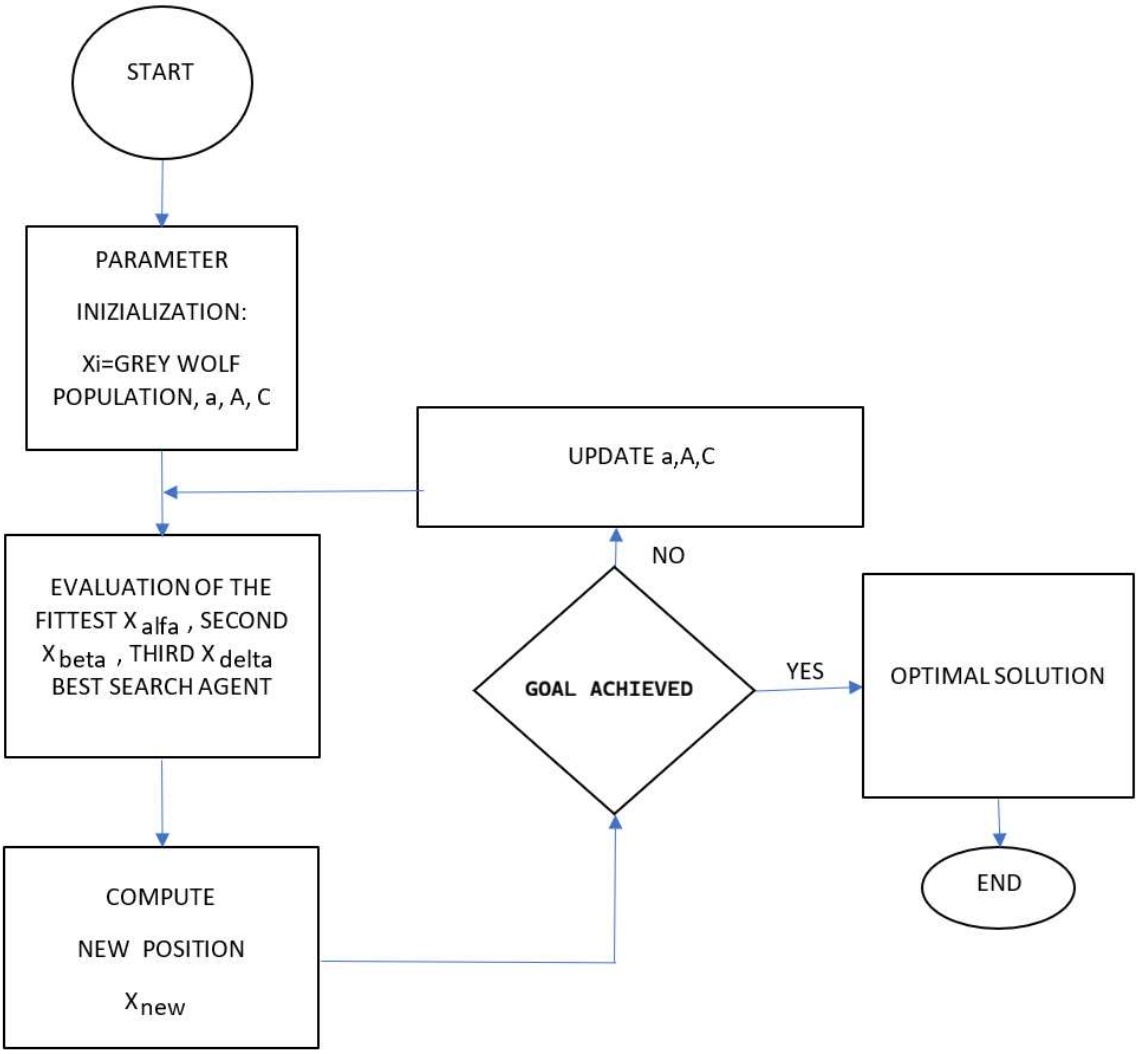


Figure 1.12- GREY WOLF OPTIMIZER FLOW CHART

Certainly, there are prospective avenues for incorporating mutation and additional evolutionary operators, with the goal of simulating the entire life cycle of grey wolves. Major details of the code are provided in Appendix 1.

1.10 APPLICATION OF GWO ALGORITHM TO ACKLEY FUNCTION

A noteworthy aspect regarding the GWO method is tied to how some Iranian researchers, Seyedali Mirjalili, Shima Moayedi, Mohammed Reza Meybodi conceived it in 2014 [12], simply observing the hunting behaviour of grey wolves in nature. This algorithm is addressed for solving complex non-linear problems with irregular search spaces and excels in multi-modal optimization scenarios and handling noisy or stochastic objective functions.

When a novel optimization algorithm is introduced, its performance is typically evaluated using a diverse suite of benchmark functions. These mathematical formulations are specifically designed to assess an algorithm's accuracy, robustness, and generalization capability across a range of problem types. The suite often includes algebraic, non-linear, stochastic, and constrained functions, enabling systematic and reproducible testing. This approach is well-established in the high-impact literature and forms a cornerstone of algorithmic validation in both theoretical and applied computational research [12], [20], [21], [22].

To better understand the performance characteristics of the GWO algorithm, the Ackley function is employed as a representative benchmark. Known for its complex and highly multi-modal landscape, the Ackley function poses a significant challenge to optimization methods, particularly in terms of exploration-exploitation balance.

In this study, the optimization problem is formulated in a dimensional search space pair to 30 (dim=30), with the objective function defined as follows:

$$f(X) = -20 \cdot \exp\left(-0.2 \cdot \sqrt{\frac{\Sigma X^2}{dim}}\right) - \exp\sqrt{\frac{\Sigma \cos(2\pi \cdot X)}{dim}} + 20 + \exp(1) \tag{1.40}$$

This function features numerous local minima, making it particularly challenging for local descent-based methods, which often become trapped in suboptimal regions. Despite its complexity, the function has a known global minimum at **X**=**0**, where f (0) = 0, providing a reliable reference point for evaluating algorithmic performance.

Because of these characteristics, the Ackley function is frequently included in standardized benchmark suites, such as the CEC benchmark sets, and serves as a rigorous test case for assessing the efficacy, convergence dynamics, and generalizability of optimization algorithms. Specifically, it enables detailed analysis of GWO's convergence behavior, its ability to escape local optima, and its overall robustness in tackling non-linear, multi-modal optimization problems.

1.11 GWO ALGORITHMS: CODE RESULTS AND DISCUSSION ON ACKLEY FUNCTION EXAMPLE

The results obtained by the Grey Wolf Optimizer (GWO) for the Ackley function indicate an exceptionally high level of precision in the optimization process (see Figure 1.13-). The best solution vector components reported are all in the order of e−14, demonstrating that the algorithm has effectively converged very close to the global optimum located at **X**=**0**.

This is further corroborated by the best objective function value found by GWO, which is approximately 8.5709e−14, a value practically indistinguishable from zero considering numerical precision limitations.

```
The best solution obtained by GWO is : -1.5723e-14 2.5701e-14 1.4161e-14  2.3187e-14
2.5111e-14 -1.6903e-14  1.3224e-14  1.6724e-14 -2.2841e-14 -2.6388e-14  1.7414e-14
2.6189e-14 -2.3691e-14  2.2088e-14  1.9665e-14  1.2492e-14  -1.814e-14
2.3912e-14 -3.0902e-14 -2.7839e-14
2.3077e-14 -3.0124e-14 -2.0215e-14 -1.5308e-14 -2.4957e-14 -1.2527e-14  1.5415e-14
2.4123e-14 -1.6976e-14  2.2453e-14

The best optimal value of the objective funciton found by GWO is : 8.5709e-14
```

Figure 1.13- GWO RESULTS DISPLAYED ON THE MATLAB COMMAND WINDOW FOR THE ACKLEY FUNCTION

Such results reflect the strong capability of GWO to navigate the complex, multi-modal landscape of the Ackley function, avoiding local minima traps and converging towards the known global minimum. The near-zero values of both the solution components and the objective function underscore the algorithm's robustness and accuracy when addressing non-linear optimization problems with numerous local optima.

Additionally, a detailed visualization of the optimization process executed by the Grey Wolf Optimizer (GWO) is presented through a three-dimensional surface plot, commonly referred to as the Fitness Landscape Plot, alongside a convergence curve to illustrate the algorithm's progression (see Figure 1.14-).

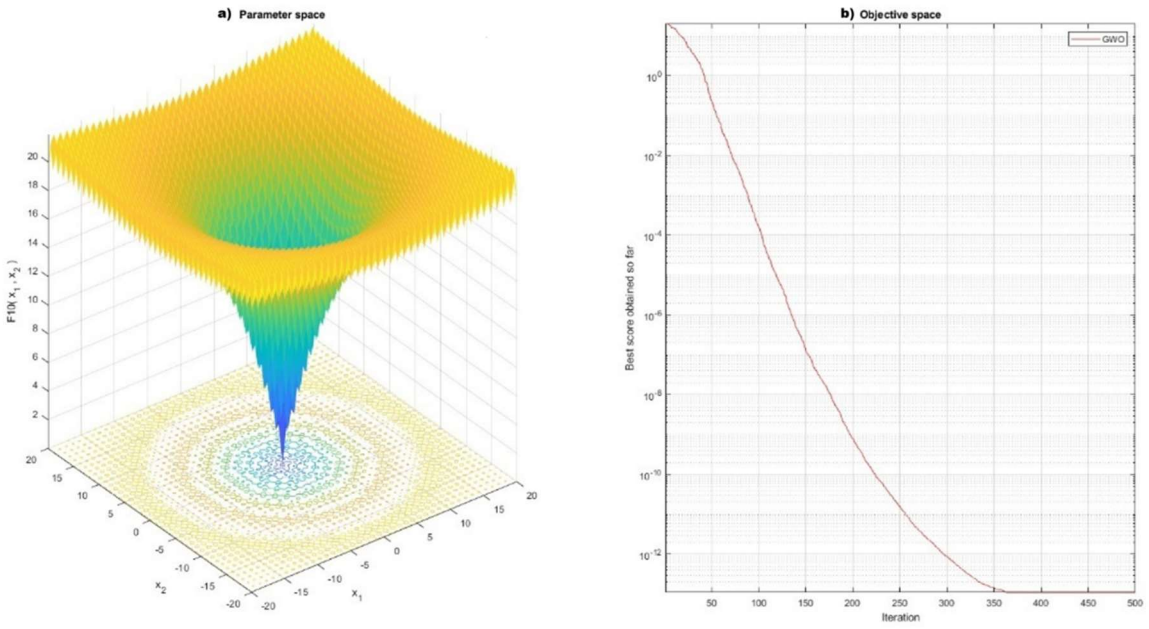


Figure 1.14- GWO ACKLEY FUNCTION OPTIMIZATION: a) FITNESS LANDSCAPE PLOT; b) CONVERGE CURVE

The three-dimensional surface plot (Figure 1.14- .a) illustrates the landscape of the Ackley function by selecting two variables for visualization while keeping the others fixed, when more than two are present. In this case, the first two decision variables are varied, and the vertical z-axis represents the corresponding objective function values, f (x1, x2). The plot clearly reveals the highly non-convex and multi-modal nature of the function, characterized by numerous local minima surrounding a single global minimum at (x1, x2) = (0,0), where f (0,0) = 0. The sharp and narrow global basin visually emphasizes the complexity of the optimization task, highlighting the necessity for robust global search strategies such as the Grey Wolf Optimizer (GWO).

The second graph (Figure 1.14- plots the convergence curve of GWO, where the x-axis denotes the number of iterations and the y-axis (in logarithmic scale) represents the best objective value (fitness score) found so far. The curve exhibits a monotonically decreasing trend, indicating consistent progress toward the global minimum. After approximately 362 iterations, the best score stabilizes around 1.10578 e-13, demonstrating GWO's high precision in approximating the known global minimum of the Ackley function. The logarithmic scaling clearly highlights the exponential rate of convergence in the early stages, followed by fine-tuning in the later phases of the optimization.

The combined visualization of parameter space and objective space effectively illustrates both the search landscape and the optimization dynamics of the GWO algorithm. The three-dimensional function plot confirms the complex nature of the problem, while the convergence plot validates the algorithm's capacity to navigate through a rugged search space and converge reliably to the optimal solution. These results provide strong empirical evidence supporting the efficiency, precision, and robustness of GWO in solving multi-modal, non-linear optimization problems such as the Ackley function.

1.12 FIRST SUMMARY AND INSIGHTS ON JAYA AND GWO

This first chapter aimed to establish a foundation for the exploration of evolutionary computation, with a particular focus on two population-based metaheuristic algorithms: JAYA and Grey Wolf Optimizer (GWO). While these algorithms do not strictly belong to the family of evolutionary algorithms, as they lack the classical evolutionary operators such as crossover, mutation, and selection, they share several key characteristics with evolutionary approaches. In particular, both JAYA and GWO operate on a population of candidate solutions, use iterative improvement mechanisms, and employ strategies that balance exploration and exploitation of the search space, core principles also found in evolutionary algorithms. Through this analysis, the distinctive features, theoretical foundations, and practical applications of each algorithm have been examined, offering a comprehensive understanding of the critical role that population-based metaheuristics play in solving complex real-world optimization problems. The analysis conducted on these metaheuristic algorithms, inspired by natural phenomena and social behaviours, has revealed their adaptability and robustness for solving optimization problems with diverse characteristics.

However, their performance is contingent upon the type of problem at hand.

The Jaya algorithm has undergone extensive and successful testing within the domain of mechanical design, showcasing notable efficacy in the context of weight minimization for structures for various structural configurations. The study proved particularly insightful when applied to the case of dam design.

In contrast, the Grey Wolf Optimizer (GWO) remains relatively uncharted within the practical domain of mechanical design, presenting a compelling opportunity for innovation and exploration. Therefore, there exists an intriguing prospect to conduct a thorough assessment of the GWO algorithm's viability and effectiveness when applied to complex structural systems, unravelling its potential contributions to this specialized domain.

Another innovative frontier of future investigations could be to explore their possible combination.

The synergy between the Grey Wolf Optimizer (GWO) and Jaya algorithms, aimed at elevating overall performance, can be strategically orchestrated through several methodologies, each intricately tailored to the nuances of the specific problem at hand. A first approach could be to set both algorithms into motion independently and collate the most exemplary solutions produced by each algorithm, weaving them into a rich tapestry of candidate solutions.

Another potential avenue could be employing a third algorithm creating a sequential execution between Jaya and GWO for a specified number of iterations, merging the solutions obtained from both algorithms to create a set of candidates and employing a selection algorithm to identify the optimal solution.

Furthermore, there is the possibility to utilize insights gained from one algorithm to guide the behaviour of the other. For instance, promising solutions generated by Jaya could influence the exploration of GWO, and vice versa.

Hence, a viable approach would involve implementing parameter tuning, which entails optimizing the parameters of both algorithms to tailor them to the specific problem. This involves adjusting parameters to align the exploration and exploitation characteristics of each algorithm with the requirements of the problem.

Finally, it could be interesting to execute the two algorithms, Jaya and GWO, in parallel, simultaneously on different instances of the problem, integrating the best solutions from both algorithms.

The examination of the logical framework governing these algorithms not only has unveiled a myriad of opportunities for exploration within the realms of artificial intelligence and data science but also has enabled the fine-tuning of exploration and exploitation phases in accordance with the available information. This analysis paves the way for a nuanced understanding of the algorithms' potential applications, propelling us towards harnessing the full spectrum of possibilities in the ever-evolving landscape of Industry 5.0, where the role of artificial intelligence, the processing of big data and the collaboration between human operators and robots become paramount.

Moving forward, further research endeavours should prioritize a clear definition of the objective function to optimize, laying the groundwork for targeted investigations and

advancements in the field. This inquiry contributes to the ongoing discourse on the application and refinement of metaheuristic algorithms in addressing complex optimization problems within diverse industrial contexts.

CHAPTER II: INTEGRATION OF METAHEURISTIC OPTIMIZATION AND AI-BASED CONTROL STRATEGIES FOR MULTI-AGENT CART-POLE SYSTEMS

As the pole sways and the cart moves, man strives to teach machines to maintain harmony where chaos reigns, echoing the subtle intelligence behind the universe's balance.

2.1 CONTROL SYSTEMS STRATEGIES

The increasing complexity of modern industrial systems has driven significant interest in the development of advanced control architectures capable of ensuring stability, robustness, and optimal performance under uncertain and dynamic operating conditions. Conventional control strategies, such as Proportional-Integral-Derivative (PID) and model-based optimal controllers, while widely adopted due to their simplicity and interpretability, often exhibit limitations in highly nonlinear, distributed, or partially observable environments [23].

In recent years, Artificial Intelligence (AI) and metaheuristic optimization algorithms have emerged as powerful tools to enhance traditional control paradigms. Reinforcement Learning (RL) has demonstrated remarkable capabilities in learning optimal policies from interaction with the environment, even in the absence of explicit system models [24]. Simultaneously, optimization methods such as the Alternating Direction Method of Multipliers (ADMM) have gained attention for their efficiency in solving large-scale, distributed, and constrained optimization problems [25].

This chapter presents an integrated control framework that combines ADMM and AI-based control strategies, aiming to advance the performance and adaptability of control systems in both single-agent and multi-agent settings. The cart-pole system, a canonical benchmark in nonlinear control and reinforcement learning research, is employed as the experimental platform due to its dynamic instability, under actuation, and relevance to a wide class of robotic and mechatronic applications [26].

Two main case studies are addressed: a) a single cart-pole system controlled via a PID regulator coordinated by an ADMM-based sensor fusion algorithm; b) a coupled dual cart-pole system stabilized using LQR control in conjunction with ADMM for distributed coordination.

Furthermore, the proposed framework explores the impact of physical design choices, such as the reduction of pole length, on closed-loop stability, demonstrating how intelligent coupling between mechanical design and control optimization can yield superior system performance.

By bridging reinforcement learning, optimal control, and distributed optimization, this research wants to contributes to merge data-driven intelligence with model-based control theory, aligning with the foundational vision of next-generation autonomous industrial systems [27].

Classical control methods, such as PID controllers, have been widely adopted due to their simplicity, ease of implementation, and effectiveness in linear, time-invariant systems.

PID controllers form the backbone of many industrial systems, due to their performance when properly tuned [28].

In contrast, optimal control methods aim to minimize a cost function representing system performance, energy, or tracking error.

A prominent example is the Linear Quadratic Regulator (LQR), which derives control laws by solving Riccati equations to achieve an optimal balance between state deviations and control effort [29].

These techniques are particularly suitable for systems with well-defined mathematical models; however, in more complex, uncertain, or high-dimensional environments, data-driven

approaches such as Reinforcement Learning (RL), or advanced robust control methods like H-infinity (H∞) control, are formulated to ensure performance and stability [30].

Furthermore, when distributed and cooperative control strategies are required, the ADMM is often employed in multi-agent and cyber-physical systems, where decentralized coordination and scalability are essential.

It is worth noting that hybrid frameworks combining classical control strategies with machine learning and optimization methods are progressively recognized as a foundational paradigm for next-generation control systems in both industrial and autonomous domains.

2.2 DYNAMICS OF A SINGLE CART-POLE MECHANISM

The inverted pendulum is a widely used laboratory setup that effectively demonstrates nonlinear control methods [31]. The basic configuration of the cart-pole system is presented in Figure 2.1- , which serves as a reference for the derivation of the system's equations of motion.

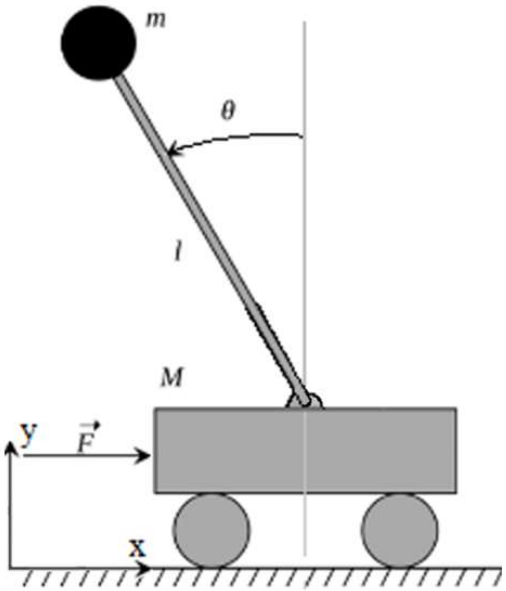


Figure 2.1- CART-POLE SYSTEM MODEL

We assume to ignore the friction forces as first approach. The following parameterization for a single cart-pole system can be defined:

x_c, y_c = horizontal and vertical position of the cart
x_p, y_p = horizontal and vertical position of the pole
θ = angle of the pendulum
m = mass of the pole
M = mass of the cart
l = length of the pole
F= external force

To give a first approximation, we can consider the whole mass of the pole concentrated at its free extremity.

The horizontal and vertical positions of the pole's mass can be defined in terms of coordinates

$$x_p = x_c + l \cdot sin\theta \tag{2.1}$$

$$y_p = l \cdot cos\theta \tag{2.2}$$

Its velocity and acceleration correspond to the first and second derivatives of these positions with respect to time:

$$\dot{x_p} = \dot{x_c} + l \cdot \dot{\theta} \cdot \cos\theta \tag{2.3}$$

$$\dot{y_p} = -l \cdot \dot{\theta} \cdot \sin\theta \tag{2.4}$$

$$\ddot{x_p} = \ddot{x}_c + l \cdot \ddot{\theta} \cdot \cos\theta - l \cdot \dot{\theta}^2 \cdot \sin\theta \tag{2.5}$$

$$\ddot{y_p} = -l \cdot \ddot{\theta} \cdot \sin\theta - l \cdot \dot{\theta}^2 \cdot \cos\theta \tag{2.6}$$

Observing that the cart moves only horizontally, it could be also useful to calculate the horizontal velocity's square of the pole finalized to a possible energetic approach, as follows:

$$\dot{x_p}^2 = \dot{x_c}^2 + l^2 \cdot \dot{\theta}^2 \cdot cos^2\theta + 2 \cdot l \cdot \dot{\theta} \cdot \dot{x_c} \cos\theta \tag{2.7}$$

The previous equation of the pole's acceleration shows that a term will depend on rotational velocity ($\dot{\theta}$) and a term will depend on rotational acceleration ($\ddot{\theta}$). The dependence on rotational velocity describes the centripetal acceleration acting on the mass m that pulls it towards the centre of rotation (joint between cart and pole).

The centripetal force and the inertial forces acting on "m" will be given by:

$$F_{coriolis} = m \cdot l \cdot \dot{\theta}^2 \tag{2.8}$$

$$F_{Inertial-x} = -m \cdot \ddot{x_p} = -m \cdot \left(\ddot{x}_c + l \cdot \ddot{\theta} \cdot \cos\theta - l \cdot \dot{\theta}^2 \cdot \sin\theta\right) \tag{2.9}$$

$$F_{Inertial-y} = -m \cdot \ddot{y_p} = m \cdot l \cdot \left(\ddot{\theta} \cdot \sin\theta + \dot{\theta}^2 \cdot \cos\theta\right) \tag{2.10}$$

At this stage, D'Alembert's principle can be applied solely along the x-axis, since the cart moves only horizontally. This involves summing all the horizontal forces and setting their total equal to zero:

$$F - M \cdot \ddot{x}_c + F_{Inertial} = F - M \cdot \ddot{x}_c - m \cdot \left(\ddot{x}_c + l \cdot \ddot{\theta} \cdot cos\theta - l \cdot \dot{\theta}^2 \cdot sin\theta\right) = 0 \tag{2.11}$$

Concerning the torques acting on the pole about a fixed axis parallel to the z-axis (perpendicular to the pla). The motion of the cart-pole system), three forces must be considered: the

gravitational force, the inertial force resulting from the horizontal acceleration of the cart (translational inertia of the mass *m*), and the torque due to the pole's rotational inertia (moment of inertia). The resulting torques will be, respectively:

$$-m \cdot g \cdot l \cdot \sin\theta \qquad \textit{GRAVITY} \tag{2.12}$$

$$m \cdot \left(\ddot{x}_c + l \cdot \ddot{\theta} \cdot \cos\theta - l \cdot \dot{\theta}^2 \cdot \sin\theta\right) \cdot (l \cdot \cos\theta) + m \cdot l \cdot (\ddot{\theta} \cdot \sin\theta + \dot{\theta}^2 \cdot \cos\theta) \cdot (l \cdot \sin\theta) = \tag{2.13}$$
$$= m \cdot l \cdot \left(\ddot{x}_c \cdot \cos\theta + l \cdot \ddot{\theta} \cdot cos^2\theta - l \cdot \dot{\theta}^2 \cdot \cos\theta \sin\theta + l \cdot \ddot{\theta} \cdot sin^2\theta + l \cdot \dot{\theta}^2 \cdot \cos\theta \cdot \sin\theta\right) =$$
$$= m \cdot l \cdot \left(\ddot{x}_c \cdot \cos\theta + l \cdot \ddot{\theta}\right) \qquad \textit{INERTIAL RESISTANCE OF m}$$

$$I \cdot \ddot{\theta}^2 \qquad \textit{MOMENT OF INERTIA} \tag{2.14}$$

where I is the pole's moment of inertia (for a material point is equal to $m\,l^2/12$).

By D'Alembert's principle, the total moment of all the forces described above about the fixed axis is set to zero:

$$-m \cdot g \cdot l \cdot \sin\theta + m \cdot l \cdot \left(\ddot{x}_c \cdot \cos\theta + l \cdot \ddot{\theta}\right) + I \cdot \ddot{\theta} = 0 \tag{2.15}$$

By explicitly expressing $\ddot{x}_c$ from equations (2.11) and (2.12), we have:

$$\ddot{x}_c = \frac{F - m \cdot l \cdot \ddot{\theta} \cdot cos\,\theta + m \cdot l \cdot \dot{\theta}^2 \cdot sin\,\theta}{M + m} \tag{2.16}$$

that is:

$$\ddot{x}_c = \frac{m \cdot g \cdot l \cdot sin\,\theta - (m \cdot l^2 + I) \cdot \ddot{\theta}}{m \cdot l \cdot cos\,\theta} \tag{2.17}$$

By equating the two expressions and isolating $\ddot{\theta}$, we obtain:

$$\ddot{\theta} = \frac{-m \cdot l \cdot \cos\theta \cdot \left(F + m \cdot l \cdot \dot{\theta}^2 \cdot \sin\theta\right) + ((M + m) \cdot m \cdot g \cdot l \cdot \sin\theta}{(M + m) \cdot (m \cdot l^2 + I) - m^2 \cdot l^2 \cdot cos^2\theta} \tag{2.18}$$

By comparing equations (2.16) and (2.17), it is evident that the expression with cos θ in the denominator becomes undefined when $\cos\theta = 0$. For this reason, equation (2.16) is preferred.

Therefore, the final equations to be considered are (2.16) and (2.18). Evaluating at each timestep $\ddot{x}_c$ and $\ddot{\theta}$, using the formulas (2.16) and (2.18) the simulation of the cart-pole can be developed.

Thus, the dynamics of the cart-pole system consists of solving a second order nonlinear equation.

Another approach to creating a concise and elegant dynamic model of the cart-pole system can be reached considering the law of conservation of energy, using the so-called Euler-Lagrange Equation, described as follow:

$$\frac{d}{dt}\left(\frac{\delta L}{\delta \dot{q}_J}\right) = \frac{\delta L}{\delta q_j} \tag{2.19}$$

where L is the Lagrange matrix (total energy= kinetic - potential energy of the system), j= 1, 2 and q represents the generalized coordinate variables of x_c and θ:

$$\begin{cases} \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x_c}}\right) - \frac{\partial L}{\partial x_c} = 0 \\ \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = 0 \end{cases} \quad (2.20)$$

The first step is to define the total energy of the system, which involves calculating the kinetic and potential energy of both the cart (K_c, U_c) and the pole (K_p, U_p), as follows:

$$K_c = \frac{M\,\dot{x_c}^2}{2} \quad (2.21)$$

$$U_c = 0 \quad (2.22)$$

$$K_p = \frac{m(\dot{x_c}^2 + l^2 \cdot \dot{\theta}^2 \cdot cos^2\,\theta + 2 \cdot l \cdot \dot{\theta} \cdot \dot{x}_c \cdot \cos\theta)}{2} \quad (2.23)$$

$$U_p = m \cdot g \cdot l \cdot \cos\theta \quad (2.24)$$

Consequently, the total energy, represented by the Lagrangian, is calculated as the difference between the kinetic and potential energies:

$$L = \frac{M\,\dot{x_c}^2 + m(\dot{x_c}^2 + l^2 \cdot \dot{\theta}^2 \cdot cos^2\,\theta + 2 \cdot l \cdot \dot{\theta} \cdot \dot{x}_c \cdot \cos\theta) - 2 \cdot m \cdot g \cdot l \cdot \cos\theta}{2} \quad (2.25)$$

Next, we proceed by calculating the following derivative:

$$\frac{\partial L}{\partial \dot{x_c}} = M\dot{x_c} + 2m\dot{x_c} + 2ml\dot{\theta}\cos\theta \quad (2.26)$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x_c}}\right) = M\ddot{x_c} + 2m\ddot{x_c} + 2ml\ddot{\theta}\cos\theta - 2ml\dot{\theta}^2\sin\theta\cos\theta \quad (2.27)$$

$$\frac{\partial L}{\partial x_c} = 0 \quad (2.28)$$

$$\frac{\partial L}{\partial \dot{\theta}} = 2ml\dot{\theta}\cos\theta + ml^2\dot{\theta}cos^3\theta + 2ml\dot{x_c}cos\theta \quad (2.29)$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) = 2ml\ddot{\theta}\cos\theta - 2ml\dot{\theta}^2 sin\theta - 2ml\dot{\theta}\dot{x_c}sin\theta - 2ml\dot{\theta}^2 cos^2\theta sin\theta \quad (2.30)$$

$$\frac{\partial L}{\partial \theta} = -2ml\dot{\theta}^2\cos\theta\, sin\theta + 2mlgsin\theta \quad (2.31)$$

Therefore, the nonlinear dynamic equations governing the motion of the cart-pole are:

$$M\ddot{x}_c + 2m\ddot{x}_c + 2ml\ddot{\theta}\cos\theta - 2ml\dot{\theta}^2\sin\theta\cos\theta = 0 \tag{2.32}$$

$$2ml\ddot{\theta}cos\theta - 2ml\dot{\theta}^2sin\theta - 2ml\dot{\theta}\dot{x}_csin\theta - 2ml\dot{\theta}^2cos^2\theta sin\theta + 2ml\dot{\theta}^2cos\theta sin\theta - 2mlgsin\theta = 0 \tag{2.33}$$

Now, linearization can be performed by introducing the state vector $X = \begin{bmatrix} x_c \\ \dot{x}_c \\ \theta \\ \dot{\theta} \end{bmatrix}$, computing the Jacobian, and expressing the system equations in the state-space form $\dot{\boldsymbol{X}} = Ax + Bu$:

$$\dot{X} = \begin{bmatrix} \dot{x_c} \\ \ddot{x}_c \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = AX + BU \tag{2.34}$$

Here, A is the system matrix, and B is the input matrix. The system includes n control inputs or external forces, denoted by u_j for j=1, 2, …, n. These inputs are grouped into the control input vector U. The matrices A, B and U are defined as:

$$A = \begin{bmatrix} \frac{\partial \dot{x}_c}{\partial x_c} & \frac{\partial \dot{x_c}}{\partial \dot{x_c}} & \frac{\partial \dot{x}_c}{\partial \theta} & \frac{\partial \dot{x}_c}{\partial \dot{\theta}} \\ \frac{\partial \ddot{x}_c}{\partial x_c} & \frac{\partial \ddot{x}_c}{\partial \dot{x_c}} & \frac{\partial \ddot{x}_c}{\partial \theta} & \frac{\partial \ddot{x}_c}{\partial \dot{\theta}} \\ \frac{\partial \dot{\theta}}{\partial x_c} & \frac{\partial \dot{\theta}}{\partial \dot{x_c}} & \frac{\partial \dot{\theta}}{\partial \theta} & \frac{\partial \dot{\theta}}{\partial \dot{\theta}} \\ \frac{\partial \ddot{\theta}}{\partial x_c} & \frac{\partial \ddot{\theta}}{\partial \dot{x_c}} & \frac{\partial \ddot{\theta}}{\partial \theta} & \frac{\partial \ddot{\theta}}{\partial \dot{\theta}} \end{bmatrix} \quad U = \begin{bmatrix} u_1 \\ u_2 \\ . \\ . \\ . \\ u_n \end{bmatrix} \quad B_{i,j} = \frac{\partial L}{\partial u_j \cdot \partial \ddot{q}_\iota} \tag{2.35}$$

where $\ddot{q}_\iota$ represents the generalized accelerations.

Considering the simplest case with very small values of θ, the following approximations can be used:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{m \cdot g}{M} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{m \cdot g + M \cdot g}{M \cdot l} & 0 \end{bmatrix} \quad U = \begin{bmatrix} u_1 \\ u_2 \\ . \\ . \\ . \\ u_n \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ \frac{1}{M} \\ 0 \\ \frac{1}{M \cdot l} \end{bmatrix} \tag{2.36}$$

with the initial conditions: $\begin{cases} x_c = 0 \\ \dot{x_c} = 0 \\ \theta = 0 \\ \dot{\theta} = 0 \end{cases}$

The model could be more accurate introducing also the inertia of pole ($ml^2/12$) so that the pole's Kinetic Energy and the Lagrangian could be defined as follow:

$$K'_p = \frac{m(12 \cdot \dot{x_c}^2 + 24 \cdot l \cdot \dot{\theta} \cdot \dot{x}_c \cdot \cos\theta + 13 \cdot \dot{\theta}^2 \cdot l^2)}{24} \tag{2.37}$$

$$L' = \frac{(M+m)\dot{x_c}^2}{2} + \frac{13\, m \cdot l^2 \cdot \dot{\theta}^2}{24} + ml \cdot \dot{\theta} \cdot \dot{x}_c \cdot \cos\theta - m \cdot g \cdot l \cdot \cos\theta \tag{2.38}$$

Thus, the nonlinear system is linearized using the following Jacobian matrix:

$$\begin{bmatrix} \dot{x_c} \\ \ddot{x}_c \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = q_{i,j} \tag{2.39}$$

Explicitly expressing the elements $q_{i,j}$:

$q_{1,1}=0$

$q_{1,2}=1$

$q_{1,3}=0$

$q_{1,4}=0$

$q_{2,1}=0$

$q_{2,2}=\frac{13\, m \cdot l \cdot \dot{\theta}^2 \cdot \sin\theta + 13F - 12\, m \cdot g \cdot \cos\theta \cdot \sin\theta}{13M + 13m - 12m \cdot \cos^2\theta}$

$q_{2,3}=\frac{\dot{\theta}\left(\dot{\theta} \cdot l \cdot \cos\theta \sin\theta - F\cos\theta + mg \cdot \sin\theta\right)}{l(13M + 13m - 12m\cos^2\theta)}$

$q_{2,4}=0$ (2.40)

$q_{3,1}=0$

$q_{3,2}=0$

$q_{3,3}=0$

$q_{3,4}=1$

$q_{4,1}=0$

$$q_{4,2}=\frac{-12\,m\cdot l\cdot\dot{\theta}^2\cdot cos\theta\cdot sin\,\theta+Fcos\theta+m\cdot g\cdot sin\,\theta(M+m)}{l(13M+13m-12m\cdot cos^2\,\theta)}$$

$$q_{4,3}=\frac{12\,m\cdot l\cdot(\dot{\theta}^2 cos^2\theta-Fcos\theta sin\theta-m\cdot g\cdot cos\theta sin\theta)}{l(13M+13m-12m\cdot cos^2\,\theta)}$$

$q_{4,4}=0$

and using the hypothesis that θ is small, the new values of A and B will be:

$$A'=\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{-12\,m\cdot g}{13M+m} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{12\,(m\cdot g+M\cdot g)}{l(13M+m)} & 0 \end{bmatrix} \qquad B'=\begin{bmatrix} 0 \\ \frac{13}{13M+m} \\ 0 \\ \frac{-12}{l\,(13M+m)} \end{bmatrix} \tag{2.41}$$

with the initial conditions (linearization): $\begin{cases} x_c=0 \\ \dot{x_c}=0 \\ \theta=0 \\ \dot{\theta}=0 \end{cases}$

Finally, having defined the dynamic of a single cart-pole system, a double model can be developed for a twin of cart-pole systems.

2.3 THE ADMM (ALTERNATING DIRECTION METHOD OF MULTIPLIERS)

The ADMM allows for the decomposition of a complex convex optimization problem into multiple, more manageable sub-problems that can be solved independently.

In the following sections, we will explore the application of ADMM to coordinate the motion of two cart-pole systems along a linear path. This approach holds potential for a wide range of applications in the context of Industry 4.0, where decentralized and cooperative control architectures are increasingly relevant.

ADMM is particularly effective for determining the optimal values of a set of variables subject to specific constraints, which may be either physical (e.g., mechanical limits) or virtual (e.g., communication or synchronization requirements). Unlike traditional centralized optimization methods, ADMM is well suited for distributed or networked systems, where each subsystem can compute locally while coordinating with others through limited information exchange.

The ADMM procedure, when applied to a generic system composed of two subsystems, each defined by a cost function, f for the first and g for the second, can be summarized as follows:

1) **Setup of the Two sub-systems.** The first step involves formulating the coordination of the two cart-pole systems as a convex optimization problem. Specifically, the problem should be defined using two convex objective functions, f and g, where each function models the behaviour or constraints of one subsystem.

To enable the application of ADMM, the problem must be decomposed into two sets of variables (or separable objectives), allowing the overall goal to be expressed as

$$\min_{x,z}\big(f(x)+g(x)\big) \text{ subject to A x + B z = c} \tag{2.42}$$

Here, x and z are decision variables associated with the two subsystems, and A, B, and c define the coupling constraint that links the subsystems (e.g., shared states or coordination requirements).

2) **Introduction of the "Augmented Lagrangian".** It incorporates the constraint $Ax + Bz = c$ using a dual variable (Lagrange multiplier) y and a quadratic penalty term that improves convergence properties:

$$L_\rho(x,z,y) = f(x) + g(x) + \ y^T \cdot (A \cdot x + B \cdot z - C) + \frac{\rho}{2} \cdot \|A \cdot x + B \cdot z - C\|^2 \tag{2.43}$$

where ρ>0 is the penalty parameter that controls the weight of the constraint violation term.

3) **ADMM:**

$$x^{k+1} \coloneqq argmin_x L_\rho(x, z^k, y^k) \quad \text{MINIMIZATION OF x} \tag{2.44}$$

$$z^{k+1} \coloneqq argmin_z L_\rho(\ x^{k+1}, z, y^k) \quad \text{MINIMIZATION OF z} \tag{2.45}$$

$$y^{k+1} \coloneqq y^k + \rho \cdot (A \cdot x^{k+1} + B \cdot z^{k+1} - C) \quad \text{DUAL UPDATE} \tag{2.46}$$

These iterations are repeated until convergence criteria are met, such as the primal and dual residuals falling below specified tolerances. The minimizations in ADMM are carried out sequentially (Gauss-Seidel fashion), rather than jointly over both x and z. If the minimization were performed jointly, the method would reduce to the classical Method of Multipliers (MoM). While MoM often achieves faster convergence for small, centralized problems, it requires solving a large, coupled optimization problem at each iteration, which is computationally expensive and unsuitable for distributed architectures. In contrast, ADMM decomposes this global problem into smaller subproblems, each solvable independently and often in parallel, making it particularly advantageous for distributed control, multi-agent coordination, and large-scale optimization where communication or computational resources are limited. This decomposition property is the main motivation for adopting ADMM in coordinating multiple cart- pole systems.The application of the ADMM procedure to the system under study can be illustrated as in **Errore. L'origine riferimento non è stata trovata.**.

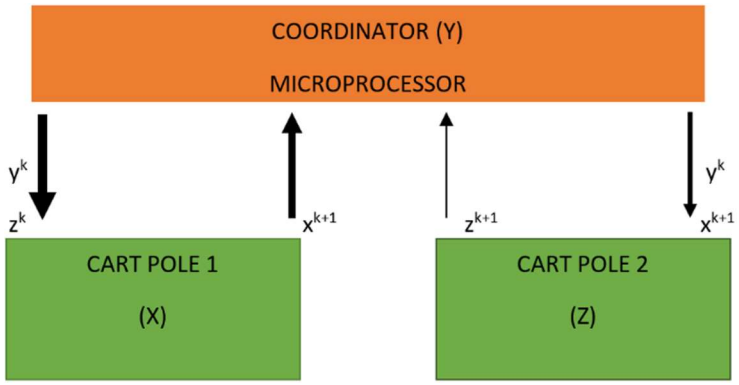


Figure 2.2- ADMM-BASED COORDINATION SCHEME FOR TWO CART-POLE SYSTEMS

Iteration k PRE-CONDITION:

Before the $k-th$ iteration begins, the coordinator distributes the current values of the decision variables and dual variables to the individual agents (subsystems). This step ensures that each agent has access to the most recent data needed for local updates.

$$x^{k+1} \coloneqq argmin\, L_\rho(x, z^k, y^k) \tag{2.47}$$

$$z^{k+1} \coloneqq argmin\, L_\rho(x^{k+1}, z, y^k) \tag{2.48}$$

$$y^{k+1} \coloneqq y^k + \rho \cdot (A \cdot x^{k+1} + B \cdot z^{k+1} - C) \tag{2.49}$$

Iteration k-step 1: FIRST AGENT UPDATES DATA

In this phase, Agent 1 (associated with variable x) performs its update using the most recent data received. The optimization is local to its subsystem and involves minimizing the augmented Lagrangian with respect to x, while keeping z^k and y^k fixed (see Equation 2.47 and FIGURE 2.3-).

Then, Agent 2 updates its own decision variable z, based on the updated value of x^{k+1} and the current dual variable y^k (see Equation 2.48 and Figure 2.4-).

Finally, the coordinator updates the dual variable y, enforcing consistency between the subsystems via the coupling constraint (see Equation 2.49 and Figure 2.5-).

$$x^{k+1} \coloneqq argmin_x L_\rho(x, z^k, y^k) \tag{2.50}$$

$$z^{k+1} \coloneqq argmin_z L_\rho(x^{k+1}, z, y^k) \tag{2.51}$$

$$y^{k+1} \coloneqq y^k + \rho \cdot (A \cdot x^{k+1} + B \cdot z^{k+1} - C) \tag{2.52}$$

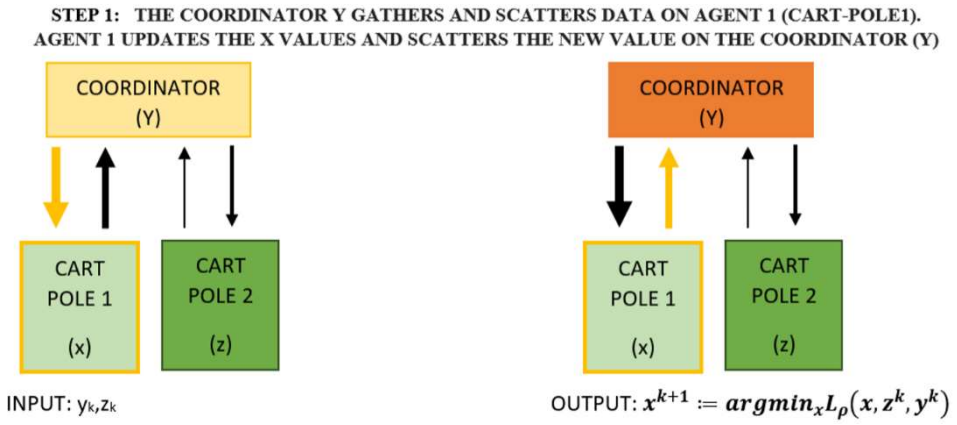


FIGURE 2.3- ITERATION k-STEP 1: FIRST AGENT UPDATES DATA

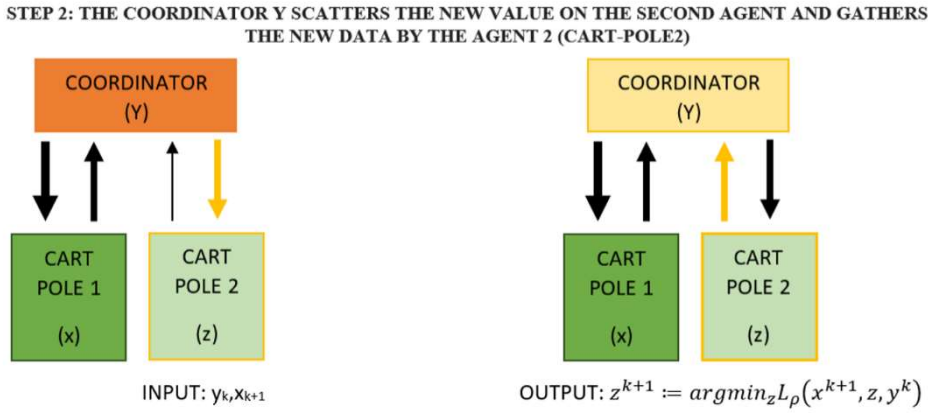


Figure 2.4- ITERATION k-STEP 2: FIRST AGENT UPDATES DATA

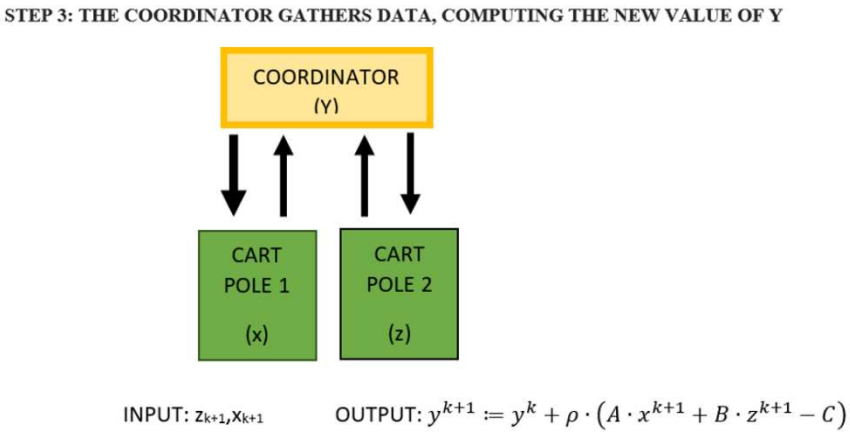


Figure 2.5- ITERATION k-STEP 3: FIRST AGENT UPDATES DATA

These steps represent one full iteration of ADMM. The key advantage of this formulation is the sequential (Gauss-Seidel) structure, allowing each agent to solve its local problem with up-to-date information from the other.

2.4. APPLICATION OF ADMM TO THE COORDINATION OF CART AND POLE SENSORS

In this example, the ADMM algorithm is employed to coordinate two distributed sensors—one tracking the cart's horizontal position and the other monitoring the pole's angular displacement. The goal is to achieve synchronized data acquisition and control between these subsystems, formulated as separable components within a convex optimization framework. While this decentralized strategy enhances real-time coordination and efficiency under the physical and communication constraints typical of Industry 4.0 systems, it's worth noting that guaranteed convergence depends on the convexity and feasibility of the underlying problem. In well-posed convex settings, ADMM offers strong convergence guarantees; however, for nonconvex or ill-conditioned scenarios, additional regularization or adaptive strategies may be required to ensure reliable performance, such as adjusting the penalty parameter dynamically, applying convex relaxations, or using warm-start techniques to improve convergence stability.

The mathematical model of the system is defined by Equation 2.41, with the control input vector represented as u= $[u_{xc} \quad u_{\theta}]$, where u_{xc} and u_{θ} denote the control actions associated with the position x_c and the pole angle θ, respectively.

A common approach for centralized control of a single cart-pole system involves using one of the following controllers [32]:

- **PID** (Proportional-Integral-Derivative) Controller
- **LQR** (Linear Quadratic Regulator)
- **MPC** (Model Predictive Control)

Each of these techniques offers different advantages in terms of performance, complexity, and robustness, depending on the characteristics and requirements of the system being controlled.

The primary objective of this example is to study how a cart-pole system can reach and maintain a desired position of the pole's tip while the cart moves along a linear path, given an initial velocity input.

To achieve this, a control strategy is implemented that combines a PID controller, responsible for regulating the cart position x_c and the pole angle θ,with an ADMM algorithm used to fuse measurements from two independent sensors for each variable, enhancing robustness and consistency.

In the control of the cart-pole system, the adopted framework is based on two main objectives: sensor coordination, to ensure consistent readings between distributed sensors, and the enforcement of physical constraint, to maintain feasible and stable system behaviour.

The two main objective functions are defined as follows:

1) The first objective function f_{coord} encourages synchronization between the two sensors, one tracking the cart's horizontal position (x_c) and the other measuring the pole's angular position (θ). This coordination is enforced by minimizing the squared difference between the two measurements:

$$f_{coord} = \alpha \cdot (x_c - \theta)^2 \tag{2.53}$$

Here α represents the weight parameter for the coordination objective.

2) The cart position constraint ensures that x_c remains within predefined physical bounds. In particular, a lower bound is imposed to maintain stability and prevent the cart from moving beyond a specific limit, defined as L_c, the length of the cart:

$$x_c \geq \left(\frac{-L_c}{2}\right) \tag{2.54}$$

To enforce both sensor coordination and constraint satisfaction, two penalty terms are introduced, one for the coordination between x_c and θ, and one for maintaining the cart position within its physical limits. The total objective is:

$$f_{total} = \alpha \cdot (x_c - \theta)^2 + \beta \cdot \max(0, |x_c| - L_c/2)) \tag{2.55}$$

where α, β represents the weight factors balancing the importance of coordination and constraint satisfaction.

The first penalty term, $\alpha \cdot (x_c - \theta)^2$, is the coordination penalty which penalizes disagreement between the cart position sensor and the pole angle sensor, promoting synchronized behaviour between the two measurements.

The second penalty term, $\beta \cdot \max(0, |x_c| - L_c/2)$, is the constraint penalty which ensures that the cart's position stays within its physical bounds by adding a cost whenever the limit is exceeded.

During the optimization loop, after calculating the control inputs and updating the system state, the penalty terms are evaluated to guide the adjustment of the control inputs. Unlike a standard Lagrangian approach (where multipliers λ are updated iteratively according to constraint violations), in this implementation the constraints are incorporated directly via soft penalties.

In the code (Appendix 3) the decision variables are not explicitly defined as separate variables. Instead, they are implicit within the control inputs u_{x_c} (control input for the cart position) and u_θ (control input for the pole angle). These control inputs are determined by the PID controller gains and are updated at each step to reach the final position of the system while respecting the penalties in Equation 2.55.

The system also incorporates several local constraints, such as:

$$|x_c| \leq L_c/2 \tag{2.56}$$

which ensures that the cart's position remains within its track limits, and the pole angle constraint, typically restricting θ between -90° and +90°, although this limit is not explicitly defined in the code. The weighting terms α and β in Equation 2.55 indirectly regulate how strongly these constraints influence the control behaviour.

It is important to note that the coupling constraints are not explicitly defined as separate mathematical equations but are implicit in the control and state dynamics of the cart-pole system. In particular, the coupling constraints are related to the physical interactions and dynamics of the systems, like the system's equation of motion, masses, lengths, forces, geometrical constraints between the cart and the pole. To accurately characterize the influence of the control inputs on the system's dynamics, it is necessary to analyse the structure of the control input matrix B and the input vector u.

The control input matrix B is a 4×2 matrix, reflecting the four-dimensional state vector (comprising cart position x_c, cart velocity $\dot{x_c}$, pole angle θ and pole angular velocity $\dot{\theta}$) and two control inputs: one for the cart position and one for the pole angle. Each element of B represents how a particular state variable is affected by a corresponding control input. The two elements of the control input vector u, u_{x_c} and u_θ, determine how much force or torque is applied to control the cart's position and the pole's angle, respectively.

The cart position input control is responsible for the cart to a desired position along the track. It represents the force or actuation applied to the cart's wheels, enabling motion to the left or right. The value of u_{x_c} is computed by a PID (Proportional-Integral-Derivative) controller, which minimizes the error between the current cart position and the desired target position.

The pole angle input control u_θ serves to stabilize the pole by exerting torque at the pivot point where the pole is connected to the cart. It counteracts disturbances and ensures the pole remains upright or follows a desired angular trajectory. As with the cart position, the control effort u_θ is also calculated via a PID controller, based on the error between the current pole angle and the desired angle. Together, these control inputs are applied through the B matrix to influence the system dynamics and achieve coordinated control of the cart-pole mechanism.

The implementation structure of the code (see Appendix 3) is organized into sequential computational stages, summarized in Figure 2.6:

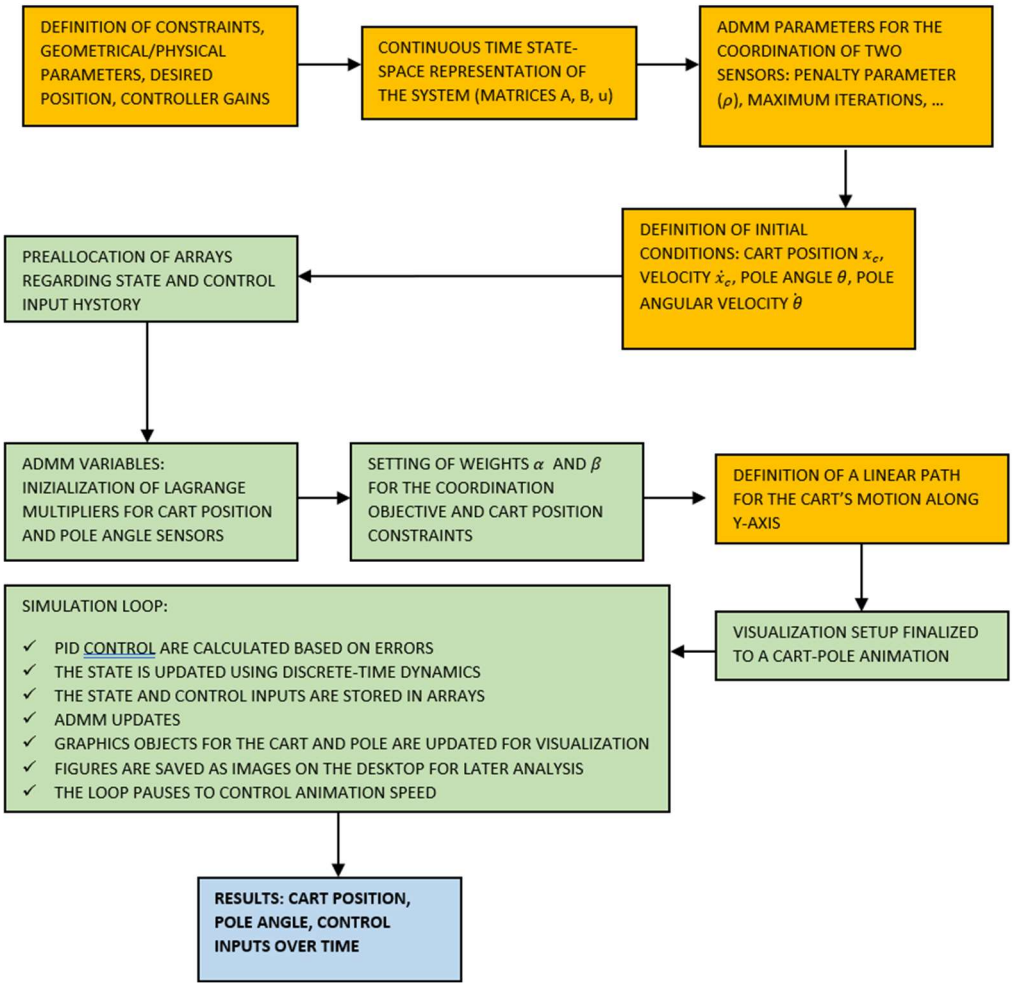


Figure 2.6- CODE STRUCTURE FOR ADMM-BASED COORDINATION OF CART AND POLE SENSORS

2.5 RESULTS OF ADMM FOR CART AND POLE SENSOR COORDINATION

This section presents four representative simulations, organized in a progressive analysis of a cart-pole system controlled under different conditions and strategies, with the objective of evaluating stability, control effort, and system responsiveness. Each simulation addresses key aspects of controller tuning and system geometry.

For all the simulation, it is specified that $\theta = 0$ corresponds to the pole in the vertical upright position, pointing upward, $\theta > 0$ is positive to the right, $\theta < 0$ indicates a tilt to the left.

In Simulation 1 (see Figure 2.7-), a baseline configuration using a classical PID controller with moderate gains is applied to both cart position and pole angle regulation. This initial test reveals limitations in pole stabilization, despite satisfactory position control, highlighting the challenges of controlling a nonlinear, underactuated system.

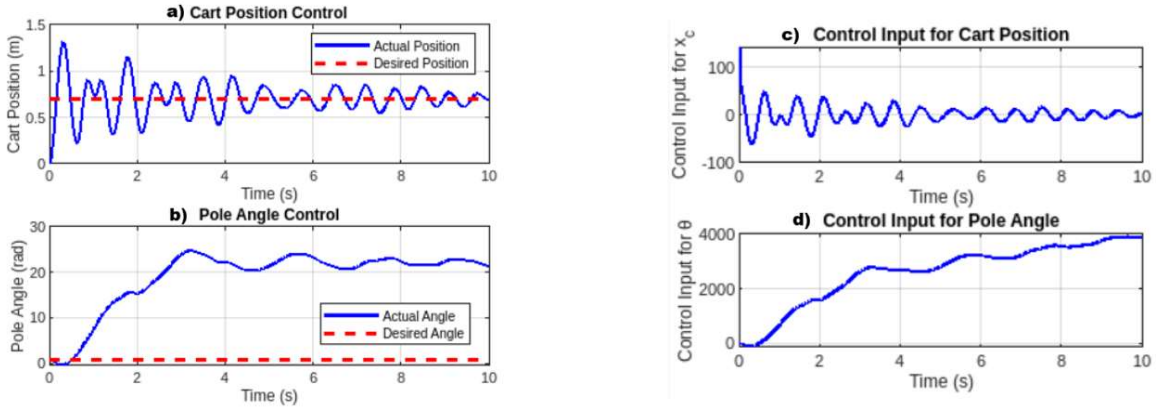


Figure 2.7- ADMM SENSOR COORDINATION UNDER PID CONTROL WITH POLE LENGTH=0.6 m, kp_{xc}, kp_{θ} = 100, ki_{xc}, ki_{θ}= 10, kd_{xc}, $kd_{\theta} = 1$; a) AND b) CART AND POLE SYSTEM RESPONSE, c) AND d) CART POSITION AND POLE ANGLE CONTROL INPUTS

Figure 2.7- .a shows the time evolution of the cart's actual position (blue) compared to the desired position (red, dashed). Initially, the system exhibits noticeable oscillations, which gradually decay over time, indicating the damping behaviour induced by the PID controller. The steady-state error approaches zero, demonstrating that the controller can regulate the cart's position effectively despite initial transients.

In Figure 2.7- .b the response of the pole angle, unlike the cart position, shows a persistent offset from the desired value (held constant at approximately zero). The actual angle stabilizes at a significantly higher value, indicating that the PID controller, with the current gain configuration, fails to achieve accurate angle tracking. This may suggest inadequate tuning, nonlinear effects, or insufficient control authority.

The control input applied to the cart (Figure 2.7-) exhibits oscillatory behaviour initially, followed by a gradual attenuation, mirroring the transient behaviour of the cart position. The amplitude of the control effort remains within ±100 units, which appears sufficient to drive the system toward the desired state without excessive actuation.

The control input associated with the pole angle (Figure 2.7- .d) increases steadily over time, reaching magnitudes exceeding 4000 units. This trend reflects the controller's continued effort to correct the pole's large angle deviation. The sustained and increasing input suggests that the controller is compensating for an unmodeled dynamic or is encountering saturation or instability in the regulation of the pole angle.

Thus, while the PID controller provides reasonable performance for cart position control, its effectiveness in regulating the pole angle is limited under the current conditions. The large control effort and persistent tracking error for the pole angle indicate a need for gain re-tuning or the use of a more advanced control strategy (e.g., nonlinear or adaptive control) to achieve full stabilization. Fine-tuning of the PID gains (especially K_i for the angle) and adjustment of the penalty weights α and β in the coordination term may improve performance.

Simulation 2 (see Figure 2.8-) investigates the impact of increasing the proportional and derivative gains, aiming to improve convergence speed and damping. The results show notable

enhancements in both position and angle control performance, especially in reducing overshoot and control effort.

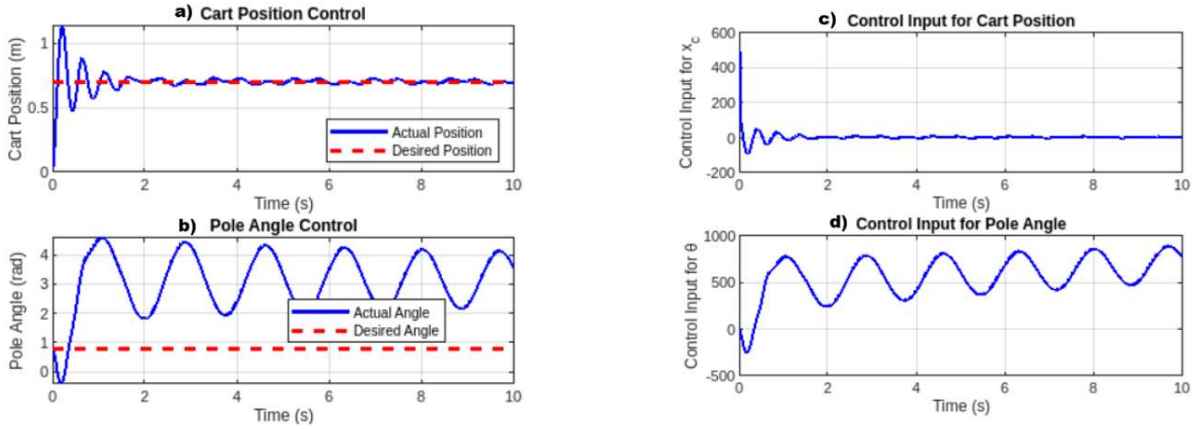


Figure 2.8- ADMM SENSOR COORDINATION UNDER PID CONTROL WITH POLE LENGTH=0.6 m, kp_{xc}, kp_{θ}= 200, ki_{xc}, ki_{θ}= 10, kd_{xc}, kd_{θ}= 5; a) AND b) CART AND POLE SYSTEM RESPONSE, c) AND d) CART POSITION AND POLE ANGLE CONTROL INPUTS

In Figure 2.8- .a the cart's actual position (blue) closely tracks the desired reference (red dashed line), showing faster convergence and reduced overshoot compared to Figure 2.7.a. The oscillations are significantly damped, and steady-state error is negligible. This indicates that the increased proportional and derivative gains improve tracking performance and system stability. The pole angle continues to oscillate around the desired reference (see Figure 2.8-). However, compared with the simulation n.1, the oscillations are now periodic and bounded, suggesting an improvement in dynamic behaviour. The persistent fluctuations may still be due to system nonlinearities or limitations in the PID structure, but their reduced amplitude indicates better disturbance rejection and responsiveness.

Figure 2.8- .c shows a quick transient rise, followed by convergence to a steady and low-amplitude signal. The input range for the cart is narrower compared to Figure 2.7.c, indicating more efficient actuation and reduced energy expenditure.

Also, the control input for pole angle (Figure 2.8- .d) now displays a well-defined periodic pattern, in contrast to the continuously increasing input observed in Figure 2.7.d). This suggests that the controller can now contain the angle deviations within a stable range, using a bounded control effort. The peak amplitude (~500±200) is considerably smaller and bounded than in the previous case, highlighting improved control efficiency and better tuning.

Thus, the increase of the proportional gain from 100 to 200 and the derivative gain from 1 to 5 has led to faster settling, reduced oscillations, and lower control effort.

These results show that the cart's position control performs well, maintaining the cart near its target with minimal oscillations. However, the pole angle control remains oscillatory and does not converge to the desired equilibrium, revealing a need for further tuning of the angular PID gains, especially the integral and derivative terms, or the inclusion of a more advanced coordination or stabilizing strategy.

The scope of the third simulation (see Figure 2.9-) is to define how the length of the pole could influence the stability of the systems at pair of other parameters related to the second simulation. The length has been halved from 0.6 meters to 0.3 meters. Halving the pole length yields faster dynamics and better angle regulation, though it also increases control input requirements due to the higher system sensitivity.

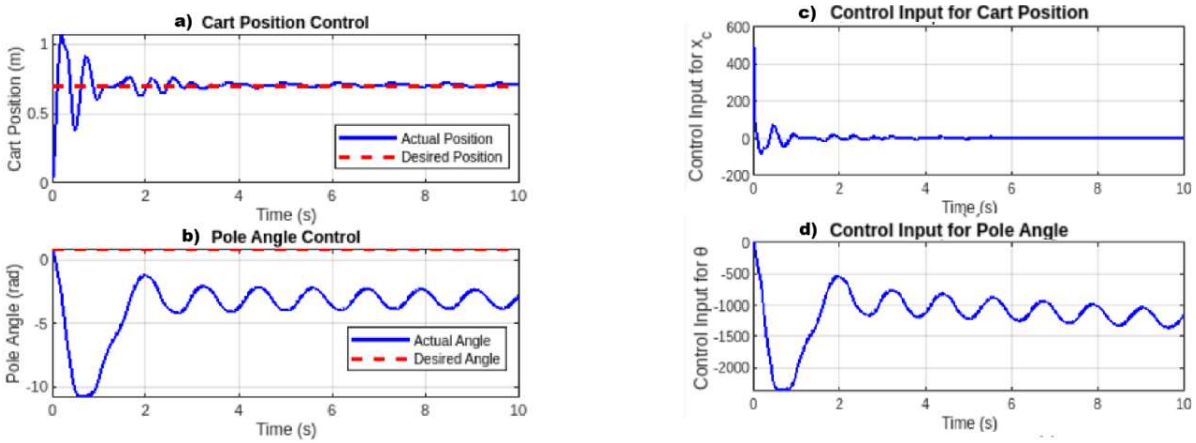


Figure 2.9- ADMM SENSOR COORDINATION UNDER PID CONTROL WITH POLE LENGTH=0.3 m, kp_{xc}, kp_{θ}= 200, ki_{xc}, ki_{θ}= 10, kd_{xc}, kd_{θ}= 5; a) AND b) CART AND POLE SYSTEM RESPONSE, c) AND d) CART POSITION AND POLE ANGLE CONTROL INPUTS

The cart position (see Figure 2.9- 2.9. a) closely tracks the desired trajectory with rapid convergence and minimal overshoot. The transient phase exhibits damped oscillations which settle quickly, reflecting a well-tuned controller. The steady-state error is negligible, indicating accurate position regulation. Considering the pole angle (see Figure 2.9-) there is a clear oscillation around the desired setpoint (held constant). Compared to Figure 2.8.b, the amplitude of the oscillations is reduced, and the frequency appears slightly increased, likely due to the shorter pole length. The response indicates improved regulation with the pole tending toward reference, though small oscillations persist. The control input for the cart (see Figure 2.9-) reaches its peak during the initial transient and then quickly decays, stabilizing around zero, but the control input required for pole angle stabilization (see Figure 2.9-) is larger in magnitude compared to Figure 2.8.d, remaining bounded and periodic. The increased peak values reflect the increased effort needed due to the higher sensitivity and faster dynamics associated with the shorter pole length. Nonetheless, the control remains effective in maintaining bounded oscillations.

It is evident that further investigations are required to improve the cart-pole system's performance. Future simulations should focus on refining controller parameters and enhancing model's accuracy.

Finally, Simulation 4 (see Figure 2.10- and **Errore. L'origine riferimento non è stata trovata.**) replaces the PID controller with an LQR (Linear Quadratic Regulator) and investigates how the system's stability is affected by the variations in the pole length.

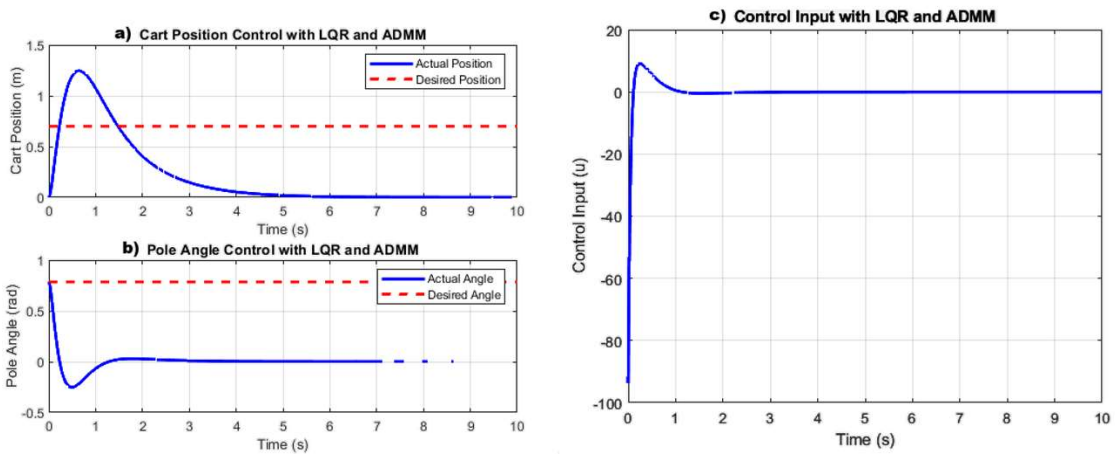
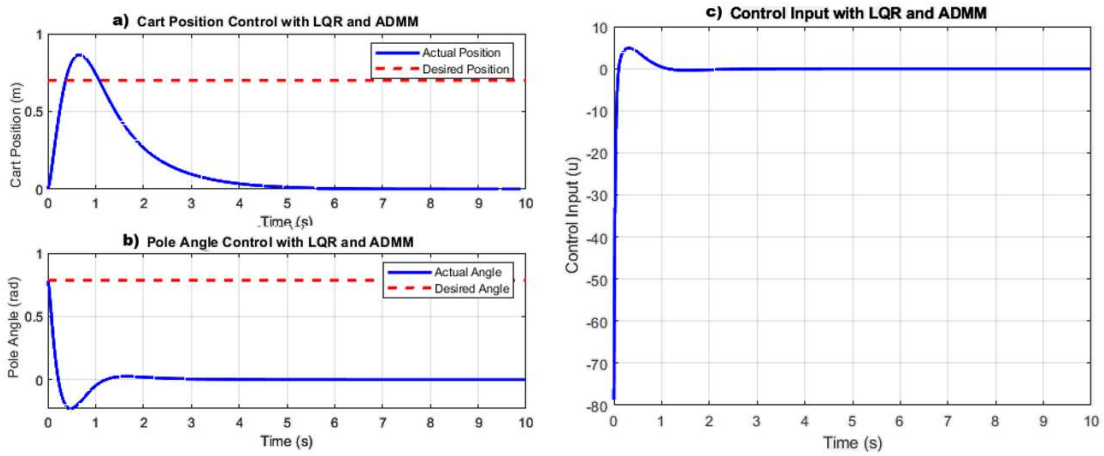


Figure 2.10- ADMM SENSOR COORDINATION UNDER LQR CONTROL WITH POLE LENGTH=0.6 m; a) AND b) CART AND POLE SYSTEM RESPONSE, c) CONTROL INPUTS



2.11- ADMM SENSOR COORDINATION UNDER LQR CONTROL WITH POLE LENGTH=0.3 m; a) AND b) CART AND POLE SYSTEM RESPONSE, c) CONTROL INPUTS

Figure 2.10 shows the system response with a pole length of 0.6 m, while Figure 2.11 presents the results when the pole length is reduced to 0.3 m. Compared with the previous PID-based simulations, the LQR controller achieves a much smoother and faster stabilization for both the cart position and the pole angle. However, the system does not fully reach the desired reference values (red dashed lines); instead, it settles slightly below them at steady state.

This behaviour occurs because the LQR is designed to minimize a quadratic cost function that balances state deviations and control effort, but it does not inherently remove steady-state errors. The LQR operates around a linearized equilibrium point. Therefore, when the desired reference differs from this equilibrium, the controller stabilizes the system close to it rather than exactly at the reference. This ensures high stability and low oscillations but allows a small steady state offset unless an integral action or reference shift is added to the control law.

Overall, the results show a well-damped and stable response, consistent with the theoretical behaviour of standard LQR controllers. In the case of the 0.6 m pole (Figure 2.10), the cart exhibits a small overshoot, peaking around 1.2 m, before settling near 0 in about 3 seconds. When the pole length is reduced to 0.3 m (Figure 2.11), the overshoot decreases (to about 0.7

m), and the system reaches steady state in similar seconds. The shorter pole enhances agility and reduces inertia, allowing the controller to achieve stability more efficiently.

The pole angle response confirms this trend. With the shorter pole, the deviation is smaller, with negligible oscillation. This demonstrates that reducing the pole length increases control authority and improves angular stability.

The control input plots (Figures 2.10.c and 2.11.c) highlight similar improvements. For the longer pole, the control input reaches around –100 units during the transient phase before settling near zero. For the shorter pole, the peak control effort decreases to approximately –80 units, indicating a more energy-efficient and responsive control action.

Compared to the previous PID-based simulations, the LQR approach offers superior damping and more balanced control effort.

However, the obtained results indicate that further refinement is necessary.

Future research should focus on fine-tuning the LQR weighting matrices to better optimize the trade-off between control effort and settling time. Moreover, extending the analysis to include nonlinear effects, model uncertainties, and external disturbances would provide a more realistic assessment of system performance. Additionally, integrating the LQR controller with adaptive or robust control strategies could enhance stability and resilience under varying pole lengths and uncertain system parameters, thereby improving the overall reliability and applicability of the cart-pole system in real-world scenarios.

2.6 APPLICATION OF ADMM AND REINFORCEMENT LEARNING TO THE COORDINATION OF TWO CART-POLE SYSTEMS

The scope of this example is to analyse the coordinated control of two cart-pole systems employing a LQR (Linear Quadratic Regulator) in conjunction with the ADMM (Alternating Direction Method of Multipliers) and reinforcement learning methods to synchronize their motion along a specified linear path.

Specifically, the optimization problem involves controlling both cart-pole systems to reach a designated target position while maintaining a prescribed pole orientation and following a predefined path. The objective is to ensure that their movements remain symmetric with respect to the z-axis, such that, in a subsequent phase of the investigation, the two systems achieve mirrored positions and coordinated dynamic behaviour. The primary goal is to minimize the overall control effort (cost function), ensuring that both the system dynamics and the positional constraints of the cart-pole configurations are simultaneously satisfied.

The objective functions are implicitly defined and closely tied to the minimization of control effort (see Appendix 4). The LQR controller operates by minimizing a quadratic cost function that balances the magnitude of control inputs with deviations of the system state variables from their desired reference values.

Regarding the local constraints, the system dynamics are captured through linear state-space equations characterized by matrices A' and B' (see Equation 2.41), derived from the underlying

ordinary differential equations governing each system. Position constraints on the carts are addressed via ADMM, which facilitates the minimization of the deviation of the carts' positions from their respective desired trajectories. Similarly, the pole angle constraints are enforced by ADMM to ensure adherence to the target pole orientation for both systems (see Figure 2.12.b).

It is noteworthy that, at this initial stage of analysis, there are no explicit coupling constraints imposed between the two cart-pole systems; each system is controlled independently. The role of ADMM here is primarily to enforce local constraints within each system in a computationally efficient manner.

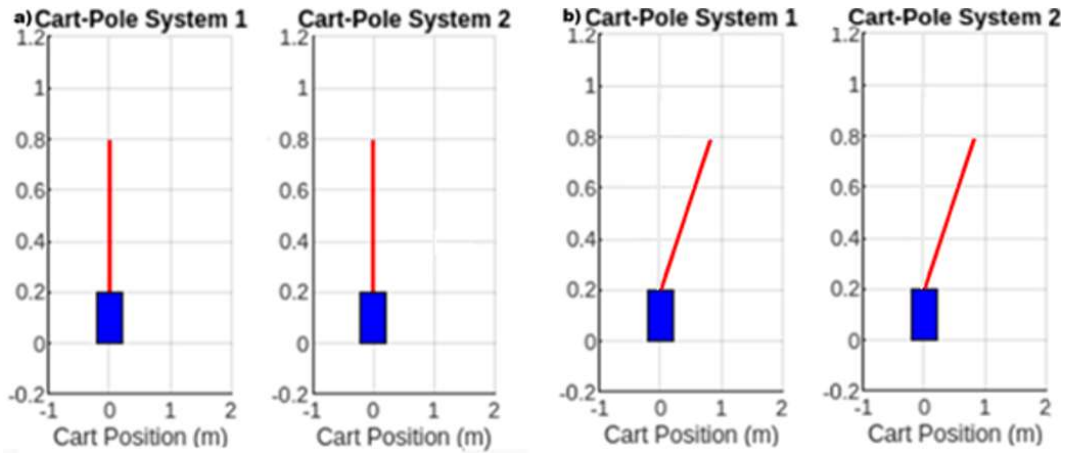


Figure 2.12- INITIAL (a) AND FINAL (b) POSITIONS OF THE TWO CART-POLE SYSTEMS IN THE RELATIVE X-Z PLANE

The two cart-pole systems move in a coordinated manner along the y-axis, starting from the origin position (see Figure 2.12-) and traveling a distance of 1 meter while maintaining a symmetric configuration, with a constant spacing of 1 meter between them. Each pole is constrained to remain in an upright position throughout the translation. The ADMM algorithm is employed to coordinate the sensor feedback used for controlling both the cart position and pole angle in each system, as well as to synchronize their translational movement. The translation velocity is set to 1 m/min.

Figure 2.13- presents the results of a preliminary investigation that requires further development.

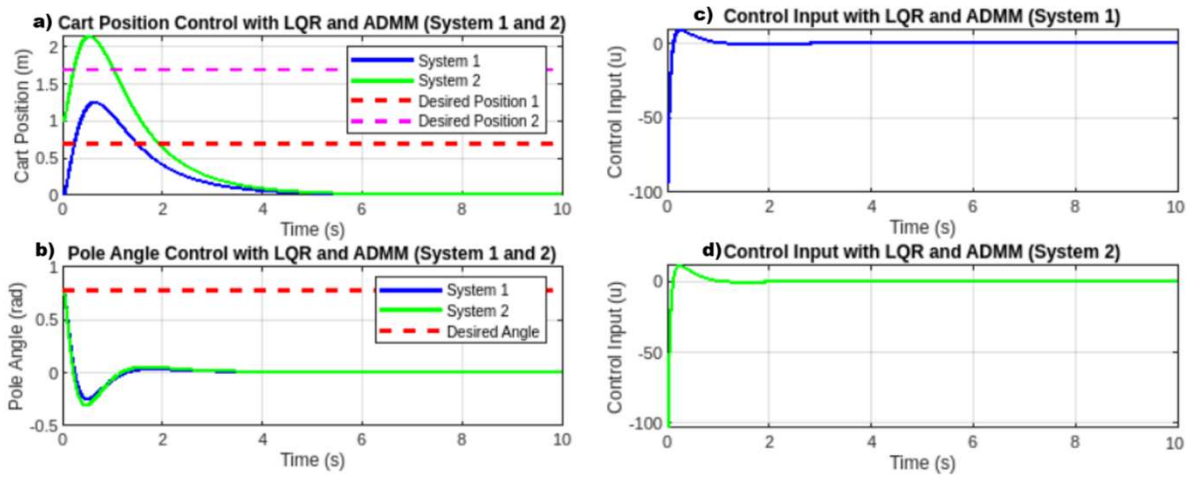


Figure 2.13- ADMM-BASED COORDINATION OF TWO CART-POLE SYSTEMS UNDER LQR CONTROL: (A) AND (B) CART AND POLE SYSTEM RESPONSES; (C) CONTROL INPUTS

The stabilization of the control input to a steady state value of zero is achieved within approximately 0.5 seconds for both systems (see Figure 2.13.c and Figure 2.13.d). Nevertheless, both the cart position control and the pole angle control exhibit notable deficiencies (see Figure 2.13.a and Figure 2.13.b), as a significant deviation between the desired and actual trajectories persists, particularly in the case of the second system. To enhance overall control performance, especially in terms of cart positioning accuracy and pole angle stabilization, several improvements can be considered. These include a systematic re-evaluation and fine-tuning of controller gains, the implementation of more advanced control strategies and the reformulation of the underlying optimization problem to more accurately capture the system's nonlinear dynamics and physical constraints. Furthermore, integrating state estimation and noise filtering techniques could improve measurement accuracy and robustness under real-world disturbances.

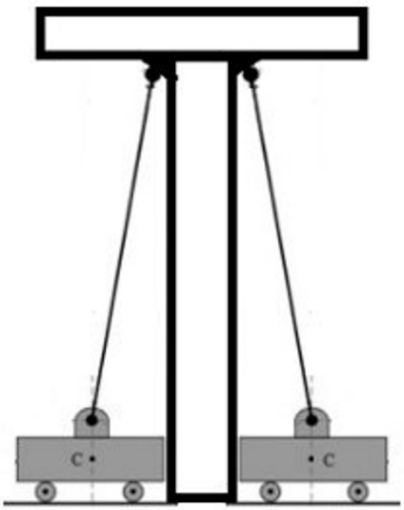


Figure 2.14- CART-POLE INSPIRED COORDINATION STRATEGY FOR SYNCHRONIZED T-JOINT WELDING

Following this initial investigation, a promising Industry 4.0 application can be identified in the field of advanced manufacturing, specifically within robotic welding processes. A particularly relevant case study involves T-joint geometries, a configuration widely used in structural and industrial fabrication. A T-joint weld is formed when one metal component is joined perpendicularly to another, creating the characteristic "T" shape. This type of joint is commonly employed in the assembly of frames, support structures, pipelines, and heavy-duty chassis due to its high structural integrity and load-bearing capacity. Despite its widespread use, T-joint welding poses several technological challenges, particularly in automated and robotic environments. The geometry of the joint generates complex thermal gradients and uneven heat distribution, which can lead to distortion or warping caused by residual stresses. Achieving uniform penetration depth and fusion quality on both sides of the joint is also difficult when welding is performed sequentially or from a single side. To address these limitations, future research could focus on the use of two collaborative robotic systems operating in a mirrored configuration (see Figure 2.14), simultaneously performing synchronized welds on opposite sides of the T-joint. This cooperative strategy could bring several benefits to robotic welding applications. By evenly distributing heat from both sides of the joint, it could enhance thermal symmetry and stress balance, helping to reduce distortion and improve dimensional accuracy. The simultaneous operation of two robots could also increase process efficiency, significantly shortening welding time and boosting overall productivity. In addition, the coordinated motion of the robotic arms could improve weld quality, ensuring more precise control of arc length, weld pool behaviour, and torch orientation, which in turn could lead to more consistent fusion and fewer defects. Finally, by integrating smart sensors and adaptive control algorithms, such as those based on LQR or ADMM coordination, the system could adapt in real time to misalignments, part variability, or environmental disturbances, resulting in a more robust and reliable welding process. Implementing such a control strategy, where each robotic system acts as an intelligent agent optimizing its local objectives while adhering to shared coordination constraints, mirrors the architecture explored in the dual cart-pole simulation could be a great object for a new future research.

It is important to emphasize that these preliminary investigations constitute an essential first step toward understanding the fundamental challenges inherent to nonlinear and coordinated control systems. Although the collected results are exploratory in nature, they have yielded valuable insights into system behaviour, stability margins, and control strategy effectiveness. Building on this foundation, future research should aim to deepen the theoretical analysis, refine the control architecture, and develop more advanced, adaptive methodologies capable of addressing the intrinsic complexity and coupling effects characteristic of modern cyber-physical systems.

2.7 SUMMARY ON METAHEURISTIC OPTIMIZATION IN CONTROL SYSTEMS

Metaheuristic optimization techniques represent powerful and versatile tools for the design, tuning, and enhancement of modern control systems.

In control system design, metaheuristics are widely applied for optimal gain tuning in PID controllers, multi-objective LQR formulations, model predictive control (MPC) horizon tuning, and robust control synthesis under uncertainty. These algorithms enable systematic exploration of large solution spaces to minimize control effort, tracking error, or energy consumption while satisfying stability margins and performance indices.

Moreover, hybrid schemes combining metaheuristics with classical control theory demonstrate superior performance in terms of convergence speed, adaptability, and robustness to disturbances and parameter variations. In safety-critical and nonlinear systems, metaheuristics facilitate constraint-handling and adaptive strategies, especially when integrated with machine learning models or surrogate approximations to reduce computational overhead.

This chapter presented an initial exploration of advanced control strategies applied to both single and multiple cart-pole systems, with a particular emphasis on coordination and optimization methodologies tailored to complex dynamic environments.

Section 2.1 offered a comprehensive review of control system strategies, covering both classical approaches, such as PID control, and modern methods including state-space representations and optimal control formulations like the Linear Quadratic Regulator (LQR). This foundational overview established the conceptual framework for the modelling and control developments addressed in subsequent sections.

In Section 2.2, the dynamics of a single cart-pole mechanism were modelled and analysed. The inherent nonlinearity of the system was emphasized through the derivation of mathematical equations governing the coupled motion of the cart and inverted pendulum. This analysis served as the basis for designing and evaluating both local and coordinated control schemes.

Section 2.3 introduced the Alternating Direction Method of Multipliers (ADMM) as a decentralized optimization framework suitable for constrained, distributed control problems. The theoretical principles of ADMM were detailed, illustrating its applicability to multi-agent coordination, particularly in systems requiring localized enforcement of global objectives.

Section 2.4 applied the ADMM algorithm to the coordination of cart and pole sensor feedback within a single cart-pole system. This implementation demonstrated how ADMM could effectively maintain sensor alignment, enforce local constraints, and enhance the stability and precision of the control response.

In Section 2.5, simulation results from the ADMM-based sensor coordination were presented and discussed. These results highlighted the algorithm's capacity to improve trajectory tracking and reduce control discrepancies, while also revealing limitations that could be addressed through further refinement of the control architecture.

Finally, Section 2.6 extended the investigation to a multi-agent scenario involving two cart-pole systems. Here, a hybrid control framework combining ADMM and Reinforcement Learning (RL) was proposed and implemented to achieve coordinated, mirrored motion along a predefined trajectory. The study showcased the potential of combining distributed optimization with data-driven learning techniques to handle high-dimensional, nonlinear coordination tasks. Such a framework demonstrated promising applicability to real-world

industrial contexts, such as robotic welding of T-joint geometries and coordinated assembly operations in Industry 4.0 environments.

Collectively, the chapter provided a coherent and systematic investigation into the integration of advanced control, optimization, and learning strategies. The findings underscored the value of hybrid approaches in managing complex dynamics, ensuring robust coordination, and paving the way for practical deployment in autonomous and intelligent manufacturing systems.

CHAPTER III: AI TOOLS FOR ADVANCED DESIGN AND PROCESSES

Through the subtle intuition of machines, man uncovers what was hidden within, weaving thoughts and knowledge into shared patterns, and together they resonate in new forms and unforeseen possibilities.

3.1 INTRODUCTION

This chapter explores the potential synergies between metaheuristic algorithms and a range of emerging systems and technologies, including social robotics, augmented and virtual reality (AR/VR), reverse engineering (RE), and additive manufacturing (AM), within the broader context of AI-driven tools for advanced design and intelligent processes.

In the domain of human-robot interaction (HRI), metaheuristic techniques are extensively utilized to optimize movement trajectories, facilitate obstacle avoidance, enhance postural and dynamic stability, and support the development of robust speech emotion recognition systems. Of particular significance is their capacity to serve as a connective framework between digital design processes, physical embodiment, and real-time interactive behaviour. Specifically in the field of social robotics, metaheuristic algorithms are instrumental in enhancing robot autonomy, adaptability, and contextual sensitivity. Such systems must be capable of responding not only to physical and environmental stimulations but also to social signals, behavioural norms, and emotional states or requirements that demand multi-objective and frequently non-deterministic optimization approaches. Beyond dyadic human-robot interactions, metaheuristics are also integral to multi-robot coordination within socially complex environments. They support adaptive role negotiation, action synchronization, and cooperative behaviour among multiple agents operating in the presence of human participants. These scenarios involve high levels of dynamic complexity, necessitating scalable and flexible optimization algorithms to effectively manage uncertainty and emergent group dynamics.

Within augmented and virtual reality (AR/VR) environments, metaheuristics play a critical role in enhancing system responsiveness, fidelity of user interaction, and spatial mapping precision, key requirements for responsive, adaptive, and immersive systems. These optimization algorithms are employed to fine-tune rendering parameters, calibrate multimodal sensor fusion, and improve user motion prediction, thereby minimizing latency and maximizing immersion. In AR-based training systems or context-aware navigation, metaheuristics contribute to the optimal placement and timing of visual signals, ensuring cognitive alignment and ergonomic effectiveness. Furthermore, in mixed-reality environments where avatars or robotic agents operate alongside human users, metaheuristics facilitate smooth, collision-free trajectory planning that preserves spatial coherence and situational awareness. When AR/VR platforms are integrated with autonomous robotic systems, such as in collaborative assembly, remote maintenance, or rehabilitation, metaheuristic approaches support the synchronization of behaviour across virtual and physical domains. This includes the alignment of sensor data with virtual representations, particularly relevant within digital twin architectures and telepresence frameworks.

AM, in addition to enabling the fabrication of geometries that are otherwise unachievable through conventional manufacturing processes, also supports the physical instantiation of socially interactive robots. These systems can be tailored to the specific needs of target user groups, including children, elderly individuals, and patients with cognitive or motor impairments. Metaheuristic algorithms aid in optimizing both mechanical aspects (e.g., joint configuration, mass distribution) and aesthetic features (e.g., facial geometry, ergonomic contours), ensuring the embodiment meets both functional and user-centred design criteria.

Furthermore, in the field of RE, metaheuristics are applied to optimize processes such as surface reconstruction, mesh refinement, and point cloud alignment, particularly in scenarios involving incomplete, noisy, or highly irregular data. These methods offer superior performance over traditional deterministic approaches when addressing non-convex, multimodal optimization problems inherent in the conversion of physical objects into accurate CAD models.

In the context of 3D printing, metaheuristics contribute to the optimization of build orientation, support structure reduction, path planning, and process variables such as thermal control and material deposition rates. These optimizations lead to reductions in production time and material waste, while simultaneously enhancing the mechanical properties and functional quality of printed components. In multi-material or functionally graded manufacturing, metaheuristics enable complex material distribution decisions through the application of evolutionary and swarm-based strategies. The integration of reverse engineering and additive manufacturing, for instance, in the reconstruction of legacy components or the customization of biomedical implants, benefits significantly from metaheuristic optimization strategies. These ensure that geometries derived from physical scans are not only dimensionally accurate but also manufacturable and structurally sound. This paradigm naturally extends to the biomedical engineering domain, where metaheuristics are applied in the optimization of exoskeletons and prosthetic devices. By tailoring mechanical properties and control algorithms to individual users, these systems enhance comfort, adaptability, and usability in everyday life, supporting greater autonomy and quality of life.

In conclusion, the integration of metaheuristic algorithms within broader artificial intelligence paradigms, in conjunction with emerging domains such as social robotics, augmented and virtual reality, reverse engineering, and additive manufacturing, delineates a fertile landscape of interdisciplinary innovation. In this context, metaheuristics function not merely as computational optimization techniques, but as core enablers of adaptive, personalized, and intelligent behaviours in autonomous systems operating across both physical and virtual domains. Their application underpins the development of AI-driven systems capable of dynamic interaction, contextual awareness, and continuous learning within increasingly complex and human-centred environments.

3.2 INVESTIGATION ON METAHEURISTIC OPTIMIZATION IN HUMAN-ROBOT INTERACTION DESIGN FOR ALPHA MINI

The Alpha Mini Robot (see Figure 3.1-) is a humanoid robot developed by UBTECH, designed for educational and interactive purposes (for technical details refer to Appendix 5). It is commonly used in schools and training programs to teach programming, robotics, and artificial intelligence (AI).



Figure 3.1- ALPHA MINI ROBOT

It also serves as a versatile platform for experimentation in advanced human-robot interaction, making it well-suited for the application of metaheuristic algorithms in areas such as motion optimization [33], adaptive behavioural control, path planning [34], and environment-aware responses [17] .

The primary objective of this investigation was to explore the integration of metaheuristic techniques into the Alpha Mini's interactive architecture, with the aim of enhancing its autonomy and responsiveness in complex, real-world scenarios. The study focused on the following key goals:

- Linguistic Adaptation: Enabling the robot to interact in Italian through the integration of the Vocs platform, which improves speech recognition and language understanding. Originally configured for English, the system was adapted for Italian, also allowing interoperability with AI-based conversational platforms such as Google Assistant and ChatGPT. This facilitates more natural, context-aware, and culturally relevant interactions.
- Speech Synthesis: Implementing Text-to-Speech (TTS) technology [35] to dynamically convert textual content into spoken Italian. This addition significantly enhances the robot's communication capabilities, enabling smooth and accessible verbal exchanges with users.
- Perceptual Augmentation: Employing computer vision techniques, particularly facial recognition and emotion detection, to enable the robot to perceive and interpret human affective states. These capabilities support more socially intelligent and empathetic interactions, in line with current trends in affective computing and human-centred robotics.

Together, these components form a multi-modal interaction framework whose performance and adaptability can be significantly improved through the application of metaheuristic algorithms.

These algorithms support the system in:

- Optimizing speech recognition thresholds and dialogue flow to handle diverse accents and interaction contexts.

- Refining emotion classification models by adjusting classifier hyperparameters.
- Tuning TTS parameters (e.g., pitch, intonation, speed) for better intelligibility and user engagement.
- Balancing processing loads across subsystems to ensure real-time responsiveness.
- Enhancing sensor fusion strategies for more accurate user state estimation through the integration of audio-visual data.

By embedding metaheuristic optimization within these AI-driven subsystems, the Alpha Mini robot can achieve adaptive, personalized, and robust behaviours, key attributes for autonomous systems operating across dynamic and human-centred environments.

The solution proposed by the computer science working group at the Department of Informatics, University of Aldo Moro of Bari, with which I collaborated during my doctoral studies, under the leadership of Prof. Berardina De Carolis, involves the development of a server implemented in Python, designed to manage natural language interaction requests originating from the robot. The operational flow proceeds as shown in Figure 3.2- : the robot listens to a spoken phrase from the user; the audio input is then transmitted to the server via an API-based architecture. The server, in turn, queries ChatGPT, specifically leveraging the custom assistant functionality, to generate a coherent and contextually appropriate response. This textual response is subsequently returned to the robot, which vocalizes it using Google Text-to-Speech (GTTS) technology. This architecture enables dynamic and customizable interactions, while maintaining a clear separation between the advanced language processing module (hosted on the server) and the robot's execution layer. Such modularity enhances scalability, maintainability, and facilitates potential extensions to multilingual or multi-agent systems.

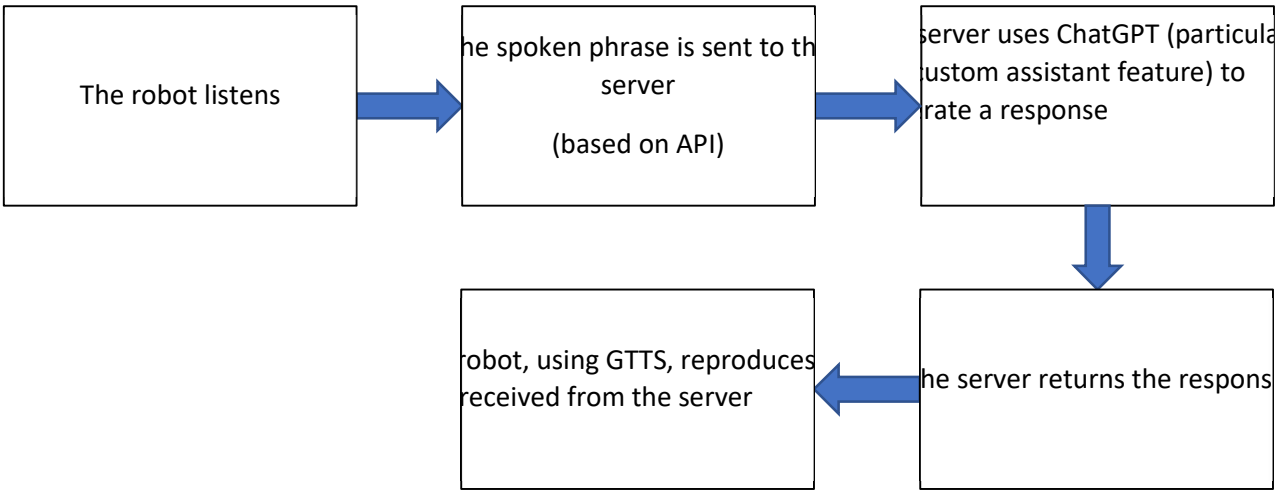


Figure 3.2- USER-ROBOT INTERACTION FLOWCHART

Docker was employed as the primary tool for the deployment of the system. Docker is an open-source platform that enables applications to be executed within containers, lightweight, isolated environments that package the software together with all its dependencies. This approach ensures consistent execution across diverse environments, significantly facilitating development, testing, and distribution processes.

The initial basis for development was a publicly available demonstration hosted on GitHub, created by a Chinese developer and specifically designed for the Alpha Mini robot. This project provided a foundational structure that was instrumental in understanding the robot's functionality and its interaction within the Android environment. The demonstration's source code was reviewed, adapted, and integrated into the current work.

During the implementation phase, particular attention was given to the modification of the MainActivity.java file, located at:

ubt_alphamini_app_demo/app/src/main/java/com/ubtrobot/mini/sdkdemo/MainActivity.java

This file constitutes one of the main entry points of the Android application and required substantial modifications to align the project's logic with the specific objectives of the study. The changes involved integration with external services, customization of the robot's voice functionalities, and the adaptation of the interaction flow with a remote server.

It is important to highlight that all APIs used in the system were developed from scratch and are hosted on a dedicated server. The Android application, developed and tested using Android Studio, communicates with this server by sending HTTP requests and receiving responses that drive the robot's interaction with the user.

Regarding the specific objectives of the project several core components have already been implemented:

- The robot's default voice output has been disabled to allow the integration of custom voice processing tools.
- Two main components were integrated for voice management: **VOSK**, an open-source offline speech recognition toolkit, and **Google Text-to-Speech (GTTS)** for speech synthesis.

The implemented voice interaction flow operates as follows: the user speaks to the robot, which uses VOSK to transcribe the audio locally into text. The transcribed text is then sent to the server, where it is processed, potentially using language models such as GPT. The server-generated textual response is subsequently converted into speech by the robot using GTTS, enabling continuous and natural verbal interaction between user and machine.

This modular, server-based architecture offers a high degree of flexibility, allowing each component (speech recognition, processing, and synthesis) to be developed and improved independently without requiring changes to the entire system.

The final architecture developed to achieve the initial tasks has been summarized in Figure 3.3- as follows:

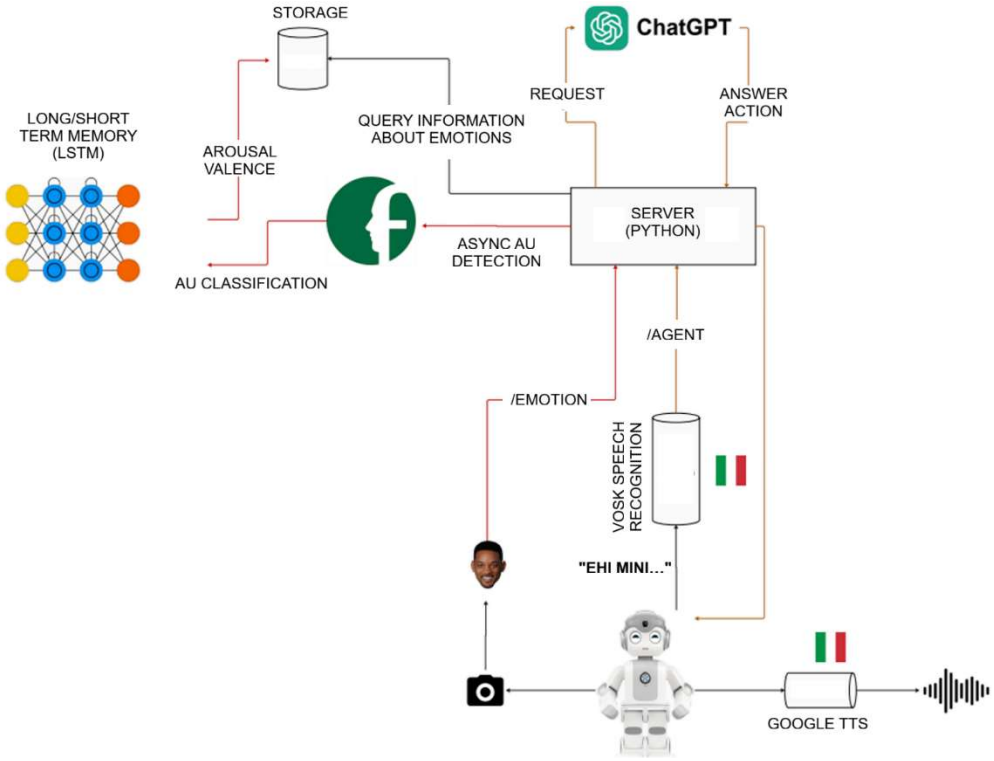


Figure 3.3- FINAL ARCHITECTURE DEVELOPED FOR LINGUISTIC ADAPTATION, SPEECH SYNTHESIS AND PERCEPTUAL AUGMENTATION

Following the evaluation of the work, enhancements are proposed through the integration of metaheuristic optimization techniques:

1) Optimization of Speech Recognition Models, implementing a genetic algorithm that automatically adjusts VOSK model hyperparameters by training on a dataset of Italian speakers with various regional accents and background noises [36] [37].
   The fitness function could be based on by measuring the Word Error Rate (WER) on a validation subset. The system could run periodic offline training sessions to evolve the best parameter sets, then update the robot's recognition model accordingly. Over multiple generations, the genetic algorithm converges on an optimal configuration that significantly reduces transcription errors in noisy classroom environments, thus improving the robot's ability to understand diverse users.
   The mathematical model could be defined as follows:

- Let $ASR(\theta)$ be the speech recognition model trained (or fine-tuned) with hyperparameter configuration θ.
- Let $D = \{(x_i, y_i)\}_{i=1}^{N}$ be the validation dataset, where x_i is the audio input, y_i is the corresponding ground-truth transcription, N= total number of words in the reference transcription
- Let $\hat{y}_i = ASR(\theta)(x_i)$ be the predicted transcription for input x_i

The objective function could be defined as:

$$f = \min_{\theta \epsilon \Theta} WER(\theta) = \frac{1}{N} \sum_{i=1}^{N} \frac{S_i + D_i + I_i}{T_i} \quad (3.1)$$

where Θ= feasible set of hyperparameter configurations, Si= Number of substitutions, Di= Number of deletions, Ii= number of insertions, Ti= number of words in the reference transcript, N= total number of words in the reference transcription, θ = model hyperparameters (learning rate layer count, dropout rate, etc.= $\{\eta, d, h, b\}$ with $\hat{\eta}$ = learning rate of acoustic model, d= dropout rate, h= number of hidden layers, b= batch size during training.

2) Optimizing facial emotion recognition for enhanced human-robot interaction with metaheuristics by tuning the hyperparameters of a convolutional neural network (CNN) to maximize the F1-score on a validation dataset of labelled samples from real human-robot interactions.
The mathematical model could be defined as follows [38] [39]:

- Let $\phi = \{k.l.f, \tau\}$ represent the set of adjustable hyperparameters with k= kernel size, l= number of convolutional layers, f= number of filters per layer, τ= classification threshold
- Let $D_{val} = \{(X_i, y_i)\}_{i=1}^{N}$ denote the labelled validation dataset with X_i= input image, y_i= ground- truth emotional label, $\widehat{y_i} = CNN_{\phi}(X_i)$ be the model prediction given ϕ.

The objective function could be the maximization of the F1-score which balances precision and recall:

$$f = \max_{\phi \in \Phi} F1(\phi) = 2 \cdot \frac{P(\phi) \cdot R(\phi)}{P(\phi) + R(\phi)} \tag{3.2}$$

where

$$P(\phi) = \frac{TP(\phi)}{T(\phi) + FP(\phi)} \tag{3.3}$$

$$R(\phi) = \frac{TP(\phi)}{T(\phi) + FN(\phi)} \tag{3.4}$$

TP, FP, FN are computed from model prediction on D_{val}.

3) Deploy metaheuristics to optimize TTS parameters such as pitch, speaking rate, and intonation style, with the aim of generating more natural and engaging Italian speech output, personalized for distinct user groups such as children or adults. For instance, the optimization model for TTS parameter tuning could be defined as the maximization of the overall speech quality as follows [40] [41]:

$$f = \max_{\phi \in \Phi} \frac{1}{N} \sum_{i=1}^{N} Q(y_i(\phi), u_i) \tag{3.5}$$

where
$\phi = \{p, r, s\}$ denote the vector of adjustable parameters in the TTS system, where p controls the pitch, r the speaking rate, and s the intonation style.
Φ is the admissible space of parameter configurations.
N represents the number of evaluations samples.
$Q(y_i(\phi), u_i)$ is the quality assessment function that evaluates how well the synthesized output matches the intended characteristics. This could be based on perceptual metrics

such as MOS (Mean Opinion Score) or objective measures like MCD (Mel Cepstral Distortion).
$y_i(\phi)$ denotes the synthesized speech output for the i-th input using parameters ϕ.
ui denotes the target user-specific speech characteristics for the i-th case (e.g., a child's preferred prosody).

4) Optimize the fusion weights assigned to audio-based sentiment analysis and visual emotion recognition outputs within the sensor fusion module, with the goal of enhancing the robot's real-time estimation of the user's emotional and cognitive state. This optimization enables more accurate, context-aware, and socially appropriate responses during human-robot interactions [42], [43].

The possible optimization problem could be to minimize the error in estimating the user's emotional and cognitive state $\hat{y}$ compared to the ground truth state y, as follows.

Let x_a , x_v be the feature vector respectively from audio-based sentiment analysis and from visual emotion recognition; $f(x_a)$, $f(x_v)$ represent the emotion/cognitive state prediction vectors from individual modalities; ω_a , ω_v be the fusion weights for audio and visual prediction; $\hat{y} = \omega_a \cdot f(x_a) + \omega_v \cdot f(x_v)$ denote the fused output prediction.

The objective function can be defined as:

$$\min_{\omega_a, \omega_v} \Gamma(\omega_a, \omega_v) = \frac{1}{N}\sum_{i=1}^{N} \left\| y^{(i)} - (\omega_a f_a\left(x_a^{(i)}\right) + \omega_v f_v(x_v^{(i)}) \right\|^2 \quad (3.6)$$

The possible constraints could be defined as: $\omega_a + \omega_v = 1, \quad \omega_a, \ \omega_v \geq 0$, obtaining a constrained quadratic optimization problem in two variables to be solved using Lagrangian multipliers or projected gradient descent or constrained convex optimization solvers.

5) Optimize Alpha Mini motion trajectories, incorporating constraints related to kinematics, dynamics, stability, and emotion expressiveness. The goal is to generate motions that are both physically feasible and socially meaningful, enabling more natural and engaging interactions with human users, as follows:

Let

- $q(t) \in R^n$: joint position at time t, n= the number of joints.
- $\dot{q}(t)$, $\ddot{q}(t)$: joint velocity and acceleration.
- $\tau(t)$: joint torque
- T: duration of the motion
- $Q = \{q(t)\}_{t=0}^{T}$: joint trajectory
- λ_1, λ_2: weight parameters
- $D_{emotion}(Q, e)$: distance between the executed motion and the prototypical emotion e
- $ZMP(q, \dot{q}, \ddot{q})$: Zero moment Point

The optimization problem could be defined as follows:

$$\min_{q(t)} J = \int_0^T \left\| q(\ddot{t}) \right\|^2 dt + \lambda_1 \int_0^T \|\tau(t)\|^2 dt + \lambda_2 D_{emotion}(Q, e) \quad (3.7)$$

with

- the kinematic constraints:

$$q_{min} \leq q(t) \leq q_{max} \text{ (joint limits)}, \|\dot{q}(t)\| \leq v_{max}, \|\ddot{q}(t)\| \leq a_{max} \quad (3.8)$$

- the dynamic constraints:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = \tau \quad (3.9)$$

- the stability constraints: ZMP must lie within the support polygon for walking:

$$ZMP(q, \dot{q}, \ddot{q}) \in P_{support} \quad (3.10)$$

- the synchronization constraints. If synchronized with sounds, define keyframes q_k at time t_k:

$$q(\tau_k) = q_k \quad (3.11)$$

3.3 INVESTIGATION ON METAHEURISTIC OPTIMIZATION IN AUGMENTED AND VIRTUAL REALITY

Immersive technologies such as Augmented Reality (AR) and Virtual Reality (VR) have witnessed rapid advancements in recent years, transforming how users perceive and interact with digital content [44]. AR (see Figure 3.4.a) enhances the physical environment by overlaying digital elements, while VR (see Figure 3.4.b) offers complete immersion in a virtual space.



Figure 3.4- COMPARATIVE OVERVIEW OF AUGMENTED REALITY (a) AND VIRTUAL REALITY TECHNOLOGIES (b)

Among AR applications, holographic displays, enabled by holographic and Meta surface technologies, allow the projection of full-colour, three-dimensional imagery directly into the user's environment, creating a seamless blend of real and virtual worlds (see Figure 3.5).

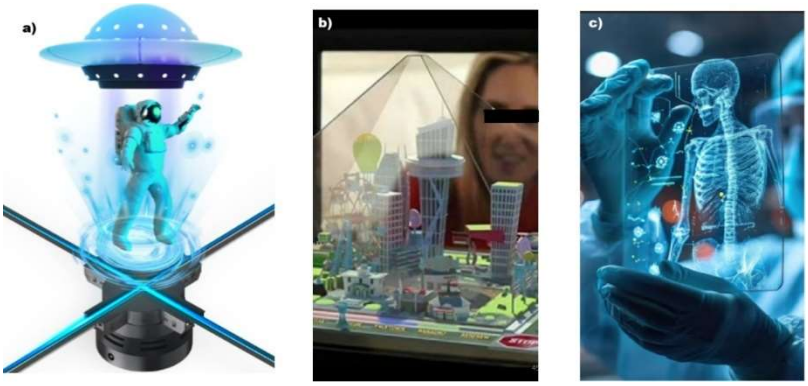


Figure 3.5- MAIN HOLOGRAM TYPES: (a) FAN HOLOGRAM, (b) REFLECTION PYRAMID HOLOGRAM, (c) COMPUTER-GENERATED HOLOGRAM

In the development of AR/VR-enabled robotic systems, real-time visual perception is a critical component for interpreting dynamic environments and enabling autonomous decision-making. One of the most interesting tools in this domain is the Open-Source Computer Vision Library (OpenCV). This library enables image processing, object detection, and camera calibration. Moreover, it provides various functions that can be used in motion tracking and real-time robotic vision pipelines, implemented in languages such as Python and C++.

To address the various challenges associated with AR and VR, metaheuristic optimization algorithms, inspired by natural processes, have emerged as effective solutions for high-dimensional, non-convex design spaces in these fields. Examples include Genetic Algorithms [45], Particle Swarm Optimization [48], Variable Neighbourhood Search [47], and Bat Algorithms[48], to optimize various aspects related to AR and VR. Additionally, both VR and AR provide powerful platforms for simulating robotic environments, enabling the testing of navigation, perception, and interaction strategies within controlled yet realistic settings. These immersive environments facilitate real-time visualization of sensor data captured by robots, enabled by advanced computer vision technologies [49], which integrates optimization problems, image processing, deep learning, and pattern recognition to interpret visual information, extract meaningful features, and support decision-making based on image or video input. Modern algorithms, such as those leveraging convolutional neural networks (CNNs) and transformers [50], give robust object detection, semantic segmentation, depth estimation, and scene reconstruction, which are essential for intelligent robotic behaviour in complex soils. To achieve high-level perception these systems rely on deep learning frameworks like TensorFlow and PyTorch [51], which support the implementation of advanced models (e.g., YOLO, Faster R-CNN) [52], and semantic segmentation networks. These models underpin core capabilities such as visual SLAM, scene reconstruction, and object pose estimation.

The Unity real-time 3D engine complements these perception algorithms and permits an extensible simulation environment suitable for AR/VR applications and robotic testing [55] [56], through modules such as AR Foundation, Unity supports deployment across ARKit, ARCore, and XR headsets. This setup may also encompass real-time sensor visualization,

vision-based interactions, the testing of robots or rovers within a variety of immersive virtual environments (see Figure 3.6-), and the development of digital twins.



Figure 3.6- REALISTIC TRAJECTORY SIMULATION IN UNITY IN DIFFERENT ENVIRONMENTS

FreeCAD [55], while not a native VR tool, functions as a parametric, open-source CAD platform that facilitates the design and optimization of mechanical components and environments. Its compatibility with common 3D exchange formats allows the direct export of CAD models into Unity, establishing a seamless pipeline from geometric design to immersive simulation. This integration is particularly beneficial for the iterative evaluation of robotic hardware, wearable systems, or optical elements used in AR/VR interfaces.

Crucially, FreeCAD's support for Python scripting permits the integration of artificial intelligence and metaheuristic optimization algorithms to automate multi-objective design processes. These methods enable the exploration of complex, high-dimensional design spaces subject to competing constraints such as structural integrity, weight minimization, and ergonomic performance. The optimized geometries can then be validated in Unity-based simulation environments under realistic operational conditions and solved with metaheuristic algorithms, as explained below.

As a next step, a mathematical model is now introduced to formally represent the key components and mechanisms outlined in the preceding analysis.

Let $O \subset R^3$ be a three-dimensional (3D) object defined by its three orthographic binary projections respectively on the xy-plane $P_{xy}: R^2 \to \{0,1\}$, xz-plane $P_{xz}: R^2 \to \{0,1\}$, yz-plane $P_{yz}: R^2 \to \{0,1\}$.

We can discretize the 3D space into a uniform voxel (volumetric pixel) grid over a finite domain $\Omega = \{0, \dots, N_x - 1\} \cdot \{0, \dots, N_y - 1\} \cdot \{0, \dots, N_z - 1\} \subset Z^3$. Each voxel is assigned to a binary variable $V(x, y, z) \in \{0,1\}, \forall (x, y, z) \in \Omega$, where $V(x, y, z) = 1$ indicates the presence of the object at voxel (x, y, z). To create consistency between the voxel volume V and the projection P, its maximum intensity along the projection axis must match the given projection.

We can define the objective function as the total loss, as follows:

$$\mathfrak{L}(V) = \lambda_{xy} \cdot \mathfrak{L}_{xy}(V) + \lambda_{xz} \cdot \mathfrak{L}_{xz}(V) + \lambda_{yz} \cdot \mathfrak{L}_{yz}(V) \tag{3.12}$$

in which the individual projection losses that penalize inconsistencies between the projections of the reconstructed volume and the ground-truth projection, can be given by:

$$\mathfrak{L}_{xy}(V) = \sum_{x=0}^{N_z-1} \sum_{y=0}^{N_y-1} |\max_z V(x,y,z) - P_{xy}(x,y)| \tag{3.13}$$

$$\mathfrak{L}_{xz}(V) = \sum_{x=0}^{N_z-1} \sum_{z=0}^{N_z-1} |\max_y V(x,y,z) - P_{xz}(x,z)| \tag{3.14}$$

$$\mathfrak{L}_{yz}(V) = \sum_{y=0}^{N_y-1} \sum_{z=0}^{N_z-1} |\max_x V(x,y,z) - P_{yz}(y,z)| \tag{3.15}$$

Furthermore, we can define the following projection constraints on the three planes:

$$\forall x,y\colon P_{xy}(x,y) = \begin{cases} 1 \; if \; \exists z \; s.t. V(x,y,z) = 1 \\ 0 \; otherwise \end{cases} \tag{3.16}$$

$$\forall x,z\colon P_{xz}(x,z) = \begin{cases} 1 \; if \; \exists y \; s.t. V(x,y,z) = 1 \\ 0 \; otherwise \end{cases} \tag{3.17}$$

$$\forall y,z\colon y_{xz}(y,z) = \begin{cases} 1 \; if \; \exists x \; s.t. V(x,y,z) = 1 \\ 0 \; otherwise \end{cases} \tag{3.18}$$

To ensure the physical plausibility and computational tractability of the reconstruction, we can impose the following additional constraints:

- Binary constraints:

$$V(x,y,z) \in \{0,1\}, \forall (x,y,z) \in \Omega \tag{3.19}$$

- Volume constraint to avoid overfitting or generating unfeasible large objects:

$$\sum\nolimits_{(x,y,z)\in\Omega} V(x,y,z) \leq V_{max} \tag{3.20}$$

- Regularization constraint to ensure smoothness/compactness:

$$\Re(V) = \sum_{(x,y,z)\in\Omega} \sum_{\delta\in N} |V(x,y,z) - V(x+\delta_x, y+\delta_y, z+\delta_z)| \tag{3.21}$$

where N is a set of neighbouring voxel offsets, the symbol δ denotes a displacement vector pointing to a neighbouring voxel relative to the current voxel (x, y, z) and $\Re(V)$ promotes spatial coherence.

Finally, the total cost function could be then expressed as:

$$\min_{V\in\{0,1\}^{\Omega}} \mathfrak{L}(V) + \lambda_{reg} \cdot \Re(V), s.t. \sum V \leq V_{max} \tag{3.22}$$

Upon formal definition of the mathematical problem, a metaheuristic algorithm may be applied to efficiently explore the solution space and derive high-quality approximate solutions.

Another intriguing application of metaheuristics in VR and AR could involve developing an advanced mathematical algorithm to integrate VR into the Digital Twin paradigm for enhanced industrial process control [56]. The proposed system could facilitate interactive and parameterized exploration of sensor configurations across real-time and post-process phases of manufacturing. Using holographic interfaces, virtual sensors may be deployed within a three-dimensional digital representation of the production environment, enabling the evaluation of system responses and the optimization of sensor layouts for enhanced monitoring and control performance. The research work could consist in formulating the sensor placement and configuration task as a constrained multi-objective optimization problem to solve with a hybrid metaheuristic model combining elements from evolutionary computation and swarm intelligence, tailored to exploit the structural and temporal nature of process data captured in the Digital Twin. The mathematic model could be developed as described below.

Let define the industrial process be represented by a digital twin $T(t, \bar{x})$, where $t \in \{0, T\}$ is the time and $\bar{x} \in R^3$ representing a generic point in the space, $S = \{s_1, \dots, s_n\}$ be a set of sensors defined by $p_i \in \Omega$ (positions), $\theta_i \in \Omega \subset R^k$(sensors parameters as range, sensitivity, frequency), $c_i(\theta_i)$ be the cost of sensor s_i, $q_i(p_i, \theta_i)$ be the quality of information obtained by sensor s_i.

The problem is to maximize the observability and minimize the total cost, redundancy and latency of the sensors:

$$\max_{p_i,\theta_i} F(S) = \alpha \cdot Q(S) - \beta \cdot C(S) - \gamma \cdot R(S) - \delta \cdot L(S) \tag{3.23}$$

where

- $Q(S) = \sum_{i=1}^{n} q_i(p_i, \theta_i)$: total information gain
- $C(S) = \sum_{i=1}^{n} c_i(\theta_i)$: total cost
- $R(S)$: redundancy penalty (overlapping fields of view)(S)
- $L(S)$: expected latency of detection and response

- $\alpha, \beta, \gamma, \delta$: user defined weights for trade-offs

The constraints can be formally expressed as follows:

- $p_i \in \Omega$: sensor positions must lie within the process space
- $\theta_i \in \Theta$ (positions): sensor parameters must be within valid operational ranges
- $C(S) \leq C_{max}$: total budget constraint
- $Q(S) \geq \varepsilon_r$ for each subregion $r \in R$: minimum information gain per subregion (coverage constraint)

After defining the objective functions and constraints, it is necessary to select a metaheuristic algorithm suited to address the multi-objective, constrained, and high-dimensional nature of the optimization problem. In this context, each candidate solution S is represented as a vector of sensor configurations $S = [(p_1, \theta_1), \ldots, (p_n, \theta_n)]$. This encoding allows for variable-length individuals, as the number of sensors n can vary between solutions.

Accordingly, the application of metaheuristics proceeds through several well-defined phases, which have been introduced in the preceding chapters and will be examined in greater detail in the following sections:

- Initialization: Random feasible sensor sets, respecting constraints
- Mutation: Perturb position and parameters
- Crossover: Combine two sensor sets via k-point recombination
- Swarm Guidance: Use PSO-like memory of best configurations to guide exploitation.
- Constraint Handling: Penalty functions + repair operators + feasibility preservation

The Algorithm Flow consists in:

- Initialize population
- Evaluate the objective function for each candidate
- Select best-performing configurations using non-dominated sorting
- Apply mutation, crossover, swarm guidance
- Update Population
- Repeat the loop until convergence.

By using a VR/holographic interface, it becomes possible to visualize sensor coverage heatmaps in real-time, parametrically move sensors and observe feedback, switch between optimization modes: cost-focused, redundancy-free, latency-minimal and override the metaheuristic locally to "nudge" the solution (human-in-the-loop optimization).

A highly relevant application in Industry 4.0, particularly within the domain of additive manufacturing, could focus on optimizing the real-time placement and configuration of temperature sensors in a laser-based metal printing process, thereby enhancing both process monitoring and control. Sensor positioning must be optimized to maximize the detection of thermal anomalies, while minimizing feedback latency, which directly impacts the dynamic control of the laser. Sensors can be virtually tested within a VR environment using a precomputed digital twin of the process, and the metaheuristic can continuously adapt and learn from simulated feedback loops. One interesting investigation could involve the integration of

reinforcement learning (RL) techniques into the sensor selection and placement policy or explore the joint optimization of sensor and actuator configurations, along with the underlying control logic. This would not only improve system responsiveness but also enable proactive interventions in the process, transforming the digital twin from a passive mirror of the physical world into an active agent capable of real-time decision support. Another critical direction could be the development of real-time reconfigurable digital twins. Unlike static or periodically updated models, a dynamically synchronized digital twin would incorporate real-time data ingestion, online parameter updating, and continuous calibration. Such capabilities are essential in high-speed manufacturing environments where rapid variations in process conditions must be mirrored immediately in the virtual model to maintain simulation fidelity and control accuracy.

Figures 3.7 and 3.8 present the FreeCAD-based optimization of a confined immersive environment, referred to as "Holobox" (for major technical details refer to Appendix 6), developed during my doctoral studies in collaboration with Prof. Gattullo from the Department of Mechanics, Mathematics, and Management at the Polytechnic University of Bari. HoloBox is a purpose-built confined room designed for the acquisition of visual and spatial data aimed at the creation of holographic digital twins. The system integrates three high-resolution RGB-D sensors (such as Intel RealSense or Microsoft Azure Kinect), strategically placed along three orthogonal planes. This configuration follows the principles of orthographic projection, enabling the simultaneous capture of depth and RGB data from multiple viewpoints. The acquired data undergoes processing and optimization to reconstruct accurate three-dimensional models, which are then used to generate immersive holographic representations of physical objects or environments.

Operating within a parametric CAD-based virtual simulation environment, the system also enables virtual prototyping and optimization of sensor placement (see Figure 3.8-), allowing for performance validation and fine-tuning without the need for physical assembly. This virtual testing approach significantly reduces time and cost associated with iterative hardware setup.

HOLOBOX:

TOTAL BULK=1000x1000x1950

MATERIAL: ALLUMINIUM

Seq. Num.	ITEM DESCRIPTION	DIMENSIONS	QTY
1	Aluminium Profile 30x30x1950 AW6060- groove 8mm	30 x 30 x 1950	4
2	Aluminium Profile 30x30x940 AW6060-grove 8mm	30 x 30 x 940	12
3	Model 3030 Corner Code-Aluminium	30x35x35	48
4	Flat Gasket M6x1-Aluminium (lot of 50 pieces)	M6x1	24
5	Hexagon Socket Screw M6-Aluminium	M6	48
6	Slider Nut for groove 8mm-Aluminium (lot of 50 pieces)	M6	48
7	Black or Green polypropylene panel for covering	1000x1000x3	1
8	Black of Green polypropylene panel for front strip + support base	1000x2000x3	3
9	Black Curtain fireproof 1000x2100	1000x2000	1
10	TP-Link TC70 Spherical Surveillance Camera IP Security Camera Indoor Ceiling/Wall		3
11	Ball Bearing Drawer Slides Guide, 10 kg Load Capacity, Full Extension, Bearings Total Extraction	50 cm	1
12	Linear Actuator 6000N 300mm / 1320lb 12 Inches, Stroke DC 12V Electric Motor	Return Length: 18,7 ich=48cm	1

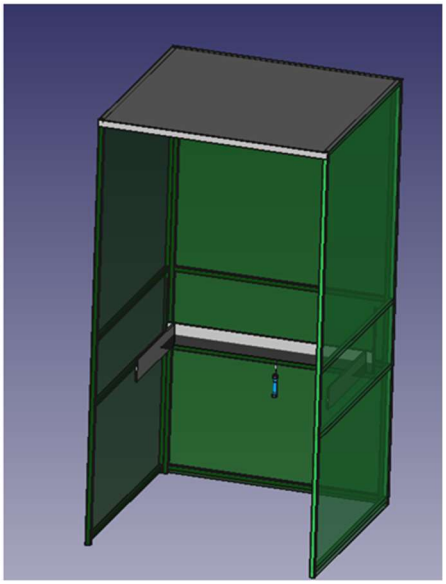


Figure 3.7- FreeCAD-BASED MECHANICAL OPTIMIZATION FOR A HOLOGRAPHIC IMMERSIVE ENVIRONMENT (HOLOBOX)

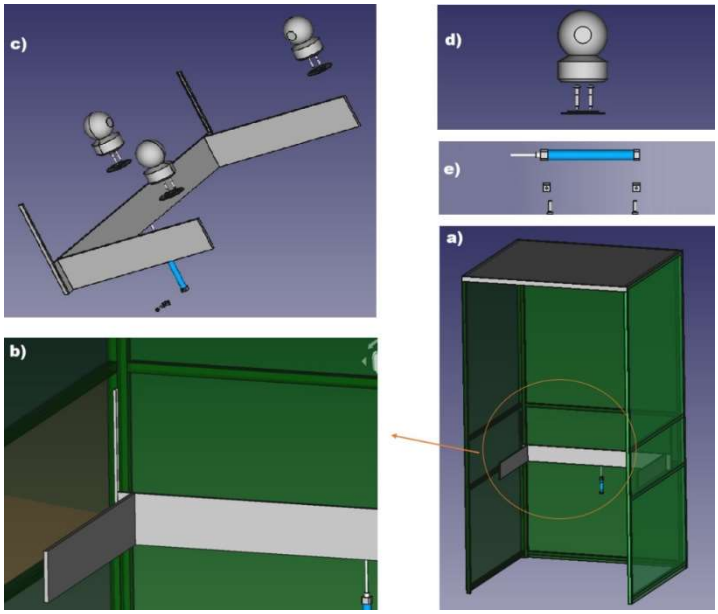


Figure 3.8- HOLOBOX COMPONENTS: a) MOTORIZED CAMERA POSITIONING SYSTEM; b) SLIDING MECHANISM DETAIL; c) LINEAR GUIDE SYSTEM FOR CAMERA ADJUSTMENT; d) CAMERA FIXTURE; e) ACTUATOR FIXTURE.

Following the acquisition of images along three orthogonal planes using RGB-D sensors, a complete holographic reconstruction of the observed object was carried out. This process, conceptually analogous to medical imaging techniques that utilize multi-planar data acquisition, was implemented through a reflective pyramid projection system, as illustrated in Figure 3.9. The resulting setup enabled the volumetric visualization of the object within a confined immersive environment, providing a spatially coherent and perceptually rich representation of holographic digital twins.

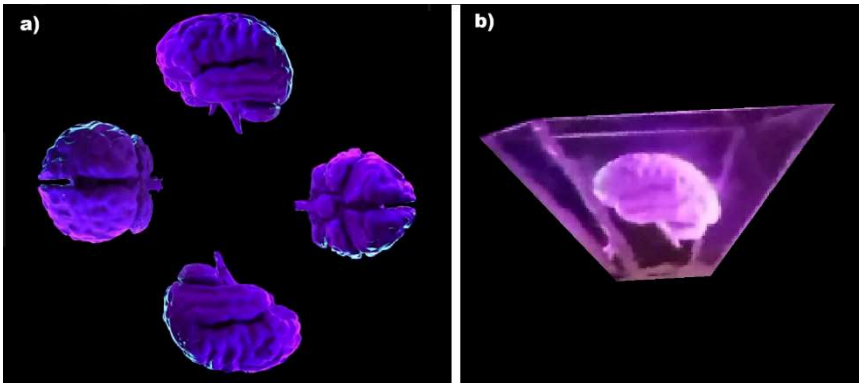


Figure 3.9- IMAGES FOR HOLOGRAM BASED ON MULTI-PLANAR DATA ACQUISITION (a); REFLECTIVE PYRAMID PROJECTION SYSTEM (b)

The reconstruction process confirmed the theoretical feasibility of using orthogonal projections for three-dimensional representation; however, further research is required to explore effective methods for image synthesis based on metaheuristic approaches to recompose 3D structures from the three orthogonal views. Specifically, generating coherent and realistic volumetric representations from front, top, and side projections remains a complex optimization problem. From a computational vision standpoint, the reconstruction of 3D volumetric models from orthographic projections is a classic but underdetermined inverse problem. The ambiguity

arises from the loss of depth information in the projections, making the problem inherently ill-posed and highly non-convex. By formulating the task as a constrained optimization over a discrete voxel grid, and by leveraging view consistency losses and structural priors, one can approximate solutions that are geometrically plausible and projection consistent. The mathematical model could be solved using a metaheuristic approach in which each candidate solution V in the population encodes a 3D binary structure. The objective function guides the search toward configurations that best match the provided projections while respecting the constraints, as described below. To sum up, a critical challenge in this domain lies in the optimization of image and sensor placement, that is, determining the most effective spatial configuration to ensure accurate, continuous, and artifact-free holographic visualizations across all viewing angles. Addressing this challenge necessitates advanced computational strategies that can be implemented with metaheuristic optimization techniques, as previously explained. It seems particularly interesting to explore these methods on Optimal Sensor Placement problems [57]. This could be realized by embedding holographic reconstructions within a CAD-based virtual prototyping environment so to iteratively simulate and refine sensor configurations prior to physical deployment, thereby minimizing time, reducing costs, and eliminating the need for repetitive hardware adjustments. These findings suggest that the synergy between immersive system design, holographic rendering, and metaheuristic-driven optimization not only advances current capabilities in digital twin generation but also opens new avenues for research in adaptive visualization, real-time simulation, and intelligent sensor networks. Further refinement of this simulation-design loop is enabled by advanced toolkits such as MATLAB's Computer Vision Toolbox and NVIDIA's DeepStream SDK, which provide capabilities for video analytics, multi-object tracking, and GPU-accelerated inference. For photorealistic simulation and synthetic data generation, NVIDIA Isaac Sim (see Figure 3.10- 0) can be integrated into the Unity workflow, supporting high-fidelity evaluation of vision-based robotic algorithms. Collectively, the interoperability between FreeCAD and Unity, combined with AI-driven optimization and advanced perception frameworks, can establish a robust flow for the design, simulation, and validation of intelligent robotic systems operating in complex, visually dynamic environments.



Figure 3.10- ISAAC SIM-ROBOTICS SIMULATION

In this context, metaheuristics are used to support the design of such systems, which involves complex, multi-objective variables: balancing display quality, ergonomics, computational intensity, and real-time responsiveness. For example, tuneable holographic meta surfaces must precisely manipulate light at the sub-wavelength scale, requiring optimization across various

parameters such as wavelength selectivity, angular response, and fabrication constraints [58]. Furthermore, the use of AR in robotics has emerged as a powerful tool to enhance perception, interaction, and control across medical, industrial, and multi-agent applications, despite ongoing challenges in localization and real-time integration [59]. For these applications, metaheuristic algorithms can also be effectively employed to optimize the mechanical design of Virtual Reality (VR) and Augmented Reality (AR) devices, with the aim of improving wearability, structural integrity, and user-specific adaptability, while simultaneously addressing a range of conflicting design constraints. However, despite the growing interest in these technologies, the application of metaheuristic optimization to the mechanical design of holographic systems remains relatively underexplored in the literature. This represents a promising area for future research, particularly in improving structural performance, wearability, and user-specific adaptability of holographic display devices.

3.4 INVESTIGATION ON METAHEURISTIC OPTIMIZATION IN REVERSE ENGINEERING AND ADDICTIVE MANUFACTURING

The continuous evolution of manufacturing technologies has ushered in a new era of innovation, where reverse engineering (RE) and additive manufacturing (AM) play a central role in redefining product development, customization, and production efficiency, as highlighted in the master's program I attended during my doctoral studies, coordinated by Prof. Casalino from the Department of Mechanics, Mathematics, and Management at the Polytechnic University of Bari, titled *"Reverse Engineering and 3D Printing"*. These technologies are foundational pillars of Industry 4.0 [60], characterized by the convergence of cyber-physical systems, the Internet of Things (IoT) [61], artificial intelligence, and advanced data analytics, all aimed at enabling intelligent, interconnected, and autonomous manufacturing environments. Beyond the automation-centric vision of Industry 4.0, the emerging paradigm of Industry 5.0 [62] introduces a more human-centric perspective, prioritizing collaboration between humans and machines, ethical considerations, personalization of products and services, and environmental sustainability. In this context, advanced technologies are not only tools for efficiency but also enablers of societal and ecological value. Looking ahead, conceptualizations of Industry 6.0 [63] envision a hyper-connected, AI-driven industrial ecosystem characterized by full autonomy, cognitive computing, and real-time decision-making across decentralized systems. In Table 3.1 4 a comparative summary is provided, outlining the main differences between Industry 4.0, Industry 5.0, and the emerging Industry 6.0, with particular emphasis on their conceptual focus, human involvement, core objectives, and technological maturity.

Table 3.1 4-Differences among Industry 4.0, 5.0, and 6.0

ASPECT	INDUSTRY 4.0	INDUSTRY 5.0	INDUSTRY 6.0
KEYWORD	Intelligent automation	Human-centricity and sustainability	Cognitive autonomy
ROLE OF THE HUMAN	Minimal	Central	Strategic but increasingly residual
PRIMARY OBJECTIVE	Efficiency and digitalization	Collaboration and product personalization	Autonomous decision-making and self-adaptability

ASPECT	INDUSTRY 4.0	INDUSTRY 5.0	INDUSTRY 6.0
TECHNOLOGICAL MATURITY	Widely implemented	Emerging	Conceptual / Visionary

Industry 4.0 represents the fourth industrial revolution and aims to establish smart factories, where machines, sensors, and systems are interconnected through networks and capable of exchanging data in real time. The primary objective is to make manufacturing processes more efficient, flexible, and automated. In contrast to Industry 4.0, which emphasizes machine autonomy, Industry 5.0 places the human being back at the centre of the production system, giving priority to creativity, critical thinking [64], and social responsibility. Technology is regarded not as a replacement for human capabilities, but as a tool that serves and enhances them. Simultaneously, the industrial sector is expected to become more sustainable and resilient, addressing environmental and social challenges more effectively. Industry 6.0 is a prospective vision of the future in which the entire industrial value chain is managed by intelligent autonomous systems capable of learning, making complex decisions, and adapting in real time. The aim is not only to enable fully automated production, but also to anticipate market demand, optimize the entire supply chain independently, and deliver fully personalized products with minimal human intervention.

Metaheuristic optimization techniques are increasingly being applied in reverse engineering and additive manufacturing to improve process efficiency, enhance geometric accuracy, and support the development of adaptive, autonomous production systems in line with Industry 4.0 principles described above. In this context, particular focus has been placed on their use in Advanced CAD Design for path optimization. This research, conducted in collaboration with Prof. Michele Fiorentino from the Department of Mechanics, Mathematics, and Management at the Polytechnic University of Bari and one of the instructors of the master's program "Reverse Engineering and 3D Printing" coordinated by prof. Casalino, explores innovative computational design strategies through a multidisciplinary approach.

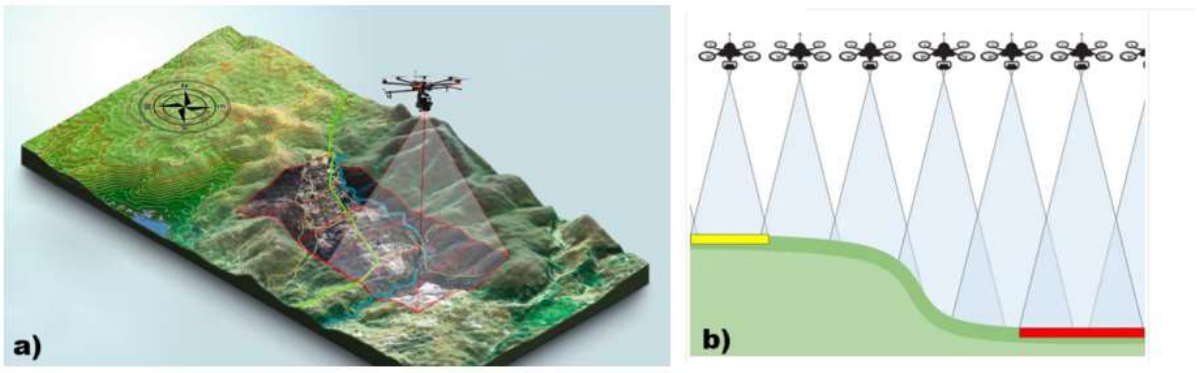


Figure 3.11- TERRAIN ANALYSIS: a) AI-ASSISTED INTEGRATION OF UAV-BASED TERRAIN ANALYSIS INTO CAD-GUIDED PATH OPTIMIZATION WORKFLOWS; b) REPRESENTATION OF LIGHT DETECTION AND RANGING (LIDAR) TECHNOLOGY

Figure 3.11.a illustrates an Unmanned Aerial Vehicle (UAV)-based topographic scanning system employed for terrain analysis and path optimization, supported by AI tools. The integration of UAV technology with advanced CAD tools enables high-resolution mapping and

supports the development of computational strategies for route planning in complex environments. Central to this approach is Light Detection and Ranging (LiDAR) technology (see Figure 3.11- b), a remote sensing method that utilizes laser pulses to accurately measure distances and capture detailed shapes of objects or surfaces. By emitting laser pulses and measuring the time interval until their reflection returns from an object, LiDAR produces highly precise three-dimensional point clouds, composed of points with XYZ coordinates representing exact spatial positions (see Figure 3.12-).

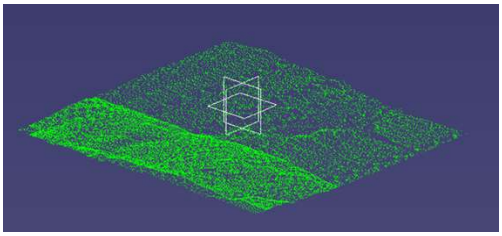


Figure 3.12- HIGH-RESOLUTION 3D POINT CLOUD WITH XYZ COORDINATES IMPORTED INTO CATIA V5 FOR GEOMETRIC ANALYSIS

Datasets for such applications can be sourced from specialized repositories such as Artec3D or iMerit (see Figure 3.13-).

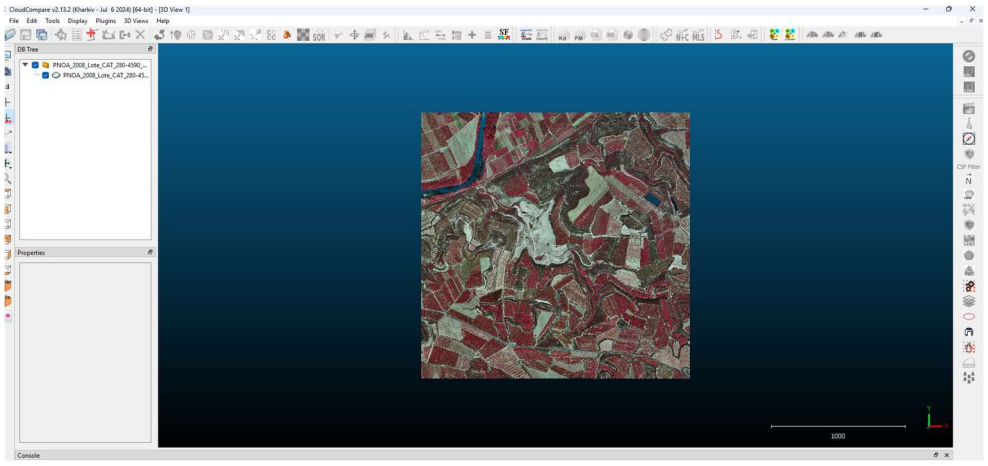


Figure 3.13- SAMPLE 3D DATASETS FROM ARTEC3D AND IMERIT, AVAILABLE FOR RESEARCH SE

The datasets utilized for the reconstruction of the scenario were originally provided in the .LAZ file format, which is a compressed form commonly used for large-scale point cloud data. These ".LAZ" files were first imported and decompressed using the open-source software Cloud Compare, which is specifically designed for handling and processing 3D point clouds. Following decompression and initial processing, the data were exported in the ".dxf" format through Rhino 8, a CAD software capable of handling complex geometries and facilitating format conversion. Subsequently, the ".dxf" files were converted to the ". igs" format, enabling seamless import and further manipulation within CATIA, a high-end CAD/CAM/CAE platform widely adopted in aerospace and industrial design. This multi-step conversion workflow was employed as the standard methodology during the practical implementation of the scenario construction phase, due to the limitations of the PC. This approach ensured the preservation of geometric fidelity while enabling integration into advanced CAD environments

for subsequent analysis and manufacturing simulation. Upon acquisition, the raw ".igs" files were processed using CATIA V5 software, where they underwent cleaning and filtering to remove noise, eliminate outliers, and segment regions of interest, thereby improving data quality(see Figure 3.14-).

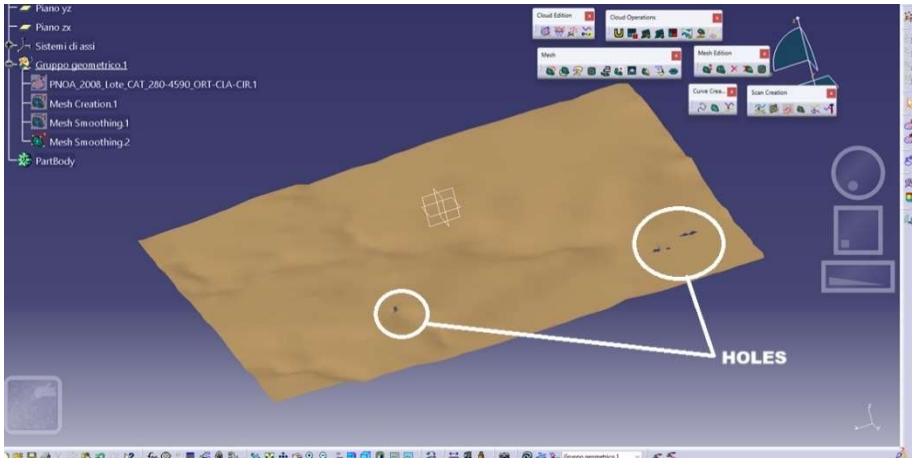


Figure 3.14- UNPROCESSED MODEL WITH NOISE AND OUTLIERS BEFORE CATIA V5 FILTERING

When multiple scans are involved (see Appendix 7), automatic alignment and registration procedures are performed to merge individual point clouds into a unified reference frame. Subsequently, these point clouds are converted into triangular meshes, as shown in Figure 3.15- , enabling continuous and comprehensive representations of the scanned environment. Finally, through reverse engineering techniques facilitated by CATIA V5, the triangular mesh is transformed into a CAD model composed of points, lines, and curves.

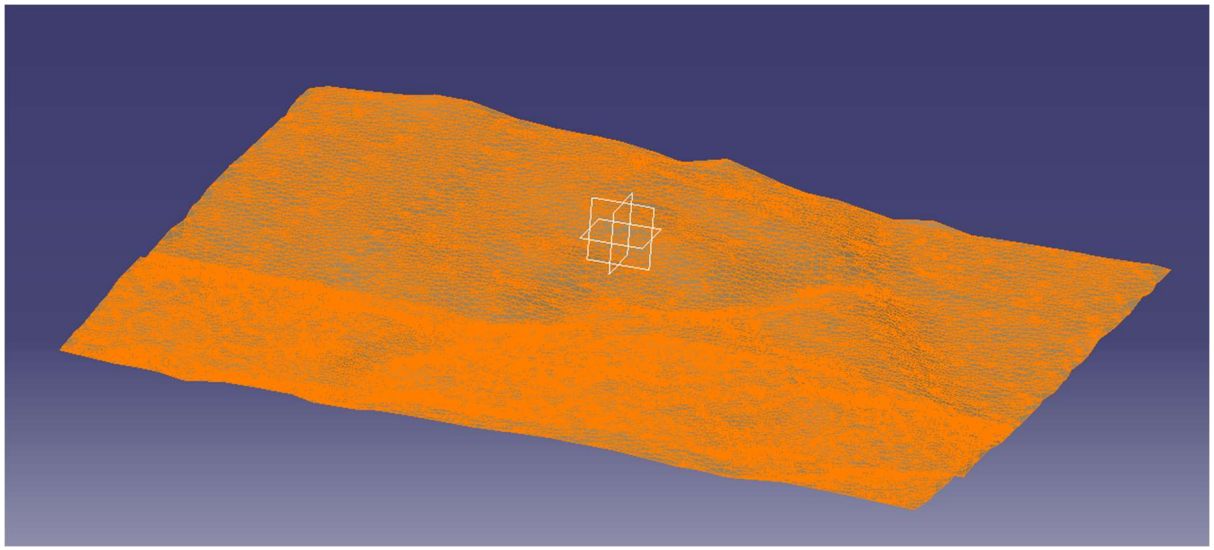


Figure 3.15- SURFACE MESH RECONSTRUCTED IN CATIA V5

This CAD model (see Figure 3.16-) serves as the foundational scenario for defining and optimizing trajectory planning, thereby supporting advanced analyses and decision-making in complex spatial contexts. Moreover, the CAD file can be utilized for creating scaled physical

models through 3D printing, or for developing immersive simulations using augmented reality (AR) or virtual reality (VR) techniques, as discussed in the preceding paragraph.

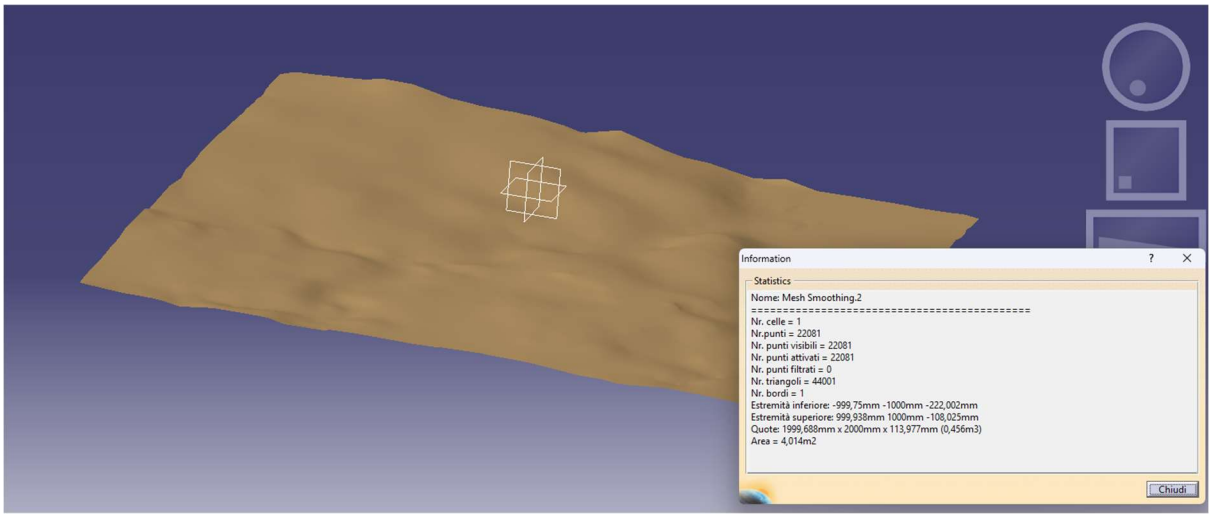


Figure 3.16- FINAL CAD MODEL OBTAINED THROUGH REVERSE ENGINEERING IN CATIA V5

After the development of the CAD model, a comprehensive analysis is undertaken to delineate critical zones and pivotal points pertinent to movement, including pathways adjacent to facades or interstitial spaces between obstacles. This analytical phase informs the formulation of preliminary trajectories; wherein initial path guidelines are established in accordance with the geometric and functional constraints intrinsic to the model.

The refinement and optimization of these trajectories are subsequently addressed through the application of metaheuristic algorithms. The primary objectives encompass minimizing the total path length, eliminating superfluous movements, and adhering to environmental constraints such as collision avoidance and allowable turning angles. Metaheuristic techniques are employed by embedding the CAD model within a simulation framework, enabling the systematic evaluation of candidate trajectories against the predefined optimization criteria. This iterative process entails multiple simulation runs; wherein diverse trajectory options are assessed based on quantitative performance indicators such as traversal time and energy expenditure. The trajectory demonstrating the most favourable balance among these metrics is selected as the optimal solution. Thereafter, the optimized trajectories undergo rigorous simulation and validation using specialized robotics simulation platforms or 3D printing simulation tools. These procedures can verify the absence of collisions and confirm compliance with precision requirements. Validation outcomes are compared against established performance benchmarks, and, if necessary, the workflow iterates through further optimization cycles to enhance trajectory efficacy.

Following the terrain reconstruction process outlined above, additional potential intersections between metaheuristic optimization techniques, reverse engineering methodologies, and additive manufacturing (3D printing) may focus on enhancing 3D printing efficiency and reducing overall production time. Key areas for optimization include the nozzle path planning, the support structure design and the point Cloud quality for the mesh definition. An AI-generated representation of these three topics is presented in Figure 3.17- , highlighting promising opportunities for future research and practical applications.

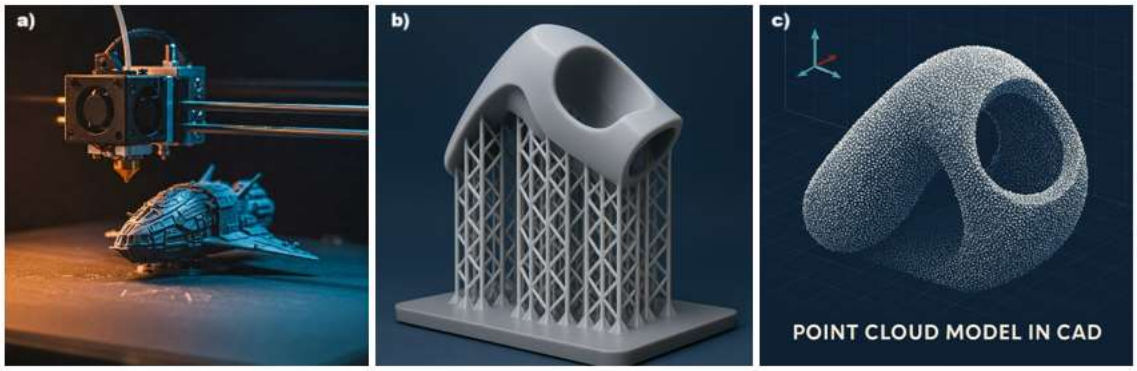


Figure 3.17- AI-GENERATED REPRESENTATION OF OPTIMIZATION STRATEGIES FOR a) NOZZLE PATH PLANNING, b) SUPPORT STRUCTURE DESIGN, AND c) POINT CLOUD PROCESSING IN 3D PRINTING

The **optimization of the nozzle trajectory** is crucial, as it directly influences print duration and quality by minimizing unnecessary movements and ensuring continuous material deposition. Effective strategies for achieving this may include:

- **Reducing Idle Movements:** Minimizing travel distance between printed sections to avoid non-productive nozzle motion.
- **Optimizing Print Paths:** Employing advanced slicing algorithms to generate smoother and more continuous trajectories, thereby improving printing efficiency.
- **Adaptive Speed and Acceleration:** Dynamically adjusting the nozzle's speed and acceleration parameters based on the geometric complexity of the model, ensuring optimal performance throughout the print.
- **Minimizing Retractions:** Reducing the number of filament retractions to prevent interruptions that can increase print time and affect surface quality.

Regarding **support structures**, these are essential for the successful fabrication of complex geometries; however, their design must carefully balance the trade-off between ensuring mechanical stability and minimizing both print time and material consumption. This balance can be achieved through:

i. **Minimization of Material Usage and Print Duration:** Utilizing advanced optimization algorithms to reduce the volume of support material without compromising structural integrity, thereby decreasing production costs and accelerating build times.
ii. **Targeted Support Placement:** Applying structural and geometric analyses to strategically position supports exclusively in critical load-bearing or overhanging areas, thus avoiding redundant scaffolding and enhancing overall process efficiency.
iii. **Reduction of Contact Interfaces:** Engineering supports to minimize both the number and surface area of contact points with the printed part, which reduces adhesion forces, facilitates easier removal, and preserves surface quality during post-processing.

Building upon the previous discussion of support structure optimization, another critical aspect for efficient 3D printing is the reduction of point clouds to generate meshes that balance detail and computational efficiency. When working with 3D scans or point cloud-derived models, it is essential to create detailed meshes without overwhelming the system with excessive data. This can be achieved through:

iv. **Intelligent Point Filtering:** Removing redundant or noisy points while preserving crucial geometric features.
v. **Optimized Resampling:** Generating smoother meshes with a minimal number of triangles to improve processing speed and reduce memory usage.
vi. **Balancing Visual Quality and Computational Efficiency:** Achieving an optimal compromise between high-fidelity mesh representation and manageable computational demands.

What follows are representative **mathematical models** demonstrating how these challenges can be tackled through quantitative approaches, revealing practical optimization opportunities for improving industrial-scale additive manufacturing processes for future research projects.

1) **Nozzle path planning.** The optimization of the nozzle's movement trajectory [65] is essential for reducing non-productive travel, minimizing idle time, and ensuring consistent material deposition. Enhancing path efficiency directly contributes to faster print cycles and improved dimensional accuracy.

The mathematical model is drafted considering the problem of planning the movement of a 3D printer's nozzle across a set of deposition point $P = \{p_1, \dots, p_n\}$, starting from an initial point p_0. The objective is to determine an optimal visitation sequence that minimizes the total printing time, accounting for both productive (material deposition) and travel movements. Let be $p_i = (x_i, y_i, z_i)$ coordinates of deposition point i, $d_{i,j}$ the Euclidean distance between points p_i and p_j, v_d the deposition speed, v_m the travel speed, t_s the time penalty for turning on/off the material flow, $s_{i,j}$ the binary indicator, 1 if the transition $i \to j$ requires a state of change(start/stop extrusion), 0 otherwise, $x_{i,j} \in \{0,1\}$ the binary variable equal to 1 if the nozzle moves directly from point i to point j, $u_i \in R$ the auxiliary variable used to eliminate sub-tours with the Miller-Tucker- Zemlin formulation (MTZ) [66]. The objective function is to minimize the total process time:

$$min \sum_{i=1}^{n} \sum_{\substack{j=0 \\ j \neq i}}^{n} \left(\frac{d_{ij}}{v_{ij}} + t_s \cdot s_{ij}\right) x_{ij} \tag{3.24}$$

where $v_{ij} = v_d$ if the material is deposited along the path or $v_{ij} = v_m$ if the nozzle travels without depositing material.

The possible constraints could be defined as follows:

- Each point is visited exactly once:

$$min \sum_{\substack{j=0 \\ j \neq i}}^{n} x_{ij} = 1, \forall\, i = 1, \dots, n \tag{3.25}$$

- The path starts from the initial position:

$$\sum_{j=1}^{n} x_{oj} = 1 \tag{3.26}$$

- Each point has one incoming way

$$\sum_{\substack{i=0 \\ i \neq j}}^{n} x_{ij} = 1, \forall\, j = 1, \dots, n \tag{3.27}$$

- Subtour elimination following the MTZ formulation

$$u_i - u_j + n x_{ij} < n - 1, \forall\, i \neq j, i, j = 1, \dots, n \tag{3.28}$$

with $1 \leq u_i \leq n, \forall\, i = 1, \dots, n$

2) **Support Structure Design.** Strategically designed support structures are vital for maintaining stability during the printing process, while also reducing material consumption and post-processing requirements. Optimized supports [67] can significantly decrease build time and facilitate easier removal, leading to streamlined production.

A possible development of the problem could be imposed as follows:

Let $V \subset R^3$ the volume of the object to be printed, $S \subset V$ the set of overhanging regions requiring support, $\Omega \subset R^3$ the candidate support region (build volume beneath overhangs), $G = (\Omega, E)$ the discretized support grid (e.g., voxel-based or tetrahedral mesh), θ_c the critical overhang angle below which a support is needed, ρ_i the material density assigned to element $i \in \Omega$, C_i the cost associated with element i, $u_i \in \{0,1\}$ the binary variable indicating whether support element i is active (1) or not (0).

The objective function will be minimizing the total cost of support structures:

$$\min_{u} \sum_{i \in \Omega} C_i \cdot u_i \tag{3.29}$$

$$C_i = \alpha_1 \cdot C_i^{(1)} + \alpha_2 \cdot C_i^{(2)} + \alpha_3 \cdot C_i^{(3)} \tag{3.30}$$

where C_i incorporate the material volume ($C_i^{(1)} \propto V_i$), the build time contribution ($C_i^{(2)} \propto height_i/v$), and the post processing complexity ($C_i^{(3)} \propto accessibility_i$).

The constraints will be defined on the support coverage, the structural stability, the printability and eventually the topology as follows:

- Each overhanging element $s \in S$ must be supported by at least one valid support path below it:

$$\sum_{i \in P_{(s)}} u_i \geq 1, \forall s \in S \tag{3.31}$$

Where $P_{(s)}$ is the set of support elements directly or indirectly under s (considering the critical angle θ_c).

- Support element must form a stable structure reaching the build plate. This can be modelled as connectivity or a force propagation constraint:

$$u_i \leq \sum_{j \in Pred_{(i)}} u_j, \forall i \notin base \tag{3.32}$$

where $Pred_{(i)}$ are the elements directly below i, and $base \subset \Omega$ are the ground-level voxel.

- Ensure that the support structure does not violate printer constraints (e.g. no floating support):

$$u_i = 0 \text{ if } i \text{ not reachable by downward path from } S \tag{3.33}$$

- To allow a continuum formulation (e.g., density-based optimization)

$$0 \leq \rho_i \leq 1, \forall i \in \Omega \tag{3.34}$$

and penalize intermediate densities using the Solid Isotropic Material with Penalization (SIMP) model [68]:

$$C_i = C_0 \cdot \rho_i^p \leq 1, p > 1 \tag{3.35}$$

3) **Point Cloud Quality for Mesh Definition**. The accuracy and resolution of the mesh are fundamentally dependent on the quality of the point cloud from which it is derived. Ensuring optimal point cloud density and applying effective noise filtering techniques are crucial for generating precise geometries, thereby improving slicing efficiency and overall print reliability.
Given a point cloud $P = \{p_1, \dots, p_N\} \subset R^3$, our goal could be to optimize point cloud parameters (density, noise level, and distribution) [69] [70] to generate a high-fidelity mesh $\mathcal{M}$, which results in efficient slicing and reliable 3D printing. Let $\rho(p_i) \in R^+$ be the local point density around point p_i, $\sigma(p_i) \in R^+$ be the local noise level at point p_i, $\theta \in R$ be the global filtering threshold (for outlier removal), $\mathcal{Q}(\mathcal{P})$ be the quality metric of point cloud (function of density and noise), $\varepsilon \in R^+$ be the allowed surface deviation between real geometry and mesh. We aim to maximize mesh fidelity while minimizing noise and ensuring an efficient point distribution

$$\max_{\rho,\sigma,\theta} (\alpha_1 \cdot \mathcal{A}(\mathcal{M}) - \alpha_2 \cdot \mathcal{N}(\mathcal{P}) - \alpha_3 \cdot \mathcal{D}(\mathcal{P})) \tag{3.36}$$

where $\mathcal{A}(\mathcal{M})$ represents the mesh accuracy, defined by Hausdorff distance [71] between mesh and real surface (lower is better), $\mathcal{N}(\mathcal{P}) = \sum_{i=1}^{N} \sigma(p_i)^2$ is the total noise energy in the cloud, $\mathcal{D}(\mathcal{P}) = \sum_{i=1}^{N} (\rho(p_i) - \bar{\rho})^2$ denotes the uniformity deviation of point density, $\alpha_1, \alpha_2, \alpha_3 \in R^+$ are the weighting factors.

The constraints could be imposed as follows:

- Minimum density constraint for surface detail preservation:

$$\rho(p_i) \geq \rho_{min}, \forall i \tag{3.37}$$

- Maximum allowable noise after filtering:

$$\sigma(p_i) \leq \sigma_{max}, \forall i \tag{3.38}$$

- Surface deviation constraint after meshing:

$$d_H(\mathcal{M}, \mathcal{S}_{ref}) \leq \varepsilon \tag{3.39}$$

where d_H is the Hausdorff distance and $\mathcal{S}_{ref}$ is the reference geometry.

- Computational cost budget (slicing efficiency/memory):

$$\mathcal{C}(\mathcal{M}) \leq C_{max} \tag{3.40}$$

These mathematical formulations offer not only a solid foundation for tackling key challenges in additive manufacturing, but also a compelling invitation to move from theory to practice. By integrating advanced optimization methods with real-world constraints, they pave the way for smarter, faster, and more resource-efficient 3D printing, particularly in high-stakes industries like aerospace. The potential impact spans from reduced production costs and improved part reliability to new frontiers in design freedom. As such, **this work represents a unique opportunity to initiate a focused research and development group aimed at translating these models into operational tools, bridging academic innovation and industrial application.** The time is ripe to bring together expertise in computational modelling, manufacturing engineering, and AI to turn these insights into tangible advances.

3.5 TOWARD INTEGRATED RESEARCH PATHWAYS: SYNERGIES IN ADVANCED DESIGN AND MANUFACTURING

The diverse investigations presented in this chapter have explored, through the lens of metaheuristic optimization, some of the most dynamic and complex areas of contemporary technological innovation. From the design of human-robot interaction strategies applied to socially assistive robotics, to the immersive challenges posed by augmented and virtual reality

environments, and the technical intricacies of reverse engineering and additive manufacturing workflows, each case study has contributed to a multifaceted understanding of how optimization algorithms can be used not only to enhance performance, but also to redefine the paradigms of intelligent design through the integration of AI-driven tools. What emerges is not a collection of isolated efforts, but the outline of a shared language, a methodological thread that binds together distinct disciplines under the sign of computational intelligence.

In the study of human-robot interaction design for Alpha Mini, metaheuristic approaches were used to fine-tune behavioural parameters, interaction timing, and user engagement strategies. These efforts revealed the potential for intelligent algorithms to bridge the gap between technical feasibility and emotional resonance in human-centred systems. In the field of augmented and virtual reality, optimization strategies allowed for the adjustment of spatial layouts, interaction responsiveness, and system adaptability in real time, factors critical to the usability and accessibility of immersive environments. Meanwhile, in reverse engineering and additive manufacturing, metaheuristics facilitated the refinement of digital workflows, from point cloud segmentation and noise reduction to the adaptive generation of tool paths and energy-efficient build strategies. In each of these domains, the application of heuristic and metaheuristic intelligence has demonstrated tangible benefits, offering a pathway to scalable, adaptive, and efficient solutions that transcend the limits of traditional design processes.

These converging lines of inquiry do not merely suggest a methodological affinity, but they point to the need for a structural evolution in how we conceive interdisciplinary research. The recurring presence of metaheuristic tools across such varied domains invites us to question the fragmentation that still characterizes many research environments. Why should algorithmic innovation developed for robot behaviour modelling not inform optimizations in additive manufacturing? Why shouldn't immersive experience design benefit from procedural learnings obtained in industrial process tuning? The time has come to move beyond isolated experimentation and toward the creation of a shared research infrastructure, capable of integrating competences, pooling resources, and cultivating a collective intelligence that is greater than the sum of its parts. This is not merely a question of efficiency; it is a question of relevance, of ensuring that research evolves in step with the complexity and interconnectedness of the technological ecosystems it seeks to serve.

In this context, we propose the launch of a joint research initiative: Metaheuristic Optimization for Human-Centred Design and Intelligent Manufacturing (MO-HuCIM). Conceived as a multi-institutional, multi-sectoral platform, MO-HuCIM would be a way to unite academic researchers, industrial partners, and technology developers around a common objective: to explore and exploit the full potential of metaheuristic optimization as a transversal enabler for innovation. The initiative would support coordinated research lines, experimental prototyping, methodological benchmarking, and the development of shared tools and datasets. Moreover, it would be strategically positioned to access national, European or International funding programs, thanks to its alignment with key thematic priorities such as AI for Industry 5.0, sustainable design, and next-generation manufacturing. More than a research project, MO-HuCIM could be a call to intellectual alliance, a structured invitation to break down silos, overcome inertia, and design, together, the intelligent systems of tomorrow.

The work developed until now is the demonstration that this convergence is not only desirable, but it is already underway. What remains is to transform it from an implicit alignment into an explicit framework for collaboration. The road forward calls for courage, coordination, and

above all, the will to act. It is an invitation to all researchers, technologists, and institutions who recognize the urgency of systemic innovation to step out of disciplinary confines and become co-authors of a shared vision. Not merely to continue what has been done, but to amplify it, to give it a shape, a direction, and a collective momentum worthy of the technological challenges we face. In this sense, MO-HuCIM is not the conclusion of this chapter, but its natural continuation, a project yet to be written, with many hands, across many domains, for a future that demands nothing less than integrated, intelligent, and human-centred design.

CHAPTER IV: SHGWJA: A SIMPLE HYBRID GREY WOLF-JAYA METAHEURISTIC ALGORITHM FOR OPTIMIZATION IN MECHANICAL AND CIVIL ENGINEERING

Through intertwined perspectives, man and machine journey across complexity: one lens draws them toward the center of each challenge, while the other seeks the view that best unveils its hidden rhythms. In their delicate balance of focus and exploration, hybrid minds discover solutions that resonate between the particular and the whole.

4.1 DETAILED EXAMINATION OF METAHEURISTIC ALGORITHMS

Metaheuristic algorithms (MAs) have become the standard approach for solving complex engineering optimization problems. Their widespread adoption is evidenced by numerous applications across science, engineering, and technology. Notable areas include structural optimization (static and dynamic) [72], [73], material characterization and structural identification (including damage detection) [74], [75], [76], vehicle routing [77], solar device optimization [78], forest fire mapping [79], urban water demand prediction [80], 3D printing process optimization [81], medical diagnostics (e.g., COVID-19 and cancer) [82], [83], and image processing [84].

Gradient-based optimizers depend on derivative information to guide the search process. In contrast, metaheuristic algorithms (MAs) are inherently stochastic and operate without requiring gradient data. They generate new trial solutions through randomized procedures inspired by the specific principles underlying each algorithm. At every iteration, the objective is to improve upon the best solution found so far. The optimization process generally unfolds in two main phases: exploration, characterized by broad perturbations aimed at identifying promising regions within the design space, and exploitation, which focuses on locally refining candidate solutions (see Figure 4.1-). Typically, exploration governs the initial stages of the search, while exploitation becomes increasingly significant as the algorithm approaches convergence.

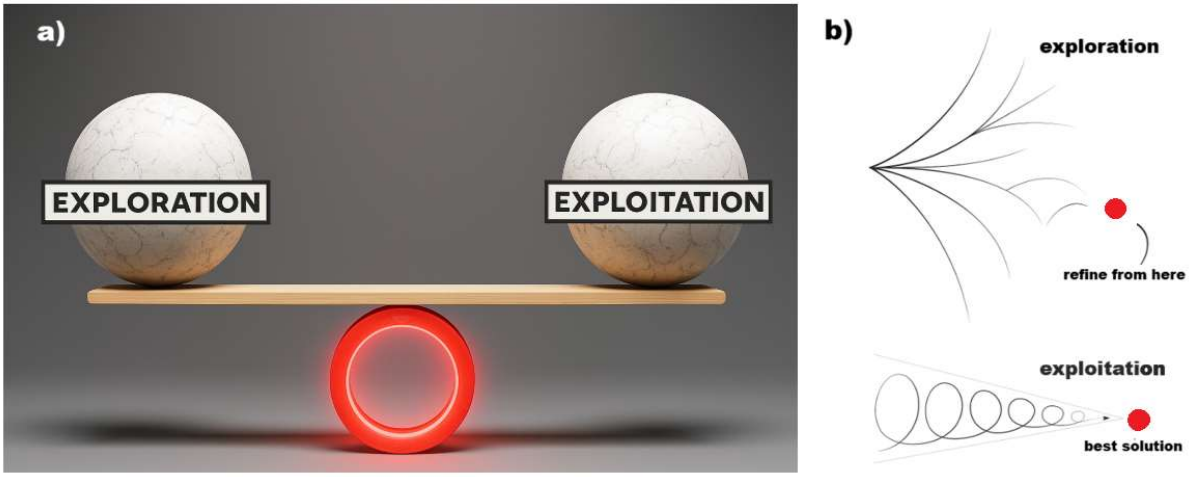


Figure 4.1- PHASES OF METAHEURISTIC OPTIMIZATION: (a) TRADE-OFF BETWEEN EXPLORATION AND EXPLOITATION; (b) TRANSITION FROM GLOBAL SEARCH TO LOCAL REFINEMENT

MAs can be broadly categorized based on the number of search agents: single-solution algorithms (e.g., simulated annealing, tabu search) and population-based algorithms, which operate on multiple candidate solutions. Most modern MAs fall into the latter category and are further classified by their inspiration source:

- **Evolutionary algorithms**
- **Science-based algorithms**
- **Human-based algorithms**
- **Swarm intelligence-based algorithms.**

Evolutionary algorithms (see Table 4.1) are inspired by natural evolution and include genetic algorithms (GAs) [85], [86], differential evolution (DE) [87], [88], evolutionary programming (EP) [89], evolution strategies (ES) [90], biogeography-based optimization (BBO) [91], and black widow optimization (BWO) [92]. Among them, GAs and DE are the most widely used, with GAs relying on selection, crossover, and mutation to evolve solutions, while DE involves population initialization, mutation, recombination, and selection. The main difference lies in how new generations are selected. The impact of these algorithms is reflected in the citation metrics, with Refs. [85], [86] collectively receiving around 7,000 citations, and Ref. [87] alone exceeding 23,000 citations over a span of 35 years (Scopus database).

Table 4.1- MAIN EVOLUTIONARY METAHEURISTIC ALGORITHMS AND THEIR MOST CITED VARIANTS ACCORDING TO SCOPUS

ALGORITHM	INSPIRATION	KEY MECHANISM	CITATION RATE
Genetic Algorithms (GAs)	Darwinian natural selection	Selection, crossover, mutation	~860/year
Differential Evolution (DE)	Evolutionary processes	Random initialization, mutation, recombination, selection	~800/year
Evolutionary Programming (EP)	Evolution	Mutation of parameters	Not specified
Evolution Strategies (ES)	Evolution	Self-adaptive mutation strategies	Not specified
Biogeography-Based Optimization	Biogeography	Habitat suitability and migration	Not specified
Black Widow Optimization (BWO)	Widow spider behavior	Mating and cannibalism	Not specified

Science-based algorithms (see Table) simulate physical, chemical, or mathematical phenomena. Notable examples include simulated annealing (SA) [93], [94], charged system search (CSS) [95], magnetic charged system search (MCSS) [96], ray optimization (RO) [97], colliding bodies optimization (CBO) [98], water evaporation optimization (WAO) [99], thermal exchange optimization (TEO) [100], equilibrium optimizer (EO) [101] and light spectrum optimizer (LSO) [102]. These methods often mimic equilibrium processes, such as energy minimization in thermal systems or light dispersion. SA remains the most cited physics-based algorithm, inspired by the annealing process in materials. It allows worse solutions to be accepted with a probability controlled by a "temperature" parameter, helping escape local minima. Moreover, chemistry-inspired algorithms include artificial chemical reaction optimization (ACROA) [103], gas Brownian motion optimization (GBMO) [104], and Henry gas solubility optimization (HGSO) [105]. These algorithms replicate molecular behaviours, such as chemical reactions or gas diffusion. For instance, HGSO mimics gas solubility equilibrium and has gained significant popularity, accumulating approximately 800 citations within five years of publication. Additional science-based algorithms have been inspired by astronomical and mathematical principles, which have played a significant role in the development of metaheuristic algorithms (MAs). Examples include the Big Bang-Big Crunch Optimization (BB-BC) [106], Gravitational Search Algorithm (GSA) [107], Galaxy-Based Search Algorithm (GBSA) [108], Black Hole Algorithm (BHA) [109], Astrophysics-Inspired Grey Wolf Algorithm (AIGWA) [110], and Supernova Optimizer [111]. These methods simulate phenomena such as cosmic expansion and contraction, gravitational interactions, stellar evolution, and orbital mechanics. Among them, GSA and BB-BC are particularly notable, with approximately 6500 and 1300 citations respectively in the Scopus database. Other science-based algorithms include mathematics-inspired MAs such as the Sine Cosine Algorithm (SCA) [112], Runge-Kutta Optimizer (RUN) [113], and Arithmetic Optimization

Algorithm (AOA) [114]. These algorithms employ mathematical models to guide the search. For instance, SCA utilizes trigonometric functions to update candidate solutions, RUN integrates slope estimation via the Runge-Kutta method, and AOA uses arithmetic operators, multiplication/division for exploration and addition/subtraction for exploitation. These approaches have gained rapid attention: AOA and RUN reached ~1900 and ~750 citations respectively within three years, while SCA garnered ~4040 citations over eight years.

Table 4.2- MAIN SCIENCE BASED METAHEURISTIC ALGORITHMS AND THEIR MOST CITED VARIANTS ACCORDING TO SCOPUS

	ALGORITHM	INSPIRATION	KEY MECHANISM	CITATION RATE
PHYSICS	Simulated Annealing (SA)	Thermodynamics	Probabilistic hill-climbing, cooling schedule	~810/year
	Charged System Search (CSS)	Electromagnetism	Coulomb 's and Newton's laws	-
	Magnetic Charged System Search	Magnetism	Magnetic field and force interactions	-
	Ray Optimization (RO)	Optics	Light ray refraction and dispersion	-
	Colliding Bodies Optimization	Mechanics	Elastic collisions	-
	Water Evaporation Optimization	Thermodynamics	Water evaporation rate	-
	Thermal Exchange Optimization	Heat transfer	Heat conduction principles	-
	Equilibrium Optimizer (EO)	Physics	Physical equilibrium states	-
	Light Spectrum Optimizer (LSO)	Light dispersion	Light spectrum and dispersion	-
CHEMISTRY	Artificial Chemical Reaction Optimization (ACROA)	Chemistry	Chemical reaction dynamics	-
	Gas Brownian Motion Optimization (GBMO)	Physics	Molecular motion and turbulence	-
	Henry Gas Solubility Optimization (HGSO)	Chemistry/Physics	Gas-liquid equilibrium based on Henry's Law	780 in 5 years
ASTRONOMY OR ASTROPHYSICS	Big Bang–Big Crunch (BB-BC)	Astrophysics	Universe expansion and contraction cycles	~1280 total
	Gravitational Search Algorithm (GSA)	Gravity	Gravitational attraction among masses	~6030 total
	Galaxy-Based Search Algorithm	Galaxies and cosmology	Spiral galaxy modelling and interactions	-
	Black Hole Algorithm	Black holes	Gravitational absorption	-
	Supernova Optimizer	Supernova explosion	Stellar material dispersion	-
MATHEMATICS	Sine Cosine Algorithm (SCA)	Mathematics	Sinusoidal movement towards/away from best solutions	~505/year
	Runge–Kutta Optimizer (RUN)	Mathematics	Runge-Kutta numerical slope method	~845/year
	Arithmetic Optimization Algorithm (AOA)	Mathematics	Arithmetic operations (addition, subtraction, etc.)	~845/year

Human-based MAs (see Table 4.3) draw inspiration from human behaviour, activities, learning processes, and sociopolitical systems. Early representatives include Tabu Search (TS) [115], [116], which prohibits the revisiting of recently explored (non-improving) solutions, and Harmony Search (HS) [117], [118], which models musical improvisation. Teaching-Learning-Based Optimization (TLBO) [6] and JAYA [13] are among the most cited in this class, with TLBO accumulating ~3920 citations (~300/year) and JAYA ~2000 citations (~250/year).

Table 4.3- MAIN HUMAN BASED METAHEURISTIC ALGORITHMS AND THEIR MOST CITED VARIANTS ACCORDING TO SCOPUS

	ALGORITHM	INSPIRATION	KEY MECHANISM	CITATION RATE
HUMAN BEHAVIOUR	Tabu Search (TS)	Human decision/memory	Tabu list, neighbourhood-based search	-
	JAYA	Striving for success, avoiding failure	Single update equation	~250/year
ACTIVITIES	Harmony Search (HS)	Musical improvisation	Memory, pitch adjustment, random selection	~235/year
LEARNING PROCESSES	Teaching–Learning-Based Optimization (TLBO)	Classroom learning	Teacher and learner phases	~300/year
	Group Teaching Optimization (GTOA)	Group education	Group-teacher improvement strategies	-
	Mother Optimization Algorithm	Maternal behavior	Mother-child interaction in education	-
	Preschool Education Optimization	Preschool learning	Teacher's influence, child's learning growth	-
	Learning Cooking Algorithm (LCA)	Cooking learning	Learning from mother and chef	-
SOCIOPOLITICAL SYSTEMS	Collective Decision Optimization	Human decision-making	Multi-phase decision model	-
	Imperialist Competitive Algorithm (ICA)	Political empires	Empire building, competition	-
	Political Optimizer (PO)	Modern politics	Election campaign, party leadership	-

These methods simulate learning dynamics in classrooms or the natural human drive to improve outcomes. Recent algorithms further explore the learning paradigm: the Group Teaching Optimization Algorithm (GTOA) [119], Mother Optimization Algorithm (MOA) [120], Preschool Education Optimization Algorithm (PEOA) [120], and Learning Cooking Algorithm (LCA) [121] all mimic specific educational or domestic learning processes. Decision-making behaviour is replicated by the Collective Decision Optimization Algorithm (CDOA) [122], which simulates phases of human reasoning including individual experience, group consensus, and innovation. Political systems have inspired methods such as the Imperialist Competitive Algorithm (ICA) [123] and Political Optimizer (PO) [124]. ICA models the dynamics between imperialist states and colonies, where empires compete for global dominance. PO simulates the political process, including constituency behaviour and learning from electoral outcomes.

Swarm intelligence-based algorithms are perhaps the most prolific category, mimicking collective behaviour in animal groups during activities such as foraging, migration, and hunting. Particle Swarm Optimization (PSO) [125], [125], [126] is the most cited MA overall, with ~73,550 citations (2535/year), inspired by the coordinated movement of bird and fish flocks. Individuals (particles) adjust their positions based on their own and the swarm's best-known positions.

Table 4.4- MAIN SWARM INTELLIGENCE-BASED ALGORITHMS AND THEIR MOST CITED VARIANTS ACCORDING TO SCOPUS

	ALGORITHM	INSPIRATION	KEY MECHANISM	CITATION RATE
FORAGING	Ant Colony Optimization (ACO)	Ant foraging	Pheromone-based search and reinforcement	~405/year
	Artificial Bee Colony (ABC)	Bee behaviour	Roles: employed, onlooker, scout bees	~370/year
	Firefly Algorithm (FFA)	Fireflies	Light intensity and attraction	~207/year
	Cuckoo Search (CS)	Cuckoo parasitism	Lévy flights, egg-laying strategies	~570/year
	Crow Search Algorithm (CSA)	Crows	Food hiding and memory recall	~220/year
	Dolphin Echolocation Algorithm	Dolphins	Echolocation-based tracking	-
MIGRATION	Particle Swarm Optimization (PSO)	Bird/Fish swarms	Position and velocity updates based on leaders	~2535/year
	Salp Swarm Algorithm (SSA)	Salps	Chain-based swarm movement	~560/year
	Starling Murmuration Optimizer	Starlings	Separation, diving, and whirling strategies	-
	Snake Optimizer (SO)	Snake mating	Mating behaviour-based perturbation	-
HUNTING	Ant Lion Optimizer (ALO)	Antlions	Trapping and selection-based search	~300/year
	Grey Wolf Optimizer (GWO)	Grey wolf hunting	Alpha-beta-delta leadership model	~1370/year
	Coyote Optimization Algorithm	Coyote behaviour	Social learning and interactions	-
	Snow Leopard Optimization Algorithm	Snow leopards	Movement, hunting, reproduction, mortality modelling	-
	Bat Algorithm (BA)	Bats	Echolocation and frequency tuning	~430/year
	Whale Optimization Algorithm (WOA)	Humpback whales	Bubble-net feeding simulation	~1225/year
	Marine Predators Algorithm (MPA)	Ocean predators	Brownian/Lévy motion for exploration	~400/year
	Giant Trevally Optimizer (GTO)	Trevally fish	Target hunting with Lévy flight	-

Insect-inspired methods include Ant System and Ant Colony Optimization (AS, ACO) [127], [128], [129], Artificial Bee Colony (ABC) [130], Firefly Algorithm (FFA) [131], [132], and Ant Lion Optimizer (ALO) [133]. Their relevance is reflected in high citation counts: ~11,320 for ACO (405/year), ~6335 for ABC (370/year), ~2895 for FFA (207/year), and ~2720 for ALO (300/year). Terrestrial animal-inspired MAs such as the Grey Wolf Optimizer (GWO) [12], Coyote Optimization Algorithm (COA) [134], Snake Optimizer (SO) [135], and Snow Leopard Optimization Algorithm (SLOA) [136] simulate predation, social dynamics, and territorial behaviours. GWO is especially influential, with ~13,700 citations since 2014 (1370/year), making it the second most cited MA after PSO. Bird-, bat-, and aquatic animal-inspired metaheuristics have also significantly contributed to the development of nature-inspired optimization methods. Among birds and bats, notable algorithms include Cuckoo Search (CS) [137], [138], Crow Search Algorithm (CSA) [139], Starling Murmuration Optimizer (SMO) [140], and Bat Algorithm (BA) [141], [142] CS simulates brood parasitism, where cuckoos lay eggs in the nests of other birds, while BA mimics the echolocation-based navigation of bats. CSA models how crows hide and retrieve food, and SMO reproduces coordinated flight dynamics in starling swarms. CS is the most widely cited among these (~8570 citations, ~570/year), followed by BA (~6030 citations, ~430/year) and CSA (~1765

citations, ~220/year, up to October 2024). In the aquatic domain, several algorithms have emerged, including the Dolphin Echolocation Algorithm, Whale Optimization Algorithm (WOA) [143], Salp Swarm Algorithm (SSA) [144], Marine Predators Algorithm (MPA) [145], and Giant Trevally Optimizer (GTO) [146]. These algorithms simulate behaviours such as prey detection via echolocation, bubble-net feeding, swarming, and Lévy-based predator search strategies. The Whale Optimization Algorithm is the most popular, with ~1300 citations/year, followed by the Salp Swarm Algorithm (~560/year) and the Marine Predators Algorithm (~400/year).

The Figure 4.2- illustrates the scientific impact of key metaheuristic algorithms based on citation data from Scopus (from their publication up to 2024). Among evolutionary algorithms, Differential Evolution (DE) leads with over 23,000 citations, followed by Genetic Algorithms (GA) with approximately 7,000 citations. In the swarm intelligence category, Particle Swarm Optimization (PSO) dominates with more than 73,000 citations, while Ant Colony Optimization (ACO) and Artificial Bee Colony (ABC) have accrued over 11,000 and 6,000 citations, respectively. Notable nature-inspired algorithms include the Grey Wolf Optimizer (GWO) with around 13,700 citations and the Whale Optimization Algorithm (WOA) with approximately 13,000 citations. Despite its recent introduction, the Henry Gas Solubility Optimization (HGSO) algorithm has already gathered close to 800 citations, reflecting growing interest in chemistry-inspired metaheuristics.

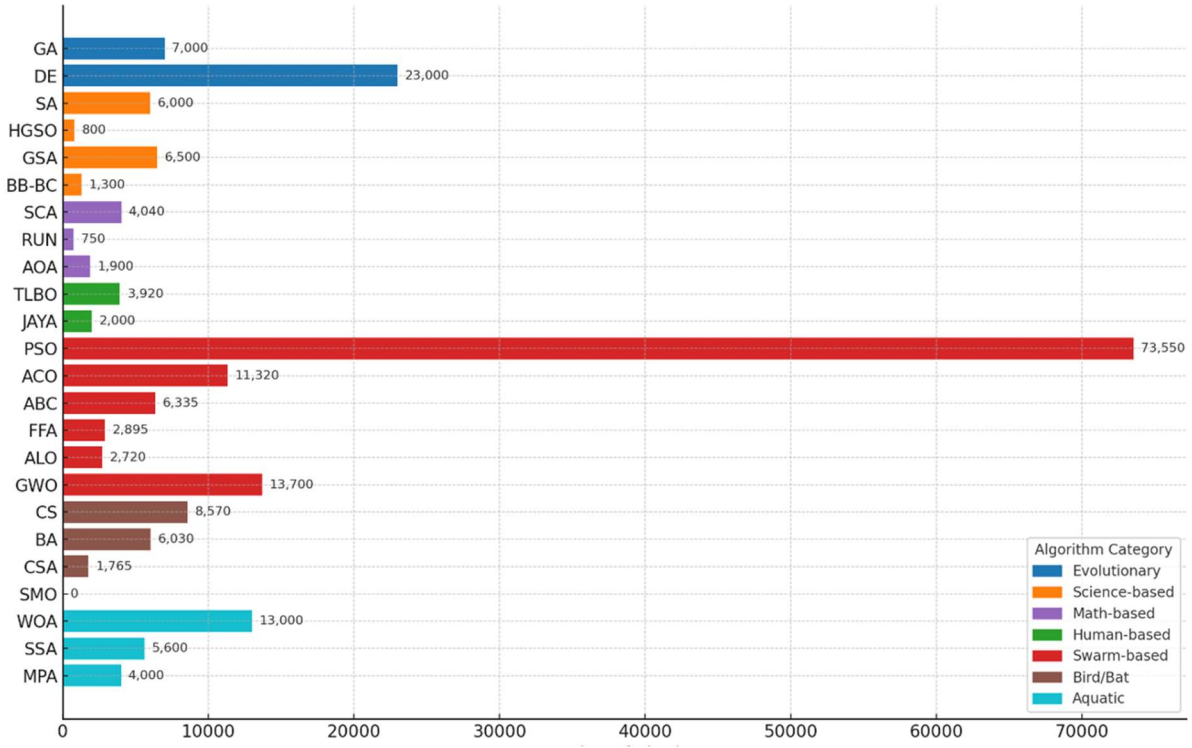


Figure 4.2- METAHEURISTIC ALGORITHMS: CITATIONS FROM THE DATE OF PUBLICATION UP TO OCTOBER 2024 (SCOPUS)

Over the past two decades, numerous improved metaheuristic variants and hybrid algorithms, combining MAs with gradient-based methods or with other Mas, have been developed (see, e.g., Refs. [147], [148], [149], [150], [151], [152], [153], [154], [155], [156], [157], [158], [159], [160], [161], [162], [163], [164], [165], [166], [167]). Common strategies include combining explorative and exploitative methods, reintroducing diversity to avoid stagnation or

premature convergence, and incorporating perturbation techniques such as chaotic maps, mutation, or Lévy flights.

Despite the continuous growth in computational power, several challenges remain in metaheuristic optimization:

- No Free Lunch Theorem [168], [169] implies that no single MA outperforms others on all problems.
- MAs typically require more function evaluations than gradient-based counterparts.
- The simplicity of basic MAs is often lost in complex variants or hybrids.
- Many MAs include multiple internal parameters, complicating tuning and implementation.

Moreover, newly developed MAs often lose relevance quickly, while classical algorithms, such as GA, DE, SA, PSO, ACO, CS, BA, HS, GSA, BB-BC, and GWO, remain dominant. Notably, only a few post-2014 algorithms (e.g., WOA, SCA, AOA/RUN, SSA, MPA, ALO, JAYA, CSA) have sustained high citation rates.

The first investigation conducted in the field of metaheuristics focused on the development of a hybrid algorithm, named **SHGWJA (Simple Hybrid Grey Wolf–JAYA Algorithm)** [170]. This algorithm was designed by combining two well-established population-based metaheuristics: **Grey Wolf Optimizer (GWO)** and **JAYA**. The objective was to address complex optimization problems in mechanical and civil engineering by reducing computational costs while maintaining robustness. The choice to merge GWO and JAYA was driven by the following considerations:

- The Grey Wolf Optimizer (GWO) [12] is the second most cited MA after PSO, with ∼1370 citations per year. It does not require the tuning of internal parameters, unlike PSO and many other MAs. GWO simulates the leadership hierarchy and hunting strategy of grey wolves, where the three best solutions guide the search process. It has been applied widely in engineering and scientific domains [171], [172].
- The JAYA algorithm [13] is one of the most cited metaheuristics of the past decade. It uses a very simple update rule that moves solutions toward the best and away from the worst using just one equation. This simplicity reduces computational effort and makes it well-suited for hybrid algorithms. Its mixed evolutionary-swarm nature also improves flexibility and adaptability [173], [174].
- Both GWO and JAYA do not require the user to set internal control parameters, aside from standard settings common to all population-based metaheuristics, such as population size and maximum number of iterations.
- Both algorithms exhibit strong exploration capabilities, while their relatively weaker exploitation performance can be effectively enhanced through elitist strategies, without increasing the algorithm's overall complexity.

In this chapter, the novel hybrid metaheuristic is evaluated on a set of benchmark problems in the fields of civil, mechanical engineering, and robotics. The JAYA update is selectively applied to the top wolves in the GWO hierarchy to prevent stagnation when the best individuals remain unchanged across iterations. Slight modifications to both algorithms aim to improve convergence and reduce function evaluations. In contrast to many GWO and JAYA variants that introduce additional mechanisms to boost diversity or exploitation, often increasing

algorithmic complexity [171], [172], [173], [173], SHGWJA [170] adopts a streamlined, elitist strategy. It focuses on improving the best solution by discarding unpromising trials before constraint evaluation, thereby reducing computational cost while preserving robustness and convergence reliability.

The effectiveness of SHGWJA were demonstrated through the following seven real-world engineering design problems, featuring up to 14 design variables and 721 nonlinear constraints:

1. Volume minimization of a concrete gravity dam
2. Weight minimization of a tension/compression spring
3. Cost minimization in welded beam design
4. Cost optimization of a pressure vessel (material, forming, welding)
5. Optimal configuration of an industrial refrigeration system
6. 2D trajectory optimization for path planning,
7. Inverse identification of mechanical properties of a flat composite panel under axial compression (by matching experimental data and finite element simulations).

Problems 2-5 belong to the CEC 2020 benchmark suite for constrained mechanical design, while problem 7 exemplifies a nonlinear inverse problem typical in aeronautical structural engineering.

To further assess scalability and global search capabilities, two classical large-scale test functions, Rosenbrock and Rastrigin, were also solved with up to 1000 variables.

The performance of SHGWJA was benchmarked against state-of-the-art metaheuristics, including CEC competition winners (e.g., LSHADE, IMODE), recent high-performing algorithms (e.g., MPA, GTO), and over ten widely recognized methods, each outperforming numerous alternatives in prior studies.

Results confirm the competitiveness and efficiency of SHGWJA [170] across diverse optimization scenarios.

The remainder of this chapter is organized as follows:

- Section 4.2 reviews the formulations of the GWO and JAYA algorithms, which serve as the foundation for the hybridization, and presents the SHGWJA algorithm in detail.
- Section 4.3 outlines the test problems and implementation aspects.
- Section 4.4 discusses the optimization results.
- Section 4.5 concludes with key findings and introduces the following topics.

This investigation demonstrated that SHGWJA offers a promising balance between exploration and exploitation, making it a competitive and reliable choice for solving complex constrained optimization problems in both engineering design and large-scale mathematical domains.

4.2 HYBRID METAHEURISTIC DESIGN: FROM GWO AND JAYA TO SHGWJA

The Grey Wolf Optimizer (**GWO**) algorithm simulates the leadership hierarchy and hunting behaviour of grey wolves. Search agents are ranked as alpha (α), beta (β), delta (δ), and omega (ω), with α, β, and δ guiding the search process, while ω agents follow. The hunting process is mathematically modelled as follows:

$$\bar{D} = \left|\bar{C} \cdot \bar{X}_p\right| - \bar{X}_{it} \tag{4.1}$$

$$\bar{X}_{it+1} = \bar{X}_p - \bar{A} \cdot \bar{D} \tag{4.2}$$

where $\bar{X}_p$ is the position vector of the prey (best solution), $\bar{X}_{it}$ is the position of a grey wolf at iteration it, $\bar{A}$ and $\bar{C}$, are coefficient vectors computed as:

$$\bar{A} = 2\bar{a} \cdot \bar{r}_1 - \bar{a} \tag{4.3}$$

$$\bar{C} = 2\bar{r}_2 \tag{4.4}$$

with $\bar{a}$ that decreases linearly from 2 to 0 over iterations, and $\bar{r}_1$, $\bar{r}_2 \in [0,1]$ that are random vectors. Since the true prey position is unknown, GWO assumes that α, β, and δ have the best estimates, and each agent updates its position based on their influence:

$$\begin{cases} \bar{D}_\alpha = |\bar{C}_1 \cdot \bar{X}_\alpha| - \bar{X}_{it} \\ \bar{D}_\beta = \left|\bar{C}_2 \cdot \bar{X}_\beta\right| - \bar{X}_{it} \\ \bar{D}_\delta = |\bar{C}_3 \cdot \bar{X}_\delta| - \bar{X}_{it} \end{cases} \tag{4.5}$$

$$\begin{cases} \bar{X}_1 = \bar{X}_\alpha - \bar{A}_1 \cdot \bar{D}_\alpha \\ \bar{X}_2 = \bar{X}_\beta - \bar{A}_2 \cdot \bar{D}_\beta \\ \bar{X}_3 = \bar{X}_\delta - \bar{A}_3 \cdot \bar{D}_\delta \end{cases} \tag{4.6}$$

$$\bar{X}_{it+1} = \frac{\bar{X}_1 + \bar{X}_2 + \bar{X}_3}{3} \tag{4.7}$$

These updates continue until the maximum number of iterations or function evaluations is reached. The coefficient $\bar{A}$ controls the balance between exploration ($\|\bar{A}\| > 1$) and exploitation ($\|\bar{A}\| < 1$). Despite its effectiveness, GWO tends to suffer from premature convergence and limited diversity due to over-reliance on the top three agents. To overcome these limitations, three new variants have been introduced in recent literature:

- GGWO (Gaze Cues Learning-based GWO) [171] introduces three candidate solutions per agent:
  1) $\bar{X}_{i-GWO}$: classic GWO update (Eq. 4.7)
  2) $\bar{X}_{i-NGCL}$: neighbour gaze cues learning, using α and the two nearest higher/lower-ranked wolves

3) $\bar{X}_{i-RG}$: random gaze cues learning, using two randomly chosen wolves.

The best among the three candidates replaces the current position if its fitness improves.

- GWO-C1 [172]: removes the coefficient vector $\bar{C}$ and absolute value in Eq. (4.5)
- GWO-C2 [172]: replaces $\bar{X}_\alpha$, $\bar{X}_\beta$, $\bar{X}_\delta$ in Eq. (4.6) with the current position $\bar{X}_{it}$.

Among these, GGWO is more suitable for this study, as it has demonstrated superior performance on real-world engineering problems [171].

However, the **JAYA** method is one of the simplest metaheuristic algorithms for optimization [13], using only one equation to update the solution and without internal parameters other than the population size $NPOP$ and the maximum number of iterations $N_{itermax}$. Let $X_{j,i,it}$ be the $j-th$ optimization variable (j=1, …, NDV) of the $i-th$ search agent (i=1, …, NPOP) at iteration it. Its update is defined by:

$$X_{j,i,it}^{new} = X_{\mathrm{j,i,it}} + r_{1,\mathrm{j,it}}\left(X_{\mathrm{j,best,it}} - \left|X_{\mathrm{j,i,it}}\right|\right) - r_{2,\mathrm{j,it}}\left(X_{\mathrm{j,worst,it}} - \left|X_{\mathrm{j,i,it}}\right|\right) \tag{4.8}$$

where $r_{1,j,it}$ and $r_{2,j,it}$ are random numbers in [0,1] and $X_{j,best,it}$ and $X_{j,worst,it}$ are the values of the $j-th$ variable for the best and worst solutions in the population at iteration it, respectively. The term $r_{1,j,it}\left(X_{j,best,it} - \left|X_{j,i,it}\right|\right)$ enables the approach to the best solution, while $-r_{2,j,it}\left(X_{j,worst,it} - \left|X_{j,i,it}\right|\right)$ helps to escape from the worst. The use of the absolute value $\left|X_{j,i,it}\right|$ enhances exploration and limits bias towards zero in mathematical problems where variables can be zero. The new solution $\bar{X}_i^{new}$ is compared with the previous one $\bar{X}_i^{pre}$: if it is better, it replaces the previous solution in the population. The process continues until reaching $N_{itermax}$, generally requiring $NPOP \cdot N_{itermax}$ function evaluations. Variants [158], [159], [160], [161], [162], [163], [164], [165], [166], [167], [168], [169], [171], [172], [173], [174], [175] like SJAYA accept $\bar{X}_i^{new}$ even if the cost is equal to $\bar{X}_i^{pre}$ [176]. Rao and Saroj [18] improved the exploration/exploitation balance by dividing the population into m adaptive subpopulations (SAMP-JAYA), adjusting m based on improvements found. The hybrid EHRJAYA [163] combines JAYA with the Rao-1 optimizer, choosing the perturbation based on probabilistic thresholds, with adaptive strategies to improve population diversity and facilitate the transition from exploration to local exploitation. The EJAYA variant [177] updates variables considering also the current mean solution and a historical population of solutions, improving the use of population information to balance exploration and exploitation. Finally, I-JAYA [178] integrates fuzzy clustering, experience learning, and Cauchy mutation to speed up convergence, balance exploration, and reduce the risk of getting stuck in local optima.

Building upon the strengths of both GWO and JAYA, a new approach was developed to further enhance optimization performance. The **SHGWJA** [170] algorithm is a simple hybrid metaheuristic that combines the exploitation capabilities of GWO with the exploration efficiency of JAYA through elitist strategies. This new algorithm effectively explores the search space while keeping the number of function evaluations low. The SHGWJA algorithm can be outlined in four main steps, as shown in Figure 1.2- and described below:

1) GWO-driven exploration
2) Elitist filtering

3) JAYA refinement and adaptive mirroring for stagnation escape
4) Robust Convergence Check.

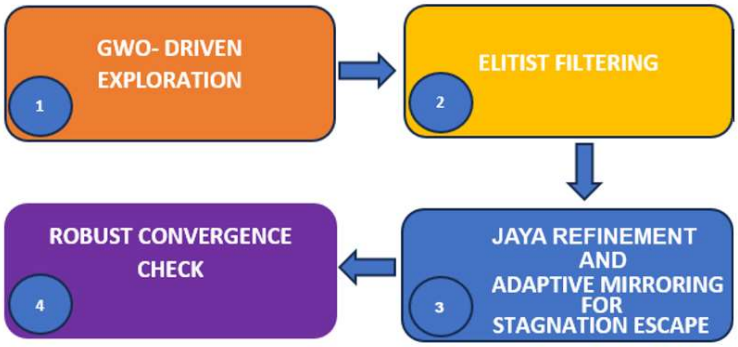


Figure 4.3- THE FOUR MAIN STAGES OF THE SHGWJA

The optimization begins with the generation of an initial population of $NPOP$ candidate solutions (or wolves), where each design vector $\bar{X}_1$ consists of NDV decision variables. The initial population is created randomly within predefined variable bounds:

$$X_j^i = X_j^L + \rho_j^i \cdot (X_j^U - X_j^L) \qquad \begin{cases} j = 1, \dots, NDV \\ i = 1, \dots, N_{POP} \end{cases} \tag{4.9}$$

where NDV is the number of optimization variables, X_j^L and X_j^U are the lower and upper bounds of the $j-th$ variable and ρ_j^i is a random number in the interval [0,1].

The **first step** involves using the classical GWO equations (4.1-4.7) to update each individual $\bar{X}_1$ in the population. To handle constraints, the standard cost function $W(\vec{X})$ is augmented with a quadratic penalty term, resulting in the penalized cost function:

$$W_p(\bar{X}) = W(\bar{X}) + p \cdot \psi \tag{4.10}$$

where p is the penalty coefficient. The penalty function ψ ensures that infeasible solutions are appropriately downgraded during selection. It is defined as follows:

$$\psi = \sum_{k=1}^{N_{con}} \left(\max\left(0, g_j\right)\right)^2 \tag{4.11}$$

The three best solutions found so far are designated as α, β and δ wolves, where $W_p(\bar{X}_\alpha) < W_p(\bar{X}_\beta) < W_p(\bar{X}_\delta)$. These acts as leaders guiding the search. The position update for each search agent produces a provisional trial design $\bar{X}_{i,tr}$. As in [67], the components of the $\bar{a}$ vector

decreases linearly from 2 to 0 throughout the optimization, enhancing the balance between exploration and exploitation.

In the **second step**, unlike the classical GWO where all trial solutions are evaluated, SHGWJA introduces a cost-based elitist filter to avoid unnecessary function evaluations and thereby reduce computational cost:

- The trial design $\bar{X}_{i,tr}$ is initially evaluated using the unpenalized cost function $W(\bar{X}_{i,tr})$.
- If $W(\bar{X}_{i,tr}) > 1.1 \cdot W(\bar{X}_{opt})$ the design is provisionally rejected and the old design $\bar{X}_i$ is retained.

  The threshold of $1.1 \cdot W(\bar{X}_{opt})$ has been adopted because it effectively balances the exploration and exploitation phases during the optimization process. During the exploration phase, the optimizer applies significant perturbations to the design variables, increasing the likelihood of solution improvement. Consequently, trial solutions rarely exceed the threshold $1.1 \cdot W(\bar{X}_{opt})$, as this phase tends to favour potentially better solutions. In the exploitation phase, however, the optimizer faces the challenge of escaping local minima to reach the global optimum. Like Simulated Annealing (SA), the optimizer may temporarily accept solutions worse than the current one to avoid stagnation in suboptimal regions and to promote a more thorough exploration of the solution space. In particular, advanced SA formulations [149], [158] employ probabilistic acceptance criteria, where the probability of accepting a worse solution can be as high as 90% when the ratio γ between the cost increase of the trial solution $\vec{X}_{i,tr}$ and the annealing temperature T is 0.1. This mechanism justifies the use of the threshold $1.1 \cdot W(\bar{X}_{opt})$, as it efficiently filters out solutions with significantly worse costs (exceeding the current best by more than 10%) while still allowing the temporary acceptance of slightly worse solutions. This fosters a balance between intensification and diversification in the search process. In other words, the threshold acts as a filter to avoid unnecessary evaluations of clearly inferior solutions, reducing computational cost without limiting the algorithm's ability to overcome local minima, thanks to a reasonable acceptance threshold inspired by the principles of Simulated Annealing.

- If the trial design is promising $W(\bar{X}_{i,tr}) \leq 1.1 \cdot W(\bar{X}_{opt})$, it is further refined using a JAYA-based strategy.

Two scenarios determine the refinement path:

1) If the GWO trial was unsuccessful (trial rejected by elitist threshold), JAYA is applied to the original solution:

$$(\bar{X}_{i,tr})' = \bar{X}_i + \bar{\lambda}_1 \cdot (\bar{X}_{opt} - \bar{X}_i) - \bar{\lambda}_2 \cdot (\bar{X}_{worst} - \bar{X}_i) \quad (4.2)$$

2) If the GWO trial was successful, JAYA is applied to the trial solution itself:

$$(\bar{X}_{i,tr})' = \bar{X}_{i,tr} + \bar{\lambda}_1 \cdot (\bar{X}_{opt} - \bar{X}_{i,tr}) - \bar{\lambda}_2 \cdot (\bar{X}_{\delta} - \bar{X}_{i,tr}) \quad (4.3)$$

In both equations $\bar{\lambda}_1$ and $\bar{\lambda}_2$ are vectors of random values in [0,1], and the absolute value of variable components is used to promote exploratory behaviour, following the logic of Equation (4.8) from the JAYA algorithm.

The refined solution $\bar{X}_{i,tr}$ is evaluated only if it satisfies the elitist threshold, and it replaces $\bar{X}_i$ if it shows better performance than both the original and the trial solutions. Equation (4.12) supports global exploration by pulling solutions toward the current best and away from the worst. Equation (4.13) focuses on local exploitation, refining already competitive designs toward elite individuals.

The **third step** focuses on updating the population and addressing stagnation using a mirror-based approach. After refinement, all trial solutions are evaluated, and the population is re-sorted based on their penalized cost W_p. The α, β and δ wolves are updated accordingly. To counteract potential stagnation, when α, β, and δ remain unchanged, the algorithm implements a mirroring mechanism. New candidate positions for β and δ are generated by reflecting them across the current best solution $\vec{X}_{opt}$:

$$\begin{cases} \bar{X}_\beta^{\,mirr} = \left(1 + h_{mirr,\beta}\right) \cdot \bar{X}_{opt} - h_{mirr,\beta} \cdot \bar{X}_\beta \\ \bar{X}_\delta^{\,mirr} = \left(1 + h_{mirr,\delta}\right) \cdot \bar{X}_{opt} - h_{mirr,\delta} \cdot \bar{X}_\delta \end{cases} \tag{4.4}$$

where $h_{mirr,\beta}, h_{mirr,\delta} \in [0,1]$ are randomly sampled to control step size and ensure feasibility. These mirror wolves are ranked together with the existing elite wolves, and the best three among $\bar{X}_\alpha, \bar{X}_\beta, \bar{X}_\delta, \bar{X}_\beta^{\,mirr}, \bar{X}_\delta^{\,mirr}$, become the new α, β and δ wolves. Figure 4.4 shown this elitist mirroring strategy implemented by SHGWJA.

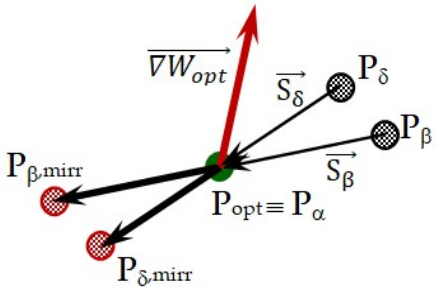


Figure 4.4- ELITIST MIRRORING STRATEGY IN SHGWJA

The two worst individuals are also evaluated for potential replacement, thus maintaining high population quality.

In the **fourth step** instead of relying on traditional stagnation checks (e.g., tracking changes in $\bar{X}_{opt}$ or cost function), SHGWJA evaluates convergence through normalized population diversity in both design and cost spaces:

$$Max\left\{\begin{matrix}\dfrac{STD\{\|\bar{X}_1-\bar{X}_{aver}\|,\|\bar{X}_2-\bar{X}_{aver}\|,\dots,\|\bar{X}_{NPOP}-\bar{X}_{aver}\|\}}{\|\overrightarrow{\bar{X}_{aver}}\|};\\ \dfrac{STD\{W_1,W_2,\dots,W_{NPOP}\}}{W_{aver}}\end{matrix}\right\}\leq\varepsilon_{conv} \tag{4.5}$$

where the convergence limit $\varepsilon_{\text{conv}}$ is equal to 10^{-7}, $\bar{X}_{aver}=1/N_{\text{POP}}\cdot\left(\sum_{i=1}^{N_{\text{POP}}}\bar{X}_i\right)$ and the average cost function $W_{\text{aver}}=1/N_{\text{POP}}\cdot\left(\sum_{i=1}^{N_{\text{POP}}}W(\bar{X}_i)\right)$. This criterion ensures that the entire population converges toward a narrow region around the global optimum, supporting the principle of collective intelligence in population-based algorithms. Traditional convergence criteria in optimization algorithms typically rely on monitoring changes in the cost function value W_{opt} and in the best solution vector $\bar{X}_{opt}$ obtained over the final iterations. The optimization process is halted when the variations in both quantities fall below a predefined convergence threshold.

This approach helps avoid premature convergence to local minima, especially when the optimizer explores regions of the design space characterized by numerous competitive solutions yielding similar cost values.

However, such criteria may be inadequate for population-based algorithms, where effective search relies on the collective behaviour of all candidate solutions within the population. In these cases, it is possible for only sub-optimal individuals to be updated while the best solution $\bar{X}_{opt}$ remains unchanged over several iterations. If convergence is assessed solely on $\bar{X}_{opt}$, the algorithm might erroneously terminate, despite the presence of substantial diversity among the remaining candidates, indicating that the search is still in the exploration phase, which typically dominates the early stages of optimization. To address this limitation, Equation (4.15) adopts a different perspective: it assumes that, in the convergence phase, all individuals in the population migrate toward the global optimum, concentrating within the most promising region of the search space.

Under this assumption, convergence is associated with a reduction in population diversity and an aggregation of solutions around $\bar{X}_{opt}$. Accordingly, Equation (4.15) introduces a dimensionless convergence metric by normalizing the standard deviation of individuals' positions with respect to the average solution. When all candidate solutions converge toward $\bar{X}_{opt}$, their dispersion diminishes, and they effectively coincide with their mean. A similar normalization is applied to the standard deviation of the cost function values with respect to their average. In this framework, the search is deemed to have converged only when both the spatial distribution of solutions and their associated costs are tightly clustered around their respective averages, indicative of genuine proximity to the global optimum.

The optimization problem continues iteratively through steps 1-3 until the convergence condition in Equation (4.15) is achieved. Once convergence is achieved, SHGWJA halts the search and records the optimal solution together with its associated objective function value and other relevant metrics.

In summary, SHGWJA is a Grey Wolf-based optimizer enhanced with JAYA-inspired operators and elitist strategies to improve both exploration and exploitation. Unlike the classical Grey Wolf Optimizer (GWO), SHGWJA updates the population by verifying whether

the α, β and δ wolves can improve the current best solution. Three key modifications are introduced: cost-based elitist filter $W(\bar{X}_{i,tr}) \leq 1.1 \cdot W(\bar{X}_{opt})$, JAYA-inspired update strategies (Equations (4.12) and (4.13)), mirroring strategy defined by Equation 4.14 to enhance population diversity.

Although simple to implement, these additions significantly improve performance without increasing algorithmic complexity.

Notably, classical GWO and JAYA lack dedicated exploitation mechanisms and often exhibit weak convergence in local regions. SHGWJA overcomes this limitation by dynamically adjusting its search behaviour based on the performance of the current trial solution: if the trial solution improves the best record, SHGWJA applies the exploitation strategy defined by Equation 4.13, if it does not, the algorithm switches to exploration using the global search strategy represented in Equation 4.12. This adaptive mechanism ensures a balance between exploration and exploitation throughout the optimization process. Moreover, continuous exploration prevents premature convergence of the α, β and δ wolves.

Due to the integration of JAYA-based modifications into the GWO structure, SHGWJA can be considered a high-level hybrid optimizer, where both components contribute to generating new designs.

Importantly, SHGWJA requires no additional control parameters beyond those of standard GWO and JAYA, a rare characteristic among hybrid metaheuristics, which often introduce extra parameters to manage component switching.

Regarding population size $NPOP$, larger populations may improve exploration but lead to a higher number of function evaluations, as total evaluations scale with $NPOP \cdot N_{itermax}$.

However, if the optimizer can consistently generate high-quality solutions, a large population becomes unnecessary. Additionally, biological inspiration supports smaller sizes: grey wolves typically hunt in packs of 5-11 members (up to 2-3 families, or ~10-20 individuals).

Based on this, all SHGWJA optimizations in this study were conducted with $NPOP = 10$. Sensitivity analyses confirmed this choice to be effective for both engineering and mathematical problems, with larger populations (e.g., 30 or 50) used for consistency with related studies on other metaheuristics.

In terms of computational cost, classical GWO has a complexity of $O(NPOP \cdot NDV + N_{iter} \cdot NPOP \cdot (NDV + \log NPOP))$.

For SHGWJA, each iteration includes $NPOP$ additional operations for the elitist filter, $NPOP \cdot NDV$ operations for each JAYA strategy (Equations (4.12) and (4.13)), $2 \cdot NDV$ operations for the mirroring strategy (Equation (4.14)).

Thus, SHGWJA performs per iteration an additional $NPOP + NDV \cdot (NPOP + 2)$ operations over classical GWO. For example, with NDV=14 and NPOP=10, SHGWJA requires at most 10+(12·14) =178 extra operations per iteration. In the worst-case scenario, the total complexity

becomes $O(140 + N_{iter} \cdot NPOP \cdot 328)$ compared to $O(140 + N_{iter} \cdot 150)$ for classical GWO.

While this is higher than efficient variants such as GGWO [171] and EHRJAYA [163], with complexities of O(140+Niter·150) and O(Niter·160) respectively, SHGWJA's additional operations are directly aimed at producing higher-quality trial solutions. This leads to improved search performance in terms of convergence and robustness, justifying the increased computational effort.

The flowchart of the SHGWJA algorithm is presented in Figure 4.5.

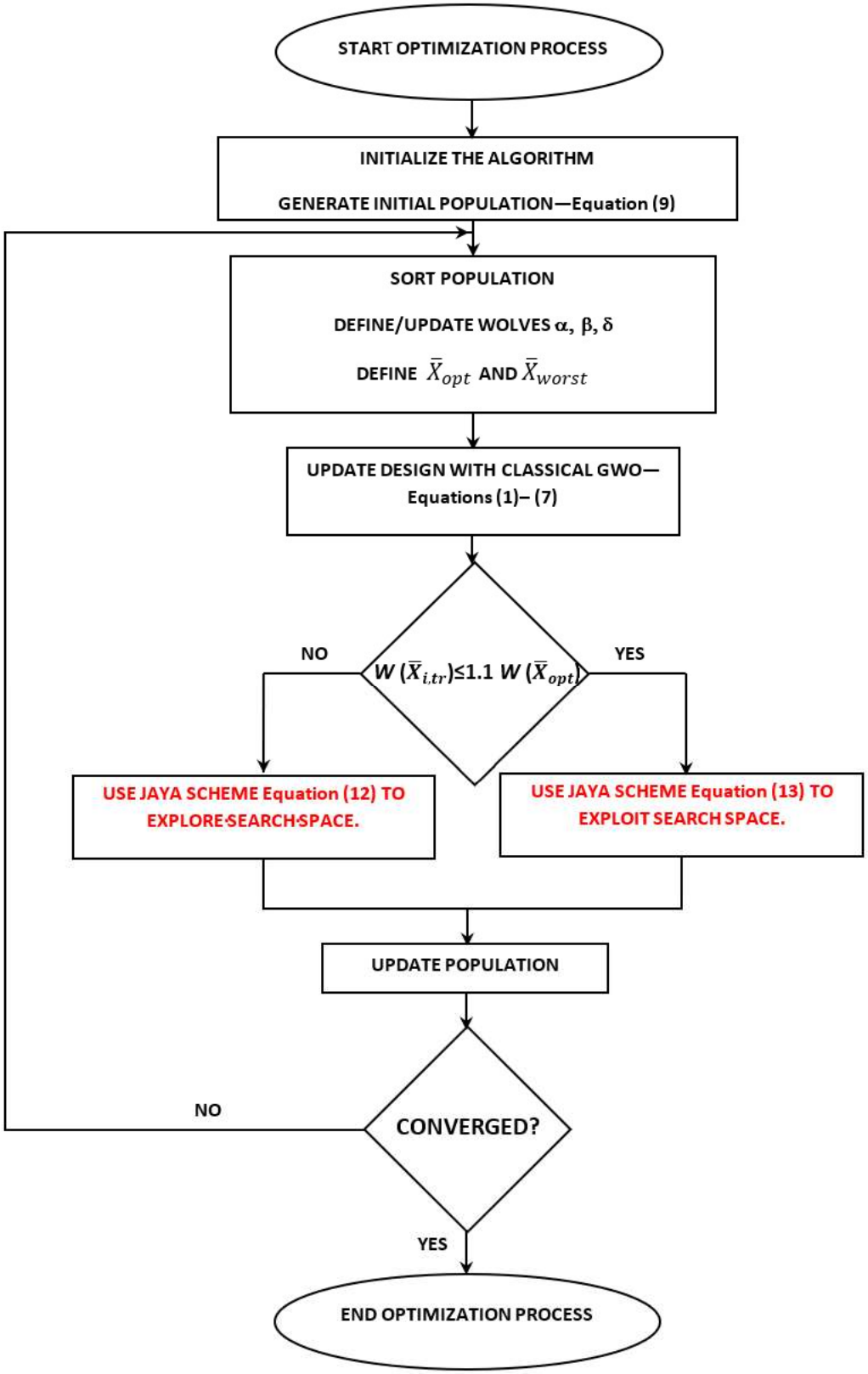


Figure 4.5- FLOWCHART OF THE SHGWJA OPTIMIZATION ALGORITHM

4.3 APPLICATION AND EVALUATION OF THE SHGWJA ALGORITHM

The SHGWJA algorithm has been evaluated on nine optimization problems, including both theoretical benchmark functions and real-world engineering applications: **Rosenbroke** and **Rastrigin** for mathematical optimization, **Concrete Gravity Dam**, **Tension/Compression Spring**, **Welded Beam**, **Pressure Vessel,** and an **industrial refrigeration system** design for engineering design optimization problems, **path planning** for robotics optimization, and **mechanical characterization of a flat composite panel under axial compression**.

4.3.1 Mathematical optimization: Rosenbrock and Rastrigin

Rosenbrock and Rastrigin functions are widely used to assess an algorithm's performance on non-convex and multimodal landscapes.

Unimodal optimization problems are used to assess the exploitation capability of metaheuristic algorithms. In contrast, multimodal problems evaluate their exploration capability, since the number of local optima increases exponentially with the number of design variables (NDV), even though convergence to the global minimum may sometimes be faster.

The **Rosenbrock** problem (see Appendix 8.1.1), called also "Banana Function", is as a non-linear unimodal problem, defined as follows:

$$Minimize\ W(\bar{X}) = \sum_{j=1}^{NDV-1} \left[100 \cdot \left(X_{j+1} - X_j^{\,2}\right)^2 \left(1 - X_j\right)^2\right]$$

$$with\ the\ following\ boundaries \tag{4.16}$$

$$-2 \le X_j \le 2$$

The global minimum of W is 0, which occurs when all optimization variables are equal to 1.

The **Rastrigin** problem (see Appendix 8.1.2) is a multimodal problem and is formulated as follows:

$$Minimize\ W(\bar{X}) = \sum_{i=1}^{NDV} (X_i^2 - 10 \cdot cos(2 \cdot \pi \cdot X_i) + 10)$$

$$with\ the\ following\ boundaries \tag{4.6}$$

$$-5.12 \le X_j \le 5.12$$

The global minimum of W is 0, attained when all variables equal 0.

The SHGWJA was thoroughly compared with several of the best-performing metaheuristic algorithms (MAs) in literature using the following combination of population size (N_{POP}) and NDV:

a) N_{POP}=10 and NDV=10, 30, 50, 100
b) N_{POP}=30 and NDV=30, 100, 500, 1000

c) N_{POP}=50 and NDV=30, 50, 100.

The comparison across the different settings was performed with respect to:

(i-ii) improved GWO [110] and JAYA [18], [160], [176], [178], which incorporate chaotic perturbation of the design variables and hybridization with sequential quadratic programming

(iii-iv) the equilibrium optimizer (EO) [101] and its enhanced version, the modified equilibrium optimizer (mEO) [157]

(v) the light spectrum optimizer (LSO) [102]

(vi) the Runge–Kutta optimizer (RUN) [113]

(vii–viii) the arithmetic optimization algorithm (AOA) [114] and its improved variant, the IAOA, which enforces a return to exploration if no design improvement occurs over a certain number of iterations [161]

(ix) the mother optimization algorithm (MOA) [120]

(x) the preschool education optimization algorithm (PEOA) [179]

(xi) the learning cooking algorithm (LCA) [121]

(xii) the snow leopard optimization algorithm (SLOA) [136]

(xiii) the giant trevally optimizer (GTO) [146]

(xiv–xvi) the basic marine predators algorithm (MPA) [145], its improved version with self-adaptive weight parameter tuning and a dynamic social mechanism (IMPA) [166], and a hybrid algorithm combining MPA with the gorilla troops optimizer (EGTO) [167]

(xvii) the hybridized differential evolution naked mole-rat algorithm (SaDN) [159]

(xviii) the modified sine cosine algorithm (MSCA) [164]

(xix) the hybrid slime mould simulated annealing (hSM-SA) algorithm [165]

(xx–xxi) particle swarm optimization variants [180], based either on the global best (GPSO) or the presence of an aging leader with challengers (ALC-PSO), as well as the NDWPSO hybrid algorithm [181], which combines PSO, differential evolution, and whale optimization—three of the most widely cited MAs

(xxii–xxv) high-performance optimizers, including the covariance matrix adaptation evolution strategy (CMA-ES) [182] and IEEE CEC competition winners, such as success-history-based adaptive differential evolution with linear population size reduction (LSHADE) [183], its variant incorporating covariance matrix learning with Euclidean neighborhood (LSHADE-cnEpSin) [184], and the hybrid LSHADE-SPACMA algorithm [185], which combines LSHADE and CMA-ES

(xxvi–xxviii) the hybrid harmony search (hybrid HS), hybrid big bang–big crunch (hybrid BBBC), and hybrid fast simulated annealing (HFSA) algorithms developed in [158].

4.3.2 Engineering design optimization

Engineering design optimization represents a critical area of research and application within the field of computational engineering, wherein mathematical programming techniques are employed to determine optimal configurations of complex systems under a set of predefined objectives and constraints to navigate through high-dimensional, nonlinear, and multi-objective design spaces.

In the following sections, several canonical engineering optimization problems are examined, including the design of concrete gravity dams, tension/compression springs, welded beams, pressure vessels, and industrial refrigeration systems. These case studies serve to illustrate the practical implementation of optimization strategies in diverse engineering contexts and highlight the trade-offs inherent in real-world design scenarios.

4.3.2.1 Concrete Gravity Dam

The first design problem focuses on optimizing the shape of a **concrete gravity dam** (See Appendix 8.6), defined by 9 design variables (see Figure 4.6), which represent the characteristic dimensions of the dam's cross-sectional shape, and 8 constraints, of which 6 are nonlinear. The objective is to minimize the volume of the dam.

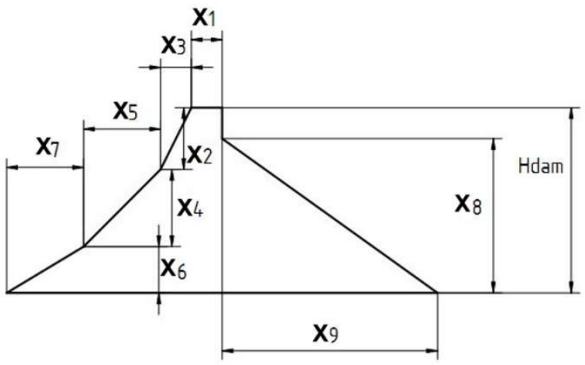


Figure 4.6- GRAVITY DAM: IDENTIFICATION OF VARIABLES.

The mathematical model is as follows:

$$Minimize$$

$$W(\bar{X}) = H_{dam} \cdot X_1 + X_3 \cdot (X_4 + X_6) + X_5 \cdot X_6 + \frac{X_2 \cdot X_3}{2} + \frac{X_4 \cdot X_5}{2} + \frac{X_6 \cdot X_7}{2} + \frac{X_8 \cdot X_9}{2} \tag{4.18}$$

$$Subject\ to\ 8\ constrains$$

$$G_1(\bar{X}) = 4 - SFS \leq 0$$

$$G_2(\bar{X}) = 1.5 - SFO \leq 0$$

$$G_3(\bar{X}) = -\sigma_u \leq 0$$

$$G_4(\bar{X}) = \sigma_u - \sigma_{max} \leq 0$$

$$G_5(\bar{X}) = -\sigma_D \leq 0$$

$$G_6(\bar{X}) = \sigma_D - \sigma_{max} \leq 0$$

$$G_7(\bar{X}) = X_2 + X_4 + X_6 - H_{dam} \leq 0$$

$$G_8(\bar{X}) = X_4 + X_6 - X_8 \leq 0$$

with the following boundaries

$$5 < X_1 < 20$$

$$20 < X_2 < 50$$

$$1 < X_3 < 20$$

$$40 < X_4 < 80$$

$$10 < X_5 < 30$$

$$30 < X_6 < 80$$

$$20 < X_7 < 40$$

$$50 < X_8 < 150$$

$$80 < X_9 < 150$$

where SFS and SFO are the safety factors against sliding and overturning of the dam, respectively; σ_u, σ_D are the vertical fatigue specific loads at the upstream and downstream faces, σ_{max} is their fatigue limit, and H_{dam} is the height of the dam.

Further details on this test problem can be found in Appendix 8.6, and in [158], [186], [187].

The SHGWJA algorithm was compared with the most effective optimizers reported in the literature for this design example: the hybrid Harmony Search (hybrid HS), hybrid Big Bang–Big Crunch (hybrid BBBC), and hybrid Fast Simulated Annealing (HFSA) algorithms presented in [158]. This comparison is particularly informative because the hybrid optimizers in [158] combine metaheuristic search with approximate line search and gradient information, whereas SHGWJA relies solely on classical GWO and JAYA operators enhanced by elitist strategies.

Additionally, SHGWJA was compared with the classical Grey Wolf Optimizer (GWO), classical and improved JAYA formulations [158], multi-level cross entropy optimizer (MCEO) [186], Flying Squirrel Optimizer (FSO) [187], modified Harmony Search with adaptive parameter updating (mAHS, derived from [188]), modified Big Bang–Big Crunch with upper bound strategy (mBBBC-UBS, derived from [189]), and modified sinusoidal differential evolution (MsinDE, derived from [190]).

4.3.2.2 Tension/Compression Spring

The **Tension/Compression Spring** design problem (See Appendix 8.2.1) is defined by 3 design variables (see Figure 4.7), X_1= d= wire diameter, X_2= D= mean coil diameter, X_3= N= number of active coils, and 4 constraints, of which 3 are nonlinear. The objective is to minimize the weight of the helical spring [191].

Despite the relatively limited number of constraints, the significance of this case study arises from the subtle yet critical issue of error propagation during approximation. The presence of exponents in the formulas amplifies the effects of truncating significant digits throughout the computational process, thereby introducing non-negligible inaccuracies. This makes the case particularly noteworthy as it highlights the delicate balance between mathematical modelling and numerical precision.

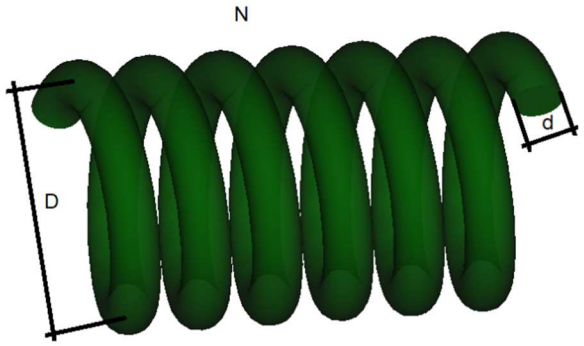


Figure 4.7- REPRESENTATION OF THE DESIGN VARIABLE USED IN THE SPRING OPTIMIZATION PROBLEM

The optimization problem is formulated as follows:

$$Minimize\ W(\bar{X}) = (X_3 + 2) \cdot X_2 \cdot {X_1}^2 \tag{4.19}$$

Subject to 4 Constrains

$$G_1(\bar{X}) = 1 - \frac{{X_2}^3 \cdot X_3}{71785 \cdot {X_1}^4} \leq 0$$

$$G_2(\bar{X}) = \frac{4 \cdot {X_2}^2 - X_1 \cdot X_2}{12566 \cdot \left({X_1}^3 \cdot X_2 - {X_1}^4\right)} + \frac{1}{5108 \cdot {X_1}^2} - 1 \leq 0$$

$$G_3(\bar{X}) = 1 - \frac{140.45 \cdot X_1}{X^2 \cdot X_3} \leq 0$$

$$G_4(\bar{X}) = 1 - \frac{140.45 \cdot X_1}{{X_2}^2 \cdot X_3} \leq 0$$

with the following boundaries

$0.05 < X_1 < 2$

$0.25 < X_2 < 1.3$

$2 < X_3 < 15$

The tension/compression spring design problem was solved using the SHGWJA algorithm, and the resulting performance was rigorously compared with a set of benchmark metaheuristic approaches:

(i) the Gaze cues learning-based Grey Wolf Optimizer (GGWO) [171], currently regarded as the most advanced GWO formulation.

(ii) the astrophysics-inspired GWO (MGWO-III) [110], a competitive variant not previously compared with GGWO in [171].

(iii) several JAYA extensions, including EHRJAYA [163] (a hybridization with Rao-1 reported as the leading method among 34 MAs), SAMP-JAYA [18] (employing an adaptive sub-population mechanism), and EJAYA [177] (which enlarges the candidate solution pool beyond the best and worst individuals).

(iv) the equilibrium optimizer (EO) [101].

(v) the light spectrum optimizer (LSO) [102], identified as the top performer among 32 Mas.

(vi–vii) the arithmetic optimization algorithm (AOA) [114] and its improved variant IAOA [161], which reactivates exploration under stagnation.

(viii) the mother optimization algorithm (MOA) [120].

(ix) the preschool education optimization algorithm (PEOA) [179].

(x) the learning cooking algorithm (LCA) [121].

(xi) the snake optimizer (SO) [135].

(xii) the starling murmuration optimizer (SMO) [140].

(xiii–xiv) the marine predators algorithm (MPA) [145] and its hybrid extension EGTO, integrating MPA with the gorilla troops optimizer [167].

(xv) the hybrid differential evolution naked mole-rat algorithm (SaDN) [159].

(xvi) the hybrid slime mould simulated annealing (hSM-SA) [165].

(xvii) the improved multi-operator differential evolution (IMODE) [192].

(xviii) the success-history based adaptive differential evolution with gradient-based repair (En(L)SHADE) [193].

(xix) nine further methods reported in Ref. [194], executed with a computational budget of 18000 function evaluations.

This comparison demonstrated the competitive performance of the proposed algorithm.

4.3.2.3 Welded Beam

The optimal design of a **welded beam** (see Appendix 8.2.3, [101] [102] [114] [120] [179] [121] [135] [140] [145] [159] [165] [167] [171] [18] [177] [181] [187] [192] [194] [195] [198]) aims to determine the minimum fabrication cost.

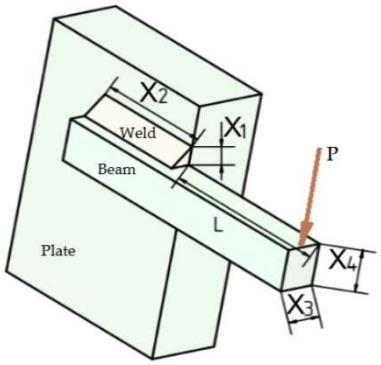


Figure 4.8- DEFINITION OF THE DESIGN VARIABLES EMPLOYED IN THE WELDED BEAM OPTIMIZATION PROBLEM

It involves 4 design variables (see Figure 4.8): thickness of the weld bead (X_1), length of the weld bead (X_2), height of the bar's cross-sectional area (X_3), thickness of the bar's cross-sectional area (X_4).

Let $\tau(\bar{X})$ denote shear stress in the bar, which must not exceed the allowable limit $\tau_{max} = 13600\ psi$; $\sigma(\bar{X})$ represent bending stress of the bar, constrained by $\sigma_{max} = 30000\ psi$; and $\delta(\bar{X})$ be the maximum deflection at the tip of the bar, which should not exceed $\delta_{max} = 0.25\ in$. The applied load $P = 6000\ lb$ must remain below the critical buckling load $P_c(\bar{X})$ of the beam, modelled as a narrow rectangular bar.

The optimization problem can be expressed as follows:

$$Minimize\ W(\bar{X}) = 1.10471 \cdot {X_1}^2 \cdot X_2 + 0.04811 \cdot X_3 \cdot X_4 \cdot (X_2 + 14) \tag{4.20}$$

Subject to 7 Constrains

$$G_1(\bar{X}) = \tau(\bar{X}) - \tau_{max} \leq 0$$

$$G_2(\bar{X}) = \sigma(\bar{X}) - \sigma_{max} \leq 0$$

$$G_3(\bar{X}) = \delta(\bar{X}) - \delta_{max} \leq 0$$

$$G_4(\bar{X}) = P - P_c(\bar{X}) \leq 0$$

$$G_5(\bar{X}) = X_1 - X_4 \leq 0$$

$$G_6(\bar{X}) = 0.125 - X_1 \leq 0$$

$$G_7(\bar{X}) = 1.10471 \cdot {X_1}^2 \cdot X_2 + 0.04811 \cdot X_3 \cdot X_4 \cdot (X_2 + 14) - 5.0 \leq 0$$

with the following boundaries

$0.1 \leq X_1 \leq 2.0$

$0.1 \leq X_2 \leq 10.0$

$10.0 \leq X_3 \leq 0.1$

$0.1 \leq X_4 \leq 2.0$

This problem was included in the mechanical engineering section of the CEC 2020 test suite on real-world constrained optimization problems from diverse fields [140] [193] [195] [196] [197]. Several variants of the problem have been proposed, resulting in different target optimal solutions.

In the classical formulation (variant 1), the moment of inertia J, the beam tip deflection $\delta(\bar{X})$, and the critical buckling load $P_c(\bar{X})$ were expressed as $J=2\sqrt{2} \cdot X_1 . X_2 \cdot \left[\frac{{X_2}^2}{12} + \left(\frac{X_1+X_3}{2}\right)^2\right]$, $\delta(\bar{X}) = \frac{6 \cdot P \cdot L^3}{E \cdot {X_3}^2 \cdot X_4}$, $P_c(\bar{X}) = \frac{4.013 \cdot E}{L^2}\sqrt{\frac{{X_3}^2 \cdot {X_4}^6}{36}}\left(1 - \frac{X_3}{2L}\right)\sqrt{\frac{E}{4G}}$ with a corresponding target optimum of 1.72485.

In a second formulation (variant 2), the moment of inertia was modified to $J = 2\sqrt{2} \cdot X_1 \cdot X_2 \cdot \left[\frac{{X_2}^2}{4} + \left(\frac{X_1+X_3}{2}\right)^2\right]$, yielding a target optimum of 1.695.

Finally, the CEC 2020 library formulation (variant 3) reported $J=2\sqrt{2} \cdot X_1 \cdot X_2 \cdot \left[\frac{{X_2}^2}{4} + \left(\frac{X_1+X_3}{2}\right)^2\right]$, $\delta(\bar{X}) = \frac{4 \cdot P \cdot L^3}{E \cdot {X_3}^2 \cdot X_4}$, $P_c(\bar{X}) = \frac{4.013 \cdot E}{L^2}\sqrt{\frac{{X_3}^2 \cdot {X_4}^6}{30}}\left(1 - \frac{X_3}{2L}\right)\sqrt{\frac{E}{4G}}$ with a corresponding target optimum of 1.670.

The solution obtained with SHGWJA algorithm was extensively compared with the most relevant Mas:

(i) the Gaze cues learning-based Grey Wolf Optimizer (GGWO) [171]

(ii) the astrophysics-inspired Grey Wolf Optimizer (MGWO-III) [110]

(iii) the Self-adaptive classification learning hybrid JAYA and Rao-1 Algorithm (EHRJAYA) [163] (reported as the best among 32 MAs)

(iv) the self-adaptive multi-population based Jaya Algorithm (SAMP-JAYA) [18]

(v) the Enhanced Jaya algorithm (EJAYA) [177]

(vi) the Equilibrium optimizer (EO) [101]

(vii) the Light spectrum optimizer (LSO) [102] (reported as the best among 27 MAs)

(viii) The arithmetic optimization algorithm (AOA) [114]

(ix) the Mother optimization algorithm (MOA) [120]

(x) the preschool education optimizer algorithm (PEOA) [179]

(xi) the Learning cooking algorithm (LCA) [121]

(xii) the Snake Optimizer (SO) [135]

(xiii) the Starling murmuration optimizer (SMO) [140]

(xiv) the Marine Predators Algorithm (MPA) [145]

(xv) the self-adaptive hybridized differential evolution naked mole-rat algorithm (SaDN) [159]

(xvi) the Hybridizing slime mould algorithm with simulated annealing (hSM-SA) [165]

(xvii) the enhanced gorilla troops optimizer (EGTO) [167]

(xviii) the hybrid PSO, differential evolution, and whale optimization (NDWPSO) [181]

(xix) the high-performance hybrid LSHADE-SPACMA [185], [195] (integrating history-based adaptive differential evolution with linear population size reduction and covariance matrix adaptation evolution strategy)

(xx) the Flying Squirrel Optimizer (FSO) [187]

(xxi) the improved multi-operator differential evolution algorithm (IMODE) [192]

(xxii) the adaptive initialization LSHADE algorithm enhanced with gradient-based repair (En(L)SHADE) [193]

(xxiii) the self-adaptive spherical search (SASS) [196], [197]

(xxiv) the Taguchi-integrated harmony search algorithm (TIHHSA) [198].

Additionally, SHGWJA was compared with the nine algorithms analysed in Ref. [194], each evaluated under a computational budget of 18000 function evaluations.

4.3.2.4 Pressure Vessel

The optimal design of a **pressure vessel** (see Appendix 8.2.4) aims to minimize material, forming, and welding costs, while ensuring compliance with the ASME Boiler and Pressure Vessel Code. The design is defined in terms of 4 variables (See Figure 4.9): sheet thickness ($X_1 = T_s$), hemispherical head thickness ($X_2 = T_h$), hemispherical head radius ($X_3 = R$), and cylindrical shell length ($X_4 = L$).

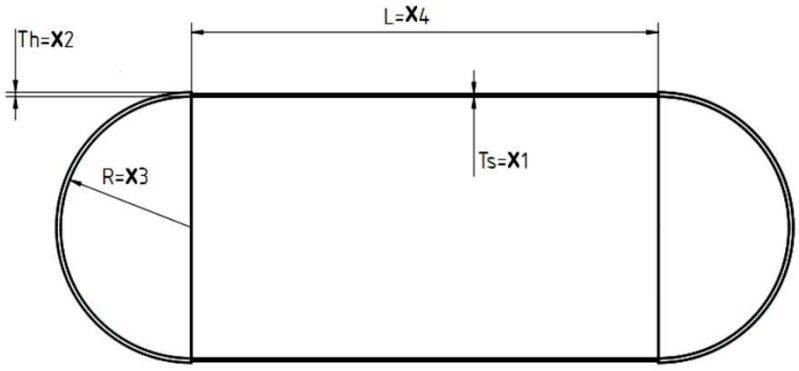


Figure 4.9- CROSS-SECTIONAL VIEW ILLUSTRATING THE DESIGN VARIABLES

The mathematical model that describes this problem is formulated as follows:

$$Minimize\ W(\bar{X}) = 0.6224 \cdot X_1 \cdot X_3 \cdot X_4 + 1.7781 \cdot X_2 \cdot {X_3}^2 + 3.1661 \cdot {X_1}^2 \cdot X_4 + 19.84 \cdot {X_1}^2 \cdot X_3 \quad (4.21)$$

$Subject\ to\ 4\ Constrains$

$G_1(\bar{X}) = -X_1 + 0.0193 \cdot X_3 \leq 0$

$G_2(\bar{X}) = -X_1 + 0.00954 \cdot X_3 \leq 0$

$G_3(\bar{X}) = -\pi \cdot {X_3}^2 \cdot X_4 - \frac{4}{3} \cdot \pi \cdot {X_3}^3 + 1296000 \leq 0$

$G_4(\bar{X}) = X_4 - 240 \leq 0$

$with\ the\ following\ boundaries$

$0 \leq X_1, X_2 \leq 100$

$10 \leq X_3, X_4 \leq 200$

This problem can be formulated in two ways: for the continuous-variable case, the boundary conditions $0 \leq X_1, X_2 \leq 100$ reported above are applicable, whereas in the mixed-variable formulation, the thickness variables X_1, X_2 must be integer multiples of 0.0625 in, corresponding to the standard increments of commercially available rolled steel plates.

Detailed discussions of this mechanical design example can be found in [16], [101], [102], [114], [120], [121], [135], [140], [145], [146], [151], [157], [159], [167], [167], [171], [177], [179], [181], [187], [194], [195], [198].

This benchmark problem was also included in the mechanical engineering section of the CEC 2020 test suite, which features real-world constrained optimization problems from various domains [140], [193], [195], [196], [197].

The comparative analysis of the SHGWJA algorithm on the pressure vessel problem was conducted with the main MAs:

(i) the Gaze cues learning-based Grey Wolf Optimizer (GGWO) [171]

(ii) the astrophysics-inspired Grey Wolf Optimizer (MGWO-III) [110]

(iii) the Self-adaptive classification learning hybrid JAYA and Rao-1 Algorithm (EHRJAYA) [163] (reported as the best among 33 MAs)

(iv) the self-adaptive multi-population based Jaya Algorithm (SAMP-JAYA) [18]

(v) the Enhanced Jaya algorithm (EJAYA) [177]

(vi) the Equilibrium optimizer (EO) [101] and mEO [157]

(vii) the Light spectrum optimizer (LSO) [102] (reported as the best among 31 MAs)

(viii) The arithmetic optimization algorithm (AOA) [114] and its improved version IAOA [161]

(ix) the Mother optimization algorithm (MOA) [120]

(x) the preschool education optimizer algorithm (PEOA) [179]

(xi) the Learning cooking algorithm (LCA) [121]

(xii) the Snake Optimizer (SO) [135]

(xiii) the Starling murmuration optimizer (SMO) [140]

(xiv) the Marine Predators Algorithm (MPA) [145]

(xv) the Giant trevally optimizer (GTO) [146]

(xvi) the self-adaptive hybridized differential evolution naked mole-rat algorithm (SaDN) [159]

(xvii) the Hybridizing slime mould algorithm with simulated annealing (hSM-SA) [165]

(xviii) the enhanced gorilla troops optimizer (EGTO) [167]

(xix) the hybrid PSO, differential evolution, and whale optimization (NDWPSO) [181]

(xx) the Flying Squirrel Optimizer (FSO) [187]

(xxi) the improved multi-operator differential evolution algorithm (IMODE) [192]

(xxii) the adaptive initialization LSHADE algorithm enhanced with gradient-based repair (En(L)SHADE) [193]

(xxiii) the self-adaptive spherical search (SASS) [196], [197]

(xxiv) the Taguchi-integrated harmony search algorithm (TIHHSA) [198]

In addition, the nine algorithms analysed in Ref.[194], which employed a computational budget of 25000 function evaluations.

4.3.2.5 Industrial Refrigeration System

The optimal design of an **industrial refrigeration system** (see Figure 4.10, Appendix 8.4, [199]) is part of the CEC-2020 benchmark suite, which comprises real-world mechanical engineering optimization problems [140], [193], [196], [197], [200], [201]. The primary objective is to minimize both the fabrication and operational costs of the refrigeration system, while defining 14 design variables and satisfying 15 non-linear constraints imposed by the system's geometry and heat transfer performance characteristics.

The 14 design variables include the shell diameter of the evaporator ($X_1 = D_{se}$), the shell diameter of the condenser ($X_2 = D_{sc}$), the shell thickness of the evaporator ($X_3 = T_{se}$), the shell thickness of the condenser ($X_4 = T_{sc}$), the tubesheet thickness for the evaporator ($X_5 = t_{se}$), the tubesheet thickness for the condenser ($X_6 = t_{sc}$), the fluid velocity in the evaporator ($X_7 = V_e$), the fluid velocity in the condenser ($X_8 = V_c$), the number of tube passes for the evaporator ($X_9 = P_e$), the number of tube passes for the condenser ($X_{10} = P_c$), the tube length for the evaporator ($X_{11} = L_e$), the tube length for the condenser ($X_{12} = L_c$), the condenser effectiveness ($X_{13} = \varepsilon$), and the inlet/outlet fluid temperature difference ($X_{14} = \Delta T_c$).

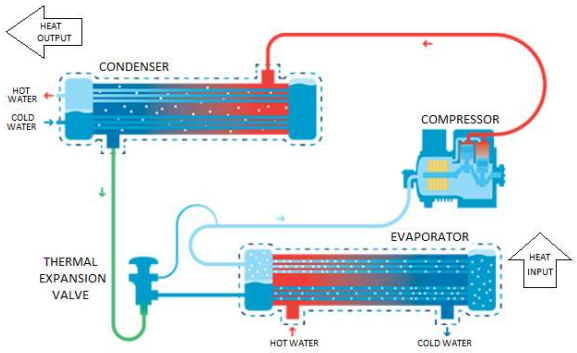


Figure 4.10- Representation of an Industrial refrigeration System

The optimization problem is formulated as follows

$$Minimize\ W(\bar{X}) = \tag{4.22}$$
$$63098.88 \cdot X_2 \cdot X_4 \cdot X_{12} + 5441.5 \cdot X_2^2 \cdot X_{12} + 115055.5 \cdot X_2^{1.664} \cdot X_6 + 6172.27 \cdot X_2^2 \cdot X_6 + 63098.88 \cdot X_1 \cdot X_3 \cdot X_{11} + 5441.5 \cdot X_1^2 \cdot X_{11} + 115055.5 \cdot X_1^{1.664} \cdot x_5 + 6172.27 \cdot X_1^2 \cdot X_5 + 140.53 \cdot X_1 \cdot X_{11} + 281.29 \cdot X_3 \cdot X_{11} + 70.26 \cdot X_1^2 + 281.29 \cdot X_1 \cdot X_3 + 281.29 \cdot X_3^2 + 14437 \cdot X_8^{1.8812} \cdot X_{12}^{0.3424} \cdot \frac{X_{10}}{X_{14}} \cdot X_1^2 \cdot \frac{X_7}{X_9} + 20470.2 \cdot X_7^{2.893} \cdot X_{11}^{0.316} \cdot X_1^2$$

$Subject\ to$ 15 constrains

$$G_1(\bar{X}) = \tau(\bar{X}) - \tau_{max} \leq 0$$
$$G_2(\bar{X}) = \sigma(\bar{X}) - \sigma_{max} \leq 0$$
$$G_3(\bar{X}) = \delta(\bar{X}) - \delta_{max} \leq 0$$
$$G_4(\bar{X}) = P - P_c(\bar{X}) \leq 0$$
$$G_5(\bar{X}) = X_1 - X_4 \leq 0$$
$$G_6(\bar{X}) = 0.125 - X_1 \leq 0$$
$$G_7(\bar{X}) = 1.10471 \cdot {X_1}^2 \cdot X_2 + 0.04811 \cdot X_3 \cdot X_4 \cdot (X_2 + 14) - 5.0 \leq 0$$
$$G(1) = \frac{1.524}{X_7} - 1 \leq 0$$
$$G(2) = \frac{1.524}{X_8} - 1 \leq 0$$
$$G(3) = 0.07789 \cdot X_1 - 2 \cdot \frac{X_9}{X_7} - 1 \leq 0$$
$$G(4) = \frac{7.05305 \cdot X_1^2 \cdot X_{10}}{X_2 \cdot X_8 \cdot X_9 \cdot X_{14}} - 1 \leq 0$$
$$G(5) = \frac{0.0833 \cdot X_{14}}{X_{13}} - 1 \leq 0$$
$$G(6) = 47.136 \cdot X_2^{0.333} \cdot \frac{X_{12}}{X_{10}} - 1.333 \cdot X_{13}^{2.1195} \cdot X_8 + \frac{62.08 \cdot X_{13}^{2.1195} \cdot X_8^{0.2}}{X_{10} \cdot X_{12}} - 1 \leq 0$$
$$G(7) = 0.04771 \cdot X_{10} \cdot X_8^{1.8812} \cdot X_{12}^{0.3424} - 1 \leq 0$$

$$G(8) = 0.0488 \cdot X_9 \cdot X_7^{1.893} \cdot X_{11}^{0.316} - 1 \leq 0$$

$$G(9) = 0.0099 \cdot \frac{X_1}{X_3} - 1 \leq 0$$

$$G(10) = 0.0193 \cdot \frac{X_2}{X_4} - 1 \leq 0$$

$$G(11) = 0.0298 \cdot \frac{X_1}{X_5} - 1 \leq 0$$

$$G(12) = 0.056 \cdot \frac{X_2}{X_6} - 1 \leq 0$$

$$G(13) = \frac{2}{X_9} - 1 \leq 0$$

$$G(14) = \frac{2}{X_{10}} - 1 \leq 0$$

$$G(15) = \frac{X_{12}}{X_{11}} - 1 \leq 0$$

$with\ the\ following\ boundaries$
$0.001 \leq X_i \leq 5$, i=1, …, 14

The global optimum reported in literature [200] is $W(\bar{X}) = 0.032213$, which corresponds to the design vector $\bar{X} = [0.001;\ 0.001;\ 0.001;\ 0.001;\ 0.001;\ 0.001; 1.524; 1.524; 5; 2;\ 0.001;\ 0.001;\ 0.0072934;\ 0.087556]$.

The performance of SHGWJA was benchmarked against several Mas, including:

(i) the hybrid Harmony Search (HS), hybrid Big Bang–Big Crunch (BBBC), and hybrid JAYA formulations as introduced in [158]

(ii) the Simplified Metaheuristic Optimization (SMO) algorithm [140]

(iii) the classical Grey Wolf Optimizer (GWO) [12]

(iv) classical and enhanced versions of the JAYA algorithm [158]

(v) the modified Adaptive Harmony Search (mAHS), derived from [188]

(vi) the modified BBBC with Uncorrelated Boundary Shrinkage (mBBBC-UBS), based on [189]

(vii) the Modified Sine Differential Evolution (MsinDE), derived from [190]

(viii) the Success-History based Adaptive Differential Evolution with gradient-based repair (En(L)SHADE) [193]

(ix) the Self-Adaptive Spherical Search (SASS) algorithm [196], [197]

(x) the hybrid SDDS-SABC algorithm, which integrates Split-Detect-Discard-Shrink procedures for search space partitioning with an enhanced Artificial Bee Colony (ABC) framework [197]

(xi) several Cuckoo Search (CS) variants, as documented in [200].

4.3.3 Robotics optimization

4.3.3.1 A six-variable path planning approach

This problem concerns a two-dimensional path planning formulation (see Appendix 8.3), frequently encountered in robotic motion planning and autonomous navigation domains. The objective is to determine the shortest admissible trajectory $\mathscr{C}$ that connects a specified starting point A to a target destination B, while rigorously avoiding collisions with multiple circular obstacles spatially distributed within the defined configuration space (see Figure 4.11).

The design domain is expressed in millimetric units, with the start and goal coordinates assigned as A (1,1) and B (29,29), respectively. The environment includes 7 circular obstacles, each defined by a center coordinate and an associated radius as follows: $Xp_1(6,6), \rho_{\mathrm{p1}} = 3\ mm,\ \mathrm{Xp}_2(7,24), \rho_{\mathrm{p2}} = 1\ mm, Xp_3(9,15), \rho_{\mathrm{p3}} = 3\ mm, Xp_4(16,23), \rho_{\mathrm{p4}} = 2\ mm,$ $Xp_5(20,11), \rho_{\mathrm{p5}} = 2\ mm,\ Xp_6(23,6), \rho_{\mathrm{p6}} = 1\ mm,$ and $Xp_7(24,23), \rho_{\mathrm{p7}} = 2\ mm.$

Each obstacle introduces a circular exclusion region that constrains the feasible search space, thereby increasing the geometric and computational complexity of the problem. As a result, due to the spatial distribution of the obstacles between points A and B, the optimal trajectory $\mathscr{C}$ deviates significantly from a straight line and assumes a complex, non-smooth profile.

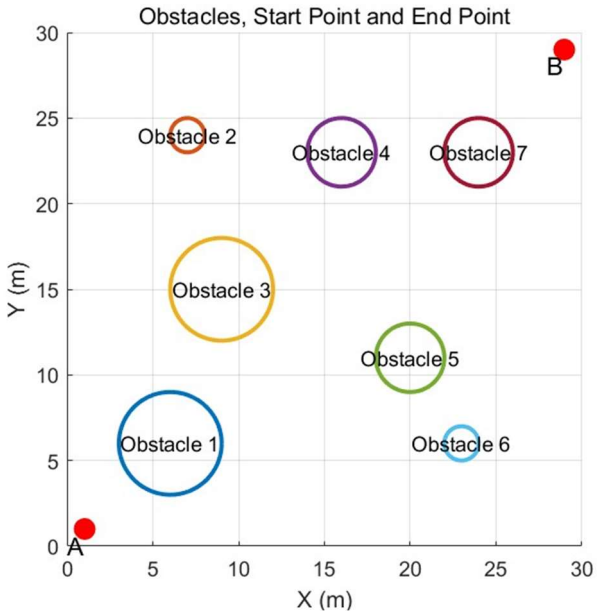


Figure 4.11- DESIGN SPACE OF THE 2D PATH PLANNING PROBLEM, WITH START (A), GOAL (B), AND CIRCULAR OBSTACLES.

Let $NBP = 3$ be the number of base points, $NDV = 2 \cdot NBP = 6$ the number of decision variables, $NOB = 7$ the number of obstacles, and $NCP = 100$ be the number of control points, with (Xc, Yc) their coordinates. The first and the last control points correspond to point

A and B, respectively. The trajectory $\mathscr{C}$ is composed of $(NCP-1)$ segments connecting these control points. Let $\bar{X}=(X_1,X_2,X_3,X_4,X_5,X_6)$ be the vector of optimization variables, where the sub vector $\overline{X_b}=(X_1,X_3,X_5)$ represents the abscissas and $\overline{Y_b}=(X_2,X_4,X_6)$ the ordinates of the three base points. Let $W(\bar{X})$ denote the length of the trajectory $\mathscr{C}$.

The mathematical model that describes this problem is formulated as follows:

$$Minimize\ W(\bar{X})=\sum_{i=1}^{NCP}\left(\sqrt{(Xc_{i+1}-Xc_i)^2+(Yc_{i+1}-Yc_i)^2}\right)+\ PENALTY \tag{4.23}$$

$$Subject\ to\ (NCP+NBP)\cdot NOB=721\ constraints:$$

$$G(i)=\frac{\rho_p(j)}{\sqrt{\left(X_c(r)-X_p(j)\right)^2+\left(Y_c(r)-Y_p(j)\right)^2}}-1<0 \qquad \text{FOR CONTROL POINTS}$$

$$G(ii)=\frac{\rho_p(j)}{\sqrt{\left(X_b(r)-X_p(j)\right)^2+\left(Y_b(r)-Y_p(j)\right)^2}}-1<0 \qquad \text{FOR BASE POINTS} \tag{4.24}$$

where i=1, ..., NCP × NOB
ii=1, ..., NBP × NOB
j= 1, 2, ..., NOB
r= 1, 2, ..., NCP
k= 1, 2, ..., NBP

$with\ the\ following\ boundaries$:

$$1\le X_1,X_2,X_3,X_4,X_5,X_6\le 29$$

The "PENALTY" term in Equation (4.23) is equals to zero if the trajectory does not collide with any of the NOB obstacles; otherwise, it aggregates the squared violations of the constraint inequalities, weighted by a factor of 1000, thereby amplifying the impact of collisions.

A comparative analysis of the SHGWJA algorithm for the path optimization problem under a 6 variables design was conducted with the main MAs:

(i) the hybrid Harmony Search (HS), hybrid Big Bang–Big Crunch (BBBC), and hybrid simulated annealing (HSA) [158]

(ii) the classical Grey Wolf Optimizer (GWO) [12]

(iii) classical and improved JAYA algorithm [158]

(iv) the modified Adaptive Harmony Search (mAHS), derived from [188]

(v) the modified BBBC with Uncorrelated Boundary Shrinkage (mBBBC-UBS), based on [189]

(vi) the Modified Sine Differential Evolution (MsinDE), derived from [190]

(vii) the dynamic differential annealed optimization (DDAO) [200].

4.3.4 Characterization of a flat composite panel subjected to axial compression

The optimization problem focuses on the inverse identification of material properties by minimizing the discrepancy between the buckling mode shape measured experimentally using projection moiré interferometry and those predicted by a finite element model simulating the buckling behaviour (see Appendix 8.2.5, [74], [202], [203], [204], [205]).

The 7 design variables are: the longitudinal Young's modulus $X_1 = E_L$, the transverse Young's modulus $X_2 = E_T$, the in-plane shear modulus $X_3 = G_{LT}$, the Poisson's ratio $X_4 = \nu_{LT}$, and the three-ply orientation variables $X_5 = \theta_0$, $X_6 = \theta_{90}$ and $X_7 = \theta_{45}$, corresponding respectively to nominal ply angles of 0°, 90° and ±45°.

Let Ω denotes the error functional quantifying the normalized discrepancy between the experimentally measured fundamental buckling mode shape $w_{norm,EXP}$, obtained via moiré interferometry, and the numerically predicted mode shape $w_{norm,FEM}$ computed from the finite element model. The comparison is performed at each measurement point p, for a total of N_{CNT} control points distributed on the panel, as shown in Figure 4.12.

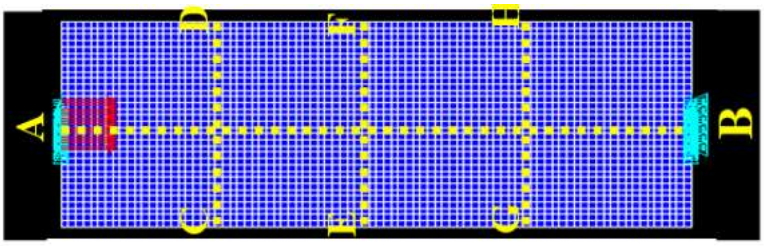


Figure 4.12- FE MODEL WITH CONTROL PATHS (AB, CD, EF, GH) USED TO EVALUATE THE ERROR FUNCTIONAL Ω THROUGH COMPARISON OF EXPERIMENTAL AND NUMERICAL DISPLACEMENTS.

The mathematical model that describes this problem is formulated as follows:

$$Minimize\ \Omega(\bar{X}) = \sqrt{\sum_{p=1}^{N_{CNT}} \left(\frac{w_{norm,FEM}^{p} - w_{norm,EXP}^{p}}{w_{norm,FEM}^{p}} \right)^2} \tag{4.25}$$

$Subject\ to\ 2\ constraints:$

$$G(1) = X_2 - X_1 < 0$$
$$G(2) = X_4 \cdot \left(\frac{X_2}{X_1}\right) - 1 < 0$$

$with\ the\ following\ boundaries$:

$$10000 \leq X_1, X_2 \leq 1000000$$
$$500 \leq X_3 \leq 20000$$

$0.001 \leq X_4 \leq 0.1$
$-50 \leq X_5 \leq 50$
$0 \leq X_6 \leq 91$
$10 \leq X_7 \leq 70$

The SHGWJA algorithm was benchmarked on this optimization problem alongside the main MAs:

(i) the hybrid fast harmony search HFHS, the hybrid fast big bang-big crunch HFBBBC and hybrid fast simulated annealing HFSA algorithms specifically developed in for material characterization inverse problems

(ii) the standard GWO and JAYA, improved JAYA [158]

(iii)) the modified Adaptive Harmony Search (mAHS), derived from [188]

(iv) the modified BBBC with Uncorrelated Boundary Shrinkage (mBBBC-UBS), based on [189]

(v) the Modified Sine Differential Evolution (MsinDE), derived from [190].

4.4 SHGWJA IMPLEMENTATION DETAILS

The SHGWJA algorithm was implemented as a standalone code in the MATLAB environment. All optimization runs were performed on a standard desktop computer equipped with 3.1 GHz AMD processor and 16 GB of RAM. To ensure statistically significant results, 20 independent runs were conducted for each test case. For the two mathematical benchmark functions (Rosenbrock and Rastrigin), the number of design variables, population size, and maximum number of iterations were selected in accordance with established literature, ensuring that all algorithms executed an equivalent number of function evaluations (see parameter settings in Section 4.2). In contrast, for the seven engineering design optimization problems, the population size and maximum number of iterations were fixed at 10 and 5000, respectively. This configuration would lead to a total computational budget of 50000 function evaluations per run, assuming full population evaluation at each iteration. However, the SHGWJA algorithm consistently achieved convergence in significantly fewer evaluations, demonstrating its computational efficiency. Table 4.5 describes the implementation parameters and computational setup described above.

Table 4.5-IMPLEMENTATION PARAMETERS AND COMPUTATIONAL SETUP

ITEM	VALUE/DESCRIPTION
PROGRAMMING ENVIRONMENT	MATLAB (STANDALONE IMPLEMENTATION)
HARDWARE	AMD 3.1 GHZ PROCESSOR, 16 GB RAM
NUMBER OF INDEPENDENT RUNS	20 PER TEST PROBLEM
DESIGN VARIABLES (MATH PROBLEMS)	AS PER LITERATURE (ROSENBROCK, RASTRIGIN)

ITEM	VALUE/DESCRIPTION
DESIGN VARIABLES (ENGINEERING PROBLEMS)	UP TO 7
POPULATION SIZE (ENGINEERING)	10
MAXIMUM ITERATIONS	5000
MAX FUNCTION EVALUATIONS	50000 (IF ALL INDIVIDUALS ARE EVALUATED IN ALL ITERATIONS)
TYPICAL CONVERGENCE	ACHIEVED WITH SIGNIFICANTLY FEWER FUNCTION EVALUATIONS

The initial populations were generated using three distinct strategies:

- Uniform sampling across the entire design space using the classical initialization Equation 4.9 for each variable X_j for "each individual" i
- Biases initialization near the lower bounds, using a perturbation of the lower bound scaled by 1/100000
- Biases initialization near the upper bounds, similarly scaled.

Some initial designs exhibited severe constraint violations, up to 1000% in the trajectory optimization problem, leading to objective function values up to two orders of magnitude higher than the known optimum. These cases were intentionally introduced to assess SHGWJA's capability to rapidly converge toward the feasible and optimal region of the search space. Table 4.6 summarizes the different strategies adopted for initializing the population in the optimization process, including uniform sampling based on Equation 4.9 and biased initializations near the design variable bounds.

TABLE 4.6-POPULATION INITIALIZATION STRATEGIES

INITIALIZATION TYPE	DESCRIPTION
UNIFORM ACROSS DESIGN SPACE	GENERATED USING THE CLASSICAL INITIALIZATION EQUATION 4. 9
NEAR LOWER BOUNDS	(EQUATION 4. 9)/100000
NEAR UPPER BOUNDS	(EQUATION 4. 9)/100000
INITIAL CONSTRAINT VIOLATIONS	UP TO 1000% (E.G., TRAJECTORY OPTIMIZATION PROBLEM)
INITIAL COST GAP FROM OPTIMUM	UP TO 100× THE OPTIMUM VALUE (TO TEST CONVERGENCE ROBUSTNESS)

Regarding constraint handling, each algorithm used the strategy specified in its original reference. In most cases, a static penalty function approach was adopted, where a constant penalty factor was applied throughout the optimization process. For SHGWJA and its comparators, the penalty function defined by Equations 4.10 and 4.11 was consistently used. A sensitivity analysis was also performed by varying the penalty coefficient in equation 4.10

across several orders of magnitude, as described in Table 4.7. This was intended to assess the robustness of each algorithm under different levels of constraint penalization.

TABLE 4.7-PENALTY FACTOR SENSITIVITY ANALYSIS

PENALTY FACTOR	SHGWJA PERFORMANCE
10^0	STABLE CONVERGENCE
10^1	STABLE CONVERGENCE
10^2	NO PERFORMANCE DEGRADATION
10^3	NO PERFORMANCE DEGRADATION
10^4	NO PERFORMANCE DEGRADATION
10^5	NO PERFORMANCE DEGRADATION
10^6	MAXIMUM DEVIATION FROM OPTIMUM < 0.001%

Notably, across all test problems, the maximum deviation from the known optimal objective value obtained by SHGWJA, regarding the penalty factor, never exceeded 0.001%, highlighting the method's stability and effectiveness.

4.5 EXPERIMENTAL RESULTS WITH SHGWJA

4.5.1. Mathematical optimization: Rosenbrock and Rastrigin

SHGWJA [170] demonstrated clearly superior performance compared to 31 other metaheuristic algorithms in solving the Rosenbrock problem, as shown in Table 4.8. Across all tested combinations of problem dimensionality (NDV) and population size (NPOP), SHGWJA consistently achieved the lowest optimized cost values, with best results never exceeding $5.179 \cdot 10^{-13}$. The standard deviation on optimized cost (reported as ST Dev in the table 4.8) also remained remarkably low, always below $2.669 \cdot 10^{-13}$.

As detailed in Table 4.8, SHGWJA's performance was practically insensitive to the choice of NDV and NPOP. For commonly used configurations in the literature, such as (NDV=30; NPOP=30) and (NDV=30; NPOP=50), the algorithm reached optimized costs as low as 10^{-15}, consistently converging within the 15000 function evaluations (NFE) limit, one of the lowest computational budgets among all compared methods.

Furthermore, when the convergence threshold ε_{conv} was reduced to 10^{-15} and the computational budget was increased to 30000 function evaluations (matching that of LCA and significantly lower than the 50000 required by MOA), SHGWJA achieved even better results. As reported in Table 4.8, both the average optimized cost and the standard deviation dropped to 10^{-28}, outperforming LCA and MOA even under more constrained computational resources.

TABLE 4.8- OPTIMIZATION RESULTS FOR THE UNIMODAL ROSENBROCK'S FUNCTION

Algorithm	*NDV*	N_{POP}	Best	Average	Worst	ST Dev	NFE
SHGWJA - Present	10	10	$5.179 \cdot 10^{-13}$	$1.044 \cdot 10^{-12}$	$2.291 \cdot 10^{-12}$	$1.117 \cdot 10^{-12}$	15000
	30	10	$4.789 \cdot 10^{-13}$	$2.832 \cdot 10^{-12}$	$3.706 \cdot 10^{-12}$	$1.029 \cdot 10^{-12}$	15000

	50	10	$4.998 \cdot 10^{-13}$	$3.959 \cdot 10^{-12}$	$6.803 \cdot 10^{-12}$	$2.669 \cdot 10^{-12}$	15000
	100	10	$8.460 \cdot 10^{-15}$	$1.032 \cdot 10^{-14}$	$1.239 \cdot 10^{-14}$	$1.606 \cdot 10^{-14}$	15000
	30	30	$1.001 \cdot 10^{-15}$	$3.846 \cdot 10^{-14}$	$7.309 \cdot 10^{-14}$	$7.545 \cdot 10^{-14}$	14187±1319
	100	30	$4.189 \cdot 10^{-13}$	$4.213 \cdot 10^{-13}$	$2.423 \cdot 10^{-12}$	$9.620 \cdot 10^{-13}$	15000
	500	30	$2.284 \cdot 10^{-14}$	$1.018 \cdot 10^{-13}$	$2.030 \cdot 10^{-13}$	$7.813 \cdot 10^{-13}$	15000
	1000	30	$7.028 \cdot 10^{-14}$	$5.407 \cdot 10^{-13}$	$6.704 \cdot 10^{-13}$	$3.875 \cdot 10^{-13}$	15000
	30	50	$1.006 \cdot 10^{-15}$	$1.072 \cdot 10^{-15}$	$1.710 \cdot 10^{-15}$	$1.563 \cdot 10^{-16}$	14233±861
	50	50	$4.541 \cdot 10^{-13}$	$3.144 \cdot 10^{-12}$	$6.331 \cdot 10^{-12}$	$1.336 \cdot 10^{-12}$	15000
	100	50	$1.000 \cdot 10^{-15}$	$1.092 \cdot 10^{-14}$	$9.215 \cdot 10^{-14}$	$2.120 \cdot 10^{-14}$	13997±1190
MGWO-III [110]	30	30	N/A	16.3	N/A	12.5	30000
	50	30	N/A	35.0	N/A	19.2	30000
	100	30	N/A	66.1	N/A	41.8	30000
CJAYA-SQP [160]	30	40	N/A	0.312	N/A	1.32	80000
	100	40	N/A	13.0	N/A	33.9	80000
SJAYA [176]	30	100	0.0015	25.453	N/A	28.876	30000
SAMP-JAYA [18]	30	Variab. Subpopul.	N/A	$1.801 \cdot 10^{-9}$	N/A	$1.844 \cdot 10^{-8}$	50000
Hybrid HS [158]	100	For all algorithms	N/A	$1.197 \cdot 10^{-11}$	N/A	Aver. of optim.	9509±420
Hybrid BBBC [158]	100	of [158]	N/A	$1.369 \cdot 10^{-11}$	N/A	runs of [158]	9865±680
HFSA [158]	100	20;200;500	N/A	$1.480 \cdot 10^{-11}$	N/A	$7.4 \cdot 10^{-12}$	9896±1406
EO [101]	30	30	N/A	25.3233	N/A	0.16958	15000*
	50	30	N/A	45.55	N/A	0.4772	15000*
	100	30	N/A	96.73	N/A	1.109	15000*
	500	30	N/A	497.6	N/A	0.1441	15000*
mEO [157]	30	30	N/A	0.07952	N/A	0.05427	15000*
	50	30	N/A	0.2291	N/A	0.2360	15000*
	100	30	N/A	0.4783	N/A	0.4249	15000*
	500	30	N/A	6.2	N/A	5.142	15000*
LSO [102]	100	20	N/A	97.0	N/A	0.618	50000
RUN [112]	30	50	22.9	24.5	N/A	1.04	25000*
AOA [114]	30	30	N/A	24.9	N/A	0.364	15000*
	100	30	N/A	97.9	N/A	0.655	15000*
	500	30	N/A	497	N/A	0.532	15000*
	1000	30	N/A	921	N/A	969	15000*
IAOA [161]	30	30	26.470	27.941	N/A	0.2065	15000*
	500	30	495.848	496.902	N/A	0.1511	15000*
	1000	30	996.137	997.156	N/A	0.1662	15000*
MOA [120]	N/A	50	0	0	0	0	50000*
PEOA [179]	N/A	50	$4.577 \cdot 10^{-7}$	$4.425 \cdot 10^{-4}$	$3.201 \cdot 10^{-3}$	$9.18 \cdot 10^{-4}$	50000*
LCA [121]	30	30	0	0	0	0	30000*
SLOA [136]	30	50	N/A	24.25997	N/A	$1.51 \cdot 10^{-14}$	50000*
GTO [146]	30	30	N/A	$2.86 \cdot 10^{-8}$	N/A	$5.221 \cdot 10^{-9}$	30000*
	100	30	N/A	$2.53 \cdot 10^{-8}$	N/A	$4.619 \cdot 10^{-9}$	30000*
	500	30	N/A	$5.1 \cdot 10^{-7}$	N/A	$9.311 \cdot 10^{-8}$	30000*
	1000	30	N/A	$3.57 \cdot 10^{-6}$	N/A	$6.518 \cdot 10^{-7}$	30000*
MPA [145]	30	50	25.943	26.618	27.502	0.3975	25000*
	50	50	43.803	45.117	47.741	0.951	25000*
IMPA [166]	50	50	N/A	$2.298 \cdot 10^{-5}$	N/A	$2.461 \cdot 10^{-5}$	50000*
EGTO [167]	30	50	$4.29 \cdot 10^{-7}$	9.2257	24.696	11.601	25000*
	50	50	$3.99 \cdot 10^{-6}$	10.7996	44.761	19.178	25000*

SaDN [159]	100	50	N/A	98.9	N/A	$3.01 \cdot 10^{-2}$	25000*
MSCA [164]	30	50	N/A	25.5	N/A	0.166	25000*
hSM-SA [165]	30	30	$1.146 \cdot 10^{-2}$	11.778	28.355	$7.059 \cdot 10^{-3}$	15000*
ALC-PSO [180]	30	20	$3.729 \cdot 10^{-7}$	7.613	N/A	6.658	6406
GPSO [180]	30	20	$2.362 \cdot 10^{-4}$	11.657	N/A	14.999	5348
NDWPSO [181]	50	70	N/A	46.2	N/A	0.65	14000*
CMA-ES [182]	30	30	1.777	27.05	3.812	0.6222	30000*
	50	50	N/A	52.11	N/A	23.15	25000*
LSHADE [183]	30	Lin. Decr. from 18*NDV*=540 to 4	3.970	46.38	84.12	29.30	30000
LSHADE-cnEpSin [184]	50	Lin. Decr. from 18*NDV*=900 to 4	N/A	59.253	N/A	29.012	25000
LSHADE-SPACMA [185]	30	Lin. Decr. from 18*NDV*=540 to 4	N/A	28.8255	N/A	0.8242	15000

*Determined as the product between the population size and the limit number of optimization iterations

The results confirm that SHGWJA was the most effective optimizer overall. Only the Mother Optimization Algorithm (MOA) [120] and the Learning Cooking Algorithm (LCA) [121] consistently reached the global optimum of 0, but at the cost of 50000 and 30000 function evaluations, respectively. By contrast, under the (NDV = 30; NPOP = 30) setting used by LCA, SHGWJA achieved best/average/worst optimized costs and standard deviations within at most 15000 evaluations, ranging from $1.001 \cdot 10^{-15}$ to $7.545 \cdot 10^{-14}$, values numerically indistinguishable from 0. Similarly, with NPOP = 50 as in MOA, SHGWJA reached best results of 10^{-15} for NDV = 30 and 100, with averages between $1.1 \cdot 10^{-15}$ and $1.1 \cdot 10^{-14}$. Although the convergence curves reported in [120,121] are of insufficient resolution for direct comparison, SHGWJA consistently attained 10^{-7} within 1000-1500 evaluations, consistent with the trends observed in those studies.

The algorithm also compared favourably with hybrid metaheuristics such as Hybrid Harmony Search (HS), Hybrid Big Bang-Big Crunch (BBBC), and Hybrid Fast Simulated Annealing (HFSA) [158]. As reported in Table 4.8, SHGWJA achieved an average optimized cost of $1.1 \cdot 10^{-12}$, compared to values consistently above $1.197 \cdot 10^{-11}$ for Hybrid HS/BBBC/SA, with a smaller average deviation ($1.478 \cdot 10^{-12}$ vs. $7.4 \cdot 10^{-12}$). Despite being executed with a stricter convergence threshold ($\varepsilon_{conv} = 10^{-15}$), the hybrid algorithms [158] stagnated at ~80% of the search due to reliance on gradient-based trial solutions, whereas SHGWJA continued improving until the final iterations.

SHGWJA also outperformed two Particle Swarm Optimization variants: GPSO and ALC-PSO [180]. Although these completed their best runs within 6500 evaluations, they did not surpass $3.7 \cdot 10^{-7}$, while SHGWJA reached values between $1.001 \cdot 10^{-15}$ and $4.789 \cdot 10^{-13}$ under comparable settings (NDV=30; NPOP=30±10). Moreover, the PSO variants produced unstable results, with average costs between 7.6-11.7 and deviations between 6.7-15, in contrast to SHGWJA's worst-case deviation of only $2.832 \cdot 10^{-12}$.

Restricting the analysis to the classical setting (NDV = 30; NPOP = 30±10) and a limited budget of 15000 evaluations, the ranking of algorithms by average optimized cost was: SHGWJA, mEO, ALC-PSO, GPSO, hSM-SA, AOA, EO, IAOA, LSHADE-SPACMA. With

30000 evaluations and NDV, NPOP ≤ 50, the ranking shifted to: SHGWJA, MOA, LCA, SAMP-JAYA, GTO, IMPA, PEOA, mEO, ALC-PSO, EGTO, GPSO, hSM-SA, MGWO-III, SLOA, RUN, AOA, EO, MPA, CMA-ES, IAOA, LSHADE-SPACMA, LSHADE, LSHADE-CnEpSin. Increasing NPOP to 70 and the budget to 80000 evaluations allowed CJAYA-SQP [160] to rank 9th and NDWPSO [181] to rank just before LSHADE [183]. Interestingly, most JAYA variants (except SJAYA [176]) consistently ranked in the top 10, while high-performance algorithms such as LSHADE variants [183] [184] [185] and the basic Marine Predators Algorithm (MPA) [145] were less effective on the Rosenbrock problem. This confirms the benefit of JAYA as a base component for hybrid metaheuristics.

Another distinctive strength of SHGWJA is its robustness to parameter settings, unlike most algorithms whose performance declines sharply with increasing dimensionality. With NPOP = 30 and NDV extended up to 1000, average costs rose dramatically for most competitors, reaching nearly 1000. Only GTO [146] and mEO [157] contained this increase to $3.57 \cdot 10^{-6}$ (NDV = 1000) and 6.2 (NDV = 500), respectively. For NDV = 100, SHGWJA ranked first, followed by Hybrid HS/BBBC/SA and GTO ($\leq 2.53 \cdot 10^{-8}$), mEO (< 0.5), CJAYA-SQP, MGWO-III [110], EO, LSO [156], and SaDN [159], which obtained higher costs (13-99).

In the Rastrigin problem (Table 4.9), SHGWJA again demonstrated excellent performance, showing insensitivity to NDV and NPOP. Unlike the Rosenbrock case, most competitors (e.g., MGWO-III [110], GTO [146], IMPA [166], IOAO [161]) converged to the global optimum of 0, often with null averages and deviations. Still, SHGWJA produced superior results, reaching $1.776 \cdot 10^{-15}$ in 8 cases, $2.842 \cdot 10^{-14}$ in 1 case, and values between $1.492 \cdot 10^{-13}$ and $2.934 \cdot 10^{-13}$ in the others, all close to 0. It also achieved zero standard deviation in 3 cases, and practically null deviations ($5.13 \cdot 10^{-16}$-$9.175 \cdot 10^{-16}$) in the remaining ones. Its overall averages ($3.171 \cdot 10^{-13}$ for cost and $6.606 \cdot 10^{-13}$ for deviation) were 1-2 orders of magnitude lower than those of Hybrid HS/BBBC/SA [158], which again stagnated before the end of optimization.

Finally, with $\varepsilon_{conv} = 10^{-15}$ and a slightly increased budget of 25000 evaluations, SHGWJA achieved costs and deviations on the order of 10^{-28}, outperforming algorithms such as MPA, EGTO, SaDN, and MSCA (25000 evaluations), MGWO-III, LCA, and GTO (30000), and significantly exceeding LSO, MOA, PEOA, SLOA, IMPA (50000), and CJAYA-SQP (80000).

TABLE 4.9- OPTIMIZATION RESULTS FOR THE MULTIMODAL RASTRIGIN'S FUNCTION

Algorithm	*NDV*	N_{POP}	Best	Average	Worst	ST Dev	NFE
SHGWJA - Present	10	10	$1.776 \cdot 10^{-15}$	$2.186 \cdot 10^{-15}$	$3.553 \cdot 10^{-15}$	$7.793 \cdot 10^{-16}$	13770±978
	30	10	$1.776 \cdot 10^{-15}$	$1.776 \cdot 10^{-15}$	$1.776 \cdot 10^{-15}$	0	13400±1253
	50	10	$1.776 \cdot 10^{-15}$	$2.265 \cdot 10^{-15}$	$3.553 \cdot 10^{-15}$	$9.175 \cdot 10^{-16}$	13960±865
	100	10	$2.842 \cdot 10^{-14}$	$3.908 \cdot 10^{-14}$	$8.882 \cdot 10^{-14}$	$3.224 \cdot 10^{-14}$	13989±288
	30	30	$1.776 \cdot 10^{-15}$	$2.131 \cdot 10^{-15}$	$3.553 \cdot 10^{-15}$	$7.493 \cdot 10^{-16}$	13736±1123
	100	30	$1.776 \cdot 10^{-15}$	$1.973 \cdot 10^{-15}$	$5.329 \cdot 10^{-15}$	$8.375 \cdot 10^{-16}$	13803±1316
	500	30	$1.492 \cdot 10^{-13}$	$2.065 \cdot 10^{-12}$	$1.114 \cdot 10^{-11}$	$3.771 \cdot 10^{-12}$	15000
	1000	30	$2.934 \cdot 10^{-13}$	$1.368 \cdot 10^{-12}$	$7.212 \cdot 10^{-12}$	$3.459 \cdot 10^{-12}$	15000
	30	50	$1.776 \cdot 10^{-15}$	$1.776 \cdot 10^{-15}$	$1.776 \cdot 10^{-15}$	0	13250±1125
	50	50	$1.776 \cdot 10^{-15}$	$1.924 \cdot 10^{-15}$	$3.553 \cdot 10^{-15}$	$5.130 \cdot 10^{-16}$	13630±724
	100	50	$1.776 \cdot 10^{-15}$	$1.776 \cdot 10^{-15}$	$1.776 \cdot 10^{-15}$	0	13578±1397
CJAYA-SQP [160]	30	40	0	0	0	0	80000
	30	100	0	0	0	0	80000

SAMP-JAYA [18]	30	Variab. Subpopul.	N/A	68.516	N/A	32.543	50000
I-JAYA [178]	30	100	$1.\cdot10^{-10}$	N/A	N/A	N/A	15700
MGWO-III [110]	30	30	0	0	0	0	30000*
	50	30	0	0	0	0	30000*
	100	30	0	0	0	0	30000*
Hybrid HS [158]	100	For all algorithms	N/A	$2.988\cdot10^{-12}$	N/A	Aver. of optim.	10877±538
Hybrid BBBC [158]	100	of [158]	N/A	$1.606\cdot10^{-11}$	N/A	runs of [158]	9703±1096
HFSA [158]	100	20;200;500	N/A	$1.192\cdot10^{-11}$	N/A	$7.3\cdot10^{-12}$	9875±677
EO [101]	30	30	0	0	0	0	15000*
	50	30	0	0	0	0	15000*
	100	30	0	0	0	0	15000*
	500	30	N/A	$9.095\cdot10^{-14}$	N/A	$2.775\cdot10^{-13}$	15000*
mEO [157]	30	30	0	0	0	0	15000*
	50	30	0	0	0	0	15000*
	100	30	0	0	0	0	15000*
	500	30	0	0	0	0	15000*
LSO [102]	100	20	0	0	0	0	50000
AOA [114]	30	30	N/A	$3.42\cdot10^{-7}$	N/A	$3.42\cdot10^{-7}$	15000*
	100	30	N/A	$8.46\cdot10^{-6}$	N/A	$9.24\cdot10^{-6}$	15000*
	500	30	N/A	$1.58\cdot10^{-2}$	N/A	$1.58\cdot10^{-2}$	15000*
	1000	30	N/A	$3.99\cdot10^{-2}$	N/A	$2.64\cdot10^{-2}$	15000*
IAOA [161]	30	30	0	0	0	0	15000*
MOA [120]	N/A	50	0	0	0	0	50000*
PEOA [179]	N/A	50	0	0	0	0	50000*
LCA [121]	30	30	0	0	0	0	30000*
SLOA [136]	30	50	0	0	0	0	50000*
GTO [146]	30	30	0	0	0	0	30000*
MPA [145]	30	50	$4.32\cdot10^{-7}$	$1.43\cdot10^{-4}$	$1.393\cdot10^{-3}$	$3.07\cdot10^{-4}$	25000*
	50	50	0	0	0	0	25000*
IMPA [166]	50	50	0	0	0	0	50000*
EGTO [167]	30	50	0	0	0	0	25000*
	50	50	0	0	0	0	25000*
SaDN [159]	30	50	0	0	0	0	25000*
MSCA [164]	30	50	0	0	0	0	25000*
hSM-SA [165]	30	30	0	0	0	0	15000*
ALC-PSO [180]	30	20	$7.105\cdot10^{-15}$	$2.528\cdot10^{-14}$	N/A	$1.376\cdot10^{-14}$	74206
NDWPSO [181]	50	70	0	0	0	0	14000*
CMA-ES [182]	30	30	141.952	157.610	165.209	7.324	30000*
	30	50	N/A	30.91	N/A	5.383	25000*
LSHADE [183]	30	Lin. Decr. from 18*NDV*=540 to 4	$1.275\cdot10^{-6}$	$4.983\cdot10^{-2}$	0.99504	0.2225	30000
LSHADE-cnEpSin [184]	50	Lin. Decr. from 18*NDV*=900 to 4	N/A	76.052	N/A	9.15	25000
LSHADE-SPACMA [185]	30	Lin. Decr. from 18*NDV*=540 to 4	N/A	67.542	N/A	10.016	15000

*Determined as the product between the population size and the limit number of optimization iterations

SHGWJA consistently required on average fewer than 14000 function evaluations to complete the optimization process for problems of up to 100 dimensions, i.e., less than the computational cost of NDWPSO, the fastest algorithm reported to reach the 0-target solution with 0 standard deviation. The fastest runs of SHGWJA converged to $1.776 \cdot 10^{-15}$ within 12877 evaluations, fewer than the 14000 evaluations needed by NDWPSO for its best setting (NDV = 50; NPOP = 70). Detailed convergence analysis further revealed that SHGWJA reliably reduced the cost to 10^{-7} within approximately 1000 evaluations, closely matching the behaviour of NDWPSO.

The I-JAYA algorithm (enhancing classical JAYA with fuzzy clustering competitive learning, experience learning, and Cauchy mutation) also showed competitive performance, reducing the cost function to 10^{-10} within 15,700 evaluations [178]. Likewise, ALC-PSO achieved a final optimized cost of $7.105 \cdot 10^{-15}$, but at a substantially higher computational expense, requiring 74206 evaluations, nearly five times more than SHGWJA (≤15000).

Given that SHGWJA practically converged to the target optimum cost of 0, with null or near-null averages and standard deviations across all settings and within the lowest number of evaluations, it can be regarded as the most effective optimizer also for the Rastrigin problem. By contrast, the metaheuristics listed in Table 4.9 that failed to reach both 0 average cost and 0 deviation within 10^{-7} occupied the lowest seven positions in the ranking: AOA, MPA, LSHADE, LSHADE-SPACMA, SAMP-JAYA, LSHADE-CnEpSin, and CMA-ES. All were tested with NDV = 30, except LSHADE-SPACMA, which was evaluated with NDV = 50.

4.5.2. Engineering design optimization

4.5.2.1 Concrete gravity dam

Table 4.10 reports the optimization results for the concrete gravity dam design problem, including the average number of function evaluations (NFEs), their standard deviations, and the fastest run for SHGWJA and selected competitors.

The Multi-Level Cross Entropy Optimizer (MCEO) [186] and the Flying Squirrel Optimizer (FSO) [187] reduced the dam's concrete volume from the original 10502.1 m³ to 7448.774 m³. The hybrid Harmony Search, Big Bang–Big Crunch and Simulated Annealing algorithms developed in [158] achieved further improvements, obtaining a volume of about 6831 m³, comparable to that of Improved JAYA [158] (a JAYA variant including line search). However, hybrid HS/BBBC/SA required fewer evaluations and converged to feasible designs, while Improved JAYA's solution violated constraints by 4.06%. In contrast, the solutions reported for MCEO and FSO presented violations of up to 32%, as the height constraint $X_2+X_4+X_6 =$ 150 m (see Section 4.3.2.1) was not enforced in [186] [187].

The proposed SHGWJA clearly outperformed all competitors, reducing the dam's volume by 2.68% relative to the best hybrid methods in [158], achieving an optimal volume of 6647.874 m³ with a negligible constraint violation of only $1.284 \cdot 10^{-3}$ %. Moreover, SHGWJA required substantially fewer evaluations: an average of 10388 analyses, approximately 50% fewer than the hybrids of [158] (14560-16200 evaluations) and 70% fewer than MCEO (35000 evaluations). Although FSO was reported in [187] to converge faster than PSO and genetic algorithms, its computational cost remained comparable to MCEO.

For comparison, standard JAYA also converged to the same optimal volume (6647.874 m³) but required 25000 evaluations, i.e., 2.5 times the cost of SHGWJA on average and nearly four times that of SHGWJA's fastest run (25000 vs. 6584 evaluations). Similarly, standard GWO obtained a slightly worse solution (6654.927 m³) with a higher constraint violation ($7.209 \cdot 10^{-3}$ %) and at a much higher computational cost (50000 evaluations, nearly five times SHGWJA's average). These results confirm the efficiency and validity of the hybrid search scheme implemented in SHGWJA.

TABLE 4.10- OPTIMIZATION RESULTS OBTAINED FOR THE CONCRETE GRAVITY DAM DESIGN PROBLEM.

Algorithm	Best (m^3)	Average (m^3)	Worst (m^3)	ST Dev (m^3)	NFE	Constraint violation (%)
SHGWJA (Present)	6647.874	6647.874	6647.874	0	10388±3628 6584	$1.284 \cdot 10^{-3}$
Standard GWO	6654.927	6752.644	6841.402	97.723	50000	$7.209 \cdot 10^{-3}$
Standard JAYA	6647.874	6722.392	6836.902	95.216	25000	$1.284 \cdot 10^{-3}$
Improved JAYA [158]	6833.391	6835.337	6839.574	1.1078	23662±8263 19531	4.06
Hybrid HS [158]	6830.976	6831.519	6832.542	0.7349	16193±965 16703	$3.398 \cdot 10^{-5}$
Hybrid BBBC [158]	6830.235	6830.998	6831.732	0.6248	15846±948 15487	$7.284 \cdot 10^{-6}$
HFSA [158]	6832.164	6833.612	6834.344	1.0277	14560±184 14447	$1.396 \cdot 10^{-5}$
mAHS (der. From [188])	6658.024	6768.352	6848.309	96.455	34159±6898 36310	$1.311 \cdot 10^{-3}$
mBBBC–UBS (der. From [189])	6656.341	6776.125	6845.265	93.866	31690±9902 32004	$3.376 \cdot 10^{-3}$
MsinDE (der. From [190])	6652.513	6771.478	6851.634	100.189	35208±7169 37545	$3.653 \cdot 10^{-3}$
MCEO [186]	7448.74	N/A	N/A	$3.274 \cdot 10^{-2}$	35000	32
FSO [187]	7448.74	N/A	N/A	N/A	N/A	32

The Modified Sinusoidal Differential Evolution (MsinDE) [190], Modified Big Bang-Big Crunch with Upper Bound Strategy (mBBBC-UBS) [189], and Modified Adaptive Harmony Search (mAHS)[188] ranked 3rd, 5th, and 6th, respectively, with optimized volumes between 6652.513 and 6658.024 m³, very close to the global optimum obtained by SHGWJA and standard JAYA (6647.874 m³). Their performance was largely attributable to the elitist strategy$W(\bar{X}_{i,tr}) \leq 1.1 \cdot W(\bar{X}_{opt})$, implemented also in SHGWJA. This mechanism enabled all three algorithms to improve upon the best results in the literature and substantially narrow the performance gap with the leading optimizer. Indeed, the solutions obtained by mAHS, mBBBC-UBS, and MsinDE were at most 0.15% larger than SHGWJA's optimum (i.e., a maximum gap of 10.150 m³), compared to the 0.89% deviation observed for the original algorithms with respect to hybrid BBBC, the best method in [158].

The optimized designs produced by mAHS, mBBBC-UBS, and MsinDE were also practically feasible, with constraint violations of $1.311 \cdot 10^{-3}$ %, $3.376 \cdot 10^{-3}$ %, and $3.653 \cdot 10^{-3}$ %, respectively, all lower than those reported in [158] ($1.437 \cdot 10^{-3}$ %, $6.733 \cdot 10^{-3}$ %, and $6.598 \cdot 10^{-3}$ %). In terms of computational effort, mAHS saved approximately 10,000 evaluations compared to [158]. Nevertheless, the three modified algorithms still required on average more than three times the function evaluations of SHGWJA (31,690–35,208 vs. only 10,388), and their best runs required 4.9–5.7 times more evaluations (32,004–37,545 vs. 6584).

By contrast, SHGWJA achieved a 100% success rate, converging in every independent run to the same optimal solution of 6647.874 m³, with zero standard deviation in optimized cost. No other algorithm in Table 4.10 achieved this consistency; none improved upon the best design reported in [93] (6830.235 m³ for hybrid BBBC) across all runs. The standard deviation of NFEs for SHGWJA was about 35% of its average, further confirming the algorithm's robustness and reliability.

Further insights into the convergence behaviour and robustness of SHGWJA can be drawn from Figure 4.13, which compares its optimization history with that of competing algorithms. The figure reports both the best and average runs of SHGWJA, which coincide since the algorithm consistently converged to the global optimum of 6647.874 m³ in all trials. For clarity, the plot is restricted to the first 18000 function evaluations.

At the start of the optimization, the best individuals across all algorithms exhibited cost function values between 13265.397 m³ and 17351.728 m³, substantially above the global optimum. SHGWJA demonstrated a superior convergence trend from the very beginning of the search. In its fastest run (completed in 6584 evaluations), SHGWJA produced a feasible intermediate design only 0.933% larger than the optimum, while at approximately 69% of the optimization process, the algorithm already generated a design within 1% of the global optimum.

The mBBBC-UBS algorithm was the only competitor showing comparable convergence speed during the very early phase (up to 630 evaluations) and in the interval between 5400 and 6600 evaluations, though it did not maintain this pace over the full search process.

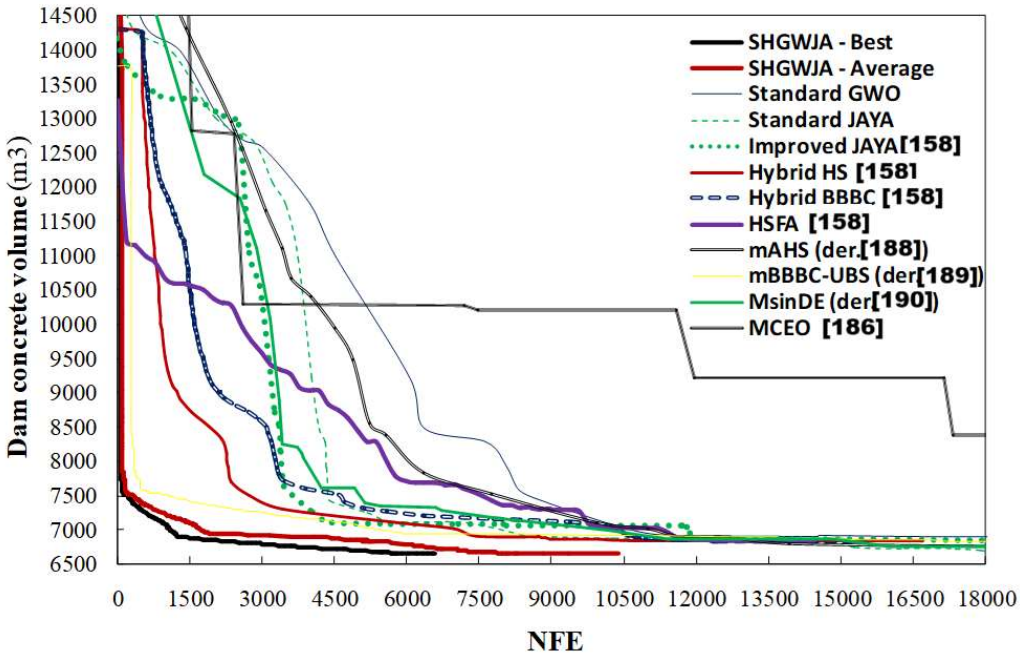


Figure 4.13- COMPARISON OF CONVERGENCE CURVES OBTAINED IN THE CONCRETE DAM PROBLEM.

Table 4.11 lists the optimized designs obtained by SHGWJA and its main competitors.

SHGWJA and standard JAYA designs coincided as these algorithms converged to the same optimal solution and were just slightly different from the optimized designs of standard GWO and MsinDE.

The new optimal solution yielding 6647.874 m^3 concrete volume was very similar to the optimal solutions of the algorithms of Ref. [158]: all variables changed by at most 5% except X_3 that was fixed to its lower bound by SHGWJA or close to the lower bound by GWO and MsinDE.

TABLE 4.11- OPTIMIZED DESIGNS FOR THE CONCRETE GRAVITY DAM PROBLEM.

Variables (m)	SHGWJA	GWO	MsinDE (der. from [124]	Hybrid HS [158]	Hybrid BBBC [158]	HFSA [158]	Impr. JA [158]	MCEO/FSO [186] [187]	Existing design
X_1	5.0000	5.0070	5.0100	5.0531	5.1417	5.0606	5.0000	5.1678	4
X_2	50.000	50.000	50.000	47.093	49.188	48.448	47.992	25.1978	42
X_3	1.0000	1.1671	1.1471	4.0802	4.2062	4.0008	4.0000	4.9556	4.2
X_4	69.799	69.808	69.801	70.771	68.177	66.920	72.007	55.7811	50
X_5	18.179	17.955	17.945	16.226	15.521	12.392	19.939	16.0874	12.5
X_6	30.000	30.001	30.001	32.136	32.627	34.860	30.000	21.0009	58

X_7	40.000	40.000	40.000	39.772	39.659	39.943	37.427	29.9036	23.2
X_8	99.787	99.799	99.811	103.56	102.76	100.48	102.01	120.169	140
X_9	80.000	80.018	80.015	81.147	82.235	87.756	80.000	85.382	105

Figure 4.14 compares the dam's optimized shapes (coordinates are expressed in meters) found by SHGWJA and MsinDE with the dam configurations reported in [158] [186] [187].

Since standard JAYA found the same optimum design as SHGWJA, while standard GWO, mAHS, mBBBC-UBS and MsinDE obtained very similar configurations to SHGWJA, only the plot relative to MsinDE (the best optimizer after SHGWJA and standard JAYA) is shown in the figure for the sake of clarity.

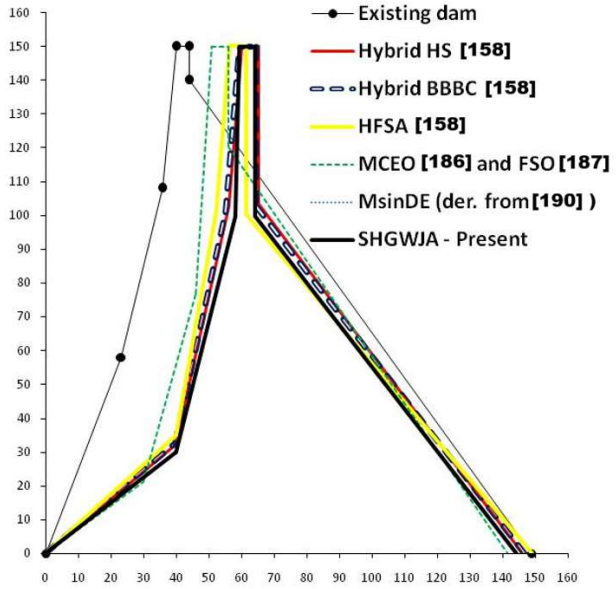


Figure 4.14- COMPARISON OF CONCRETE DAM OPTIMIZED SHAPES OBTAINED BY DIFFERENT ALGORITHMS.

Remarkably, the dam safety factors against sliding and overturning corresponding to the SHGWJA's optimal solution became 6.318 and 3.515, respectively, vs. their counterpart values of 6.213 and 3.453 quoted in m^3 to 6647.874 to m^3 not only resulted in a more convenient design also allowed a higher level of structural safety to be achieved. This proves the suitability of the proposed SHGWJA algorithm for civil engineering design problems.

4.5.2.2. Tension/compression spring

Table 4.12 reports the optimization results for the spring design problem, comparing SHGWJA with its competitors. For each algorithm, the table presents the best, average, and worst optimized costs, the standard deviation across independent runs, the number of function evaluations (NFE), and the corresponding optimal design parameters. The standard deviation

values reported for the reference algorithms are reproduced exactly as provided in the original literature.

TABLE 4.12- OPTIMIZATION RESULTS FOR THE TENSION/COMPRESSION SPRING DESIGN PROBLEM.

Algorithm	Best	Average	Worst	ST Dev	NFE	X_1 (in)	X_2 (in)	X_3
SHGWJA - Present	0.0126665	0.012682	0.012692	$1.11 \cdot 10^{-5}$	6347±2246♣	0.0514648	0.35134343	11.611476
GGWO [171]	0.0126652	0.0127	0.0127	0.0000	22640	0.0516	0.3540	11.4515
MGWO-III [110]	0.0126662	0.012679	0.012721	$1.4 \cdot 10^{-5}$	30000*	0.051658	0.355975	11.33351
EHRJAYA [163]	0.012665	0.012716	0.012869	$5.24 \cdot 10^{-5}$	2000	0.0517098	0.357218	11.2597
SAMP-JAYA [18]	0.012665	0.012714	0.013193	$9.252 \cdot 10^{-5}$	6861 (aver)	N/A	N/A	N/A
EJAYA [[177]	0.012665	0.012668	0.012687	$4.633 \cdot 10^{-6}$	15000	0.0517431	0.3580205	11.213015
EO [101]	0.012666	0.013017	0.013997	$3.91 \cdot 10^{-4}$	15000*	0.0516199	0.35505438	11.387968
LSO [102]	0.0126652	N/A	N/A	N/A	5000**	0.051689	0.35671	11.290
AOA [114]	0.012124	N/A	N/A	N/A	15000*	0.05	0.349809	11.8637
IAOA [161]	0.012018	N/A	N/A	N/A	15000*	0.0500825	0.363061	11.197508
MOA [120]	0.012665	0.012665	0.012665	$9.85 \cdot 10^{-19}$	N/A	N/A	N/A	N/A
PEOA [179]	0.012667	0.01268	0.01272	$2 \cdot 10^{-5}$	30000*	0.05161	0.35472	11.4068
SO [135]	0.0126725	0.013634	0.017773	$1.2 \cdot 10^{-3}$	30000*	0.0511	0.3418	12.2222
SMO [140]	0.0126652	N/A	N/A	N/A	30000*	0.0516759	0.35640097	11.307562
MPA [145]	0.012665	0.012665	0.012665	$5.55 \cdot 10^{-8}$	25000*	0.0517245	0.35757003	11.239196
EGTO [167]	0.0126652	0.012666	0.012668	$1.13 \cdot 10^{-6}$	25000*	0.0516840	0.35659664	11.296069
SaDN [159]	0.012665	0.012678	0.012872	$6.73 \cdot 10^{-8}$	15000	0.051622	0.355105	11.384415
MSCA [164]	0.0126668	0.012817	0.013342	$1.90 \cdot 10^{-4}$	25000*	0.051782	0.3589448	11.160789
hSM-SA [165]	0.012715	0.013757	0.017731	$1.561 \cdot 10^{-3}$	15000*	0.050059	0.3187474	13.919190
IMODE [[192]	0.012665	0.0135	0.0145	$6.32 \cdot 10^{-4}$	30000	0.0519	0.3623	11
En(L)SHADE [193]♦	$1.27 \cdot 10^{-2}$	$1.27 \cdot 10^{-2}$	$1.27 \cdot 10^{-2}$	0	100000	N/A	N/A	N/A
QSA (best alg. [194])	0.0126652	0.012667	0.012674	$2.589 \cdot 10^{-6}$	18000	0.051688	0.3567	11.28999

♣The fastest optimization run of SHGWJA converging to the best solution required only 2247 function evaluations
*Determined as the product between the population size and the limit number of optimization iterations
**Maximum number of function evaluations indicated in the convergence curve plot
♦The number of significant digits quoted in Ref. [193] was only 3

It can be seen that the SHGWJA algorithm proposed in this study performed very well: its best solution, 0.0126665, practically coincided with the optimized costs of MGWO-III - the astrophysics-based GWO variant of Ref.[110] (0.0126662), preschool education optimization algorithm (PEOA) [179] (0.012667), and modified sine cosine algorithm (MSCA) [164] (0.012668). SHGWJA's optimized cost was only 0.0121% larger than the 0.012665 global optimum obtained by the JAYA variants EHRJAYA [163], SAMP-JAYA [18] and EJAYA [177], mother optimization algorithm (MOA) [120], marine predators algorithm (MPA) [145], hybridized differential evolution naked mole-rat algorithm (SaDN) [159] and improved multi-operator differential evolution (IMODE) [192]. The gaze cues learning-based grey wolf optimizer (GGWO [171], the so-far known best GWO variant), light spectrum optimizer (LSO) [102], starling murmuration optimizer (SMO) [140], hybrid EGTO algorithm combining marine predators and gorilla troops optimizer[167] and the queuing search algorithm (QSA, which mimics the human activities in queuing and was the best amongst the nine MAs compared in [194]), converged to 0.0126652, the same as for EHRJAYA, MOA, MPA, SaDN

and IMODE up to the 5th significant digit. The high-performance algorithm success history based adaptive differential evolution with gradient-based repair (En(L)SHADE) was reported in [193] to obtain an optimized cost of $1.27 \cdot 10^{-2}$, corresponding to the round up of the 0.0126652 cost.

The arithmetic optimization algorithm (AOA) [114] and its improved variant IAOA [161] found the smallest optimized costs respectively of 0.012124 and 0.012018 but the corresponding optimal designs were infeasible violating the 2nd constraint equation – g2($\vec{X}$) in Eq. (19) – by 8.05% and 10.8%. All other algorithms compared in Table 4.10 obtained practically feasible solutions violating constraints by less than 0.143%. The learning cooking algorithm (LCA) was reported in [121] to converge within 30000 function evaluations to the optimized cost of 0.0125 for the corresponding solution $\bar{X}_*$(0.05566; 0.4591;7.194). However, the actual value of cost function computed by giving in input $\bar{X}_*$ to Eq. (19) is 0.013107, much higher than the 0.012665 optimum quoted in Table 4.12.

The rate of success of SHGWJA was 100% also for this test problem: in fact, the present algorithm found very close designs to the global optimum in all independent optimization runs. SHGWJA obtained the 7th lowest standard deviation on optimized cost out of 29 MAs compared in this study (21 listed in Table 4.10 and other 8 of Ref. [194], while no information was available for LSO and its 31 competitors compared in [102]), thus ranking in the 1st quartile. The standard deviation on optimized cost was only 0.0876% of the best design found by SHGWJA. Furthermore, average optimized cost and worst optimized cost by SHGWJA, respectively, were at most 0.134% and 0.213% larger than the 0.012665 target value. Statistical dispersion on optimized cost was very similar to its counterpart seen for MGWO-III, EHRJAYA and PEOA. However, EHRJAYA exhibited a larger average optimal cost than SHGWJA while the worst solutions of MGWO-III and PEOA achieved a larger cost than that of SHGWJA.

As far as it regards computational cost, SHGWJA was on average the 3rd fastest optimizer overall after EHRJAYA and LSO. However, the fastest optimization run of SHGWJA converging to the best solution of 0.0126665 was completed within only 2247 function evaluations, practically the same as for EHRJAYA while the LSO's convergence curve indicated in [102] for the best optimization run covered 5000 function evaluations. Among other JAYA variants, SAMP-JAYA was just slightly slower than SHGWJA (6861 function evaluations vs. only 6347). Furthermore, the present algorithm was on average about (i) 4.7 times faster than MGWO-III, LCA, PEOA, SO, SMO and IMODE, (ii) 4 times faster than MPA, EGTO and MSCA, (iii) 3.6 times faster than GGWO, (iv) 2.8 times faster than QSA and other algorithms of Ref. [194], and (v) 2.4 times faster than EJAYA, EO, AOA, IAOA, SaDN and hSM-SA. It should be noted that Table 4.12 compares the average number of function evaluations required by SHGWJA either with the fixed computational budget of all optimizations runs or the computational cost of the best optimization run. Furthermore, SHGWJA was robust enough in terms of computational cost: standard deviation on the number of function evaluations was about 35.4% of average number of evaluations.

Figure 4.15 shows a more detailed comparison of the optimization history of SHGWJA with those of EGTO [167] and QSA [194]. For the sake of clarity, the plot is limited to the first 3000 function evaluations. Convergence curves of best and average optimization run recorded for

the present algorithm practically coincided after only 700 function evaluations, that is at about 31% of the optimization history of the best run. Interestingly, EGTO started its optimization search from a much better initial design than those of SHGWJA and EGTO: the initial cost evaluated for its best individual was only 0.016 vs. 0.06 for QSA, 0.1 for the best SHGWJA run and 0.1528 for the average SHGWJA run. Despite this, the present algorithm recovered the gap with respect to EGTO already in the early optimization cycles and the convergence curves of EGTO and the best SHGWJA's run crossed each other after 650 function evaluations. Furthermore, SHGWJA generated on average better feasible intermediate designs than QSA after only 135 function evaluations keeping this ability over the first 750 function evaluations. After about 2500 function evaluations, the cost function values of 0.012673 and 0.012697 respectively achieved by EGTO and QSA in their best optimization runs still were higher than the best cost of 0.0126665 finally reached by SHGWJA after only 2250 function evaluations.

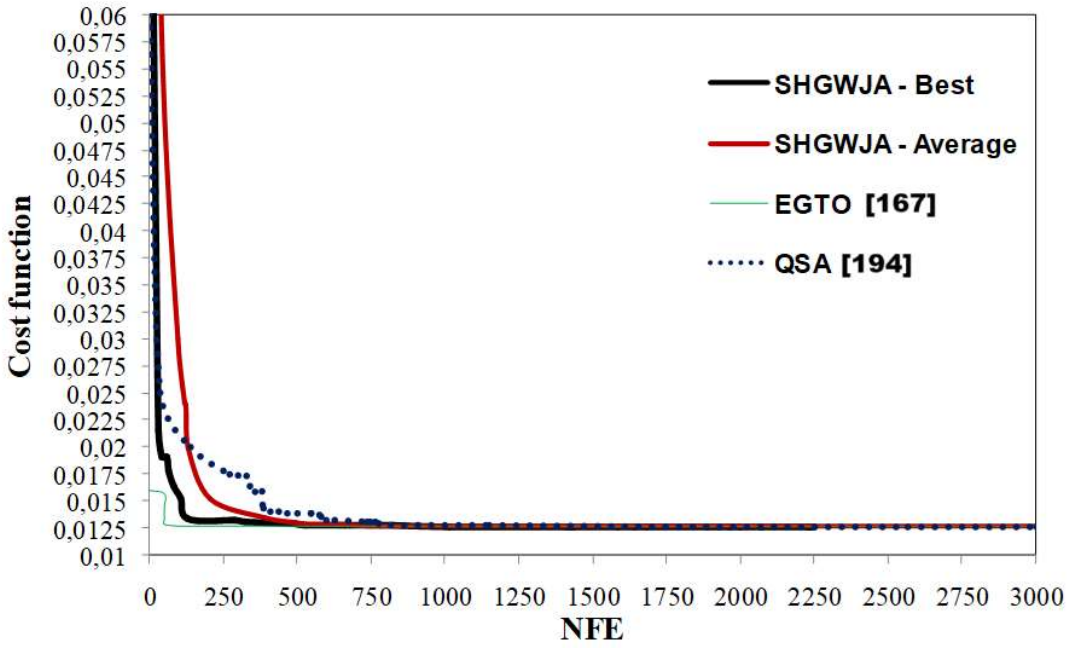


Figure 4.15- COMPARISON OF CONVERGENCE CURVES OBTAINED IN THE TENSION/COMPRESSION SPRING PROBLEM.

The data presented in this section demonstrate with no shadow of doubt the computational efficiency of SHGWJA. In fact, the present algorithm resulted very competitive with 21 state-of-the art MAs each of which was in turn proven in the literature to outperform from 5 to 34 other metaheuristic algorithms and their variants.

4.5.2.3. Welded beam

The optimization results for the welded beam design problem are shown in Table 4.13. The values of standard deviation on optimized cost listed in the table for referenced algorithms are exactly those reported in literature. The proposed SHGWJA algorithm was the best optimizer overall. In fact, the optimum cost of 1.724852 corresponds to the target solution of the classical formulation (see Section 4.4) and practically coincided with the optimized cost values obtained

by all SHGWJA competitors except for the hybrid JAYA variant EHRJAYA [163], arithmetic optimization algorithm (AOA) [114], hybridized differential evolution naked mole-rat algorithm (SaDN) [159], improved multi-operator differential evolution (IMODE) [192], and Taguchi method integrated harmony search algorithm (TIHHSA) [198]. IMODE and SaDN converged to the largest costs, respectively, about 63.8% and 14.6% higher than the optimum cost of 1.72485. The Taguchi method integrated harmony search algorithm (TIHHSA) [198] converged to an optimized cost of 1.74026, close enough to the global optimum reached by SHGWJA.

The arithmetic optimization algorithm (AOA) [114] converged to an optimized solutions yielding a slightly lower cost than SHGWJA: 1.7164 vs. 1.72485. However, this solution violates the shear stress constraint of the classical problem formulation, namely constraint $G_1(\bar{X})$ in Eq. (4.20), by 25.3%. All other algorithms of Table 4.13 solving the classical problem formulation converged to feasible designs.

TABLE 4.13- OPTIMIZATION RESULTS FOR THE WELDED BEAM DESIGN PROBLEM.

Algorithm	Best	Average	Worst	ST Dev	NFE	X_1 (in)	X_2 (in)	X_3 (in)	X_4 (in)
SHGWJA (present) Problem Variant 1	1.724852	1.724852	1.724852	0	4276±509♣	0.20573	3.470489	9.036624	0.20573
GGWO [171]	1.72485	1.72485	1.72485	0	13920	0.2057	3.4705	9.0366	0.2057
MGWO-III [110]	1.724984	1.725156	1.725420	$1.37{\cdot}10^{-4}$	30000*	0.20567	3.4719	9.03668	0.20573
SAMP-JAYA [18]	1.724852	1.724852	1.724852	$6.7{\cdot}10^{-16}$	3618 (aver)	N/A	N/A	N/A	N/A
EJAYA [177]	1.724852	1.724852	1.724852	$5.563{\cdot}10^{-10}$	24000	0.20573	3.470489	9.036624	0.20573
EO [101]	1.724853	1.726482	1.736725	$3.257{\cdot}10^{-3}$	15000*	0.2057	3.4705	9.03664	0.2057
LSO [102	1.724866	N/A	N/A	N/A	5000**	0.20572	3.4707	9.0366	0.20573
AOA [114]	1.7164	N/A	N/A	N/A	15000*	0.194475	2.57092	10.000	0.201827
MOA [120]	1.724852	1.724852	1.724852	$6.9{\cdot}10^{-16}$	N/A	N/A	N/A	N/A	N/A
PEOA [179]	1.724856	1.724892	1.724948	$3.11{\cdot}10^{-5}$	30000*	0.20573	3.47048	9.03664	0.20573
SO [135]	1.724852	1.769949	2.455685	0.137	30000*	0.2057	3.4705	9.0366	0.2057
SMO [140]	1.724852	N/A	N/A	N/A	30000*	0.20573	3.47049	9.03662	0.20573
MPA [145]	1.724853	1.724861	1.724873	$6.41{\cdot}10^{-6}$	25000*	0.20573	3.47051	9.03662	0.20573
EGTO [167]	1.724852	1.724852	1.724852	$5.32{\cdot}10^{-16}$	25000*	0.20573	3.47049	9.03662	0.20573
SaDN [159]	1.9773	1.9828	1.9947	$5.5{\cdot}10^{-6}$	15000	0.2444	6.21787	8.2915	0.2444
hSM-SA [165]	1.725134	1.778021	2.321843	0.143522	15000*	0.20573	3.47113	9.03548	0.20578
IMODE [192]	2.82578	4.44	7.55	1.01	30000	0.2344	4.9662	7.5300	0.3674
QSA (best alg. [194])	1.724852	1.724852	1.724852	$1.121{\cdot}10^{-15}$	18000	0.20573	3.47049	9.03662	0.20573
TIHHSA [198]	1.74026	N/A	N/A	N/A	N/A	0.19823	3.64539	9.02857	0.206407
SHGWJA (present) Problem Variant 2	1.69525	1.69525	1.69525	0	4201±406♣♣	0.20573	3.25312	9.036624	0.20573
EHRJAYA [163]	1.6952	1.696	1.6989	$9.44{\cdot}10^{-4}$	2000	0.20573	3.2531	9.0366	0.20573
MSCA [164]	1.6971	1.70209	1.72216	$5.816{\cdot}10^{-3}$	25000*	0.20519	3.26607	9.033801	0.20591
NDWPSO [181]	1.6978	N/A	N/A	$2.76{\cdot}10^{-9}$	20000*	0.206	3.25	9.04	0.2066
FSO [187]	1.69525	1.69673	1.70139	$7.24{\cdot}10^{-4}$	4500	0.20573	3.25312	9.036624	0.20573
SHGWJA (present) Problem Variant 3	1.670218	1.670218	1.670218	0	4438±424♣♣♣	0.198832	3.337365	9.192024	0.198832
En(L)SHADE [193]♦	1.670	1.670	1.670	0	100000	N/A	N/A	N/A	N/A

LSHADE-SPACMA [185] [195]	1.67021	1.67021	N/A	$1.977 \cdot 10^{-16}$	40000	N/A	N/A	N/A	N/A
SASS [196] [197]	N/A	1.670218	N/A	$2.27 \cdot 10^{-16}$	20000	N/A	N/A	N/A	N/A

♣The fastest optimization run of SHGWJA for problem variant 1 required only 3670 function evaluations
♣♣The fastest optimization run of SHGWJA for problem variant 2 required only 3635 function evaluations
♣♣♣The fastest optimization run of SHGWJA for problem variant 3 required only 3791 function evaluations
* Determined as the product between the population size and the limit number of optimization iterations
**Maximum number of function evaluations indicated in the convergence curve plot
♦The number of significant digits quoted in Ref. [193]was only 4

SHGWJA found the target optimum cost of 1.6702 in all optimization runs (zero standard deviation) also in the problem variant 3 of the CEC 2020 library including $J=2\sqrt{2} \cdot x_1 x_2 \cdot \left[\frac{{x_2}^2}{4} + \left(\frac{x_1+x_3}{2}\right)^2\right]$, beam tip displacement $\delta(\overline{X}) = \frac{4PL^3}{E{x_3}^2 x_4}$, and critical load $P_c(\overline{X}) = \frac{4.013E}{L^2}\sqrt{\frac{{x_3}^2 {x_4}^6}{30}}\left(1 - \frac{x_3}{2L}\right)\sqrt{\frac{E}{4G}}$. The high-performance algorithms En(L)SHADE [193], LSHADE-SPACMA [185] [195] and SASS [196] [197] also converged to the same target optimal solution with 0 or nearly 0 standard deviations. However, SHGWJA required on average less than 4450 function evaluations vs. from 20000 to 100000 evaluations required by SASS, LSHADE-SPACMA and En(L)SHADE, thus confirming the computational efficiency of the proposed algorithm.

The rate of success of SHGWJA was again 100%: in fact, it converged to the global optima of the three problem variants (i.e. 1.72485; 1.69725; 1.6702) in all independent optimization runs. The zero or "practically zero" dispersion on optimized cost was achieved only by (i) SHGWJA, GGWO, SAMP-JAYA, EJAYA, MOA, EGTO and QSA in the classical problem variant 1; (ii) SHGWJA and NDWPSO in problem variant 2; (iii) SHGWJA, En(L)SHADE, LSHADE-SPACMA and SASS in problem variant 3. These data confirm the robustness of the present algorithm. SHGWJA was robust also in terms of computational cost. In fact, the ratio between standard deviation on the number of function evaluations and the corresponding average value was only 11.9%, 9.7% and 9.6% for problem variants 1, 2 and 3, respectively. In problem variant 1 (classical formulation), the present algorithm was the fastest optimizer together with SAMP-JAYA completing its best optimization run within only 3670 function evaluations, 37.5% less than in the case of the 3nd fastest optimizer LSO which required 5000 function evaluations. Furthermore, SHGWJA was on average 3.3 times faster than GGWO and 7 times faster than MGWO-III. The EO, AOA, SaDN and hSM-SA algorithms ranked 5th because they were slightly slower than GGWO; however, AOA converged to an infeasible solution while SaDN missed the global optimum. All other MAs listed in Table 4.13 were from 4.2 to 7 times slower than SHGWJA requiring between 18000 (e.g. QSA and all other MAs of [194]) and 30000 (i.e. the high-performance algorithm IMODE) function evaluations. In problem variant 2, SHGWJA was on average slightly faster than FSO (i.e. 4201 vs. 4500 function evaluations) but also about 6 times faster than MSCA and about 5 times faster than NDWPSO. Its fastest optimization run required only 3635 function evaluations, hence close enough to the 2000 function evaluations required by EHRJAYA. The other JAYA variants of Ref. [163] also were slower than SHGWJA as their optimization runs required between 4000 and 5000 function evaluations. Finally, in problem variant 3, SHGWJA required on average only 4438 function evaluations vs. the fixed computational budgets of 20000, 40000 and 100000 evaluations,

respectively, set in [196] [197] [185] [195] and [193] for the high-performance algorithms SASS, LSHADE-SPACMA and En(L)SHADE. Figure 4.15 presents a detailed comparison of the convergence curves of SHGWJA, LSO [102], EGTO [167], FSO [187] and QSA [194] in the three problem variants. Like the dam case, the best optimization runs of SHGWJA correspond also to the fastest runs as the present algorithm achieved 0 standard deviation on optimized cost in all problem variants. For the sake of clarity, the plot is limited to the first 4500 function evaluations. All algorithms except LSO started from very large costs compared to the target optima: (i) between 2.6 and 6.273 vs. only 1.72485 for problem variant 1; (ii) between 3.144 and 4.675 vs. only 1.69525 for problem variant 2; (iii) between 3.465 and 4.5 vs. only 1.6702 for problem variant 3. The analysis of convergence curves plotted in Figure 4.16 and its insert confirms the superiority of SHGWJA over its competitors also for this test problem. In fact, the best and average optimization runs' convergence curves of SHGWJA practically coincided after only 2000 (i.e. 54.5% of the best run's optimization history), 2100 (i.e. 57.8% of the best run's optimization history) and 1700 (i.e. 44.8% of the best run's optimization history) function evaluations, respectively, for problem variants 1, 2 and 3.

In problem variant 1, LSO started from the cost function value of 1.745, which is just 1.168% larger than the target optimum cost of 1.72485, while SHGWJA started from 6.273. However, SHGWJA recovered such a large gap in cost function with respect to LSO at 60% of its optimization history and then reduced the cost function to a slightly lower value than LSO (i.e. 1.724852 vs. 1.724866). EGTO started its best run from the cost of 2.6, about 58.5% lower than SHGWJA's initial cost. Also in this case, the present algorithm recovered the gap before completing the 68% of its optimization history. QSA's best run started from a lower cost than SHGWJA (i.e. 4 vs. 6.273) but the present algorithm generated better feasible intermediate designs since the first optimization iteration and for the first 400 function evaluations. The convergence curves of QSA and SHGWJA then crossed each other with the former algorithm generating better intermediate designs until about 2000 function evaluations: in particular, SHGWJA reached the cost of 1.72507 after 1968 function evaluations while QSA's cost was 1.725 after 2000 function evaluations. However, SHGWJA returned better than QSA from that point to the end of search process further reducing cost function to the target cost of 1.72485 over the remaining 1692 function evaluations while QSA completed such a task over further 5000 function evaluations.

In problem variant 2, SHGWJA and FSO practically started from the same cost (4.237 vs. 4.675) but the present algorithm approached the target solution of 1.69525 much before FSO. SHGWJA generated in its best optimization run a feasible intermediate design only 0.6% higher in cost than the target minimum (i.e. 1.70526 vs. 1.69525) already at the 46% of its optimization history (i.e. after only 1674 function evaluations) while FSO's cost still was 2.26653. The optimization histories coincided after 2200 function evaluations with both algorithms practically converging to the target cost of 1.69525.

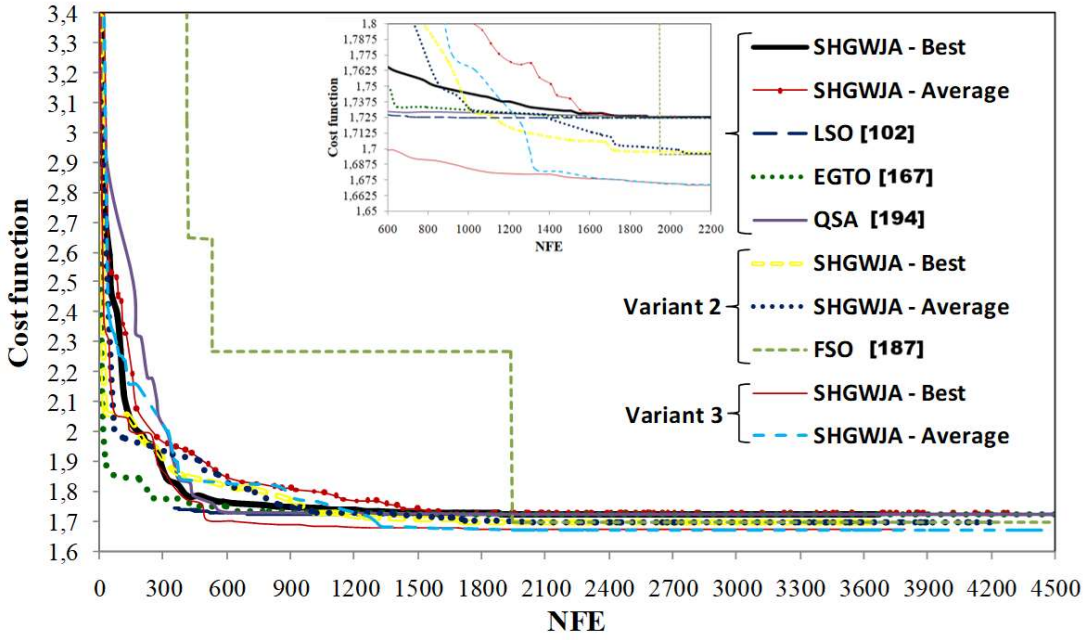


Figure 4.16- COMPARISON OF CONVERGENCE CURVES OBTAINED IN THE WELDED BEAM DESIGN PROBLEM.

The data presented in this section clearly proved the computational efficiency of SHGWJA also in this test problem. In fact, the present algorithm resulted very competitive with 24 state-of-the art MAs each of which was in turn proven in the literature to outperform from 5 to 32 other metaheuristic algorithms and their variants.

4.5.2.4 Pressure vessel

Table 4.14 presents the optimization results for the pressure vessel design problem. The values of standard deviation on optimized cost listed in the table for referenced algorithms are exactly those reported in literature. In the continuous version of the problem, the SHGWJA algorithm developed in this study and the hybrid algorithm EGTO combining the marine predators algorithm and gorilla troops optimizer [167] found the lowest cost overall of 5885.331 in all their independent optimization runs (zero standard deviation). This fully feasible solution was practically the same as those obtained by gaze cues learning-based grey wolf optimizer (GGWO [171], the so-far known best GWO variant, 5885.3328), starling murmuration optimizer (SMO, 5885.3329) [140], EJAYA [177] (5885.333 for this JAYA variant utilizing a larger sets of solutions than classical JAYA to form new trial solutions), marine predators algorithm (MPA, 5885.3353) [145], and equilibrium optimizer (EO) [102] (5585.329, with just 0.000042% violation on constraint $G_3(\bar{X})$ in Eq. (4.21)).

The light spectrum optimizer (LSO) [102], learning cooking algorithm (LCA) [121], hybrid slime mould simulated annealing (hSM-SA) [165], snake optimizer (SO) [135], giant trevally optimizer (GTO) [146] and NDWPSO [181] (combining PSO, differential evolution and whale optimization) ranked right after SHGWJA and the other top MAs for this example as they found optimized cost values ranging from 5585.434 to 5890. The modified sine cosine

algorithm (MSCA) [164], Taguchi method integrated harmony search algorithm (TIHHSA) [198], and arithmetic optimization algorithm (AOA) [114] converged to significantly larger optimized costs than SHGWJA and EGTO: from 5917.5 to 6059.7 vs. only 5885.331. Interestingly, LSO missed the global optimum of the continuous problem although it was superior to other 18 MAs (see Ref. [102] for details).

The optimized designs reported in Table 4.14 for the continuous version of the pressure vessel problem were all feasible except for those algorithms that converged to lower costs than SHGWJA and EGTO: (i-ii) the mother optimization algorithm (MOA) [179], preschool education optimization algorithm (PEOA) [121] (i.e. 5882.901 with 0.040% violation on constraint g3($\vec{X}$) in Eq. (21)); (iii) the astrophysics based GWO variant MGWO-III [110] (5884.0616 with 0.039% violation on constraint g3($\vec{X}$) in Eq. (21)); (iv) the hybrid JAYA variant EHRJAYA [163] (5734.9132 with 3.57% violation on constraint g1($\vec{X}$) in Eq. (21)); (v) the improved arithmetic optimization algorithm (IAOA) [161] (5813.551 with 3.85% violation on constraint g1($\vec{X}$) in Eq. (21)). SAMP-JAYA [18], the JAYA variant using sub-populations to optimize balance between exploration and exploitation, converged to an unfeasible design of cost 5872.213, practically within the same number of function evaluations as the SHGWJA's fastest optimization run (i.e. 6513 vs. 6732).

TABLE 4.14- OPTIMIZATION RESULTS FOR THE PRESSURE VESSEL DESIGN PROBLEM.

Algorithm	Best	Average	Worst	ST Dev	NFE	X_1 (in)	X_2 (in)	X_3 (in)	X_4 (in)
SHGWJA (present) Continuous variables	5885.331	5885.331	5885.331	0	9773±2561♣	0.778168	0.384649	40.31962	200
GGWO [171]	5885.3328	5885.3328	5885.3328	0	12860	0.7782	0.3846	40.3196	200
MGWO-III [110]	5884.0616	5884.7820	5886.3961	0.977	30000*	0.778197	0.38468	40.31549	199.9593
EHRJAYA [163]	5734.9132	5735.3873	5741.0678	1.5336	4000	0.742434	0.370198	40.31962	200
EJAYA [177]	5885.333	5885.886	5894.777	1.734	16000	0.778169	0.384649	40.31962	199.9999
EO [101]	5885.3279	N/A	N/A	N/A	15000*	0.778169	0.384649	40.31962	199.9999
LSO [102]	5585.434	N/A	N/A	N/A	5000**	0.77818	0.38466	40.320	200
AOA [114]	6048.7844	N/A	N/A	N/A	15000*	0.830374	0.416206	42.75127	169.3454
IAOA [161]	5813.551	N/A	N/A	N/A	15000*	0.763721	0.370546	41.5666	184.1352
MOA [120]	5882.901	5882.901	5882.901	$1.89{\cdot}10^{-12}$	N/A	N/A	N/A	N/A	N/A
PEOA [179]	5882.901	5883.043	5884.245	0.316128	30000*	0.778027	0.384579	40.31228	200
LCA [121]	5886	5887	5890	0.9632	30000*	N/A	N/A	N/A	N/A
SO [135]	5887.5298	5989.8091	6247.6170	104	30000*	0.7819	0.3857	40.5752	196.5499
SMO [140]	5585.3329	N/A	N/A	N/A	30000*	0.778169	0.3846492	40.31962	199.9999
MPA [145]	5885.3353	N/A	N/A	N/A	25000*	0.778169	0.3846497	40.31962	199.9999
EGTO [167]	5885.331	5885.331	5885.331	$1.43{\cdot}10^{-12}$	25000*	0.778168	0.384649	40.31962	200
GTO [146]	5889.5	5967.464	6175.568	14.2397	90000*	0.778834	0.385442	40.34289	199.6848
MSCA [164]	5917.5098	6029.244	6396.551	113.247	25000*	0.780583	0.391756	40.41908	198.9641
hSM-SA [165]	5885.789	7318.734	6182.999	451.541	15000*	0.778348	0.384789	40.32887	199.8715
NDWPSO [181]	5890	N/A	N/A	$4.973{\cdot}10^{-5}$	20000*	0.778	0.385	40.3	200
IMODE [192]	6499.445	N/A	N/A	N/A	30000	0.9480	0.4735	48.8727	114.1672
TIHHSA [198]	5959.86	N/A	N/A	N/A	N/A	0.814181	0.403799	42.1533	176.032

SHGWJA (present) Discret./Contin. Variables	6059.714	6059.714	6059.714	0	9712±2730♣♣	0.8125	0.4375	42.0984	176.6372
EO [101]	6059.714	6668.114	7544.493	566.24	15000	0.8125	0.4375	42.0984	176.6372
mEO [157]	6059.714	N/A	N/A	N/A	15000	0.8125	0.4375	42.0984	176.6372
SaDN [159]	6059.714	6128.732	6416.353	46.482	15000	0.8125	0.4375	42.0984	176.6372
FSO [187]	6059.714	6059.717	6060.184	0.2469	7500	0.8125	0.4375	42.0984	176.6366
En(L)SHADE [193]♦	$6.06 \cdot 10^3$	$6.06 \cdot 10^3$	$6.09 \cdot 10^3$	11.5	100000	N/A	N/A	N/A	N/A
QSA (best alg. [194])	6059.714	6078.319	6370.780	61.563	25000	0.8125	0.4375	42.09844	176.6365
SASS [196][197]	N/A	6059.714	N/A	$3.71 \cdot 10^{-12}$	20000	N/A	N/A	N/A	N/A

♣The fastest optimization run of SHGWJA in the continuous problem required only 6732 function evaluations
♣♣The fastest optimization run of SHGWJA in the mixed variable problem required only 7604 function evaluations
* Determined as the product between the population size and the limit number of optimization iterations
**Maximum number of function evaluations indicated in the convergence curve plot
♦The number of significant digits quoted in Ref. [193] was only 3

SHGWJA was the fastest optimizer overall, up to one order of magnitude faster than the giant trevally optimizer (GTO) [146] that even missed the target optimum cost of 5885.331 and exhibited the 3rd worst standard deviation on optimized cost. In particular, SHGWJA required on average: (i) 25% less function evaluations than GGWO; (ii) about 1.5 times less function evaluations than EJAYA, EO and hSM-SA; (iii) about 2 times less function evaluations than MPA, EGTO and MSCA; (iv) about 2.5 times less function evaluations than MPA and EGTO; (v) about 3 times less function evaluations than MGWO-III, PEOA, LCA, SO, SMO and IMODE.

It should be noted the fastest optimization run of SHGWJA was completed within only 6732 function evaluations, very close to: (i) the 4000 function evaluations required by EHRJAYA [165] to prematurely converge to an infeasible solution (the other JAYA variants analysed in [165] required between 8000 and 10000 function evaluations), and (ii) the 5000 function evaluations covered by the LSO's convergence curve indicated in [102].

However, in its fastest optimization run, SHGWJA generated a feasible intermediate design better than the optimized cost of LSO (only 5585.431 vs. 5585.434) within only 4880 function evaluations.

In the mixed variable version of the pressure vessel problem, SHGWJA and its competitors all converged to the target optimum cost of 6059.714. However, the zero or "almost 0" standard deviation on optimized cost was achieved only by SHGWJA and self-adaptive spherical search (SASS) [196][197], not even by the high-performance algorithm En(L)SHADE [193].

This occurred in spite of the fact that the present algorithm was on average (i) one order of magnitude faster than En(L)SHADE; (ii) about 54% faster than EO [101], its improved version mEO [157], and hybridized differential evolution naked mole-rat algorithm (SaDN) [159]; (iii) about 2 times faster than SASS; (iv) about 2.6 times faster than the queuing search algorithm (QSA, the best amongst the nine MAs compared in [194]).

The fastest optimization run of SHGWJA required 7604 function evaluations, practically the same as the flying squirrel optimizer (FSO) [187].

It should be noted that SHGWJA did not implement any specific strategy for handling discrete variables X_1 and X_2 but rounding the continuous values generated for these variables to their nearest multiples of 0.0625.

SHGWJA was robust enough also with respect to computational cost because the ratio between standard deviation on the number of function evaluations and the corresponding average value was respectively 26.2% and 28.1% for the continuous and mixed variables problem versions.

The excellent convergence behaviour of SHGWJA is confirmed by the optimization histories shown in Figure 4.16 for the best and average optimization runs recorded for the two problem variants with continuous variables or mixed variables.

Again, the best SHGWJA's runs correspond to the fastest ones since the present algorithms achieved null standard deviation on optimized cost in both problem variants.

The plot is limited to 7500 function evaluations for the sake of clarity. Best and average optimization runs' convergence curves of SHGWJA practically coincided after only 4500 (i.e. 66.8% of best run's optimization history) and 4000 (i.e. 52.6% of best run's optimization history) function evaluations, respectively, for problem variants 1 and 2. Such a behaviour is highlighted in detail by the insert of Figure 4.17.

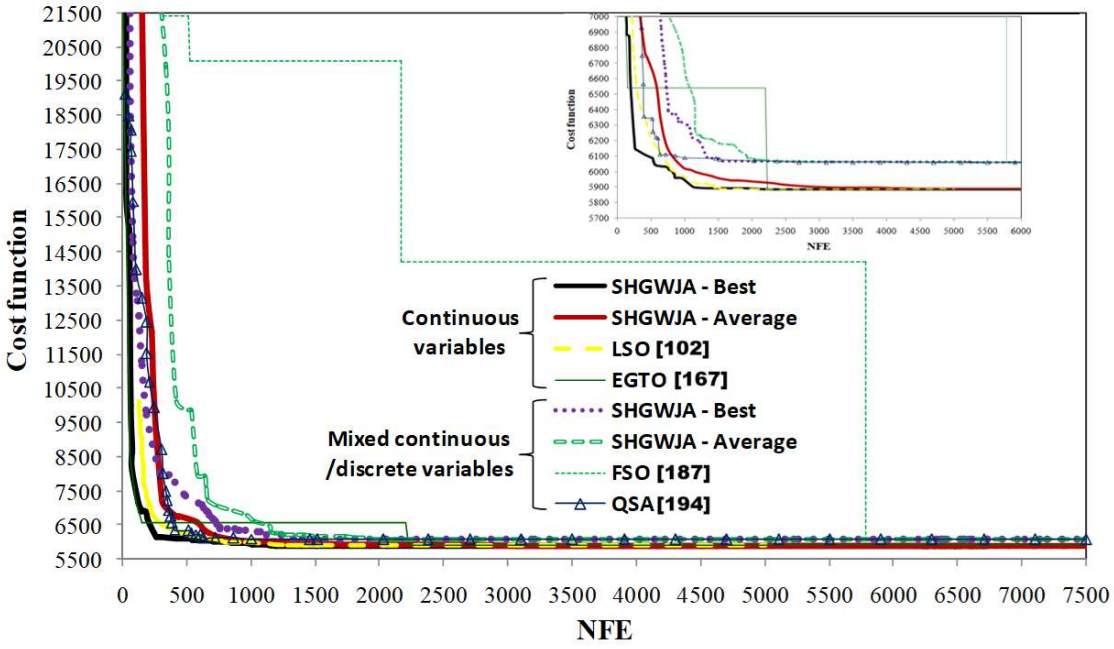


Figure 4.17- COMPARISON OF CONVERGENCE CURVES OBTAINED IN THE PRESSURE VESSEL DESIGN PROBLEM.

The optimization histories of SHGWJA, LSO [102] and EGTO [167] recorded for problem variant 1 are compared in Figure 4.16. LSO and EGTO started their optimizations from 10128.57 and 22233.47, respectively, while SGHJA started its search process from the much

larger cost of 526765. Despite such a large gap, FSO generated worse intermediate designs than the present algorithm over the whole optimization history and converged to a higher optimized cost than the target solution reached by SHGWJA (i.e. 5884.434 vs. 5885.331). Furthermore, optimization histories of SHGWJA and EGTO practically coincided for the first 100 function evaluations and then EGTO reduced cost function to 6359.256 at 150 function evaluations while SHGWJA obtained a similar cost function value at 200 function evaluations. However, the present algorithm kept reducing cost while EGTO exhibited a large step in cost function up to 2200 function evaluations: at this NFE, SHGWJA had already reduced cost to about 5887.3 (just 0.034% more than the 5885.331 target optimum) while cost function values of EGTO's intermediate designs remained fixed at 6359.256.

The optimization histories recorded for SHGWJA, FSO [187] and QSA [194] in problem variant 2 again started from very larger costs than the 6059.714 target: 263805.7, 35000 and 19097.22, respectively (see Figure 4.16). FSO was much slower than the present algorithm and recorded a cost of 14179 after about 5780 function evaluations, that is when SHGWJA's intermediate designs had already reduced cost function to 6061.85, just 0.035% higher than target optimum. SHGWJA recovered the initial gap with respect to QSA already within the first 60 function evaluations and the optimization histories of the two algorithms practically coincided after 3500 function evaluations.

The data presented in this section demonstrate with no shadow of doubt the computational efficiency of SHGWJA also in this test problem. In fact, the present algorithm resulted very competitive with 27 state-of-the art MAs each of which was in turn proven in the literature to outperform from 5 to 31 other metaheuristic algorithms and their variants.

4.5.2.5. Industrial refrigeration system

Table 4.15 presents the optimization results for the industrial refrigeration system design problem. The values of standard deviation on optimized cost listed in the table for referenced algorithms are exactly those reported in literature. All algorithms listed in the table converged to fully feasible solutions.

The new algorithm SHGWJA was very competitive with the other optimizers selected for comparison. In fact, it reached the global optimum cost of 0.032213 (and the target design POPT[0.001; 0.001; 0.001; 0.001; 0.001; 0.001; 1.524; 1.524; 5; 2; 0.001; 0.001; 0.0072934; 0.087556]) along with standard JAYA and improved JAYA [158] variants, starling murmuration optimizer (SMO) [140], hybrid fast simulated annealing (HFSA) [158], success history based adaptive differential evolution with gradient-based repair En(L)SHADE [184], self-adaptive spherical search (SASS) [196][197], and improved cuckoo search (CS-DP1) [200].

The hybrid HS and hybrid BBBC algorithms developed in [158], mBBBC-UBS (derived from [189]), and MsinDE (derived from [190]) ranked right after the best algorithms achieving best optimized costs of 0.032214, 0.032215, 0.032218 and 0.032220, respectively. The mAHS algorithm (derived from [188]) captured the target optimum only up to the third significant digit (i.e. 0.032239 vs. 0.032213). Similar to the gravity dam design problem, the elitist strategy $W(\bar{X}_{i,tr}) \leq 1.1 \cdot W(\bar{X}_{opt})$ implemented in SHGWJA was effective also for mBBBC-UBS,

mAHS and MsinDE that improved their optimized solutions with respect to the original formulations but not enough to outperform SHGWJA. Hybrid Split-Detect-Discard-Shrink - Sophisticated Artificial Bee Colony algorithm (SDDS-SABC) [197] and standard grey wolf optimizer (GWO) found the worst solutions overall with a maximum cost of 0.033740.

The rate of success of SHGWJA was above 80% considering only the solutions yielding optimized costs between the global minimum of 0.032213 and the average cost 0.032215, hence within only 0.00621% penalty with respect to global minimum. Overall, SHGWJA was enough competitive with all other algorithms in terms of robustness. In fact, the average optimized cost coincided with its counterpart for the other algorithms up to the fourth significant digit (i.e. 0.03221).

For SHGWJA, the ratio of standard deviation on optimized cost to average optimized cost was only 0.0114%. Standard GWO and CS-DP1 exhibited the largest dispersions on optimized cost. The high-performance algorithms En(L)SHADE [193] and SASS [196][197] achieved zero standard deviation on optimized cost, but their computational budgets were respectively 100000 (fixed in the CEC 2020 benchmark) and 20000 function evaluations vs. only 8517 function evaluations required on average by SHGWJA.

Interestingly, SHGWJA completed on average within 10000 function evaluations all optimization runs converging to the target global optimum of 0.032213 with null standard deviation when the convergence limit $\varepsilon_{\text{conv}}$ in Eq. (14) was reduced to 10^{-10}.

TABLE 4.15- OPTIMIZATION RESULTS FOR THE INDUSTRIAL REFRIGERATION SYSTEM.

Algorithm	Best	Average	Worst	ST Dev	NFE
SHGWJA (pres.)	0.032213	0.032215	0.032219	$3.671{\cdot}10^{-6}$	8517♣±2370
Standard GWO	0.033740	0.034672	0.036780	$1.292{\cdot}10^{-3}$	50000
Standard JAYA	0.032213	0.032215	0.032233	$6.583{\cdot}10^{-6}$	20255
Improved JAYA [158]	0.032213	0.032213	0.032213	$1.10{\cdot}10^{-10}$	8672
SMO [140]	0.032213	N/A	N/A	N/A	N/A
Hybrid HS [158]	0.032214	0.032214	0.032215	$4.714{\cdot}10^{-7}$	4975
Hybrid BBBC [158]	0.032215	0.032218	0.032220	$2.055{\cdot}10^{-6}$	5784
HFSA [158]	0.032213	0.032218	0.032221	$2.966{\cdot}10^{-6}$	5371
mAHS (der. From [188])	0.032239	0.032263	0.032278	$1.61{\cdot}10^{-5}$	7990
mBBBC–UBS (der. From [189])	0.032220	0.032224	0.032230	$4.113{\cdot}10^{-6}$	8243
MsinDE (der. from [190])	0.032218	0.032222	0.032227	$2.39{\cdot}10^{-5}$	8468
En(L)SHADE [193]♦	$3.22{\cdot}10^{-2}$	$3.22{\cdot}10^{-2}$	$3.22{\cdot}10^{-2}$	0	100000
SASS [196][197]	0.032213	0.032213	0.032213	$1.42{\cdot}10^{-17}$	20000
SDDS-SABC [197]	N/A	0.033414	N/A	0.2644	20000
CS-DP1 [200]	0.032213	0.040964	N/A	$3.274{\cdot}10^{-2}$	10000

♣The fastest optimization run of SHGWJA converging to target optimum required only 5165 function evaluations
♦The number of significant digits quoted in Ref. [193] was only 3

It can be seen from Table 4.15 that SHGWJA required a considerably lower number of function evaluations than standard GWO and JAYA to complete the optimization process. Convergence rate increased almost by about 2.4 times with respect to standard JAYA and by a factor 6 with

respect to standard GWO. The hybrid GWO-JAYA scheme implemented by SHGWJA was hence very effective in increasing diversity of trial solutions from α, β and δ wolves in the exploration phase and enhancing the exploitation phase through the generation of very high-quality designs.

SHGWJA was robust enough in terms of computational cost limiting the ratio between standard deviation on the number of function evaluations and the corresponding average value to 27.8%. SHGWJA required on average 1.6 times more function evaluations than the hybrid HS/BBBC/SA algorithms of Ref. [158], about 4% more function evaluations than mAHS, mBBBC-UBS and MsinDE, and slightly less function evaluations than improved JAYA. However, the fastest optimization run of SHGWJA converging to the global optimum of 0.32213 was completed within only 5165 function evaluations, practically at the same convergence rate of the fastest algorithms reported in literature. This confirms the high computational efficiency of the present algorithm.

The superior convergence behaviour of SHGWJA with respect to its competitors is confirmed by Figure 4.18, which compares the optimization histories of the best runs for the algorithms listed in Table 4.15. The average convergence curve of SHGWJA also is shown in the figure. The plot is limited to 9000 function evaluations for the sake of clarity. Best and average optimization runs' convergence curves of SHGWJA practically coincided after only 2700 function evaluations, that is at 52.3% of the best run's optimization history. All algorithms started their optimization runs from much larger initial costs than the target optimum: from 50 to 235.562 vs. only 0.032213. SHGWJA rapidly approached the target design reducing the cost function to 0.0325 (less than 1% more than the target cost) within only 900 function evaluations vs. the 3000 function evaluations performed by hybrid fast simulated annealing (HFSA) [158], the fastest competitor to achieve the same intermediate solution of 0.0325.

The mAHS algorithm (derived from [188]) was faster than the present algorithm over the first 170 function evaluations but then exhibited two large steps respectively lasting about 2950 function evaluations where cost function decreased from 0.0612 to 0.04 and other 3050 function evaluations where cost function decreased from 0.04 to 0.034.

The elitist strategy $W(\bar{X}_{i,tr}) \leq 1.1 \cdot W(\bar{X}_{opt})$ of SHGWJA allowed to significantly improve convergence speed in the early optimization iterations also for mBBBC-UBS (derived from [189]) and MsinDE (derived from [190]).

In particular, the average optimization history of SHGWJA and the best optimization run of mBBBC-UBS practically coincided between 320 and 1200 function evaluations, while the best optimization runs of SHGWJA and MsinDE overlapped for the first 120 function evaluations.

However, like what reported above for mAHS, the optimization histories of mBBBC-UBS and MsinDE presented large steps yielding very little improvement in cost function. In summary, the $W(\bar{X}_{i,tr}) \leq 1.1 \cdot W(\bar{X}_{opt})$ strategy is very effective when the cost function must be rapidly reduced, but it should be completed by other elitist strategies such those implemented in SHGWJA.

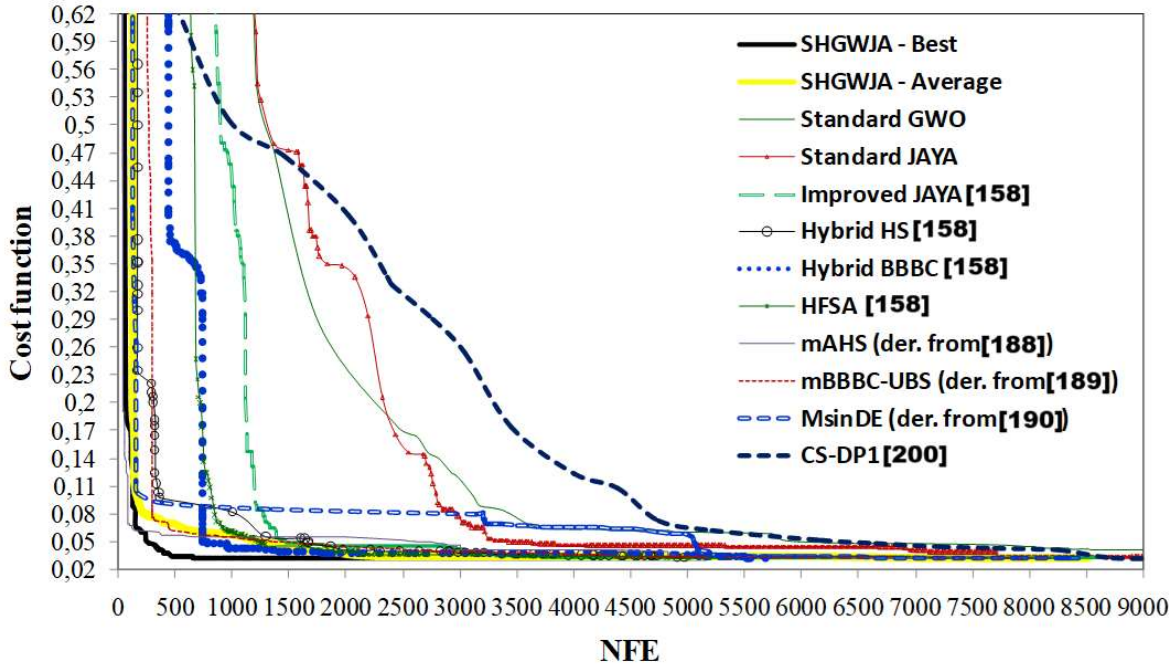


Figure 4.18- COMPARISON OF CONVERGENCE CURVES OBTAINED IN THE INDUSTRIAL REFRIGERATION SYSTEM PROBLEM.

4.5.3 Robotics optimization

4.5.3.1 A six variable path planning

Table 4.16 presents the optimization results for the sixth engineering design problem solved in this study, the 2-D path planning. All data listed in the table correspond to feasible solutions. The new algorithm SHGWJA was the best optimizer overall and designed the shortest path of 41.057 mm. The component algorithms of SHGWJA, the standard grey wolf optimizer (GWO) and standard JAYA were less efficient than the hybrid algorithm developed here: they designed longer trajectories than SHGWJA, respectively, 41.116 mm and 41.083 mm vs. only 41.057 mm.

The other algorithms compared with SHGWJA ranked as follows: hybrid BBBC [158], improved JAYA [158], mBBBC-UBS (derived from [189]), mAHS (derived from [188]), hybrid harmony search (hybrid HS) [158], hybrid fast simulated annealing (HFSA) [158], MsinDE (derived from [190]) and dynamic differential annealed optimization (DDAO) [206]. Optimized trajectory lengths ranged from 41.083 mm to 41.104 mm except for DDAO which designed the very long trajectory of 42.653 mm. Remarkably, the simple formulation of SHGWJA was able to solve a highly nonlinear optimization problem such as 2-D path planning in a more efficient way of algorithms including approximate line search and gradient information. This happened because SHGWJA is inherently able to balance exploration and exploitation phases: the JAYA operator enhances the exploration capability of α, β and δ wolves and forces the optimizer to search very high-quality solutions in the exploitation phase. Like the industrial refrigeration system design problem, the high nonlinearity of the 2-D path planning problem was handled better by SHGWJA than by other HS/BBBC/SA hybrid algorithms including line search and gradient information. The approximate line searches and gradient information are introduced in the optimization formulation trying to minimize the

number of function evaluations. However, this may limit the quality of the approximation and hence the ability of the optimizer to generate trial solutions that are effectively located in regions of search space where the current best record can be improved. Similar to the gravity dam and refrigeration system design problems, the elitist strategy $W(\bar{X}_{i,tr}) \leq 1.1 \cdot W(\bar{X}_{opt})$ implemented in SHGWJA was effective also for mBBBC-UBS, mAHS and MsinDE that improved their optimized solutions with respect to the original BBBC-UBS, SAHS and sinDE algorithms. However, the overall rank BBBC-JAYA-HS-SA-GWO-DE of the base algorithms after SHGWJA was substantially confirmed.

TABLE 4.16-OPTIMIZATION RESULTS FOR THE 2-D PATH PLANNING PROBLEM.

Algorithm	**Best** (mm)	**Average** (mm)	**Worst** (mm)	**ST Dev** (mm)	**NFE**
SHGWJA (pres.)	41.057	41.066	41.081	$7.445 \cdot 10^{-3}$	3541♣±271
Standard GWO	41.116	41.128	41.322	$8.896 \cdot 10^{-2}$	50000
Standard JAYA	41.083	41.095	41.147	$2.635 \cdot 10^{-2}$	15191
Improved JAYA [158]	41.087	41.093	41.097	$4.722 \cdot 10^{-3}$	3613
Hybrid HS [158]	41.092	41.099	41.113	$9.723 \cdot 10^{-3}$	3122
Hybrid BBBC [158]	41.083	41.095	41.119	$1.756 \cdot 10^{-3}$	3482
HFSA [158]	41.093	41.101	41.119	$1.531 \cdot 10^{-2}$	3615±346
mAHS (der. From [188])	41.090	41.114	41.126	$3.205 \cdot 10^{-2}$	5260
mBBBC–UBS (der. From [189])	41.087	41.103	41.115	$2.066 \cdot 10^{-2}$	3934
MsinDE (der. From [190])	41.104	41.110	41.118	$3.354 \cdot 10^{-3}$	5378
DDAO [206]	42.653	45.861	47.421	1.4902	10000

♣The fastest optimization run of SHGWJA converging to the best solution required only 3416 function evaluations

The rate of success of SHGWJA was 100% considering that the solution of all independent optimization runs correspond to shorter trajectories than the best solutions obtained by its competitors. The present algorithm was very competitive with all other optimizers in terms of robustness. In fact, it achieved the best average optimized path length of only 41.066 mm followed by 41.093 mm of improved JAYA, and 41.095 mm of standard JAYA and hybrid BBBC. Standard deviation of optimized path length was only 0.0181% of the shortest path length designed by SHGWJA. Amongst the top 5 algorithms that designed the shortest trajectories (i.e. SHGWJA, standard JAYA, hybrid BBBC, improved JAYA, and mBBBC-UBS), SHGWJA presents the lowest ratio between worst and best solutions. Interestingly, mBBBC-UBS ranked 3rd in terms of best solution together with improved JAYA but only 7th in terms of average optimized path length with 41.103 mm. Standard GWO again exhibited the largest dispersion on optimized cost also considering that only 10 independent optimization runs were executed for DDAO in [206] vs. the 20 runs carried out for standard GWO. SHGWJA was computationally efficient also for this test problem. In this regard, Table 4.16 shows that the proposed SHGWJA algorithm required on average much less function evaluations than standard GWO and JAYA to converge to the optimal solution. In fact, SHGWJA was about 4.3 and 14.1 times faster than standard JAYA and standard GWO, respectively. The

computational speed of SHGWJA was practically the same as for the HS/BBBC/SA hybrid algorithms and improved JAYA. The optimization run of SHGWJA converging to the global optimum path length of 41.057 mm required 3416 function evaluations, which falls in the middle of the 3122 to 3615 intervals recorded for the function evaluations of the fastest algorithms reported in the literature. The mBBBC-UBS algorithm ranked 6th overall (right after HFSA, and on average about 11% slower than SHGWJA) while MsinDE and DDAO were the slowest optimizers requiring about 50% more function evaluations than SHGWJA. The present algorithm was very robust also with respect to computational cost: just 7.6% dispersion on the number of function evaluations, followed by hybrid fast simulated annealing with 9.6%.

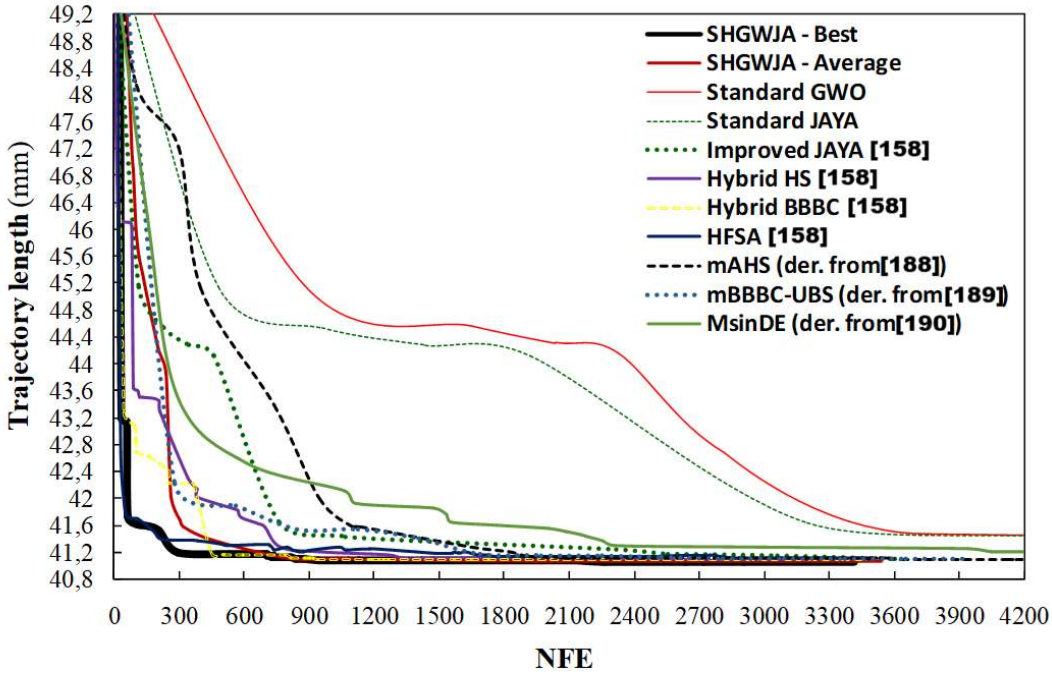


Figure 4.19- COMPARISON OF CONVERGENCE CURVES OBTAINED IN THE 2D PATH PLANNING PROBLEM.

The excellent convergence behaviour of SHGWJA is confirmed by Figure 4.19 that compares the optimization histories of the best runs for the algorithms listed in Table 4.16. The average convergence curve of SHGWJA also is shown in the figure. The plot is limited to 4200 function evaluations for the sake of clarity. The best and average optimization runs' convergence curves of SHGWJA practically coincided after only 1450 function evaluations, that is at 42.5% of the best run's optimization history. The best optimization run of SHGWJA started from the very large cost of 254.713 mm (i.e. about 6.2 times the globally optimum length of 41.017 mm found by the present algorithm) while the initial cost for all other optimizers ranged between 50.01 and 59.62 mm (i.e. at most 45.2% longer than the shortest trajectory designed by SHGWJA). Like the other test problems discussed in this section, the present algorithm immediately recovered the initial gap in cost function with respect to its competitors. The hybrid simulated annealing (HFSA) and hybrid big bang □ big crunch algorithms of Ref. [158]

were the only algorithms to compete in convergence speed with SHGWJA, respectively, for the first 230 and 450 function evaluations.

The optimized trajectories obtained in the best optimization runs of SHGWJA and its competitors are compared in Figure 4.20. SHGWJA, hybrid HS/BBBC/SA, mBBBC-UBS and improved JAYA reduced substantially the path length between the obstacles O1 and O7 with respect to the solution of DDAO. Furthermore, the present code reduced the curvature of the central part of the designed trajectory with respect to hybrid HS/BBBC/SA, improved JAYA and mBBBC-UBS (trajectories relative to mAHS and MsinDE are not shown in the figure because are very similar to the one plotted for mBBBC-UBS) thus shortening the path length. This is confirmed by comparing the optimal solution of SHGWJA with that of hybrid BBBC, the 2nd best solution overall. The optimal positions of base points found by SHGWJA were (10.001;5.778), (17.183;16.589) and (19.696;20.495) mm while they were (12.362;8.940), (15.060;13.460) and (18.198;18.446) mm for hybrid BBBC, which obtained the best solution quoted in the literature so far [158]. The base points of SHGWJA may be fitted in the XY plane by a linear regression with R2=1 while for the base points of hybrid BBBC the correlation coefficient is only R2=0.999.

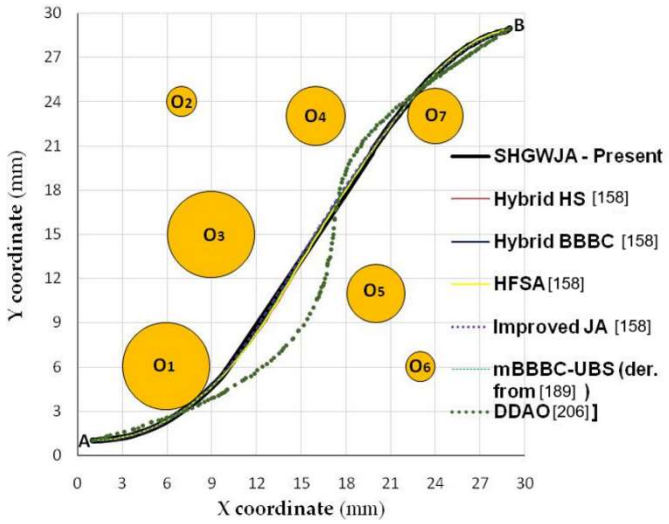


Figure 4.20- COMPARISON OF OPTIMAL TRAJECTORIES OBTAINED IN THE 2-D PATH PLANNING PROBLEM BY THE DIFFERENT MAs.

4.5.4. Characterization of a flat composite panel subjected to axial compression

The results obtained for this engineering problem are presented in Table 4.17. The values of material properties and ply orientations of the axially compressed flat composite panel identified by SHGWJA and its competitors in their best optimization runs are listed in the table. The average and maximum errors on the identified parameters and the panel's fundamental buckling mode shape for the best optimization runs of all algorithms also are listed in the table along with the number of finite element (FEM) analyses required in the identification process.

SHGWJA identified the panel's properties more accurately than its competitors. In fact, the average error on material/layup parameters recorded for SHGWJA was only 0.03888% while the corresponding errors for the other algorithms ranged between 0.197 (hybrid fast simulated

annealing, HFSA, of Ref. [74]) and 5.214 (improved JAYA variant of Ref. [74]). The largest error on parameters for the present algorithm was only 0.07080% (on the longitudinal modulus EL), while the other algorithms could not reduce this error below 0.389% (the best SHGWJA's competitor again was HFSA). The superiority of SHGWJA on its competitors emerges clearly from the analysis of identified parameters. SHGWJA identified the 0° ply angle at the 4th digit while the other algorithms could at most identify up to the 3rd digit. Furthermore, the errors on the 90° and ±45° ply directions were only 0.01222% and 0.008889%, respectively.

TABLE 4.17- RESULTS OF THE AXIALLY COMPRESSED FLAT COMPOSITE PANEL'S IDENTIFICATION PROBLEM.

Properties	E_L (MPa)	E_T (MPa)	G_{LT} (MPa)	ν_{LT}	θ_o	θ_{90}	θ_{45}	Error (%) on parameters	Error (%) on buckling mode	FEM analyses
SHGWJA (pres.)	148405	7448	4138	0.01511	0.000581	90.011	45.004	Aver: 0.03888 Max: 0.07080	Aver: 0.674 Max: 1.035	384±39 366
Standard GWO	148635	7468	4317	0.01516	0.007469	89.125	45.357	Aver: 1.151 Max: 4.275	Aver: 3.581 Max: 4.421	5000
Standard JAYA	148562	7474	4308	0.01517	0.005758	89.214	45.121	Aver: 1.027 Max: 4.058	Aver: 3.269 Max: 4.074	2000
Impr. JAYA [74]	149703	7451	4999	0.01525	0.01586	85.102	46.171	Aver: 5.124 Max: 20.749	Aver: 4.058 Max: 4.903	975
HFHS [74]	147928	7491	4145	0.01510	0.003492	89.904	45.113	Aver: 0.213 Max: 0.550	Aver: 1.680 Max: 2.504	478
HFBBBC [74]	148343	7499	4160	0.01509	0.003243	90.274	44.975	Aver: 0.266 Max: 0.658	Aver: 1.889 Max: 2.735	431
HFSA [74]	148708	7479	4140	0.01508	0.001021	90.223	44.938	Aver: 0.197 Max: 0.389	Aver: 1.554 Max: 2.281	596
mAHS (der. From [188])	148593	7429	4109	0.01513	0.007373	89.306	45.279	Aver: 0.470 Max: 0.771	Aver: 2.075 Max: 2.993	1044
mBBBC–UBS (der. From [189])	147793	7437	4108	0.01523	0.008162	89.426	45.314	Aver: 0.581 Max: 0.861	Aver: 2.146 Max: 3.087	819
MsinDE (der. From [190])	148164	7483	4139	0.01528	0.005173	89.468	45.329	Aver: 0.402 Max: 0.731	Aver: 1.996 Max: 3.006	968

The maximum residual error on panel's buckling mode shape evaluated for the SHGWJA's solution was only 1.035% vs. 2.281% of HFSA, while the other algorithms obtained error values ranging from 2.504% (hybrid fast harmony search optimization, HFHS, of Ref. [74]) to 4.421% (standard GWO). The average error on buckling mode shape for the present algorithm was only 0.674 vs. at least 1.554% for its competitors. The largest errors on w-displacements were localized in correspondence of the regions near points A and B of the vertical control path sketched in Figure 4.11, that is where the panel is fixed to the testing machine, and

displacements should be close to zero. The selected mesh size allowed to avoid numerical noise in the finite element analyses performed in the optimization search. Interestingly, SHGWJA minimized the error functional Ω of Eq. (25) considering 170 control points while the hybrid fast HS/BBBC/SA algorithms used in Ref. [74] considered only the 86 control points lying on the path AB parallel to the loading direction. Despite this, SHGWJA significantly reduced the error on the buckling pattern with respect to the optimizers of Ref. [74].

SHGWJA outperformed its component algorithms standard GWO and JAYA, which failed in identifying the target value of shear modulus GLT (respectively, 4317 and 4308 MPa with 4.275% and 4.058% error on this elastic property). Furthermore, the present algorithm required only 366 finite element analyses vs. 5000 and 2000 analyses respectively required by standard GWO and JAYA. The improved JAYA variant used in [74] required 975 analyses but prematurely converged to the worst solution overall with 20.749% error on shear modulus. The hybrid GWO-JAYA formulation implemented by SHGWJA was hence very effective also in this highly nonlinear optimization problem.

Similar to the gravity dam, refrigeration system design and 2D path planning problems, the elitist strategy $W(\bar{X}_{i,tr}) \leq 1.1 \cdot W(\bar{X}_{opt})$ implemented in SHGWJA was effective also for mAHS (derived from [188]) and mBBBC-UBS (derived from [189]). In fact, mAHS and mBBBC-UBS significantly improved their performance with respect to the original AHS and BBBC-UBS algorithms used in Ref. [74]. MAHS reduced the identification errors on transverse elastic modulus E_T and Poisson's ratio ν_{LT}, respectively, from 2.470% to only 0.282% and from 1.589% to only 0.199%. Ply angles also were identified much more accurately than in [74]: errors on 90° and ±45° orientations decreased from 2.631% to 0.771% and from 1.796% to 0.620%, respectively; the 0° ply angle was identified up to the 3rd digit reducing its value by more than one order of magnitude with respect to [74]: 0.007373° vs. 0.08884°. Furthermore, mAHS completed its search process within only 1044 finite element analyses while the original AHS algorithm used in [74] required 1650 analyses.

The mBBBC-UBS algorithm reduced the identification errors on transverse elastic modulus E_T, shear modulus G_{LT} and Poisson's ratio ν_{LT}, respectively, from 1.503% to only 0.175%, from 3.478% to 0.773%, and from 1.987% to 0.861%. Ply angles also were identified much more accurately than in [74]: errors on 90° and ±45° orientations decreased from 1.652% to 0.638% and from 2.227% to 0.698%, respectively; the 0° ply angle was identified up to the 3rd digit reducing its value by almost 48%, from 0.03891° vs. 0.008162°. Furthermore, mBBBC-UBS completed the identification process within only 819 finite element analyses while the original BBBC-UBS algorithm required 1250 analyses in [74].

The MsinDE algorithm (derived from [190]) augmented by the $W(\bar{X}_{i,tr}) \leq 1.1 \cdot W(\bar{X}_{opt})$ SHGWJA's elitist strategy also performed well ranking overall 5th in terms of accuracy in panel's property identification and buckling pattern reconstruction after SHGWJA and hybrid fast HS/BBBC/SA. MsinDE ranked overall 6th in terms of computational cost after SHGWJA, hybrid fast HS/BBBC/SA and mBBBC-UBS completing the identification process within 968 finite element analyses.

Statistical dispersion on identified panel's properties and residual error on buckling pattern evaluated over the 20 independent runs remained below 0.75% thus confirming the robustness

of SHGWJA. In the worst optimization run, SHGWJA still was able to limit maximum errors on panel's properties and buckling mode to 0.08% and 1.0625%, respectively, while its best competitor, HFSA, recorded 0.389% and 2.281% for the same quantities. The present algorithm was very robust also in terms of computational cost: in fact, standard deviation on the number of finite element analyses required by SHGWJA in the identification process for the different runs was only 10.2% of the corresponding average number of analyses. The present algorithm was on average respectively 35.6%, 19.7% and 10.9% faster than hybrid fast SA, hybrid fast HS and hybrid fast BBBC, and from 2.1 to 2.7 times faster than mBBBC-UBS, MsinDE and mAHS.

More information on convergence behaviour of SHGWJA and its competitors are available from Figure 4.21 that compares convergence curves relative to the best optimization runs of the algorithms listed in Table 4.17. The average convergence curve of SHGWJA also is shown in the figure. The plot is limited to 1050 function evaluations for the sake of clarity. It can be seen from the figure that SHGWJA generated better intermediate designs than its competitors over the whole search process. The best and average optimization runs' convergence curves of SHGWJA practically coincided after 275 function evaluations, that is at 75.1% of the best run's optimization history. The convergence curve recorded for the best optimization run of the hybrid fast harmony search (HFHS) algorithm was close enough to the average convergence curve of SHGWJA for the first 40 finite element analyses and from 150 to 210 analyses. HFHS and HFBBBC (hybrid fast big bang-big crunch) were the only algorithms able to reduce the error functional value below 10^{-4} within 350-360 structural analyses, that is when SHGWJA concluded its optimization process with the final cost function value $\Omega_{OPT} = 1.97 \cdot 10^{-6}$. Since SHGWJA started its best run from an initial population including individuals with at least Ω=2.049 while the best optimization runs of HFHS and HFBBBC started from populations with at least Ω=0.891, the present algorithm proved once again its inherent ability to quickly approach the global optimum solution.

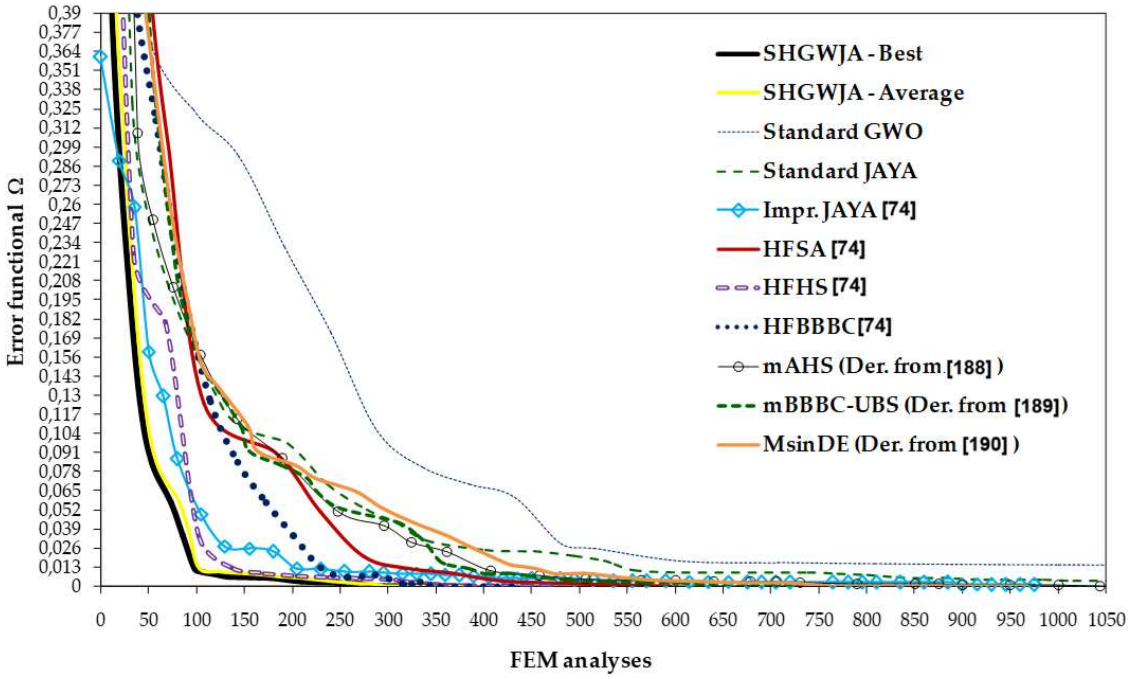


Figure 4.21- COMPARISON OF CONVERGENCE CURVES OBTAINED IN THE IDENTIFICATION PROBLEM OF THE AXIALLY COMPRESSED FLAT COMPOSITE PANEL.

To check whether residual errors on panel buckling pattern evaluated for each optimization run were not caused by some inherent limitation of SHGWJA formulation, the identification problem stated by Eq. (25) was solved in silico. For that purpose, the target buckling pattern of the panel was computed by ANSYS for the target panel's properties. SHGWJA hence performed the optimizations to reconstruct the target displacement field generated via FEM. Remarkably, the present algorithm always converged to target properties reproducing the panel displacement field with 0 residual error in all independent optimization runs. The 0.08% maximum errors on buckling pattern mentioned above for the 20 SHGWJA's optimization runs were hence caused by the uncertainties usually entailed by the hybrid characterization process, a very complicated task matching experimental data and FEM results. Optically measured displacements certainly are a good target to be selected because the double projection moiré is a highly sensitive full field measurement technique. However, in this study, control points were located at critical locations where even small changes in measured quantities may affect to a great deal the success of identification process. Hence, the present results are very satisfactory.

4.6 STATISTICAL ANALYSIS OF THE RESULTS OF ENGINEERING OPTIMIZATION PROBLEMS

In addition to the classical statistical analysis commonly employed in engineering optimization practice, based on the best, average, and worst solutions, as well as the corresponding standard deviation of the optimized cost across independent optimization runs, the optimized designs obtained by SHGWJA and its competing algorithms- standard GWO, standard JAYA, improved JAYA [158], hybrid HS/BBBC/SA [158], hybrid fast HS/BBBC/SA (specifically developed for inverse problems) [74], mAHS (derived from [188]), mBBBC-UBS (derived from [189]) and MsinDE (derived from [190]), in the independent runs were also analysed by

performing a Wilcoxon test with a level of significance of 0.05. Interestingly, p-values determined for each pair of algorithms' solutions always were smaller than 0.05, thus confirming the superiority of SHGWJA over its competitors.

In the second part of statistical analysis, SHGWJA and its competitors were ranked with respect to five performance criteria:

i. best optimized cost (BEST)
ii. average optimized cost (AVERAGE)
iii. worst optimized cost (WORST)
iv. standard deviation on optimized cost (STD)
v. number of function evaluations (or finite element analyses) (NFE).

Table 4.18 lists the corresponding ranks achieved by SHGWJA in each engineering optimization problem. The table includes 10 entries for each ranking criterion because the welded beam and pressure vessel design problems have 3 and 2 variants, respectively. For each test problem, the total score (TOTSCO) assigned to SHGWJA (also reported in Table 4.18) was determined by summing over the ranks achieved for the different criteria: TOTSCO=5 may be obtained only if the optimizer ranks 1st with respect to all performance criteria.

Overall, SHGWJA achieved the highest rank in 9 out of 10 cases for best optimized cost, in 8 out of 10 cases for average and worst optimized cost, in 7 out of 10 cases for the standard deviation of the optimized cost, and in 6 out of 10 cases for the number of function evaluations. The comparatively lower ranks obtained by SHGWJA in the tension/compression spring design problem—14th out of 22 MAs for best optimized cost, 8th for average optimized cost, 6th for worst optimized cost, and 7th for the standard deviation of the optimized cost—are not particularly concerning. This is because SHGWJA's best solution (0.0126665) was virtually identical to the target optimum (0.012665), and its average and worst optimized costs were also very close to this reference value. A similar situation was observed in the refrigeration system design problem, where SHGWJA ranked 5th and 7th (out of 15 MAs) with respect to average/worst optimized cost and the standard deviation of the optimized cost, respectively. In this case, the average and worst optimized costs found by SHGWJA were 0.032215 and 0.032219, values practically indistinguishable from the target optimum of 0.032213. Furthermore, it should be emphasized that two algorithms reporting lower standard deviations than SHGWJA never converged to the target optimum in any of their independent runs. Finally, in the 2D path planning problem, SHGWJA ranked 4th out of 11 MAs for the standard deviation of the optimized cost. However, since the worst solution obtained by SHGWJA outperformed the best solutions produced by all competing algorithms, its effective rank in this problem should be regarded as first.

TABLE 4.18- SHGWJA RANKS AND TOTAL SCORE IN THE SEVEN ENGINEERING PROBLEMS

Design example	BEST	AVERAGE	WORST	STD	NFE	TOTSCO
Dam	1	1	1	1	1	5
Tension/compression spring	14	8	6	7	2	37
Welded beam	1	1	1	1	1	5

	1	1	1	1	2	6
	1	1	1	1	1	5
Pressure vessel	1	1	1	1	1	5
	1	1	1	1	1	5
Refrigeration system	1	5	5	7	2	20
2D path planning	1	1	1	4	2	9
Panel identification	1	1	1	1	1	5

Table 4.18 shows that SHGWJA obtained a very large value of TOTSCO (i.e. 37) only in the tension/compression spring design problem and a quite large total score (i.e. 20) in the industrial refrigeration system problem. In view of this, the SHGWJA's competitors that obtained lower cost function values than 0.0126665 were analysed with respect to all performance criteria and total score: the corresponding results are listed in Table 4.19. SHGWJA practically reached the same overall performance of GGWO and EHRJYA, the most advanced GWO and JAYA variants considered for this test problem: 37 vs. 31 and 36 total score values. While GGWO and EHRJYA converged to the target optimum, they were less efficient than SHGWJA with respect to 3 out of the other 4 performance indicators. Interestingly, SHGWJA would be the 5th best algorithm overall if the contribution of the best convergence run is removed from the total score (see the last column of the table), thus recovering the small gaps from GGWO and EHRJYA and reducing its original 22 points gap from the top ranked algorithms MPA and En(L)SHADE to only 9 points.

The same analysis of Table 4.19 was then carried out for the welded beam and the pressure vessel design problems limiting the set of SHGWJA's competitors to GGWO, EHRJAYA, EJAYA, MPA, EGTO, SaDN, En(L)SHADE and QSA. These algorithms were selected because (i) all necessary data for determining performance indicators were available from literature for at least one test problem, and (ii) they had achieved better values of total score than SHGWJA in the tension/compression spring problem. Table 4.20 compares the performance indicators and the total score values obtained by SHGWJA and its competitors in the welded beam and pressure vessel problems. The first entries of each row in the table refer to the welded beam problem results. Since some algorithms were run only for one of the welded beam problem or pressure vessel problem variants, values of performance indicators and total score reported for SHGWJA represent the average of the corresponding values listed in Table 4.19.

TABLE 4.19- COMPARISON OF RANKS AND SCORES ACHIEVED IN THE TENSION/COMPRESSION SPRING PROBLEM.

Algorithm	BEST	AVERAGE	WORST	STD	NFE	TOTSCO	(TOTSCO–BEST)
SHGWJA - Present	14	8	6	7	2	37	23
GGWO [171]	1	10	7	1	12	31	30
MGWO-III [110]	13	7	10	8	14	52	39
EHRJAYA [163]	1	13	11	10	1	36	23
SAMP-JAYA [18]	1	12	13	11	4	41	29
EJAYA [177]	1	6	4	7	4	22	21
MPA [145]	1	1	1	3	9	15	14

EGTO [167]	2	3	3	4	9	21	19
SaDN [159]	1	7	10	4	4	26	25
En(L)SHADE [193]	1	1	1	1	11	15	14
QSA (best alg. [194])	2	4	4	5	10	23	21

It can be seen from Table 4.20 that SHGWJA significantly improved its rank with respect to the other algorithms. En(L)SHADE remained the best algorithm over the spring/welded beam/vessel problem, but SHGWJA reduced the gap to only 7.3 points (47.3 vs. 40), equalling almost the 2nd rank position of GGWO (47.3 vs. 45). SHGWJA actually ranks above both MPA, which reached a total score of 44 for only the tension/compression spring and welded beam problems, and EHRJAYA that converged to an infeasible solution in the pressure vessel problem obtaining a total score of 45 for only the tension/compression spring and welded beam problems: even adding only 5 points (best case scenario) the total score would raise to 49 for MPA and 50 for EHRJYA, above the 47.3 score of SHGWJA.

TABLE 4.20- COMPARISON OF RANKS AND SCORES ACHIEVED IN THE WELDED BEAM (1ST ENTRY OF EACH ROW) AND PRESSURE VESSEL (2ND ENTRY OF EACH ROW) DESIGN PROBLEMS BY SHGWJA AND ITS BEST COMPETITORS, ALONG WITH CUMULATIVE RANKS FOR SPRING/WELDED BEAM/VESSEL PROBLEMS.

Algorithm	BEST	AVERAGE	WORST	STD	NFE	TOTSCO	Cumulative Total Score
SHGWJA - Present	1	1	1	1	1.3	5.3	47.3
	1	1	1	1	1	5	
GGWO [171]	1	1	1	1	3	7	45
	1	1	1	1	3	7	
EHRJAYA [163]	1	2	2	3	1	9	45
	Infeasible	---	---	---	---	---	
EJAYA [177]	1	1	1	1	8	12	57
	1	4	4	9	5	23	
MPA [145]	1	1	8	8	11	29	44
	N/A	N/A	N/A	N/A	N/A	N/A	
EGTO [167]	1	1	1	1	11	16	50
	1	1	1	2	8	13	
SaDN [159]	17	14	13	8	5	57	103
	1	6	5	5	3	20	
En(L)SHADE [193]	1	1	1	1	4	8	40
	1	1	3	4	8	17	
QSA (best alg. [194])	1	1	1	1	6	10	52
	1	5	4	2	7	19	

Interestingly, SHGWA and En(L)SHADE can be compared also in the industrial refrigeration system design problem while no data are available for the other algorithms. The ranks of En(L)SHADE were $\{1, 1, 1, 1, 14\}$ for a total score of 18, less than the score 20 obtained by SHGWA (see Table 4.18). However, En(L)SHADE results were collected for a computational budget of 100000 function evaluations while the present algorithm always required on average only 8517 function evaluations to complete its optimization runs. As mentioned in Section 5.6,

SHGWJA completed on average within only 10000 function evaluations (yet one order of magnitude less than En(L)SHADE) all optimization runs always converging to the target global optimum of 0.032213 with null standard deviation when the convergence limit $\varepsilon_{\text{conv}}$ in Eq. (14) was reduced to 10^{-10}. Hence, the total score of SHGWJA in the industrial refrigeration system problem would be 5. Consequently, the total score of SHGWJA in the four CEC2020 test problems (i.e. tension/compression spring, welded beam, pressure vessel, industrial refrigeration system) would be only 37+5.3+5+5=52.3, better than the total score 15+8+17+18=58 of En(L)SHADE. This confirms the high efficiency of the present algorithm.

The last part of statistical analysis regarded the trajectory of optimization variables during search process. However, it should be noted that analysing the trajectory of only a few (very often just 1 or 2) optimization variables of one individual of the population as it is usually done in literature may provide only partial information on the convergence behaviour and robustness of an optimization algorithm if the selected variables do not drive the search process. For example, some optimization variable may fluctuate by a large extent over the whole search process while another variable may converge to its target value within just a few iterations. The former may occur if the optimizer gets stuck in a region of search space hosting many competitive solutions weakly dependent on the fluctuating variable(s). To overcome this limitation, the worst-case scenario was considered in this study by monitoring the evolution of the largest percent variation experienced by optimization variables of the whole population as search progressed. If the optimizer truly converged to optimal solution, design variables should not fluctuate. The selected representation is a sort of normalized trajectory of the variables that cycle by cycle have most difficulty to converge. The normalized trajectory is hence formed by the trajectories of many different variables. The maximum variation of optimization variables over the whole population also gives information on population diversity, which will tend to zero as the global optimum is reached.

Figure 4.22 shows the plots relative to the best runs of SHGWJA for the engineering optimization problems solved in this study. In order to evaluate the overall behaviour of SHGWJA, normalized trajectories were plotted with respect to the percent fraction of best run's convergence history: (i) Figure 4.22.a presents the data obtained for the tension/compression spring, welded beam and pressure vessel test problems; (ii) Figure 4.22.b shows the data obtained for the concrete dam, industrial refrigeration system, 2D path planning and composite panel characterization test problems. For the sake of clarity, the scale of the plot vertical axes was limited to 45%.

It can be seen from Figure 4.22 that the largest variation of optimization variables became smaller than 1-2% at about half of the optimization history. Such a behaviour was observed for all engineering design test problems solved in this study. Oscillations of normalized trajectory (with peaks up to 45%) were relevant over the first 10% of optimization history for the spring/beam/vessel problems while they lasted until 25-30% of optimization history for the dam/refrigeration system/2D path planning/composite panel problems. This was expected in view of the larger number of optimization variables and higher complexity of the latter group of test problems. In all cases, oscillations were smoothed as the optimization search progressed. For example, in the variant 2 of pressure vessel problem, the largest percent variation of variables was 45% at 3% of search, 25% at 6%, 15% at 9%, and only 2% at 17% of search. In

the composite panel identification problem, the largest percent variation of variables was 42% at 4% of search, 20% at 14%, 8% at 20%, 5% at 25% and only 2.5% at 30% of search.

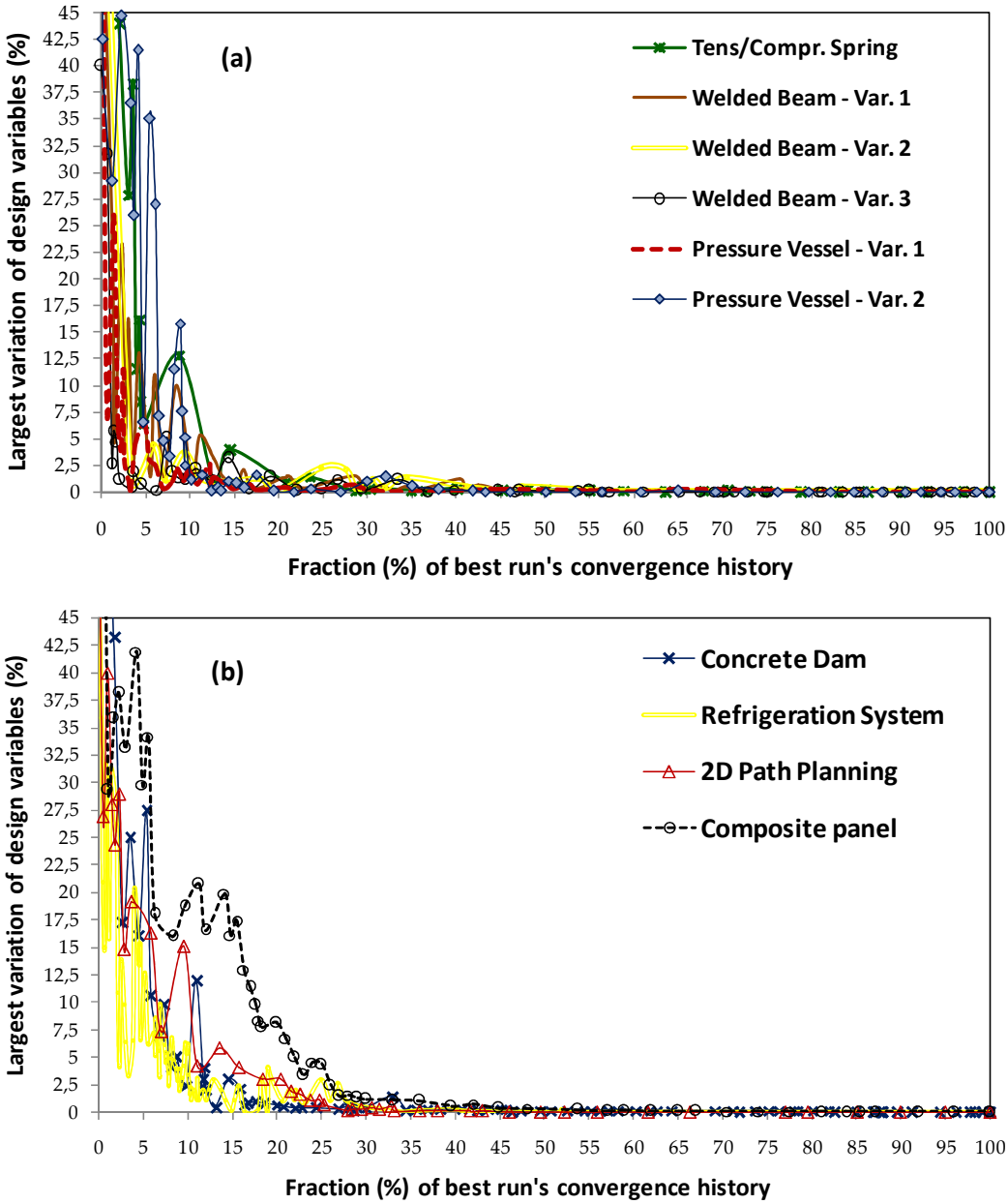


Figure 4.22- CONVERGENCE FRACTION VS. LARGEST DESIGN VARIABLE VARIATION

4.7 FINAL EVALUATION AND RESEARCH SIGNIFICANCE OF SHGWJA

This Chapter has introduced SHGWJA, a novel hybrid metaheuristic algorithm combining the Grey Wolf Optimizer (GWO) and the JAYA algorithm. This method employs an elitist strategy together with JAYA operators to reduce the number of function evaluations while achieving an effective balance between exploration and exploitation. The guiding principle of this work

was to design a hybrid optimization algorithm that is both computationally efficient and conceptually simple, making it suitable for a wide range of engineering optimization tasks.

The proposed algorithm was tested extensively on two classical mathematical benchmarks (Rosenbrock and Rastrigin functions, with up to 1000 variables) and on seven real-world engineering optimization problems. Four of these problems were drawn from the CEC 2020 benchmark suite, while the remaining cases spanned diverse engineering domains: civil engineering (shape optimization of a concrete gravity dam), mechanical engineering (tension/compression spring, welded beam, pressure vessel, refrigeration system), aeronautical engineering (mechanical characterization of a flat composite panel under axial compression), and robotics (2D path planning). The engineering problems considered involved up to 14 optimization variables and as many as 721 nonlinear constraints, thereby providing a demanding and diverse testing ground.

The comparative study was deliberately rigorous. SHGWJA was evaluated against between 10 and 31 state-of-the-art metaheuristics in each problem, including algorithms that had already outperformed numerous competitors and those recognized as CEC competition winners. Many of these methods were improved variants of widely used MAs, recently proposed high-performance algorithms, or hybrid approaches incorporating gradient-based or line-search components. Against this challenging benchmark set, SHGWJA consistently converged to the global optima reported in the literature or to solutions practically indistinguishable from them. For the mathematical benchmarks, best costs were in the order of 10^{-15}, with standard deviations between 10^{-13} and 10^{-1} . In the real-world engineering problems, deviations from the global optimum were negligible, with the largest observed discrepancy being only 0.0121% in the spring design case. Overall, SHGWJA systematically ranked first or second in terms of optimized cost.

Beyond accuracy, SHGWJA also demonstrated computational superiority. Its performance consistently surpassed that of standard and advanced GWO and JAYA formulations, ranking first or second overall in terms of average computational speed. Its fastest runs were competitive with or superior to the best performers in literature. Robustness was another notable strength: SHGWJA achieved a high success rate across all test problems, with near-zero standard deviations on optimized cost in most cases. The variability in the number of function evaluations was modest, never exceeding 35% of the average.

Taking together, these results confirm that SHGWJA is a highly effective tool for engineering design optimization. The algorithm exhibits no theoretical limitations that would undermine its search capability or robustness. Nevertheless, the engineering test problems considered in this Chapter involved at most 14 decision variables. Current research therefore focuses on extending SHGWJA to larger-scale constrained problems, for example by integrating JAYA operators more deeply into the definition of α, β and δ wolves. Preliminary studies on a 200-bar truss structure with 29 variables and 1200 constraints have already confirmed the potential of this approach, with SHGWJA converging to the known optimum.

Further validation using the complete CEC 2020 test suite of 19 benchmark problems, including cases with discrete variables, equality constraints, and topology optimization, showed that SHGWJA reached the target optima within a significantly smaller number of

function evaluations than the budgets specified for the competition. Investigations into scalability have also demonstrated that SHGWJA performs effectively on large structural optimization problems, such as the 200-bar truss with up to 200 design variables and 3500 constraints, achieving structural weights lower than the best previously reported values while remaining consistent with problem formulations.

Although this Chapter focused on single-objective optimization, SHGWJA is readily extendable to multi-objective frameworks, given that both its component algorithms (GWO and JAYA) have already been successfully applied in this context. Future work will explore a multi-objective variant of SHGWJA, potentially using multiple subpopulations to solve independent single-objective tasks and the construction of composite solutions to address the multi-objective formulation.

In conclusion, the investigations presented in this Chapter highlight SHGWJA as a robust, accurate, and computationally efficient hybrid metaheuristic, with strong potential for application in both classical and large-scale engineering optimization problems. The results also indicate promising directions for future research, particularly in scaling the algorithm to more complex constrained problems and extending it to multi-objective optimization, themes that will be further developed in subsequent Chapters of this volume.

CHAPTER V: FHGWJA: AN EFFICIENT FAST HYBRID GREY WOLF-JAYA METAHEURISTIC ALGORITHM FOR DESIGN OPTIMIZATION

Shifting perspectives allows man and machine to see challenges anew, uncovering hidden patterns that simpler views miss. Mirroring strategies guide the search, reflecting possibilities along unseen paths and revealing solutions with speed and elegance.

5.1 INTRODUCTION TO THE FAST HYBRID GREY WOLF-JAYA ALGORITHM (FHGWJA)

In optimization, the goal is to determine the extrema of a function $W(\bar{X})$ with NDV variables subject to $NCON$ constraints, expressed as inequalities $G(\bar{X})$ or also equalities $H(\bar{X})$. Two main algorithmic classes are employed: gradient-based algorithms (GBA), which use derivatives, and metaheuristic algorithms (MAs), which rely on population-based search strategies. MAs are particularly effective for nonlinear and non-convex problems, balancing exploration and exploitation.

This chapter introduces the Fast Hybrid Grey Wolf JAYA Algorithm (FHGWJA) [207], a novel framework developed to improve the Simple Hybrid Grey Wolf JAYA Algorithm (SHGWJA).

FHGWJA merge the Grey Wolf Optimizer (GWO) and the JAYA algorithm with new components to enhance convergence reliability, control the balance between exploration and exploitation, and reduce the computational burden of constraint evaluations.

FHGWJA leverages the strengths of GWO and JAYA while maintaining a streamlined structure. Unlike SHGWJA, FHGWJA uses a multi-stage process: initialization, GWO-based trial generation, rank-based refinement, elitist evaluation and repair, population reordering, convergence check, and termination. It introduces rank-proportional refinement to mitigate alpha-bias, elitist and repair mechanisms to reduce constraint evaluations, and a mirroring strategy to redirect non-improving solutions. These mechanisms enable dynamic adaptation of perturbation magnitude and direction, explicitly controlling the transition from exploration to exploitation. The algorithm is largely parameter-free, requiring only population size and iteration limit, with the latter often optional due to dynamic convergence criteria.

Several algorithms inspired the FHGWJA design. The Harris Hawks Optimization (HHO) algorithm and its variants [208], [209] implement cooperative hunting strategies that dynamically balance exploration and exploitation. Metaheuristic frameworks can adopt parallel or serial architectures [210] to combine multiple algorithms efficiently. Novel hybrid GWO models integrated with algorithms such as PSO, Biogeography-Based Optimization, Differential Evolution, HHO, and the Whale Optimization Algorithm have demonstrated enhanced robustness, faster convergence, and improved adaptability [209], [211], [212], [213], [214], [215], [216], [217], [218], [219], [220]. Similarly, JAYA has been hybridized with methods including Harmony Search, Genetic Algorithms, Crow Search, Rao-1 Optimizer, Teaching-Learning-Based Optimization, and Differential Evolution [163], [221], [222], [223], [224], employing strategies such as sub-populations, inclusion of average and historical solutions, probabilistic candidate selection, fuzzy clustering, competitive and experience learning, Cauchy mutation, selective population updating, and rejection of heavier designs to improve exploration, exploitation, and computational efficiency.

FHGWJA was validated on seven challenging engineering problems with up to 29 variables and 1,200 nonlinear constraints, achieving global or near-global optima with negligible variance and ranking first or second in computational efficiency among state-of-the-art metaheuristics. Its performance gains are attributed to refined exploitation control, constraint-aware repair mechanisms, and mirroring-based descent search.

This chapter is structured as follows: Section 5.2 presents a detailed formulation of the Fast Hybrid Grey Wolf-JAYA Algorithm (FHGWJA); Section 5.3 discusses the algorithm test problems and the corresponding optimization results; finally, Section 5.4 summarizes the main findings and outlines future research perspectives.

5.2 DETAILED FORMULATION OF THE FAST HYBRID GREY WOLF-JAYA ALGORITHM

The new hybrid metaheuristic algorithm FHGWJA developed in this study is now described in detail. The algorithm is comprised of seven steps: (i) initialization; (ii) preliminary definition of trial solutions with classical GWO; (iii) rank-based refinement of preliminary trial solutions defined with classical GWO; (iv) evaluation/repair of trial solutions with elitist strategies and JAYA-based schemes; (v) population reordering to define the new best and worst individuals, and update α, β, δ wolves trying to avoid stagnation; (vi) convergence check; (vii) end of optimization search and output.

Step 1. Initialization

FHGWJA randomly generates a population of NPOP candidate designs (i.e. wolves) following the Equations 4.9, 4.10 and 4.11.

Step 2. GWO phase: preliminary generation of new trial designs

FHGWJA utilizes the classical GWO scheme to preliminarily generate the new trial designs. The best three individuals stored in the population are ranked as wolves α, β and δ. The other individuals of the population are ranked as wolves ω. Wolves encircle the prey during the hunt. Such a behavior is mathematically described by the Equations 4.1 and 4.2.

While grey wolves recognize prey's location in the hunting phase, optimum position (prey) usually is not known a priori in the optimization search. To reproduce wolves' behaviour in the GWO algorithm, it is assumed that the three best solutions of population (i.e. α, β and δ wolves) have a better knowledge of the potential prey's location (i.e. the target optimal solution). The rest of population must be updated using the $\bar{X}_{\alpha}$, $\bar{X}_{\beta}$ and $\bar{X}_{\delta}$ positions of the best three search agents. The generic solution $\bar{X}_{it}$ of population is updated to $\bar{X}_{i,tr}$ as described in Equations 4.5, 4.6 and 4.7.

Step 3. Rank-based refinement of preliminary trial designs

In classical GWO, the new population formed by the trial solutions $\bar{X}_{i,tr}$ obtained by perturbing the individuals $\bar{X}_{it}$ stored in the population in the previous iteration are sorted in terms of penalized cost function values to update positions $\bar{X}_{\alpha}$, $\bar{X}_{\beta}$ and $\bar{X}_{\delta}$. The process ends upon reaching the limit number of iterations or function evaluations defined by the user. However, the classical GWO search scheme has two inherent drawbacks: (i) since the position of the prey (optimal solution) is not known *a priori*, the search is directly biased toward the α wolf that has a null distance from the current best record $\bar{X}_{best}$ as it corresponds to the current best record

itself in view of population sorting; (ii) there is no guarantee that each new trial solutions $\bar{X}_{i,tr}$ may improve the current best record $\bar{X}_{best}$ or at least the corresponding individual $\bar{X}_{it}$ stored in the previous population.

To solve these issues, FHGWJA refines the preliminary trial solutions $\left(\overline{X}_{i,tr}\right)^{prel}$ obtained in Step 1 by means of a rank-based criterion. The population defined in the previous iteration is sorted with respect to the penalized cost function values $W_p(\overline{X}_{it})$. The best solution is assigned the rank 1 while the worst solution is assigned the rank 1/NPOP: let rank(i)=1/i be the rank of the generic individual $\bar{X}_{it}$. The average rate of variation for the penalized cost function with respect to the current best record $\gamma^{(\mathrm{i})} = \frac{W_p^{(i)}}{\Delta S^{(i)}}$ is evaluated for each solution $\bar{X}_{it}$. The $\Delta W_p^{(i)} = [W_p(\overline{X}_{it}) - W_p(\overline{X}_{best})]$ numerator represents the increase in penalized cost function that occurs when design is perturbed by $\Delta S^{(i)} = \|\overline{X}_{it} - \overline{X}_{best}\|$ from the current best record to the generic candidate solution $\bar{X}_{it}$. The average rate of variation $\gamma^{(\mathrm{aver})} = \sum_{\mathrm{i}=1}^{\mathrm{N_{POP}}} \gamma^{(\mathrm{i})} / \mathrm{N_{POP}}$ is hence defined. The refinement step size $\mu^{(\mathrm{i})}$ is defined for each individual $\bar{X}_{it}$ as:

$$\mu^{(\mathrm{i})} = Min\left\{\left(1 - rank^{(\mathrm{i})}\right); \left(\frac{\gamma^{(\mathrm{i})}}{\gamma^{(\mathrm{aver})}}\right)\right\} \tag{5.1}$$

The preliminary trial solution $\left(\overline{X}_{i,tr}\right)^{prel}$ is adjusted to the trial solution $\bar{X}_{i,tr}$ using the equation:

$$\bar{X}_{i,tr} = \left(\bar{X}_{i,tr}\right)^{prel} + \frac{(\bar{X}_{best} - \bar{X}_{it})}{\|\bar{X}_{best} - \bar{X}_{it}\|} \cdot \mu^{(\mathrm{i})} \tag{5.2}$$

The rationale behind Equations 5.1 and 5.2 is the following. All trial solutions generated by GWO must be evaluated to determine whether they improve the design. Individuals that were ranked as high-quality solutions in the previous iteration are likely to maintain their rank in the current iteration. However, a solution's rank may decrease if the cost function changes sharply about the perturbed solution, for instance, if the perturbation leads to an infeasible design and the associated penalty increases. To address this, Equation 5.1 sets small refinement step sizes $\mu^{(i)}$, which helps preserve the overall ranking of the population. Since improvement occurs when moving from $\bar{X}_{it}$ to the current best solution $\bar{X}_{best}$, the vector $\bar{S}_{it} = \overline{X}_{best} - \overline{X}_{it}$ represents a descent direction for the i-th individual. Accordingly, FHGWJA perturbs the design along $\bar{S}_{it}$ to further reduce the cost function, at least relative to the original position $\bar{X}_{it}$.

Step 4. Evaluation and modification/repair of new trial designs: elitist strategies and JAYA scheme

The trial designs $\bar{X}_{i,tr}$ generated by FHGWJA in Steps 1 and 2 are subsequently evaluated. In the classical GWO, if the new design $\bar{X}_{i,tr}$ improves the old design, it directly replaces it in the population. However, this approach requires a new constraint evaluation for each trial design to compute its penalized cost function, which significantly increases the computational burden.

Rather than evaluating all trial designs through costly constraint checks, FHGWJA first applies an elitist pre-selection step to identify candidates with real improvement potential. At this stage, the unpenalized cost $W(\bar{X}_{i,tr})$of each trial solution is compared with the penalized cost $W_p(\bar{X}_{best})$of the current best record: if $W(\bar{X}_{i,tr}) > W_p(\bar{X}_{best})$, the trial solution is immediately rejected, as it cannot outperform the current best even without penalties; if $W(\bar{X}_{i,tr}) < W_p(\bar{X}_{best})$, then $W(\bar{X}_{i,tr})$is compared with $1.1\,W(\bar{X}_{best})$. When $W(\bar{X}_{i,tr}) \leq 1.1\,W(\bar{X}_{best})$, the new solution is provisionally accepted and included in the new population. The 1.1 threshold was introduced to emulate the probabilistic acceptance concept used in Simulated Annealing (SA) [149]. During the exploration phase, large improvements are more probable; hence, the condition $W(\bar{X}_{i,tr}) > 1.1W(\bar{X}_{best})$ rarely occurs. Conversely, in the exploitation phase, the algorithm may need to bypass local minima to progress toward the global optimum. As in SA, FHGWJA allows a limited acceptance of slightly worse candidates to escape stagnation. In classical SA, the acceptance probability of a worse solution is about 0.9 when the ratio between the cost increment $[W(\bar{X}_{i,tr}) - W(\bar{X}_{best})]$and the annealing temperature T is 0.1, meaning that solutions up to 10% worse than $\mathbf{X}_{best}$can still lead to improvement in later iterations. Since each trial design $\bar{X}_{i,tr}$ provisionally included in the new population is generated through the classical GWO mechanism based on the positions of the α, β, and δ wolves, FHGWJA employs a JAYA-based refinement step to prevent stagnation. Accordingly, when $W(\bar{X}_{i,tr}) \leq 1.1W(\bar{X}_{best})$, a new, refined trial design $\bar{X}'_{i,tr}$ is defined as:

$$\bar{X}'_{i,tr} = \bar{X}_{i,tr} + \bar{\lambda}_1 \cdot (\bar{X}_{best} - \bar{X}_{i,tr}) - \bar{\lambda}_2 \cdot (\bar{X}_{\delta} - \bar{X}_{i,tr}) \tag{5.3}$$

where $\bar{\lambda}_1$ and $\bar{\lambda}_2$ are two vectors of NDV random numbers generated in the [0,1] interval. In contrast to Equation 4.8, FHGWJA treats the wolf δ as the worst individual, a strategy that deliberately confines the search to promising regions of the design space to refine solution quality.

The new trial solution $\bar{X}'_{i,tr}$ is compared with $\bar{X}_{i,tr}$ to select the solution that finally updates population. If both conditions $W(\bar{X}'_{i,tr}) < 1.1W(\bar{X}_{best})$ and $W(\bar{X}'_{i,tr}) < W(\bar{X}_{i,tr})$ hold, $\bar{X}'_{i,tr}$ replaces $\bar{X}_{i,tr}$ in the population. Conversely, if $W(\bar{X}'_{i,tr}) < 1.1W(\bar{X}_{best})$ but it occurs $W(\bar{X}'_{i,tr}) > W(\bar{X}_{i,tr})$, the original $\bar{X}_{i,tr}$ is retained. Finally, when $W(\bar{X}'_{i,tr}) > 1.1W(\bar{X}_{best})$, $\bar{X}'_{i,tr}$ is discarded and $\bar{X}_{i,tr}$ remains in the population.

When $W(\bar{X}_{i,tr}) > 1.1W(\bar{X}_{best})$, FHGWJA adopts a repair strategy. In this situation, the trial design $\bar{X}_{i,tr}$ is good enough to avoid immediate rejection, since $W(\bar{X}_{i,tr}) < 1.1W(\bar{X}_{best})$, but not sufficiently promising to be accepted.

The algorithm then applies a JAYA-based correction defined as:

$$\bar{X}'_{i,tr} = \bar{X}_{it} + \bar{\lambda}_1 \cdot (\bar{X}_{best} - \bar{X}_{it}) - \bar{\lambda}_2 \cdot (\bar{X}_{worst} - \bar{X}_{it}) \quad (5.4)$$

where $\vec{l_1}$ and $\vec{l_2}$ are two vectors of NDV random numbers like those of Equation 5.3.

Equation 5.4 enhances exploration by moving each individual toward the best solution and away from the worst one. This step engages the entire population, compensating for cases where the leading wolves (α, β, δ) in the GWO phase fail to guide other agents toward promising regions near the prey (i.e., the current best record). Population renewal proceeds along the descent direction $\bar{X}_{best} - \bar{X}_{it}$, which is likely to improve the current design, while simultaneously avoiding the influence of $\bar{X}_{worst}$, which cannot provide beneficial guidance. Each new trial design $\bar{X}'_{i,tr}$ is initially compared against $1.1W(\bar{X}_{best})$, and retained in the population only if $W(\bar{X}'_{i,tr}) \leq 1.1W(\bar{X}_{best})$. If this JAYA-based refinement fails, three suboptimal solutions, $\bar{X}_{it}$, $\bar{X}_{i,tr}$, $\bar{X}'_{i,tr}$, remain. In such cases, FHGWJA applies a *mirroring strategy* that reverses the non-descent directions relative to the gradient of the cost function ΔW_{best} at $\bar{X}_{best}$, thereby defining new descent directions that are more likely to reduce the objective function.

The mirrored trial solution is defined as:

$$\bar{X}''^{(mirr)}_{i,tr} = \bar{X}_{best} + \eta_{mirr,i} \cdot \bar{S}^{(mirr)}_{it} + \eta_{mirr,i-tr} \cdot \bar{S}^{(mirr)}_{i,tr} + \eta_{mirr,(i-tr)'} \cdot \bar{S}^{(mirr)}_{(i,tr)'} \quad (5.5)$$

where $\eta_{mirr,i}, \eta_{mirr,i-tr}, \eta_{mirr,(i-tr)'} \in [0,1]$ are random coefficients. By combining multiple descent directions, the solution $\bar{X}''^{(mirr)}_{i,tr}$ (represented by $P''^{(mirr)}_{i,tr}$ in Figure 5.1 is highly likely to improve the design. Its evaluation follows the same procedure as other trial solutions within FHGWJA.

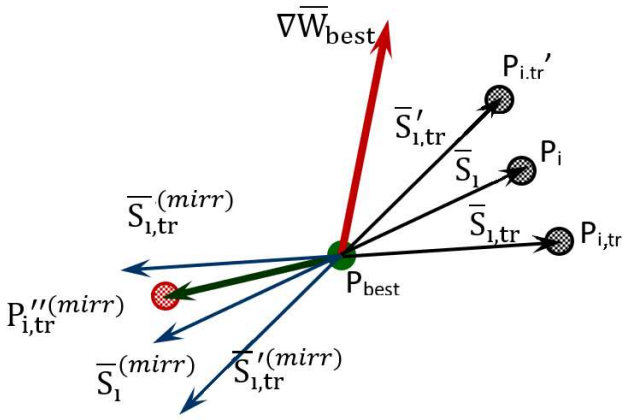


Figure 5.1. SCHEMATIC OF THE "MIRRORING" STRATEGY USED BY FHGWJA IF TRIAL SOLUTIONS $\bar{X}_{it}$, $\bar{X}_{i,it}$, AND $\bar{X}'_{it}$, DO NOT IMPROVE THE CURRENT BEST RECORD.

Step 5. Resort population, update α, β, δ wolves, $\bar{X}_{best}$ and $\bar{X}_{worst}$.

Once FHGWJA updates all candidate solutions $\bar{X}_{it}$, the population is resorted, and the N_{POP}individuals are ranked according to their penalty function values. If any of $W_p(\bar{X}_{i,tr})$, $W_p(\bar{X}'_{i,tr})$ exceeds $W_p(\bar{X}_{it})$, the corresponding new trial solutions generated or refined for $\bar{X}_{it}$ are discarded, and the previous design is retained within the population. Nevertheless, the elitist strategies employing the $1.1 \cdot W(\bar{X}_{best})$ threshold, together with the JAYA-based mechanisms described in Equations 5.3, 5.4 and the "mirroring" approach of Equation 5.5, significantly enhance the likelihood that the updated population will contain designs superior to those in the previous iteration.

The top three individuals of the new population are then designated as the α, β, and δ wolves, associated with the design vectors $\bar{X}_{\alpha}$, $\bar{X}_{\beta}$, and $\bar{X}_{\delta}$, respectively. The overall best and worst solutions are defined as $\bar{X}_{best}$and $\bar{X}_{worst}$.

To prevent stagnation, the ranking of the α, β, and δ wolves is reassessed using an additional elitist criterion whenever FHGWJA fails to update their positions during the current iteration. This criterion again relies on the principle of the descent direction. The vectors $\bar{S}_{\beta} = \left(\bar{X}_{best} - \bar{X}_{\beta}\right)$and $\bar{S}_{\delta} = (\bar{X}_{best} - \bar{X}_{\delta})$represent descent directions relative to the positions $\bar{X}_{\beta}$ and $\bar{X}_{\delta}$ of wolves β and δ, respectively, implying that design improvement occurs when moving toward the α wolf. Consequently, FHGWJA perturbs $\bar{X}_{\beta}$ and $\bar{X}_{\delta}$ along the descent directions $\bar{S}_{\beta}$and $\bar{S}_{\delta}$. The positions of β and δ are then mirrored with respect to $\bar{X}_{best}$ as follows:

$$\begin{cases} \bar{X}_{\beta}^{mirr} = \bar{X}_{best} + \left(1 + \mathrm{h}_{mirr,\beta}\right) \cdot \bar{X}_{best} - \mathrm{h}_{mirr,\beta} \cdot \bar{X}_{\beta} \\ \bar{X}_{\delta}^{mirr} = \bar{X}_{best} + \left(1 + \mathrm{h}_{mirr,\delta}\right) \cdot \bar{X}_{best} - \mathrm{h}_{mirr,\delta} \cdot \bar{X}_{\delta} \end{cases} \tag{5.6}$$

where $\mathrm{h}_{mirr,\beta}$ and $\mathrm{h}_{mirr,\delta}$ are two random numbers in the interval (0,1) that limit step sizes to reduce probability of generating infeasible positions. Among the five candidate positions, $\bar{X}_{\alpha}$, $\bar{X}_{\beta}$, $\bar{X}_{\delta}$, $\bar{X}_{\beta}^{mirr}$, and $\bar{X}_{\delta}^{mirr}$, the three best are selected as the new α, β, and δ wolves for the next iteration. The two worst positions are then compared with the remaining population and may replace $\bar{X}_{worst}$ and the second-worst design from the previous generation. This procedure also accounts for cases where the mirrored positions fail to improve any of the top three wolves.

Figure 5.2 illustrates the mirror strategy adopted by FHGWJA. Specifically, it depicts: (i) the initial locations of wolves α, β, and δ, corresponding to points $P_{opt} \equiv P_{\alpha}$, P_{β}, and P_{δ}in the search space; (ii) the mirrored counterparts $\beta^{mirr}$and δ^{mirr}, defined by Equation 5.16 and corresponding to $P_{\beta,mirr}$ and $P_{\delta,mirr}$; and (iii) the cost function gradient vector ∇W_{best} evaluated at $\bar{X}_{best}$. The mirrored wolves lie along descent directions and may even yield improvements over $\bar{X}_{best}$, as indicated by the scalar products

$\nabla W_{best} \cdot \left(\bar{X}_{\beta}^{mirr} - \bar{X}_{opt}\right) < 0$ and $\nabla W_{best} \cdot (\bar{X}_{\delta}^{mirr} - \bar{X}_{opt}) < 0$.

Since $W_p(\bar{X}_\delta) > W_p(\bar{X}_\beta) > W_p(\bar{X}_{best})$, the direction associated with $\bar{X}_\delta^{mirr}$ is typically steeper, giving wolf δ a higher chance than β to replace α, despite initially occupying a less favorable position.

This elitist mechanism compels FHGWJA to explore new regions of the search space rather than over-exploiting stagnant solutions. By simultaneously discarding the two poorest designs, the algorithm enhances the average quality of the population and increases the likelihood of generating superior trial solutions in subsequent iterations.

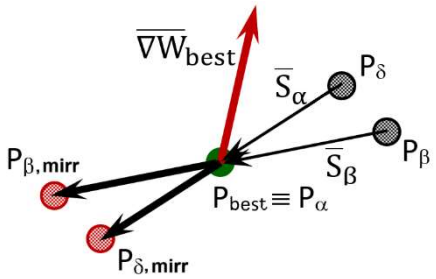


Figure 5.2. SCHEMATIC OF THE ELITIST MIRROR STRATEGY USED BY FHGWJA TO AVOID STAGNATION OF WOLVES α, β AND δ.

Step 6. Check for convergence

As the optimization progresses toward the global optimum, the standard deviations of both the design variables and the penalty function values of the candidate solutions within the updated population are expected to decrease. To monitor this behaviour, FHGWJA normalizes these standard deviations with respect to the average design $\bar{X}_{aver} = \frac{1}{N_{POP}} \sum_{i=1}^{N_{POP}} \bar{X}_{it}$ and the average penalty function value$W_{p,aver} = \frac{1}{N_{POP}} \cdot \sum_{i=1}^{N_{POP}} W_p(\bar{X}_{it})$

The convergence criterion is expressed as:

$$Max \left\{ \begin{matrix} \dfrac{STD\{\|\bar{X}_1 - \bar{X}_{aver}\|, \|\bar{X}_2 - \bar{X}_{aver}\|, \ldots, \|\bar{X}_{NPOP} - \bar{X}_{aver}\|\}}{\|\bar{X}_{aver}\|} \\ \dfrac{STD\{W_{p,1}, W_{p,2}, \ldots, W_{p,NPOP}\}}{W_{p,aver}} \end{matrix} \right\} \leq \varepsilon_{conv} \tag{5.7}$$

where $\varepsilon_{conv} = 10^{-7}$.

Since convergence is expected during the exploitation phase of the search, Equation 5.7 ensures that all agents converge toward the same region containing the global optimum. The first term normalizes the positional standard deviation to quantify population diversity, which must decrease as agents cluster around $\bar{X}_{best}$. The second term applies the same principle to the

penalty function values, ensuring that only solutions effectively close to the global optimum are retained.

Including penalty function values in the convergence test also accounts for potential local infeasibility caused by changes in the set of active constraints near the optimum. In the case of feasible solutions or unconstrained problems, these values reduce to the corresponding cost function values.

Steps 2 to 6 are iteratively executed until FHGWJA achieves convergence toward the global optimum.

Step 7. Terminate optimization process

FHGWJA concludes the optimization by storing the optimal design and the corresponding cost function value in the results file. Figure 5.3 presents the flowchart of the proposed algorithm, which constitutes an enhanced version of the Grey Wolf Optimizer (GWO). At each iteration, the α, β, and δ wolves are updated to ensure the continuous improvement of the current best solution.

Through the integration of JAYA-based schemes and elitist strategies, FHGWJA achieves an effective balance between exploration and exploitation, enhancing population diversity while producing high-quality trial designs with fewer function evaluations. All descent directions are determined without additional constraint evaluations, as algorithmic decisions rely solely on cost function comparisons.

In contrast to many hybrid metaheuristics, FHGWJA does not introduce additional control parameters beyond those of classical GWO and JAYA algorithms, requiring only the population size (N_{POP}) and the maximum number of iterations ($N_{iter,max}$). Its master-slave architecture assigns GWO as the master algorithm responsible for generating trial solutions, while JAYA acts as the slave algorithm refining or correcting them to accelerate convergence.

Typically, GWO and JAYA implementations in the literature require approximately $N_{POP} \cdot N_{iter,max}$ function evaluations (also referred to as function calls or analyses, including constraint evaluations). In contrast, FHGWJA's capability to continuously generate high-quality trial solutions significantly reduces the overall computational cost. Regarding the population size, increasing N_{POP} generally enhances exploration but also results in a substantial rise in computational effort. However, large populations are unnecessary if the optimization engine consistently produces high-quality solutions, as achieved by FHGWJA. Biologically, grey wolves naturally hunt in packs of about 5-11 individuals (occasionally up to 20). Reflecting this behaviour, all FHGWJA optimizations performed in this study employed a population of 10 individuals. Sensitivity analyses confirmed the adequacy of this setting.

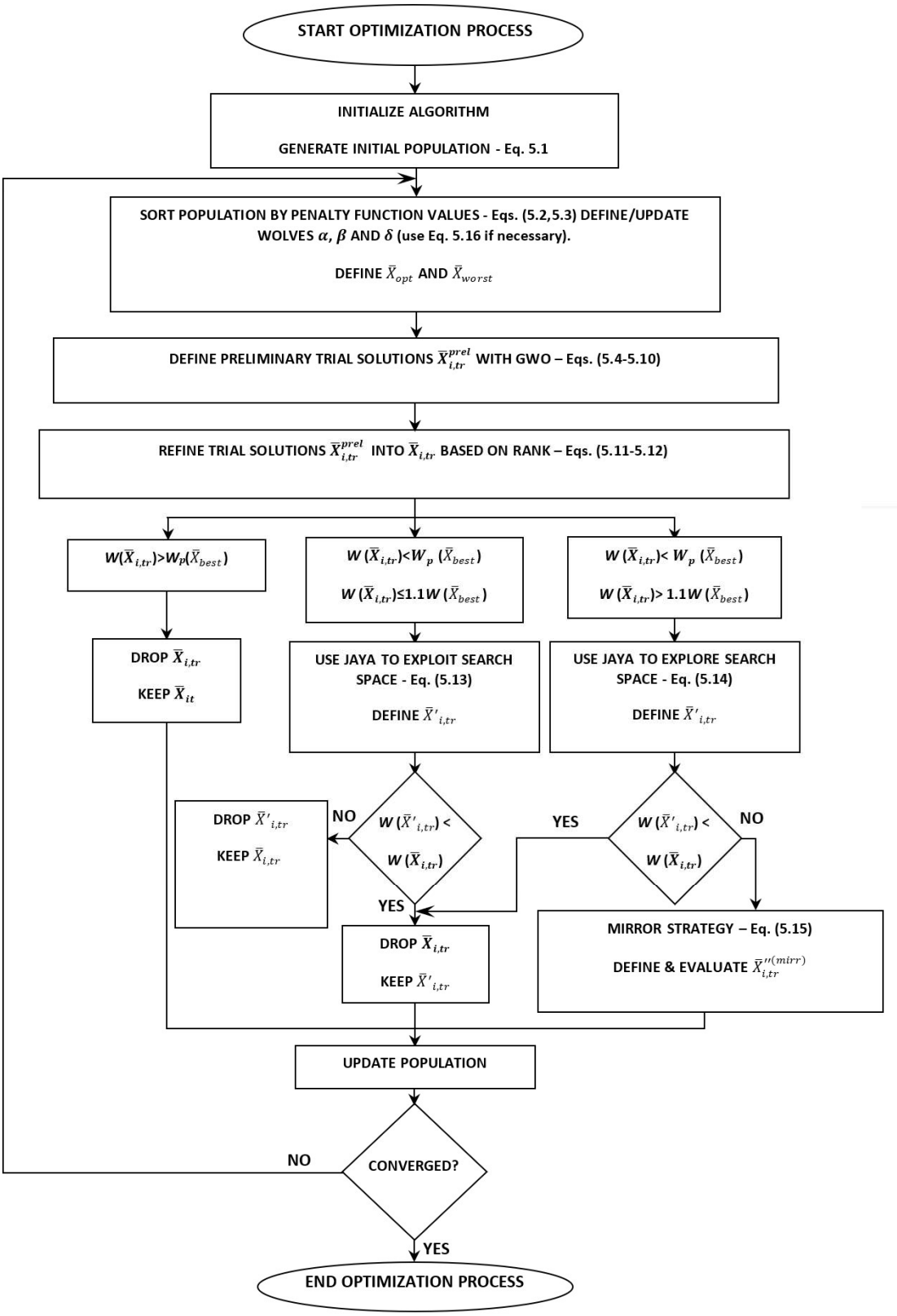


Figure 5.3. FLOW-CHART OF THE FHGWJA ALGORITHM

5.3 FHGWJA TEST PROBLEMS AND OPTIMIZATION RESULTS

The FHGWJA algorithm was validated on seven engineering optimization problems involving up to 29 design variables and 1200 nonlinear constraints.

The test cases included:

(i–ii) 2D path planning (trajectory-length minimization with 7 or 10 obstacles).

(iii) Shape optimization of a concrete gravity dam under earthquake loading (volume minimization).

(iv–v) Calibration of nonlinear hydrologic models (Muskingum problem with 3 and 25 unknown parameters).

(vi) Crashworthiness optimization of a vehicle under side impact.

(vii) Weight minimization of a planar 200-bar truss under three independent loading conditions.

FHGWJA was implemented in MATLAB as a standalone optimization tool and executed on a standard PC equipped with a 3.1 GHz AMD processor and 16 MB RAM. Random numbers were generated using MATLAB's Mersenne Twister (MT19937) algorithm, which ensures high-quality uniform randomness with a period of $2^{19937} - 1$.

To assess the algorithm's statistical performance and robustness, 20 independent runs were performed for each test problem using different initial populations. The population size and maximum iteration limit were set to 10 and 5000, respectively. Initial populations included designs with up to 1000% constraint violation to evaluate FHGWJA's ability to rapidly converge toward feasible and optimal regions.

FHGWJA consistently completed all optimizations in far fewer analyses than the theoretical computational budget of 10 × 5000 = 50000 evaluations required by classical GWO and JAYA. Approximately 90% of its computational cost was attributed to function evaluations. For the largest test case, the 200-bar truss weight minimization problem (29 variables), optimization runs were completed in an average of 25 minutes of wall-clock time.

5.3.1 2D path planning

The theoretical framework for trajectory optimization is presented in Appendix 8.3.

Figure 5.4 illustrates the design space for the two path-planning problems analysed, with all coordinates expressed in millimetres.

In the first case, the objective is to connect the start and end points $A(1,1)$ and $B(29,29)$ in the presence of seven circular obstacles (NOB = 7) located at $O_1(6,6)$, $O_2(7,24)$, $O_3(9,15)$, $O_4(16,23)$, $O_5(20,11)$, $O_6(23,6)$, $O_7(24,23)$. These obstacles define exclusion zones with radius of 3, 1, 3, 2, 2, 1, and 2 mm, respectively.

The corresponding optimization problem includes 14 design variables, the x- and y-coordinates of seven base points (NBP = 7), which can vary between 1 and 29 mm. It also involves NCP =

100 control points and a total of 749 nonlinear constraints, computed as (NCP + NBP) × NOB = (100 + 7) × 7 = 749.

In the second case, three additional obstacles are introduced at $O_8(12{,}10)$, $O_9(14{,}16)$, $O_{10}(19.5{,}17.5)$, each defining circular exclusion zones with a radius of 2 mm. The resulting optimization problem therefore includes 20 design variables (2 × NBP = 2 × 10), NOB = 10 obstacles, NCP = 100 control points, and 1100 nonlinear constraints, according to (NCP + NBP) × NOB = (100 + 10) × 10 = 1100.

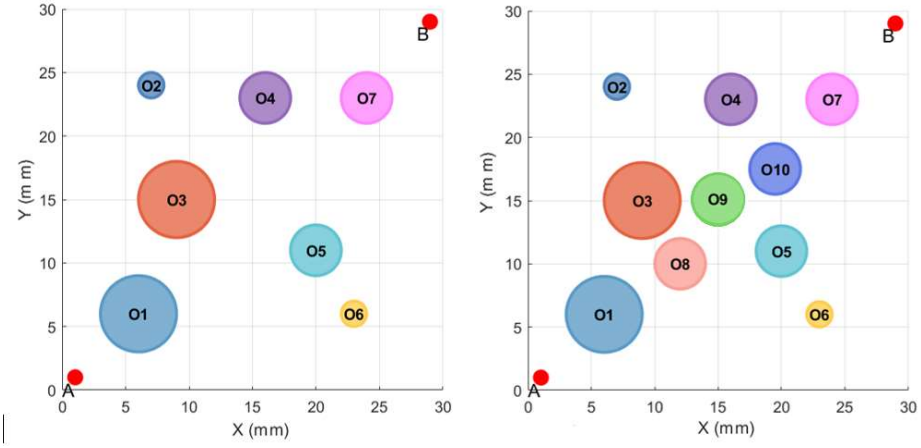


Figure 5.4- 2D PATH-PLANNING DESIGN SPACES WITH 7 AND 10 OBSTACLES.

In addition to 20 independent optimization runs, one further run was conducted with the initial base-point positions coinciding with the obstacle centers (offset by 0.001 mm) to prevent constraint singularities. This configuration represents the worst-case scenario, where the trajectory initially intersects all obstacles. The corresponding path lengths were 231.2 mm (7 base points) and 238.5 mm (10 base points), with initial constraint violations of 159,816% and 204,741%, respectively.

For both problem variants, the proposed FHGWJA hybrid optimizer was compared with leading methods from the literature, hybrid harmony search (HS), hybrid big bang-big crunch (BBBC), and hybrid fast simulated annealing (HFSA), as well as the classical and improved JAYA and grey wolf optimizer (GWO) algorithms. Three additional modified optimizers, mAHS, mBBBC-UBS, and MsinDE, were further enhanced with FHGWJA's elitist strategy $W(\overline{X}_{i,tr}) \leq 1.1 \cdot W(\overline{X}_{opt})$ to assess its contribution to performance.

A population size of 10 was used for all new optimization runs to ensure comparability with FHGWJA and its component algorithms, while all other parameters followed those in the original references.

Table 5.1 summarizes the results for the 2D path-planning problem with 14 design variables (7 base points, 7 obstacles). All listed data refers to feasible solutions, with the number of analyses corresponding to nonlinear constraint evaluations. FHGWJA achieved the shortest trajectory (40.811 mm), outperforming standard GWO (41.104 mm) and standard JAYA (41.050 mm). Its optimal base-point coordinates were (3.66, 1.39), (6.59, 2.79), (16.56, 11.71), (21.23, 17.08), (24.19, 19.83), (27.86, 26.25), and (28.63, 28.20) mm.

The performance ranking was: FHGWJA > hybrid HS [81] > MsinDE [96] > hybrid BBBC [81] > HFSA [81] > mAHS [94] > mBBBC–UBS [95] > improved JAYA [81].

FHGWJA achieved a 100% success rate, with all runs yielding shorter trajectories than the best solutions for competing algorithms. Its worst result (40.890 mm) still outperformed the best of the others (40.983-41.104 mm). This consistency stems from its balanced exploration-exploitation mechanism, which enhances the search capabilities of α, β, and δ wolves while avoiding stagnation. The elitist strategy also improved the performance of mAHS, mBBBC-UBS, and MsinDE, with the latter advancing to third place overall.

In terms of computational efficiency, FHGWJA was the fastest optimizer, requiring on average 4581 analyses, compared to 50000 for GWO and 17204 for JAYA, approximately 42% faster than improved JAYA. Its fastest run converged after 3837 analyses, comparable to hybrid HS (4150), BBBC (3611), and HFSA (4127). Notably, FHGWJA produced a feasible intermediate trajectory of 40.961 mm after only 2875 analyses, already outperforming the final paths obtained by the hybrid algorithms (40.983-40.994 mm).

Table 5.1. Optimization results for the 2D path planning problem solved with 14 design variables.

Algorithm	Best (mm)	Average (mm)	Worst (mm)	Std. Dev. (mm)	Number of Analyses
FHGWJA (present)	40.811	40.859	40.890	3.33×10^{-2}	4581 ± 1127♣
Standard GWO	41.104	41.112	41.129	1.34×10^{-2}	50000
Standard JAYA	41.050	41.092	41.162	6.08×10^{-2}	17204
Improved JAYA [158]	41.082	41.104	41.119	1.85×10^{-2}	6523
Hybrid HS [158]	40.983	41.040	41.105	6.13×10^{-2}	4150
Hybrid BBBC [158]	40.986	41.078	41.101	6.11×10^{-2}	3611
HFSA [158]	40.994	41.048	41.113	6.04×10^{-2}	4127
mAHS (from [186])	41.059	41.067	41.095	1.89×10^{-2}	7237
mBBBC–UBS (from [188])	41.081	41.099	41.112	1.14×10^{-2}	5724
MsinDE (from [189])	40.985	41.002	41.026	2.12×10^{-2}	6291

♣The fastest optimization run of FHGWJA requires only 3837 analyses

FHGWJA ranked sixth in terms of the standard deviation of optimized path length, following mBBBC-UBS, standard GWO, improved JAYA, mAHS, and MsinDE. However, these algorithms required more function evaluations and ultimately converged to inferior designs. FHGWJA also demonstrated strong robustness in computational cost, exhibiting only 24.6% dispersion in the number of analyses across the 20 independent runs.

Figure 5.5 confirms the superior convergence behaviour of FHGWJA compared to its competitors. The figure shows the optimization histories of the best runs for all algorithms in Table 5.1, together with the average FHGWJA convergence curve. For clarity, the plot is limited to the first 4500 analyses and the 40.5-48.5 mm trajectory-length range. Both the best

and average FHGWJA curves consistently lie below those of the other algorithms, indicating faster and more stable convergence.

FHGWJA's best run started from a cost of 225.35 mm, approximately 5.5 times higher than its own global optimum of 40.811 mm, while the initial costs for competing algorithms ranged from 59.62 to 184.21 mm. Despite this disadvantage, FHGWJA rapidly closed the performance gap in the early iterations. Among the competitors, only HFSA [81] and mBBBC-UBS (enhanced with FHGWJA's $W(\overline{X}_{i,tr}) \leq 1.1 \cdot W(\overline{X}_{opt})$ elitist strategy) briefly matched FHGWJA's initial convergence speed, up to approximately 150 and 190 analyses, respectively.

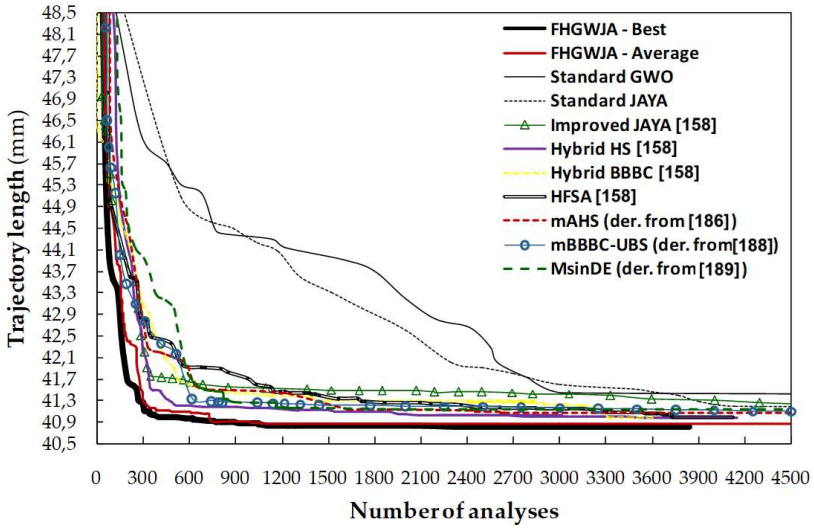


Figure 5.5- CONVERGENCE CURVES OF FHGWJA AND COMPETITORS FOR THE 2D PATH PLANNING PROBLEM(14 DESIGN VARIABLES)

Table 5.2 summarizes the optimization results for the 2D path-planning problem with 20 design variables (10 base points and 10 obstacles). Once again, FHGWJA achieved the best overall performance, producing the shortest trajectory with a length of 40.898 mm. In comparison, the best solutions obtained by the other algorithms were all longer, ranging from 40.997 mm (hybrid HS) to 41.141 mm (improved JAYA).

The worst FHGWJA solution still corresponded to a path length of 40.924 mm, which remained shorter than the best results achieved by any competing optimizer.

The optimal base-point coordinates identified by FHGWJA were (3.9165, 1.7729), (7.9192, 3.6484), (20.853, 16.019), (25.143, 21.071), (27.178, 24.426), (27.185, 24.444), (27.188, 24.449), (27.967, 26.268), (27.998, 26.345), and (28.981, 28.578) mm.

Table 5.2. Optimization results for the 2D path planning problem solved with 20 design variables.

Algorithm	Best (mm)	Average (mm)	Worst (mm)	Std. Dev. (mm)	Number of Analyses
FHGWJA (present)	40.898	40.901	40.924	1.16×10^{-2}	5153 ± 248♣
Standard GWO	41.134	41.146	41.165	1.28×10^{-2}	50 000
Standard JAYA	41.112	41.160	41.187	3.10×10^{-2}	11 778
Improved JAYA [158]	41.141	41.184	41.209	2.81×10^{-2}	11 393
Hybrid HS [158]	40.997	41.053	41.116	4.86×10^{-2}	7408
Hybrid BBBC [158]	41.034	41.091	41.108	3.17×10^{-2}	7542
HFSA [158]	41.004	41.060	41.127	5.03×10^{-3}	8245
mAHS (from [186])	41.075	41.092	41.116	1.68×10^{-2}	8541
mBBBC–UBS (from [188])	41.091	41.123	41.150	2.41×10^{-2}	7492
MsinDE (from [189])	41.030	41.077	41.092	2.64×10^{-2}	8762

♣The fastest optimization run of FHGWJA requires only 5065 function evaluations

FHGWJA was significantly faster than all competing algorithms, completing optimization runs on average within 5153 analyses, compared to 7408 for hybrid HS and 50000 for standard GWO. Standard JAYA was slightly more efficient than improved JAYA, while standard GWO was the least efficient. FHGWJA also demonstrated superior robustness, achieving the lowest standard deviation in path length and only 4.8% dispersion in the number of analyses.

The elitist strategy $W(\bar{X}_{i,tr}) \leq 1.1 \cdot W(\bar{X}_{best})$ implemented in FHGWJA substantially improved the performance of mAHS, mBBBC-UBS, and MsinDE, reducing their average computational cost by 36% compared to the original AHS, BBBC-UBS, and sinDE algorithms [186], [188], [189]. Notably, FHGWJA handled the high nonlinearity of the 2D path-planning problems with 14 and 20 design variables more effectively than the hybrid HS/BBBC/SA algorithms of Ref. [158]. Although these hybrid algorithms leveraged gradient information and multiple line searches to reduce the number of analyses, this approach limited approximation quality and the ability to generate trial solutions in regions likely to improve $\bar{X}_{best}$.

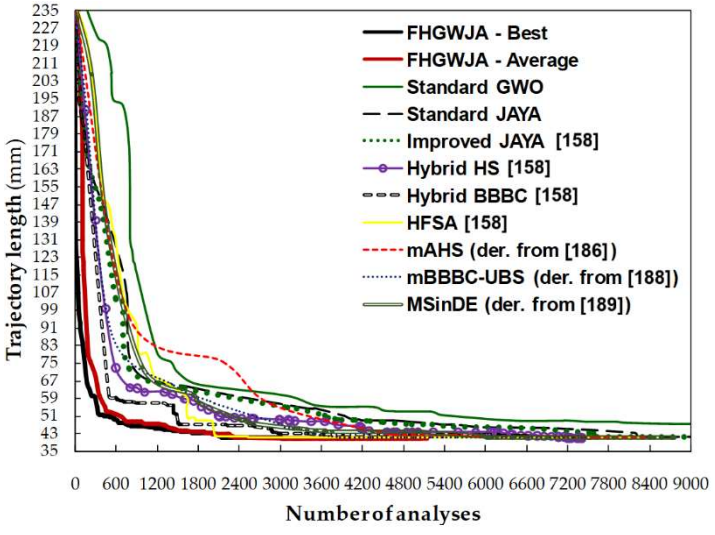


Figure 5.6- CONVERGENCE CURVES OF FHGWJA AND COMPETITORS FOR THE 2D PATH PLANNING PROBLEM (20 DESIGN VARIABLES)

Figure 5.6 presents the convergence histories of the best runs of FHGWJA and competing algorithms for the 2D path planning problem with 20 design variables; the average convergence curve of FHGWJA is also included. For clarity, the plot is limited to the first 9000 analyses. As in the 14-variable case, both the best and average convergence curves of FHGWJA consistently lie below those of the competing algorithms. All optimizers began their best runs from a high initial cost of 238.54 mm, approximately 5.83 times the optimal trajectory length of 40.898 mm achieved by FHGWJA. However, FHGWJA reduced the cost function toward the optimum much more rapidly than the other algorithms. The hybrid simulated annealing algorithm (HFSA) [158] was the only method approaching FHGWJA's convergence rate, and only after 2000 analyses, when the cost was within 5% of the optimum.

Figure 5.7.a compares the optimized trajectories obtained by FHGWJA and its competitors for the 14-variable variant. For clarity, only the six best solutions reported in Table 5.1 are shown. All optimized paths involve obstacles 1 and 7, and the segment between the start point A and obstacle 1 is nearly identical for all algorithms. Trajectories diverge between obstacles 1 and 7, where hybrid HS, HFSA, and standard JAYA exhibit steeper slopes than FHGWJA, hybrid BBBC, and MsinDE. The latter four algorithms produced nearly identical path lengths (40.983-40.994 mm), with trajectories symmetric about the line joining the divergence point near obstacle 1 and endpoint B. Because the path generated by standard JAYA was more curved between obstacles 1 and 7, its total length increased to 41.050 mm.

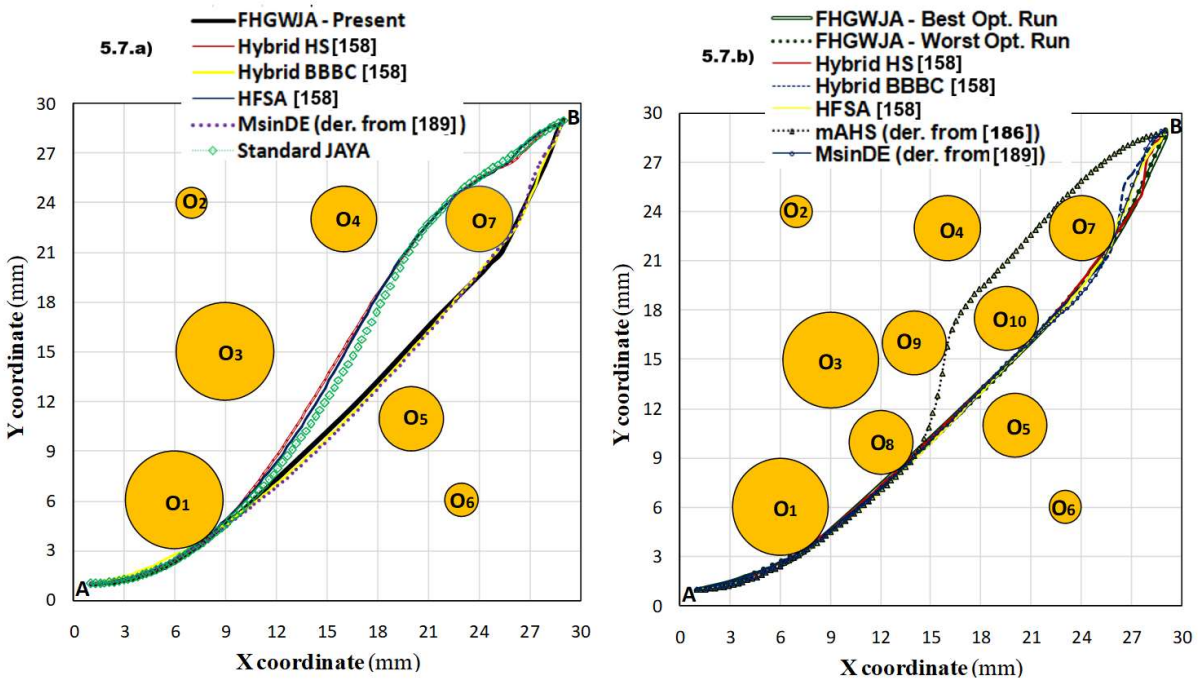


Figure 5.7. COMPARISON OF OPTIMIZED TRAJECTORIES FOUND BY FHGWJA AND ITS COMPETITORS IN THE 2D PATH PLANNING PROBLEM: (a) 7 BASE POINTS AND 14 VARIABLES; (b) 10 BASE POINTS AND 20 VARIABLES.

Detailed trajectory analysis revealed that path length minimization requires rectilinear segments between obstacles 1 and 7 and from obstacle 7 to endpoint B, conditions met exclusively by FHGWJA. The FHGWJA trajectory between obstacles 1 and 7 can be approximated by a linear regression with $R^2 = 0.999$, compared to $R^2 = 0.992$ and $R^2 = 0.994$ for hybrid HS (like HFSA) and hybrid BBBC (like MsinDE), respectively. In the final branch, FHGWJA achieved $R^2 = 0.998$ versus 0.991 and 0.994 for hybrid HS and hybrid BBBC. The trajectories generated by the other algorithms exhibited higher curvature and concavity changes, deviating from the desired straight-line behavior.

Figure 5.7.b compares the optimized trajectories for the 20-variable problem. The figure includes FHGWJA's best and worst runs along with the five best solutions from Table 5.2. The newly introduced obstacles 8 and 10 were active for all algorithms, while obstacle 9 was approached only by mAHS. Consequently, the optimized path lengths of hybrid HS, HFSA, MsinDE, and hybrid BBBC varied by at most 0.037 mm (40.997-41.034 mm), whereas mAHS yielded a slightly longer path of 41.075 mm. Again, FHGWJA produced the most linear trajectories between obstacles 1-7 and 7-B, for both its best and worst runs. In contrast, the hybrid HS, BBBC, SA, and MsinDE trajectories displayed pronounced changes in slope and curvature, particularly between obstacles 10 and 7 and along the final branch. The final FHGWJA path segment achieved a linear fit with $R^2 = 0.997$, compared to 0.965, 0.940, and 0.978 for hybrid HS, hybrid BBBC, and HFSA, respectively.

In conclusion, the FHGWJA algorithm developed in this study demonstrates high efficiency and robustness in solving nonlinear 2D path planning problems in robotics. It consistently achieved shorter trajectories and required fewer analyses to converge than all competing methods.

5.3.2 Shape optimization of a concrete gravity dam under multiple earthquake loads

The test case (iii) solved in this study was the shape optimization of a concrete gravity dam under multiple earthquake loads.

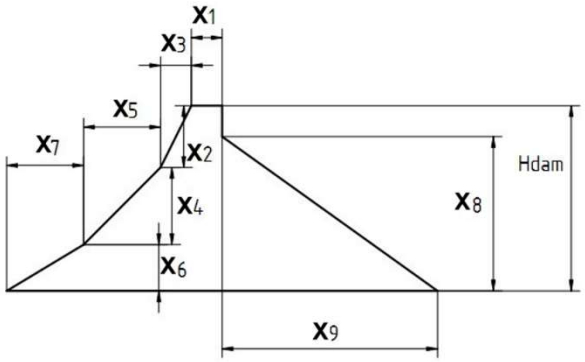


Figure 5.8. CROSS-SECTION OF THE CONCRETE GRAVITY DAM INDICATING SHAPE VARIABLES AND MAIN GEOMETRIC DIMENSIONS.

The optimization problem was stated as:

$$Min\ W(\bar{X}) = H_{dam} \cdot X_1 + X_3 \cdot (X_4 + X_6) + X_5 \cdot X_6 + \frac{X_2 \cdot X_3}{2} + \frac{X_4 \cdot X_5}{2} + \frac{X_6 \cdot X_7}{2} + \frac{X_8 \cdot X_9}{2} \tag{5.8}$$

$$Subject\ to \begin{cases} G_1(\bar{X}) = 4 - \mathrm{SFS} \leq 0 \\ G_2(\bar{X}) = 1.5 - \mathrm{SFO} \leq 0 \\ G_{3-4}(\bar{X}) = 0 \leq \sigma_U \leq \sigma_{MAX} \\ G_{5-6}(\bar{X}) = 0 \leq \sigma_D \leq \sigma_{MAX} \\ G_7(\bar{X}) = X_4 + X_6 - X_8 \leq 0 \\ H_8(\bar{X}) = (X_2 + X_4 + X_6 - H_{dam})^2 = 0 \end{cases} \tag{5.9}$$

Theoretical background is discussed in Appendix 8.6.

The concrete volume per unit width of the actual dam (10502.1 m³) was significantly reduced to 7448.774 m³ by the multi-level cross-entropy optimizer (MCEO) [190] and the flying squirrel optimizer (FSO) [118]. However, neither study accounted for the equality constraint H_8 nor for the horizontal earthquake load $2F_e$. These aspects were included in Ref. [158], which reported results for the hybrid HS, hybrid BBBC, hybrid fast simulated annealing (HFSA), adaptive harmony search (AHS) [186], big bang–big crunch with upper bound strategy (BBBC-UBS) [188], and sinusoidal differential evolution (sinDE) [189]. Accordingly, the FHGWJA algorithm developed in this study was compared with hybrid HS/BBBC/SA, as well as with modified versions, mAHS, mBBBC-UBS, and MsinDE. Like the 2D path-planning problem, these modified algorithms incorporated the elitist strategy $W(\bar{X}_{i,tr}) \leq 1.1 \cdot W(\bar{X}_{best})$ used in FHGWJA to enhance their performance.

As in Ref. [158], a population size of 20 was adopted for hybrid HS, hybrid BBBC, and improved JAYA, while mAHS, mBBBC-UBS, and MsinDE were executed with $N_{POP} = 10$. Table 5.3 summarizes the optimization results for the concrete gravity dam problem. The number of structural analyses corresponds to the evaluations of constraints $G_{1-7}(\bar{X})$and $H_8(\bar{X})$. The table also reports the mean and standard deviation of the number of analyses, along with the minimum value achieved during the fastest run when available.

Table 5.3. Optimization results of the concrete gravity dam design problem.

Algorithm	Best (m³)	Average (m³)	Worst (m³)	ST Dev (m³)	Structural Analyses	Constraint Violation (%)
FHGWJA (Present)	7079.558	7079.558	7079.558	0	3834 ± 1597 (2207)	1.441×10^{-3}
Standard GWO	7117.441	7119.288	7125.568	3.4995	50000	2.423×10^{-3}

Algorithm	Best (m³)	Average (m³)	Worst (m³)	ST Dev (m³)	Structural Analyses	Constraint Violation (%)
Standard JAYA	7121.134	7125.213	7129.822	5.3946	25000	1.338×10^{-3}
Improved JAYA [158]	7117.437	7119.141	7120.890	1.7269	22537 ± 1996 (19917)	4.076×10^{-4}
Hybrid HS [158]	7109.608	7112.340	7113.474	2.5882	18487 ± 3226 (17708)	3.641×10^{-5}
Hybrid BBBC [158]	7108.772	7111.175	7112.275	1.8606	18074 ± 2692 (18128)	3.029×10^{-5}
HFSA [158]	7113.456	7114.225	7114.533	1.5385	19390 ± 1324 (18232)	1.343×10^{-3}
mAHS [186]	7109.419	7111.476	7113.292	2.0719	24687 ± 2916 (22172)	3.333×10^{-3}
mBBBC-UBS [188]	7109.122	7110.306	7111.004	0.7912	18376 ± 1737 (16924)	6.067×10^{-3}
MsinDE [189]	7113.123	7114.350	7115.267	0.8983	24196 ± 2048 (22809)	4.145×10^{-3}
MCEO [190]	7448.74	N/A	N/A	3.274×10^{-2}	35000	32
FSO [118]	7448.74	N/A	N/A	N/A	N/A	32

FHGWJA achieved the best overall performance, yielding the minimum concrete volume per unit width of 7079.558 m³, which is 0.411% lower than the best result previously reported for hybrid BBBC [158] (7108.772 m³). All algorithms significantly outperformed MCEO [190] and FSO [118], whose optimized volumes (7448.774 m³) were markedly higher. The solutions listed in Table 5.3 satisfied all design constraints $G_1(\vec{X}) - G_7(\vec{X})$ and nearly satisfied the equality constraint $H_8(\vec{X})$, with maximum violations below 6.067×10^{-3} %. Based on the optimized volumes, the algorithms ranked as follows: FHGWJA > hybrid BBBC > mBBBC-UBS > mAHS > hybrid HS > MsinDE > HFSA > improved JAYA > standard GWO > standard JAYA.

FHGWJA was also the fastest and most robust algorithm, converging to the global optimum (7079.558 m³) in every run with zero standard deviation. None of the competing methods achieved such consistency. On average, FHGWJA was 4.7-6.5 times faster than its competitors, requiring only 2207 structural analyses in its fastest run, compared to 16924 for mBBBC-UBS, which nevertheless produced a higher volume.

The hybrid formulation of FHGWJA proved substantially more efficient than its component algorithms. Standard GWO and JAYA converged to 7117.441 m³ and 7121.134 m³, respectively, while improved JAYA [158], which employed a simplified elitist rule $W(\overline{X}_{i,tr}) \leq 1.1 \cdot W(\overline{X}_{it})$, did not achieve volumes below 7117.44 m³. Classical GWO required approximately 13 times more analyses than FHGWJA, and classical and improved JAYA were 6.5 and 5.9 times slower, respectively.

Finally, the inclusion of the elitist strategy $W(\overline{X}_{i,tr}) \leq 1.1 \cdot W(\overline{X}_{best})$ significantly enhanced the performance of the modified algorithms mAHS, mBBBC-UBS, and MsinDE relative to

their original versions [186],[188], [189]. Their best results became comparable to those of hybrid HS/BBBC/SA, while average computational costs decreased by 16.5%, 6.1%, and 35.2%, respectively.

Figure 5.9 clearly demonstrates the superior convergence behaviour of the proposed FHGWJA algorithm compared to its competitors. The figure shows the optimization histories of the best runs for the algorithms listed in Table 5.3, along with the average convergence curve of FHGWJA. For clarity, the plot is limited to the first 21000 structural analyses and to the concrete volume interval between 7000 and 14000 m³. It is evident that both the best and average convergence curves of FHGWJA consistently lie below those of the other algorithms. FHGWJA began its best run from a relatively large initial cost of 18372.761 m³, approximately 2.6 times the globally optimal value of 7079.558 m³, whereas the other algorithms started between 13265.001 and 14209.219 m³. Despite this disadvantage, FHGWJA rapidly closed the performance gap and outperformed all competitors. The hybrid fast simulated annealing (HFSA) algorithm of Ref. [158] was the only method to approach FHGWJA's convergence rate, and only during the first 100 structural analyses.

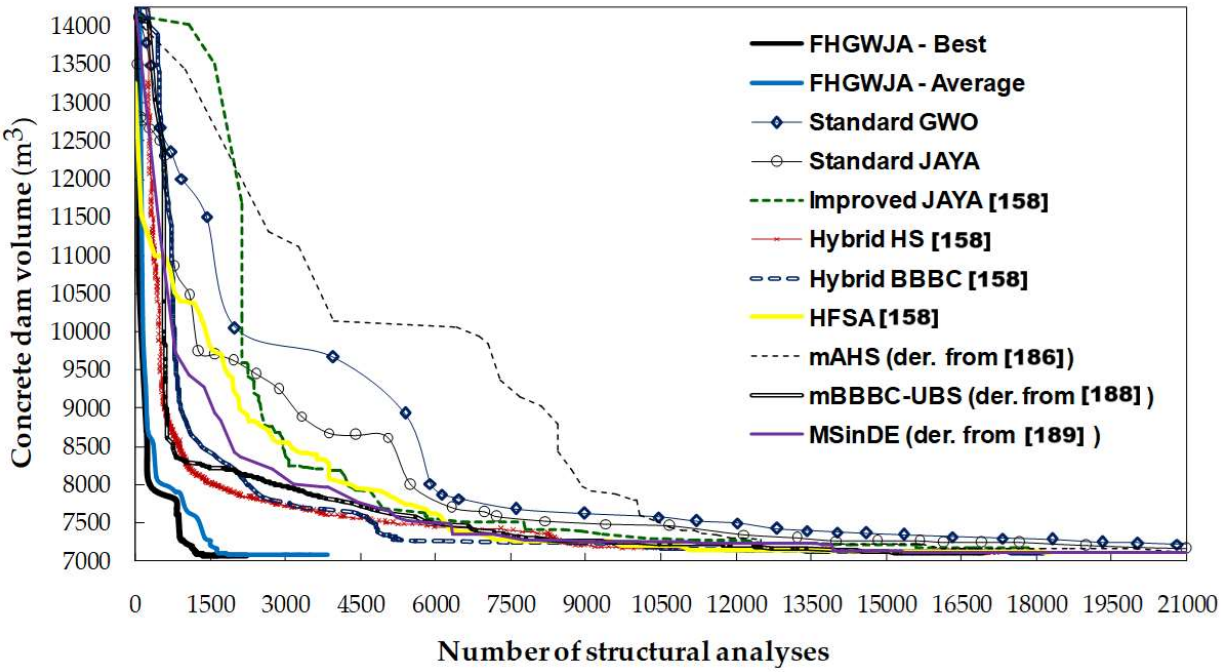


Figure 5.9. COMPARISON OF CONVERGENCE CURVES OF FHGWJA AND ITS COMPETITORS FOR THE CONCRETE DAM OPTIMIZATION

Table 5.4 lists the optimized dam designs obtained by FHGWJA and its principal competitors, with all coordinates expressed in meters. For brevity, results from standard GWO, standard JAYA, and improved JAYA are omitted since they yielded the least effective designs. The optimal design identified by FHGWJA (7079.558 m³) closely resembles those obtained by the other algorithms. All design variables differed by less than 5.6%, except for X_3, which was fixed by FHGWJA at its lower bound (1 mm) but ranged between 4.0001 m and 4.1358 m for the other methods.

Table 5.4. Comparison of optimized designs obtained by FHGWJA and its competitors in the concrete gravity dam problem.

Variables	FHGWJA	Hybrid HS [81]	Hybrid BBBC [81]	HFSA [81]	mAHS [94]	mBBBC-UBS [95]	MsinDE [96]	MCEO/FSO [97,98]	Existing dam
X_1	5.0000	5.0000	5.0006	5.0004	5.0394	5.0175	5.0256	5.1678	4.000
X_2	50.000	49.670	49.554	49.924	49.211	49.333	49.497	25.198	42.000
X_3	1.0000	4.0132	4.0043	4.0001	4.1358	4.0846	4.0866	4.9556	4.200
X_4	69.784	70.084	70.309	68.244	70.006	70.053	69.483	55.781	50.000
X_5	24.835	24.499	24.539	23.955	23.973	24.175	24.066	16.087	12.500
X_6	30.000	30.239	30.131	31.673	30.696	30.924	31.090	21.001	58.000
X_7	40.000	39.997	40.025	39.984	39.974	39.968	39.919	29.904	23.200
X_8	99.770	100.39	100.49	99.241	100.51	100.75	100.26	120.169	140.000
X_9	80.000	80.261	80.164	81.874	80.566	81.348	80.947	85.382	105.000

Figure 5.10 compares the optimized dam profiles (in meters) obtained by FHGWJA and its competitors with the real structure and with the optimal design reported in Refs. [190], [118]. For clarity, the figure presents the average configurations of hybrid HS/BBBC/HFSA and mAHS/mBBBC-UBS/MsinDE. All optimized geometries maintain a base-to-height aspect ratio close to unity, resembling the real dam. However, the optimized profiles are slightly shifted in the positive X-direction relative to the real structure and to the designs in [190], [118], compensating for the additional horizontal earthquake load $2 \cdot F_E$ introduced in the present formulation. The smaller inclination of the upstream bottom segment also increases the reservoir's storage capacity compared to the original dam.The safety factors against sliding and overturning for the FHGWJA-optimized structure were 5.738 and 3.470, respectively, values consistent with those reported in [158]. Thus, the reduction in concrete volume did not compromise structural safety, confirming the effectiveness and reliability of the proposed FHGWJA algorithm for civil engineering design applications.

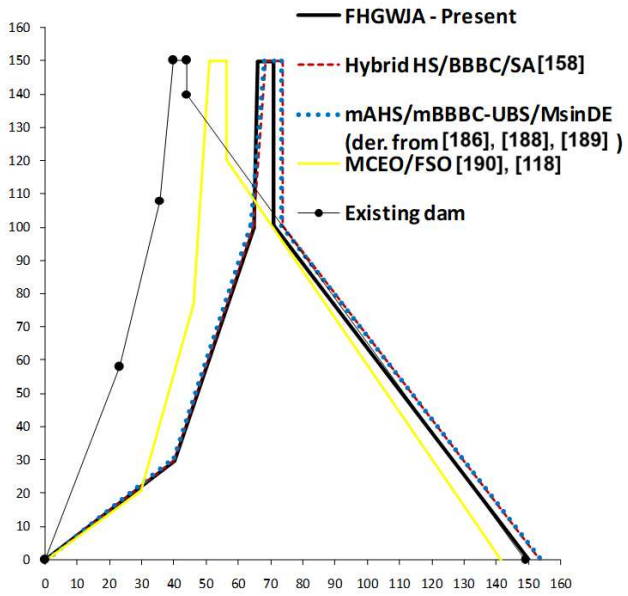


Figure 5.10. COMPARISON OF CONCRETE DAM OPTIMIZED SHAPES OBTAINED BY FHGWJA AND ITS COMPETITORS.

5.3.3 Calibration of nonlinear hydrologic models

Test problems (iv–v) addressed the calibration of nonlinear hydrologic models involving either 3 or 25 unknown parameters, a typical inverse problem in hydrology [225], [226], [227].

Theoretical foundations of the problem are presented in Appendix 8.5.1 and Appendix 8.5.2

Two formulations of the calibration problem were examined. The first corresponds to the classical version with three optimization variables, the Muskingum model parameters K, h, and r.[118], [228], [229], [230], [231] The second, proposed in [158], extends the formulation by incorporating 22 storage values S_t as additional decision variables, resulting in a total of 25 optimization parameters.

The FHGWJA algorithm was benchmarked against several state-of-the-art methods from the literature. These included:(i) metaheuristic algorithms such as the hybrid HS/BBBC/SA schemes [158], other HS variants [118], improved immune clonal selection (IICSA) [228], backtracking search with orthogonal initialization and chaotic map (COBSA) [229], genetic algorithms (GA) [229], differential evolution (DE) [229], and particle swarm optimization (PSO) [229]; (ii) modified adaptive harmony search (mAHS), modified big bang-big crunch with upper bound strategy (mBBBC-UBS), and modified sinusoidal differential evolution (MsinDE), each enhanced by the FHGWJA elastic rule elitist rule $W(\overline{X}_{i,tr}) \leq 1.1 \cdot W(\overline{X}_{best})$ to improve their original versions [188], [189], [190]; (iii) the gradient-based Broyden-Fletcher-Goldfarb-Shanno (BFGS) method [230]; and (iv) the zero-order Nelder-Mead Simplex (NMS) algorithm [231].Comparisons were also made with standard GWO and both standard and improved JAYA algorithms [158].

In consistency with previous experiments, the population size was set to 20 for hybrid HS, hybrid BBBC, and improved JAYA [158], while mAHS, mBBBC-UBS, and MsinDE were executed with $N_{POP} = 10$.

Table 5.5 summarizes the optimization results for the 3-variable variant. The table reports the average and standard deviation of the number of cost function evaluations (NFE), together with the number of evaluations required by the fastest optimization run, when available. FHGWJA consistently converged to the minimum objective value $SSQ = 36.760$ across all independent runs, achieving zero standard deviation and confirming its robustness. Although several algorithms, standard and improved JAYA, modified AHS/BBBC-UBS/sinDE, COBSA, and NMS, also reached the global optimum, none maintained this performance across all runs. Their standard deviations in optimized SSQ ranged from 1.442×10^{-4}(improved JAYA) to 0.01 (COBSA). Other competitors achieved slightly inferior results, with objective values between $SSQ = 36.761$(standard GWO, hybrid HS/BBBC/SA) and $SSQ = 36.77$(HS, IICSA, GA, DE, PSO). In terms of efficiency, FHGWJA significantly outperformed its base components, GWO and JAYA. Standard GWO required 50000 function evaluations to reach $SSQ = 36.761$, approximately 25 times slower than FHGWJA, while standard JAYA converged to $SSQ = 36.760$ but was 2.7 times slower on average. Improved JAYA was about 20.3% slower, though its fastest run (1766 evaluations) was comparable to that of FHGWJA (1637 evaluations). Nevertheless, even in its fastest execution, improved JAYA reached a slightly inferior value of $SSQ = 36.762$, whereas FHGWJA achieved the global optimum in all runs. The superior efficiency of FHGWJA's search operators is further demonstrated by the performance of the elitist strategy $W(\overline{X}_{i,tr}) \leq 1.1 \cdot W(\overline{X}_{best})$. This mechanism enabled mAHS, mBBBC-UBS, and MsinDE to reach the global optimum $SSQ = 36.760$ while reducing computational cost by 9.1%, 11%, and 35.6%, respectively, relative to their original versions[188], [189], [190]. Notably, mAHS and mBBBC-UBS exhibited performance close to FHGWJA, requiring on average 2368 and 2344 evaluations, respectively, compared with only 2005 for FHGWJA, and achieving comparable speeds in their fastest runs (1670 and 1711 vs. 1637 evaluations).

The hybrid HS/BBBC/SA algorithms of Ref. [158] that utilized gradient information and approximate line searches could even complete their fastest optimization runs within less function evaluations than FHGWJA (i.e. between 1281 and 1331 vs. 1637 of FHGWJA) but converged to higher values of SSQ than the global optimum 37.760 found by FHGWJA.

Table 5.5- Optimization results of the hydrologic model calibration problem solved with 3 unknown parameters.

Algorithm	k	h	m	Best SSQ	Average SSQ	Worst SSQ	Std. Dev.	NFE	NFE (best run)
FHGWJA (Present)	0.08620	0.2869	1.8681	36.760	36.760	36.760	0	2005 ± 386	1637
Standard GWO	0.08621	0.2869	1.8681	36.761	36.762	36.764	1.26×10^{-3}	50 000	—
Standard JAYA	0.08620	0.2869	1.8681	36.760	36.761	36.763	1.50×10^{-3}	5438 ± 756	4331

Algorithm	k	h	m	Best SSQ	Average SSQ	Worst SSQ	Std. Dev.	NFE	NFE (best run)
Improved JAYA [81]	0.08620	0.2869	1.8681	36.760	36.761	36.762	1.44×10^{-4}	2412 ± 684	1766
Hybrid HS [81]	0.08633	0.2869	1.8677	36.761	36.762	36.762	4.89×10^{-4}	2547 ± 1035	1331
Hybrid BBBC [81]	0.08618	0.2869	1.8682	36.761	36.761	36.762	1.77×10^{-5}	2468 ± 1102	1281
HFSA [81]	0.08632	0.2869	1.8678	36.761	36.762	36.762	3.89×10^{-4}	2755 ± 1344	1326
mAHS (der. [94])	0.08620	0.2869	1.8681	36.760	36.761	36.762	1.54×10^{-4}	2368 ± 967	1670
mBBBC-UBS (der. [95])	0.08620	0.2869	1.8681	36.760	36.761	36.762	2.04×10^{-4}	2344 ± 835	1711
MsinDE (der. [96])	0.08620	0.2869	1.8681	36.760	36.762	36.763	2.12×10^{-3}	2966 ± 1293	2213
HS [102]	0.0870	0.2870	1.8661	36.77	N/A	N/A	N/A	20 000	—
IICSA [103]	0.0865	0.2870	1.8675	36.77	N/A	N/A	N/A	N/A	—
COBSA [104]	0.0864	0.2869	1.8678	36.76	36.77	36.77	0.01	100 000	—
GA [104]	0.0865	0.2869	1.8674	36.77	36.85	37.24	0.13	N/A	—
DE [104]	0.0863	0.2869	1.8680	36.77	37.09	38.04	0.36	N/A	—
PSO [104]	0.0867	0.2867	1.8668	36.77	36.84	37.25	0.14	N/A	—
BFGS [105]	0.0863	0.2869	1.8679	36.768	N/A	N/A	N/A	N/A	—
NMS [106]	0.0862	0.2869	1.8681	36.76	N/A	N/A	N/A	—	221 *

* Number of optimization iterations

Furthermore, hybrid HS/BBBC/SA required on average more function evaluations than FHGWJA (i.e. between 2468 and 2755 vs. only 2005 of FHGWJA). Again, in a highly nonlinear problem like the hydrologic model calibration, the elitist strategies implemented by FHGWJA were extremely efficient in generating high quality trial solutions over the whole search process. This happened because generation of trial solutions in FHGWJA is not biased to any extent by the quality of gradient information that may instead drive hybrid HS/BBBC/SA towards poor trial solutions should the cost function gradient not be well approximated. However, looking at Table 5.5, it appears that computational cost was not an issue for this test problem: HS [118] and COBSA [229], respectively, required 20000 and 100000 function evaluations vs. only 2000 of FHGWJA while no information on the effective number of function evaluations were given in Refs. [228], [229], [230], [231] for metaheuristic (i.e. IICSA, GA, DE, PSO) as well as gradient-based algorithms (BFGS) and zero-order heuristic methods (NMS).

Figure 5.11 confirms the superiority of FHGWJA over its competitors in terms of convergence behaviour also for this test problem. The figure compares the optimization histories of the best runs for the algorithms listed in Table 5.5; the average FHGWJA convergence curve also is

shown in the figure. Since FHGWJA always converged to the global optimum of 36.760 in all optimizations runs, its best run corresponds to the fastest one. The plot is limited to the first 4500 function evaluations of optimization history and to the 35-70 cost function interval for the sake of clarity. It can be seen from the figure that the best and average optimization runs' convergence curves of FHGWJA lie below those of the other algorithms practically over the whole search history also for this test problem. FHGWJA started its best optimization run from the very large cost of 1520.145 (i.e. about 41.3 times the target optimum of 36.760 quoted in Table 5.5) while the initial cost for all other optimizers ranged between 181.683 and 1122.263. However, the present algorithm immediately recovered the initial gap in cost function with respect to its competitors reducing cost to 37.5 (just 2% more than the target optimum) within only 280 function evaluations while all other algorithms required at least 520 function evaluations to reach the same intermediate cost function value. The hybrid harmony search algorithm of Ref. [158] was competitive with FHGWJA only for the first 150 function evaluations, while the hybrid simulated annealing algorithm (HFSA) of Ref. [158] went close to the average convergence curve of FHGWJA at about 325 function evaluations. The $W(\overline{X}_{i,tr}) \leq 1.1 \cdot W(\overline{X}_{opt})$ elitist strategy of FHGWJA allowed mAHS, mBBBC-UBS and MsinDE to approach the cost function reduction rate of FHGWJA from 400 to 700 function evaluations, while the hybrid big bang - big crunch algorithm of Ref. [158] showed a similar convergence speed to FHGWJA after about 750 function evaluations.

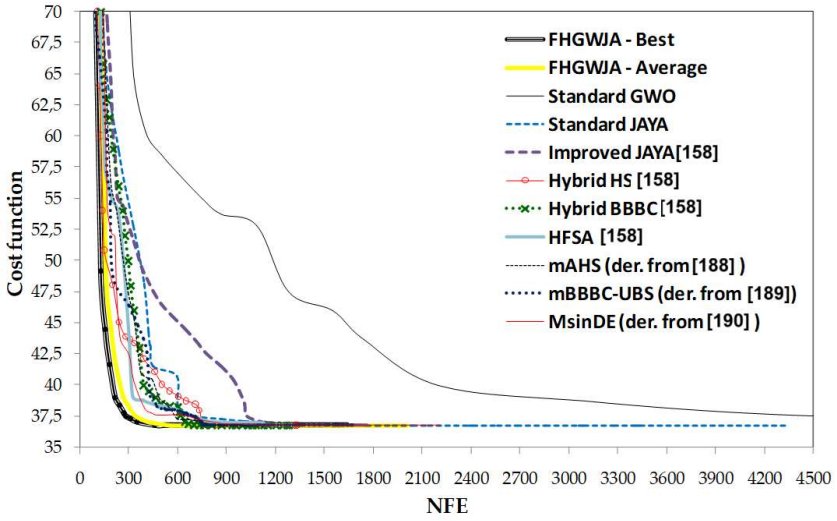


Figure 5.11- COMPARISON OF CONVERGENCE CURVES OF FHGWJA AND ITS COMPETITORS FOR THE 3-VARIABLE HYDROLOGIC MODEL CALIBRATION PROBLEM.

Table 5.6 presents the results obtained by FHGWJA and its competitors in the problem variant with 25 variables; the same nomenclature of Table 5.5 is utilized. For the sake of clarity, the table reports only the optimal values of channel storage S_t (included as optimization variables 4 through 25) found by FHGWJA. The present algorithm was the best optimizer also in this test problem. In fact, it converged to the lowest value overall of SSQ=36.761, followed by the HFSA and hybrid HS/BBBC algorithms of Ref. [158] that obtained SSQ=36.762 and 36.763, respectively. Standard JAYA ranked 5th overall in terms of SSQ but required on average 5.1 times more function evaluations than FHGWJA to complete optimization runs. Improved

JAYA and standard GWO found the worst solutions amongst all optimizers (respectively, SSQ=36.806 and 36.818), requiring on average, respectively, about 1.64 and 11.3 times more function evaluations than the present algorithm. These results confirm the validity of the hybridization scheme implemented in FHGWJA.The $W(\overline{X}_{i,tr}) \leq 1.1 \cdot W(\overline{X}_{best})$ elitist strategy of FHGWJA significantly improved the performance of the AHS/BBBC-UBS/sinDE algorithms of Refs. [188], [189], [190] also in the 25-variables problem variant. In particular, in their best optimization runs, mAHS reduced the optimized cost from 37.078 of AHS [188] to 36.774 (i.e. the 6th best result amongst those quoted in Table 5.6), mBBBC-UBS from 36.826 of BBBC-UBS [189] to 36.786, and MsinDE from 37.101 of sinDE [190] to 36.786. Furthermore, mAHS, mBBBC-UBS and MsinDE reduced average computational cost by 32.7%, 34.4% and 28.4% with respect to the original algorithms of Refs. [188], [189], [190], thus becoming competitive enough with FHGWJA in terms of average computational speed.

Table 5.6. Optimization results of the hydrologic model calibration problem solved with 25 variables.

Algorithm	k	h	m	Best SSQ	Average SSQ	Worst SSQ	Std. Dev.	NFE	NFE (best run)
FHGWJA (Present) *	0.08611	0.2869	1.8681	36.761	36.761	36.761	2.92×10^{-12}	4438 ± 544	4122
Standard GWO	0.08576	0.2870	1.8694	36.818	36.870	36.934	1.40×10^{-2}	50 000	—
Standard JAYA	0.08620	0.2868	1.8677	36.765	36.766	36.769	1.44×10^{-3}	22 663 ± 8359	13 820
Improved JAYA [81]	0.08591	0.2866	1.8689	36.806	36.806	36.807	1.08×10^{-7}	7276 ± 2367	5476
Hybrid HS [81]	0.08614	0.2869	1.8705	36.763	36.763	36.763	3.33×10^{-11}	4744 ± 1954	3277
Hybrid BBBC [81]	0.08567	0.2868	1.8718	36.763	36.763	36.763	8.42×10^{-11}	4863 ± 1607	3656
HFSA [81]	0.08599	0.2867	1.8670	36.762	36.762	36.762	1.03×10^{-10}	5144 ± 1733	3887
mAHS (der. [94])	0.08624	0.2862	1.8678	36.774	36.774	36.775	1.26×10^{-8}	5385 ± 760	4610
mBBBC-UBS (der. [95])	0.08621	0.2872	1.8693	36.786	36.786	36.787	1.15×10^{-8}	5146 ± 797	4488
MsinDE (der. [96])	0.08604	0.2864	1.8676	36.786	36.786	36.787	1.46×10^{-8}	5848 ± 1016	5261

* The optimal channel storage values S_t found by FHGWJA are $\{27.723;26.729;35.245;80.441;147.329;199.195;244.112;279.252;\ 296.016;300.904;291.677;263.154;229.724;187.670;146.693;107.274;75.785;54.842;41.014;30.810;25.732;20.491\}$ m^3.

The superiority of FHGWJA over competing algorithms is evident from its exceptionally low standard deviation of the optimized cost (2.92×10^{-12}), compared with values ranging from 3.33×10^{-11} to 1.40×10^{-2} achieved by the other methods.

Notably, the worst solution obtained by FHGWJA was virtually identical to its best one and consistently superior to the best results of all competitors. In terms of computational efficiency, FHGWJA completed the 20 independent optimization runs with an average of only 4438 function evaluations, outperforming the hybrid HS/BBBC/SA and modified AHS/BBBC-UBS/sinDE algorithms, which required between 4744 and 5848 evaluations.

Although the hybrid HS, BBBC, and SA methods of Ref. [81] reached their fastest runs with fewer evaluations (3277, 3656, and 3887, respectively, versus 4122 for FHGWJA), the present algorithm achieved intermediate solutions equivalent to the best results of hybrid HS and BBBC (SSQ = 36.763) within only 3200 evaluations, and matched the best HFSA result (SSQ = 36.762) within 3700 evaluations.

Figure 5.12 compares the optimization histories of the best runs for the algorithms listed in Table 5.6; the average FHGWJA convergence curve also is shown in the figure. The plot is limited to the first 4500 function evaluations of optimization history and to the 35-80 cost function interval for the sake of clarity.

The convergence behaviour depicted in the figure resembles that observed for the 3-variable problem variant. The best and average optimization runs' convergence curves of FHGWJA again lie below those of the other algorithms practically over the whole search history. FHGWJA started its best optimization run from the very large cost of 3547.602 (i.e. about 96.5 times the target optimum of 36.761 quoted in Table 5.6) while the initial cost for all other optimizers ranged between 461.502 and 2063.123, except the improved JAYA variant of Ref. [158] that started from 75.965.

However, the present algorithm immediately recovered the initial gap in cost function with respect to its competitors reducing cost to 37.2 (just 1.2% more than the target optimum) within only 320 function evaluations while all other algorithms required at least 995 function evaluations to reach the same intermediate cost function value.

The hybrid big bang- big crunch algorithm of Ref. [158] was competitive with FHGWJA only for the first 190 function evaluations. The $W(\bar{X}_{i,tr}) \leq 1.1 \cdot W(\bar{X}_{opt})$ elitist strategy of FHGWJA allowed mAHS and MsinDE to approach the cost function reduction rate of FHGWJA from 220 to 310 function evaluations, while mBBBC-UBS generated the closest intermediate designs to those of FHGWJA between 650 and 1000 function evaluations.

The excellent performance of FHGWJA in the two variants of the hydrologic model calibration problems confirms the suitability of the proposed algorithm for highly nonlinear optimization problems. Interestingly, the higher level of design freedom introduced in the optimization search by the additional 22 variables did not affect at all the robustness of FHGWJA that was able to converge practically to the same values (with at most 0.105% difference) of Muskingum model parameters (i.e. K, h and m) and channel storage values S_t found in the 3-variables problem variant.

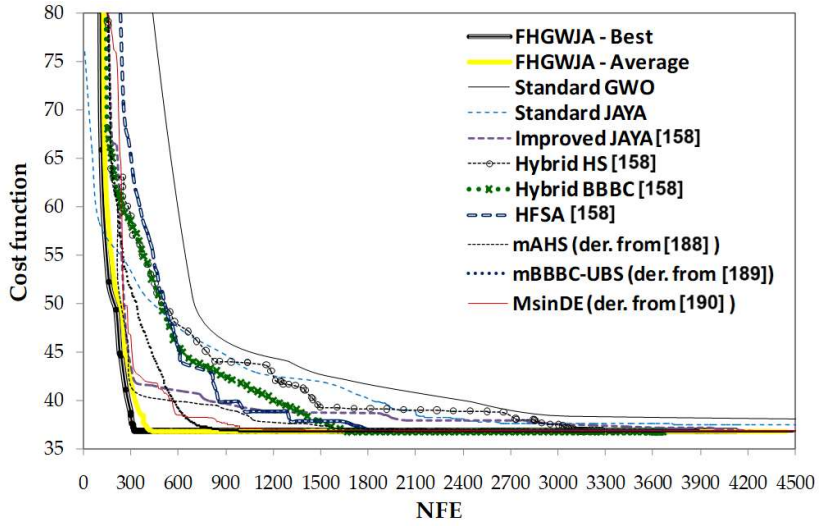


Figure 5.12- COMPARISON OF CONVERGENCE CURVES OF FHGWJA AND ITS COMPETITORS FOR THE 25-VARIABLE HYDROLOGIC MODEL CALIBRATION PROBLEM.

5.3.4 Optimal crashworthiness design of a vehicle subject to side impact

The test case (vi) addressed in this study concerns the optimal crashworthiness design of a vehicle, and the corresponding details are provided in Appendix 8.2.2. The objective is to minimize the vehicle weight under side-impact conditions. Crashworthiness optimization aims to ensure that the vehicle structure effectively absorbs and dissipates impact energy, thereby minimizing the forces transmitted to the occupants. The structural configuration must provide maximum occupant protection without substantially increasing the vehicle's weight or compromising other performance aspects [232], [233], [234]. Figure 5.13 illustrates the structural components considered in the optimization process.

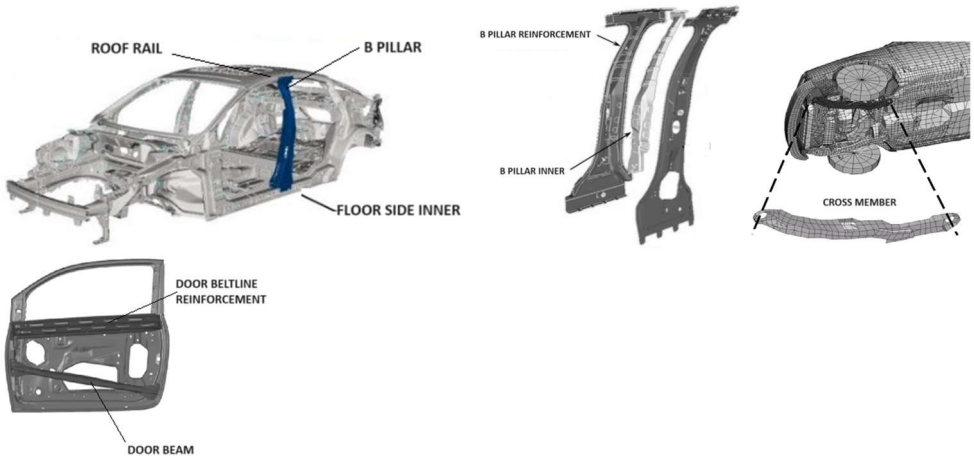


Figure 5.13. Schematic of vehicle parts to be optimized.

The optimal crashworthiness design problem is formulated as follows:

$$Minimize\ W(\bar{X}) = 1.98 + 4.90X_1 + 6.67X_2 + 6.98X_3 + 4.01X_4 + 1.78X_5 + 2.73X_7 \tag{5.10}$$

Subject to 10 Constrains

$$\begin{aligned}
G_1(\bar{X}) &= 1.16 - 0.3717X_2X_4 - 0.00931X_2X_{10} - 0.484X_3X_9 + 0.01343X_6X_{10} - 1 \le 0\\
G_2(\bar{X}) &= 0.261 - 0.0159X_1X_2 - 0.188X_1X_8 - 0.019X_2X_7 + 0.0144X_3X_5 + 0.0008757X_5X_{10} +\\
&\quad +0.08045X_6X_9 + 0.00139X_8X_{11} + 0.00001575X_{10}X_{11} - 0.32 \le 0\\
G_3(\bar{X}) &= 0.214 + 0.00817X_5 - 0.131X_1X_8 - 0.0704X_1X_9 + 0.03099X_2X_6 - 0.018X_2X_7 +\\
&\quad +0.0208X_3X_8 + 0.121X_3X_9 - 0.00364X_5X_6 + 0.0007715X_5X_{10} - 0.0005354X_6X_{10} +\\
&\quad +0.00121X_8X_{11} + 0.00184X_9X_{10} - 0.02{X_2}^2 - 0.32 \le 0\\
G_4(\bar{X}) &= 0.074 - 0.61X_2 - 0.163X_3X_8 + 0.001232X_3X_{10} - 0.166X_7X_9 + 0.227{X_2}^2 - 0.32 \le 0\\
G_5(\bar{X}) &= 28.98 + 3.818X_3 - 4.2X_1X_2 + 0.0207X_5X_{10} + 6.63X_6X_9 - 7.7X_7X_8 + 0.32X_9X_{10} - 32 \le 0\\
G_6(\bar{X}) &= 33.86 + 2.95X_3 + 0.1792X_{10} - 5.057X_1X_2 - 11.0X_2X_8 - 0.0215X_5X_{10} - 9.98X_7X_8 +\\
&\quad +22.0X_8X_9 - 32 \le 0\\
G_7(\bar{X}) &= 46.36 - 9.9X_2 - 12.9X_1X_8 + 0.1107X_3X_{10} - 32 \le 0\\
G_8(\bar{X}) &= 4.72 - 0.5X_4 - 0.19X_2X_3 - 0.0122X_4X_{10} + 0.009325X_6X_{10} + 0.000191{X_{11}}^2 - 4 \le 0\\
G_9(\bar{X}) &= 10.58 - 0.674X_1X_2 - 1.95X_2X_8 + 0.02054X_3X_{10} - 0.0198X_4X_{10} + 0.028X_6X_{10} - 9.9 \le 0\\
G_{10}(\bar{X}) &= 16.45 - 0.489X_3X_7 - 0.843X_5X_6 + 0.0432X_9X_{10} - 0.0556X_9X_{11} - 0.000786 - 15.7 \le 0
\end{aligned} \tag{5.11}$$

with the following boundaries

$$0.5 \le X_1, X_2, X_3, X_4, X_5, X_6, X_7 \le 1.5$$

$$X_8, X_9 \in \{0.192; 0.345\}$$

$$-30 \le X_{10}, X_{11} \le 30$$

The objective function $W(\bar{X})$ represents the structural weight of vehicle components to be minimized, while the constraints $G_i(\bar{X})$ were modelled using response surface approximations [233].

Each constraint corresponds to a specific safety measure: G_1limits the force on the dummy's abdomen; G_2, G_3, and G_4constrain the velocities of the dummy's upper, middle, and lower chest; G_5, G_6, and G_7restrict rib deflections; G_8bounds the force at the pubic symphysis; and

$G_9$and G_{10}regulate velocities at the B-pillar's middle and front-door points, respectively. Forces are given in kN, velocities in mm/s, and deflections in mm.

The best-known solution in the literature was achieved using the multi-strategy fusion improved Gray Wolf Optimization (IGWO) algorithm, yielding a minimized structural weight of 21.39473 kg [215]. IGWO outperformed chaotic GWO, CSO, and the tunicate swarm algorithm (TSA), which achieved optimized weights between 21.46164 and 22.70296 kg. The adaptive self-learning GWO (ASGWO) [216] produced a slightly higher weight (22.87188 kg), while the Starling Murmuration Optimizer (SMO) [140] and the improved continuous ant colony optimizer (LIACOR) yielded similar results (22.84298 kg and 22.84299 kg, respectively). Other algorithms, including KH, ABC, HHO, CLPSO, and WOA, reported optimized weights ranging from 22.88596 to 23.12717 kg.

Table 5.7 compares the optimal solution obtained FHGWJA algorithm developed in this study with those of the optimizers. The results are grouped as follows: (i) FHGWJA and its component algorithms standard GWO and JAYA; (ii) IGWO and its three best competitors compared in [215]; (iii) ASGWO [216]; (iv) SMO and its six best competitors compared in [140]. All designs reported in the table are practically feasible as the maximum constraint violation was 0.001%. Further details on computational cost and statistical performance are provided below the table.

Overall, the proposed FHGWJA algorithm achieved the best performance, converging to an exceptionally low structural weight of 21.38340 kg. In comparison, other algorithms produced weights between 21.39473 kg (IGWO) and 23.12717 kg (WOA). The chaotic GWO ranked third (21.46164 kg), only marginally worse than IGWO and FHGWJA.

However, the optimized discrete variables X_8 and X_9 obtained by IGWO and chaotic GWO do not correspond to the admissible set $\{0.192, 0.345\}$. Once corrected to feasible values, their designs violate constraints $G_6 - G_9$ by up to 13.9-23.7%, rendering them infeasible.

Table 5.7. Comparison of optimized designs obtained by FHGWJA and its competitors in the optimal crashworthiness problem.

Algorithm	X_1 (mm)	X_2 (mm)	X_3 (mm)	X_4 (mm)	X_5 (mm)	X_6 (mm)	X_7 (mm)	X_8 (mm)	X_9 (mm)	X_{10} (mm)	X_{11} (mm)	Structural weight (kg)
FHGWJA* (Present)	0.50000	1.21204	0.50000	0.77908	0.50000	1.49004	0.50000	0.345	0.345	−28.9781	0.00010	**21.38340**
Standard GWO*	0.50002	1.11343	0.50000	1.30794	0.50041	1.50000	0.50011	0.345	0.192	−20.0960	−0.86260	22.84755
Standard JAYA*	0.50000	1.22573	0.50000	1.20711	0.50000	1.49194	0.50000	0.345	0.345	0.00000	0.00000	23.19115
IGWO♣ [215]	0.50000	0.88641	0.50000	1.25781	0.64809	0.91372	0.50000	1.00000	0.52483	1.94790	15.3719	21.39473
Chaotic GWO♣	0.84252	0.50000	0.50000	1.36186	0.83852	0.86445	0.57679	0.94066	0.24622	4.55062	12.3084	21.46164

Algorithm	X_1 (mm)	X_2 (mm)	X_3 (mm)	X_4 (mm)	X_5 (mm)	X_6 (mm)	X_7 (mm)	X_8 (mm)	X_9 (mm)	X_{10} (mm)	X_{11} (mm)	Structural weight (kg)
CSO♣	0.78544	0.56882	0.50000	1.34514	0.82107	0.86975	0.74586	0.89302	0.39281	0.89510	1.17997	22.00444
TSA♣	0.50315	0.89471	0.50000	1.40533	0.86396	0.84245	0.59580	0.86533	0.06579	0.12364	2.06338	22.70296
ASGWO♦ [216]	0.50004	1.13454	0.50009	1.27905	0.50020	1.49996	0.50005	0.34496	0.33248	−16.3332	−2.14912	22.87188
SMO** [140]	0.50000	1.11634	0.50000	1.30224	0.50000	1.50000	0.50000	0.345	0.345	−19.566	0.000001	22.84298
LIACOR**	0.50000	1.11593	0.50000	1.30293	0.50000	1.50000	0.50000	0.192	0.345	−19.640	−0.000003	22.84299
KH**	0.50000	1.14747	0.50000	1.26118	0.50000	1.50000	0.50000	0.345	0.345	−13.998	−0.89840	22.88596
Best ABC**	0.50000	1.30539	0.50000	1.10312	0.50000	0.50000	0.50000	0.345	0.345	14.213	20.3306	22.88605
HHO**	0.50000	1.15627	0.50000	1.27133	0.50000	1.47770	0.50000	0.345	0.192	−14.592	−2.4898	22.98537
CLPSO**	0.50610	1.17379	0.50130	1.24706	0.50370	1.49560	0.50000	0.345	0.345	−9.5985	3.3627	23.06244
WOA**	0.50000	1.09276	0.50000	1.41233	0.50000	1.45497	0.50000	0.345	0.192	−24.038	−3.1789	23.12717

* Optimized weight: 21.3905±0.007125 kg for FHGWJA; 22.9318±0.1423 kg for standard GWO; 23.2067±0.02431 kg for standard JAYA.
Computational cost: 1309±343 structural analyses for FHGWJA; 50000 for standard GWO; 7560±998 for standard JAYA.
The best optimization run of FHGWJA was completed within 1367 structural analyses.
♣ Computational cost: 30000 structural analyses, determined as the product between the population size NPOP=30 and the limit number of iterations Nitermax=1000. No statistical information on optimized weight and computational cost was given in [215].
♦ Computational cost: 15000 structural analyses, determined as the product between the population size NPOP=30 and the limit number of iterations Nitermax=500. No statistical information on optimized weight and computational cost was given in [216].
** Computational cost: 30000 structural analyses, determined as the product between the population size NPOP=20 and the limit number of iterations Nitermax=1500. No statistical information on optimized weight and computational cost was given in [140].

The FHGWJA algorithm achieved an average optimized structural weight of 21.3905 kg, outperforming the best solutions of both IGWO (21.39473 kg) and Chaotic GWO (21.46164 kg). Moreover, it required on average only 1309 structural analyses, compared to the 30,000 evaluations reported in [215] for IGWO and Chaotic GWO.

The proposed hybrid optimizer also surpassed standard GWO and standard JAYA, which yielded heavier designs (22.84755 kg and 23.19155 kg, respectively) and required substantially more analyses (50,000 and 7560). Similarly, the ASGWO [216] algorithm produced a higher optimized weight (22.87188 kg) than standard GWO while reducing computational cost to 15,000 analyses.

In summary, FHGWJA's performance advantage arises from its use of simple elitist strategies and an adaptive trial-solution refinement mechanism that consistently drives the search along descent directions. Unlike other GWO-based hybrids relying on perturbations or multiple metaheuristic components, FHGWJA continuously generates high-quality trial solutions, enabling both superior convergence and efficiency.

The Starling Murmuration Optimizer (SMO) and the algorithms from [216] were also less effective, requiring approximately 30,000 structural analyses to obtain heavier designs

(22.84298–23.12717 kg), compared with only 21.38340 kg achieved by FHGWJA in 1309 analyses.

From a statistical standpoint, FHGWJA demonstrated exceptional robustness: the standard deviation of the optimized weight was only 0.0333% of its mean, while the coefficient of variation for the number of analyses was 26.2%. Comparable statistical data for competing algorithms were unavailable in the literature, likely due to the structure of the crashworthiness design problem, in which the cost function $W(\bar{X})$depends only on variables $X_1, X_2, X_3, X_4, X_5, X_7$, and the constraints primarily involve X_{10}. Consequently, the search space contains many near-optimal yet diverse solutions, leading to significant variation among optimized variables, up to 27.8% for X_1–$X_7$and 259.5% for X_8–X_{11}, the largest fluctuation occurring in X_{10}.

Despite this complexity, FHGWJA consistently identified the global optimum region across all runs, effectively balancing exploration and exploitation. This capability drastically reduced the dispersion of optimized outcomes.

The algorithm's convergence stability and robustness are illustrated in Figure 14, which compares the optimization histories of FHGWJA, GWO, and JAYA. The curves for FHGWJA's best and average runs nearly coincide after 830 analyses (≈ 60.7% of the total process), highlighting its consistency. Starting from an initial structural weight of 40.032 kg, about 87.2% above the final optimum, FHGWJA achieved a design only 1.48% worse than the global optimum after merely 350 analyses (≈ 26% of the process), confirming its fast convergence and high efficiency.

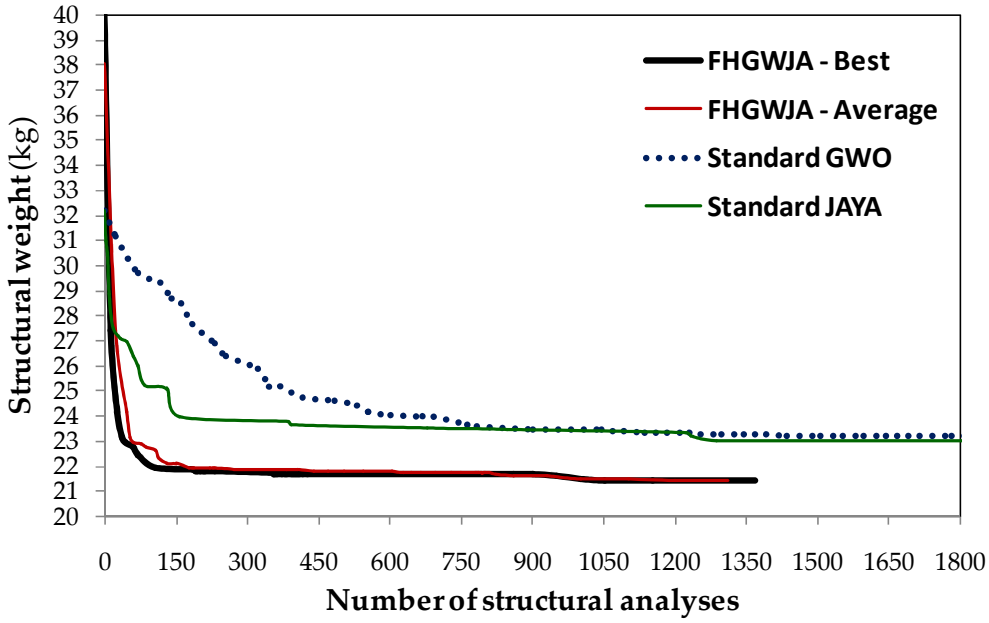


Figure 5.14. Comparison of convergence curves of FHGWJA and its competitors for the optimal crashworthiness problem.

The results presented in this section further confirm the effectiveness and robustness of FHGWJA in addressing complex engineering optimization problems. This conclusion is reinforced by its comparison against 14 state-of-the-art metaheuristic algorithms, or 28 in total, considering all methods evaluated in Refs. [215], [216], [140], including three advanced GWO variants developed only a few months prior to the present study.

5.3.5 Weight minimization of a planar 200-bar truss structure

The final benchmark analysed in this study concerns the weight minimization of a planar 200-bar steel truss (see Appendix 8.7), illustrated in Figure 5.15. The structure comprises 200 elements (NEL) and 77 nodes, and it is made of steel with a Young's modulus of 206.91 GPa and a mass density of 7833.413 kg/m³. Due to structural symmetry, the elements are organized into 29 groups, each sharing the same cross-sectional area, as reported in Table 5.8. Accordingly, the optimization problem involves 29 design variables, $X_i = A_i (i = 1, \dots, 29)$, where A_i represents the cross-sectional area of the i^{th}group.

The truss is subjected to three independent loading conditions:

(a) a 4.45 kN (1000 lbf) horizontal load acting in the positive X-direction at nodes 1, 6, 15, 20, 29, 34, 43, 48, 57, 62, and 71.

(b) a 44.497 kN (10,000 lbf) vertical load acting in the negative Y-direction at nodes 1-6, 8, 10, 12, 14-20, 22, 24, 26, 28-34, 36, 38, 40, 42-48, 50, 52, 54, 56-62, 64, 66, 68, 70-75.

(c) a combined loading case, in which both (a) and (b) act simultaneously.

The optimization aims to minimize the total truss weight subject to 1200 nonlinear stress constraints, ensuring that the axial stress in each element remains within ±68.97 MPa (±10,000 psi) under both tension and compression.

The optimization problem can thus be formally stated as:

$$Minimize\ W(\bar{X}) = \rho\gamma \sum_{j=1}^{NEL} A_j\, L_j \tag{5.12}$$

$$Subject\ to\ the\ following\ Constrains$$

$$\begin{cases} G(\bar{X}) = \frac{\sigma_{j,ilc}}{\sigma_{t,lim}} - 1 \le 0 \\ G(\bar{X}) = \frac{\sigma_{j,ilc}}{\sigma_{c,lim}} - 1 \le 0 \end{cases} \qquad \begin{cases} j = 1, \dots, NEL \\ ilc = 1,2,3 \end{cases} \tag{5.13}$$

$with\ the\ following\ boundaries, for\ i = 1, \ldots, 29$

$0.1 \leq X_i \leq 100\ (\ in^2)$

$or\ equivalently$

$6.4516 \cdot 10^{-5}\ \leq X_i \leq 6.4516 \cdot 10^{-2}\ \ (\ m^2)$

In Equation (5.12), ρ and γ denote, respectively, the mass density and the specific weight of the material. NEL represents the number of truss elements, while A_j and L_j correspond to the cross-sectional area and length of the j^{th}element, respectively. The design vector $\overline{\text{X}}$ contains the 29 sizing variables, each defining the cross-sectional area associated with one of the 29 groups of the 200 truss elements.

In Equation (5.13), $\sigma_{j,ilc}$ denotes the axial stress in the j^{th} element under the ilc^{th} loading condition, while $\sigma_{t,lim} = 10000$ psi and $\sigma_{c,lim} = -10000$ psi represent the allowable stresses in tension and compression, respectively.

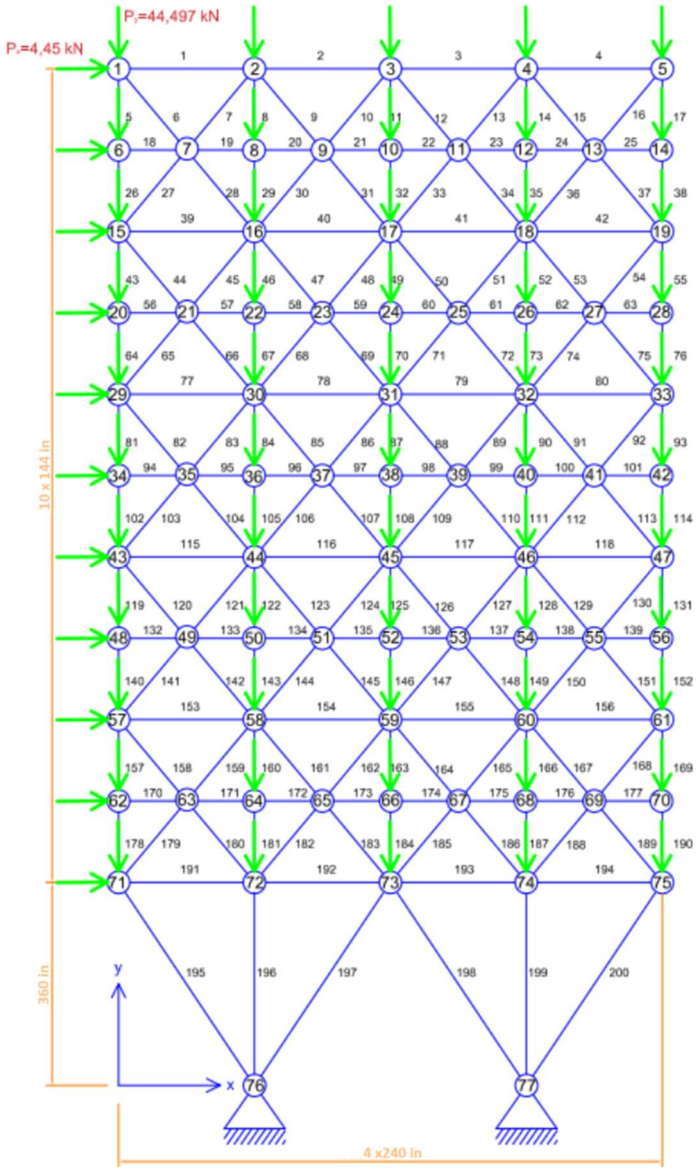


Figure 5.15. Schematic of the planar 200-bar truss including applied loads and kinematic constrains.

Table 5.8. Groups of elements defined in the 200-bar truss design problem.

Group	Elements	Group	Elements
1	1, 2, 3, 4	16	82, 83, 85, 86, 88, 89, 91, 92, 103, 104, 106, 107, 109, 110, 112, 113
2	5, 8, 11, 14, 17	17	115,116,117,118
3	19, 20, 21, 22, 23, 24	16	119, 122, 125, 128, 131
4	18, 25, 56, 63, 94, 101, 132, 139, 170, 177	19	133, 134, 135, 136, 137, 138
5	26, 29, 32, 35, 38	20	140, 143, 146, 149, 152

6	6, 7, 9, 10, 12, 13, 15, 16, 27, 28, 30, 31, 33, 34, 36, 37	21	120, 121, 123, 124, 126, 127, 129, 130, 141, 142, 144, 145, 147, 148, 150, 151
7	39, 40, 41, 42	22	153, 154,155, 156
8	43,46,49,52,55	23	157, 160, 163, 166, 169
9	57,58,59,60,61,62	24	171, 172, 173, 174, 175, 176
10	64, 67, 70, 73, 76	25	178, 181, 184, 187, 190
11	44, 45, 47, 48, 50, 51, 53, 54, 65, 66, 68, 69, 71, 72, 74, 75	26	158, 159, 161, 162, 164, 165, 167, 168, 179, 180, 182, 183, 185, 186, 188, 189
12	77, 78, 79, 80	27	191, 192, 193, 194
13	81, 84, 87, 90, 93	28	195, 197, 198, 200
14	95, 96, 97, 98, 99, 100	29	196,199
15	102, 105, 108, 111, 114		

This case study represents a medium-scale structural optimization problem that has been widely examined in the literature. The non-convex search space makes this problem particularly challenging for metaheuristic algorithms. The global optimum weight is 11542.4 kg, while a local optimum exists at 11544 kg.

Among previously reported methods, the hybrid Harmony Search / Big Bang-Big Crunch / Simulated Annealing (HS/BBBC/SA) algorithms [158] achieved 11542.409 kg, effectively matching the global optimum. Other high-performing optimizers include: ECM, based on the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [235]: 11543.521 kg; LSHADE-Epsin, a self-adaptive differential evolution variant [236]: 11543.980 kg; MCOA, a modified Coyote Optimization Algorithm using chaotic sequences [237]: 11544.007 kg; CNNT-PSO, a particle swarm optimizer using cyclic neighborhood topology [238]: 11545.330 kg; hGMO, a hybrid Geometric Mean Optimizer with variable neighborhood search [239]: 11545.568 kg.

Some advanced metaheuristics produced lighter but infeasible designs, violating stress limits. For example, the CMLPSA algorithm [149] achieved 11542.137 kg with a 0.0709% constraint violation, while HPSACO [150]obtained 11410.737 kg but violated stress limits by 9.97%. Similarly, Improved HHO [240], PSOHHO [241], and Biogeography-Based Optimization (BBO) [242] yielded structural weights of 11533.983 kg, 11374.477 kg, and 11320.758 kg, with violations of 9.55%, 27.6%, and 11.33%, respectively.

For fair comparison, the FHGWJA algorithm proposed in this study was benchmarked against the best feasible optimizers, namely, HS/BBBC/SA [158], ECM [235], LSHADE-Epsin [236], MCOA [237], CNNT-PSO [238], and hGMO [239], as well as standard and improved versions of GWO [243] and JAYA [158]. The Political Optimizer (PO) [244] was also included due to its performance being comparable with classical GWO and JAYA. These comparisons are particularly meaningful, as each of the selected algorithms was previously reported to outperform between 5 and 76 other state-of-the-art metaheuristics.

Reported population sizes varied significantly, 50 for HS [158], MCOA [237], and hGMO [239]; 100 for BBBC [158]; 20 for improved JAYA [158]; 14 for ECM [235]; 1000 for CNNT-PSO [238]; and 121 for PO [244], highlighting the inconsistent computational demands of existing methods. Notably, FHGWJA required the smallest population, owing to its intrinsic ability to consistently generate high-quality trial solutions throughout the optimization process.

Table 5.9 summarizes the optimization outcomes for all methods. It reports optimized cross-sectional areas (in in^2), the best, average, and worst structural weights, associated standard deviations, and the number of structural analyses (NSA) performed. For brevity, algorithms producing infeasible designs are [150], [241], [240], [242] excluded.

Results confirm that FHGWJA outperformed all competitors, achieving the lowest structural weight of 11541.380 kg, marginally below the global optimum. All solutions generated by FHGWJA were feasible, with maximum element stresses of 10000.412 psi, effectively satisfying the 10000 psi limit. The hybrid HS/BBBC/SA algorithms [158] were also competitive, reaching the target optimum with minimal variation (standard deviation 0.00622 kg vs. 0.4157 kg for FHGWJA). The average computational cost of FHGWJA, 3356 structural analyses, was comparable to the mean of the hybrid HS (2736), BBBC (1676), and fast SA (5806) algorithms, confirming its strong balance between accuracy, robustness, and efficiency.

The best-performing competitors of FHGWJA, namely ECM [235], LSHADE-Epsin [236], MCOA [237], CNNT-PSO [238], and hGMO [239], achieved optimized weights ranging from 11543.521 kg (ECM) to 11545.568 kg (hGMO). However, they required between 20000 (hGMO) and 290000 (LSHADE-Epsin) structural analyses, compared with only 3356 analyses on average for FHGWJA.

FHGWJA also clearly outperformed its component algorithms. The best designs obtained by improved JAYA [158], standard JAYA, and standard GWO weighed 11550.054 kg, 11553.355 kg, and 11558.650 kg, respectively, all heavier than the 11541.380 kg achieved by FHGWJA. Moreover, these algorithms required substantially higher computational effort, 31580, 40941, and 50000 structural analyses, respectively, versus only 3356 for the proposed method. The improved GWO (IGWO) variant of Ref. [243], employing an exponential variation scheme for the control vectors $\bar{A}_1, \bar{A}_2, \bar{A}_3$ in Equations 4.3 and 4.4, prematurely converged to 11689.883 kg within 23760 analyses. Similarly, the Political Optimizer (PO) [244] produced a slightly lighter design than standard GWO, 11557.384 kg versus 11558.650 kg, but still required 28000 analyses, well above the computational effort of FHGWJA.

Table 5.9. Comparison of optimized designs obtained by FHGWJA and its competitors in the 200-bar truss problem.

Design variable	FHGWJA (Present)	Standard GWO	Standard JAYA	Improved JAYA [81]	Hybrid HS/BBBC/ SA [81]	ECM [110]	LSHADE-Epsin [111]	MCOA [112]	CNNT-PSO [113]	hGMO [114]	PO [120]
1	0.1484	0.1467	0.1473	0.1473	0.1484	0.1471	0.14838	0.1390	0.1482	0.1484	0.13911
2	0.9447	0.9415	0.9417	0.9404	0.9445	0.9399	0.94448	0.9355	0.9405	0.9408	0.96277
3	0.1000	0.1001	0.1002	0.1001	0.1000	0.1000	0.10000	0.1000	0.1000	0.1000	0.10996
4	0.1000	0.1006	0.1001	0.1001	0.1000	0.1000	0.10000	0.1000	0.1000	0.1000	0.10000
5	1.9434	1.9419	1.9423	1.9412	1.9445	1.9399	1.94448	1.9355	1.9408	1.9409	1.94300
6	0.2976	0.2965	0.2970	0.2968	0.2980	0.2965	0.29796	0.2909	0.2975	0.2975	0.29526
7	0.1000	0.1001	0.1001	0.1001	0.1000	0.1000	0.10000	0.1000	0.1000	0.1000	0.10006
8	3.1177	3.1147	3.1076	3.1067	3.1227	3.1049	3.12260	3.0816	3.1067	3.1097	3.09870
9	0.1000	0.1001	0.1001	0.1001	0.1000	0.1000	0.10000	0.1000	0.1000	0.1000	0.13761
10	4.1149	4.1416	4.1093	4.1081	4.1227	4.1049	4.12260	4.0816	4.1067	4.1097	4.09920
11	0.3989	0.4022	0.4034	0.4040	0.3990	0.4037	0.39897	0.3967	0.4057	0.4047	0.41926
12	0.1000	0.1871	0.1931	0.1931	0.1000	0.1906	0.10044	0.2959	0.1897	0.1722	0.15592
13	5.3783	5.4324	5.4406	5.4342	5.3934	5.4298	5.39336	5.3854	5.4343	5.4303	5.43440
14	0.1000	0.1001	0.1001	0.1001	0.1000	0.1006	0.10000	0.1000	0.1000	0.1179	0,10437
15	6.3731	6.4298	6.4360	6.4342	6.3934	6.4298	6.39336	6.3853	6.4340	6.4303	6.43550
16	0.5262	0.5728	0.5745	0.5753	0.5264	0.5739	0.52663	0.6332	0.5745	0.5745	0.56508
17	0.4521	0.1456	0.1352	0.1355	0.4353	0.1332	0.43485	0.1842	0.1366	0.1332	0.15753
18	7.9242	7.9712	7.9825	7.9802	7.5904	7.9744	7.95063	8.0396	7.9803	7.9762	7.96720
19	0.1000	0.1001	0.1002	0.1002	0.1000	0.1000	0.10000	0.1000	0.1000	0.1001	0.10012
20	8.9151	8.9715	8.9829	8.9804	8.5904	8.9744	8.95063	9.0395	8.9802	8.9762	8.96750
21	0.8691	0.7147	0.7092	0.7090	0.8592	0.7064	0.85901	0.7460	0.7109	0.7096	0.72223
22	0.1555	0.4572	0.4363	0.4372	0.1500	0.4339	0.14995	0.1306	0.4659	0.4441	0.48433
23	10.9621	10.8968	10.8940	10.8912	10.9977	10.8790	10.9977	10.9114	10.9110	10.8931	10.9130
24	0.1219	0.1001	0.1001	0.1002	0.1000	0.1000	0.100074	0.1000	0.1000	0.1000	0.10472
25	11.9512	11.8973	11.8948	11.8914	11.9977	11.8790	11.9977	11.9114	11.9112	11.8931	11.9140
26	0.9359	1.0598	1.0394	1.0491	0.9125	1.0453	0.91252	0.8627	1.0712	1.0548	1.08420
27	6.5048	6.5394	6.6153	6.6106	6.6620	6.6300	6.66179	6.9169	6.5030	6.5761	6.50930
28	10.8708	10.8368	10.8721	10.7791	10.8061	10.7827	10.8061	10.9674	10.7210	10.7574	10.7000
29	13.8713	13.9062	13.8872	13.8783	13.8236	13.8691	13.8237	13.6742	13.9310	13.8942	13.9520
Best (kg)	11541.380	11558.650	11553.355	11550.054	11542.409 11542.409 11542.409	11543.521	11543.980	11544.007	11545.330	11545.568	11557.384
Average (kg)	11541.716	11572.101	11564.050	11556.377	11542.410 11542.410 11542.411	11580.980	11545.681	11576.616	11548.048	11609.697	11618.419
Worst (kg)	11542.390	11590.931	11579.700	11569.059	11542.411 11542.410 11542.415	N/A	11548.448	11592.701	11551.227	N/A	N/A
ST DEV (kg)	0.4157	37.109	25.403	10.941	0.00141 0.00010 0.00622	19.125	0.9888	21.600	1.431	39.694	170.601
NSA	3356 ± 431 3464	50000	40941	31580	2736 ± 283* 1676 ± 137* 5806 ± 304*	96600	290000	27720	150000	20000	27984

*Best optimization run completed by Hybrid HS in 2912 structural analyses; by Hybrid BBBC in 1669 analyses; by HFSA in 5600 analyses.

The FHGWJA ranked fourth overall in terms of standard deviation of optimized weight yet achieved a 100% success rate in identifying feasible solutions lighter than the target optimum (11542.4 kg) and consistently avoided convergence to the local minimum (11544 kg). The ratio between the standard deviation and the mean computational cost was only 12.8%, confirming its numerical stability.

The elitist strategy $W(\bar{X}_{i,tr}) \leq 1.1\, W(\bar{X}_{best})$ again proved effective, significantly enhancing the performance of the AHS, BBBC-UBS, and sinDE algorithms of Refs. [188],[189], [190]. These methods, originally reported in [158] to achieve structural weights of 11542.410 kg, 11542.410 kg, and 11555.006 kg with respective constraint violations of 0.9998%, 1.076%, and 0.806%, were improved to fully feasible solutions equivalent to the hybrid HS/BBBC/SA design listed in Table 5.9 (11542.409 kg). Their computational cost decreased by approximately 30% compared to the original algorithms but remained 3.5- 8.5 times higher than that of FHGWJA. For brevity, these additional results are not reported in Table 5.9.

Figure 5.16 illustrates the optimization histories of the best runs for the algorithms listed in Table 5.9, including the average convergence curve of FHGWJA. For clarity, the plot is limited to the first 8000 structural analyses and the 11000- 31000 kg weight interval.

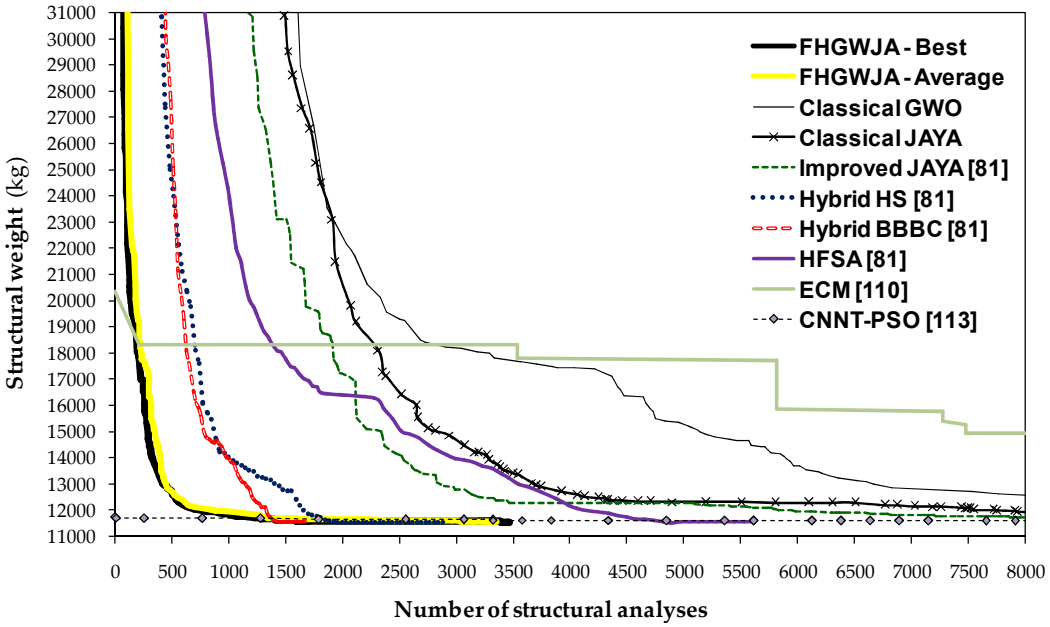


Figure 5.16. Comparison of convergence curves of FHGWJA and its competitors for the 200-bar optimization problem.

FHGWJA began its best optimization run from a notably large initial cost of 208700.225 kg, approximately 18.1 times the target optimum of 11541.380 kg reported in Table 5.9. In contrast, the initial costs of the other optimizers ranged between 157,234.459 and 185,304.986 kg, except for ECM [235] and CNNT-PSO [238], which started from 20437.901 kg and 11754.998 kg, respectively. Despite its much higher starting cost, FHGWJA rapidly closed the initial performance gap, within only 210 and 1330 structural analyses compared with ECM and CNNT-PSO, respectively.

Throughout most of the optimization history, both the best and average convergence curves of FHGWJA consistently remained below those of all other algorithms, confirming its superior search efficiency and convergence stability. The hybrid HS and BBBC algorithms of Ref. [158] exhibited competitive performance, approaching the best FHGWJA run after approximately 2000 and 1400 structural analyses, respectively, near the end of their optimization histories. However, while the hybrid HS/BBBC algorithms plateaued at their best weight of 11542.409 kg, FHGWJA continued improving, ultimately reaching the lowest structural weight of 11541.380 kg.

Notably, the best and average FHGWJA convergence curves nearly coincided after about 1750 analyses, corresponding to only 50.5% of the total optimization history, demonstrating its robustness and consistency across independent runs.

The results in this section confirm the tendency of metaheuristic optimizers to prematurely converge to suboptimal designs in the 200-bar truss weight minimization problem. In contrast, FHGWJA once again demonstrated superior performance, owing to its intrinsic ability to generate high-quality trial solutions throughout the optimization process. Notably, the proposed algorithm outperformed advanced high-performance optimizers, including ECM and LSHADE-Epsin, both recognized for their competitiveness with CEC (IEEE Congress on Evolutionary Computation) benchmark winners.

5.4 CONCLUSIONS ON FHGWJA

In this chapter, the Fast Hybrid Grey Wolf-JAYA Algorithm (FHGWJA) [207] has been introduced as an advanced and efficient hybrid metaheuristic optimization method derived from the combination of the Grey Wolf Optimizer (GWO) and the JAYA algorithm. Conceptually, FHGWJA can be regarded as an enhanced version of the classical GWO, where elitist strategies and JAYA-inspired updating schemes are integrated to systematically refine the population and improve the current best solution at each iteration. Table 5.10 provides a comparison between the Simple Hybrid Grey Wolf-JAYA Algorithm (SHGWJA) [170] and the novel FHGWJA [207].

TABLE 5.10. COMPARATIVE ANALYSIS BETWEEN SHGWJA [170] AND FHGWJA [207].

FEATURE	SHGWJA (Simple Hybrid Grey Wolf JAYA)	FHGWJA (Fast Hybrid Grey Wolf JAYA)
ALGORITHM TYPE	Simple hybrid of GWO and JAYA	Fast hybrid of GWO and JAYA
MAIN OBJECTIVE	Develop a highly efficient yet simple hybrid optimizer	Enhance speed and robustness through elitist and repair strategies
EXPLORATION PHASE	Based on classical GWO search operators	Based on GWO with dynamic leader updating and mirroring strategies
EXPLOITATION PHASE	Improved via simple elitist strategies	Improved through rank-based elitist and repair strategies
INTERNAL PARAMETERS	None (only population size and iteration limit)	None (only population size and iteration limit)
DISTINCTIVE STRATEGIES	Rejection of unpromising designs	Rank-proportional variation and mirroring to avoid stagnation

FEATURE	SHGWJA (Simple Hybrid Grey Wolf JAYA)	FHGWJA (Fast Hybrid Grey Wolf JAYA)
BENCHMARK TESTS	Seven real-world engineering problems and two mathematical benchmarks	Seven complex engineering design problems with up to 1200 nonlinear constraints
PERFORMANCE SUMMARY	High robustness and low dispersion; competitive with advanced MAs	Always converged to the global optimum; high computational efficiency

The hybridization process was carefully designed to preserve the algorithm's structural simplicity while considerably strengthening its balance between exploration and exploitation. The algorithm enhances exploration through adaptive diversity-preserving mechanisms that dynamically update the leaders of the wolf hierarchy, thus preventing premature convergence and stagnation in local minima. At the same time, exploitation is reinforced by the inclusion of rank-based elitist and repair strategies, which promote high-quality trial solutions and ensure convergence toward feasible and globally optimal designs. The overall search process is highly efficient, as FHGWJA performs most of its decision-making operations based on the evaluation of the cost function, thereby limiting the number of computationally expensive constraint evaluations.

The FHGWJA algorithm was thoroughly validated across seven real-world engineering design problems, encompassing diverse application domains such as robotics (2D path planning with multiple obstacles), hydrology (calibration of nonlinear hydrologic models), hydraulics, structural and mechanical design (shape optimization of a concrete gravity dam under seismic load, crashworthiness optimization of a vehicle subject to side impact), and civil engineering (weight minimization of a planar 200-bar truss structure). The test problems considered included up to 29 optimization variables and 1200 nonlinear constraints. In all cases, FHGWJA demonstrated superior performance relative to the most advanced metaheuristic algorithms available in the literature, consistently achieving higher solution accuracy, faster convergence rates, and greater computational efficiency.

The algorithm systematically ranked first or second in computational speed and proved to be significantly faster than both the standard GWO and JAYA formulations. Moreover, the results exhibited very low standard deviations in both optimized cost and the number of function evaluations, confirming the remarkable robustness and repeatability of the proposed approach.

CHAPTER VI: HALSGWJA: AN HYBRID APPROXIMATE LINE SEARCH GREY WOLF JAYA METAHEURISTIC ALGORITHM FOR STRUCTURAL OPTIMIZATION

As flocks of birds follow subtle shifts in the wind, and packs of wolves sense the faintest traces in the terrain, so hybrid minds move guided by an intuitive sense of direction. The approximate gradient acts as a sensory lens, orienting exploration toward hidden and optimal patterns.

6.1 INTRODUCTION TO THE HYBRID APPROXIMATE LINE SEARCH GREY WOLF-JAYA ALGORITHM (HALGWJA)

Real-world optimization problems are often complex, involving multiple design variables, nonlinear constraints, and discrete or non-convex search spaces. Gradient-based optimization algorithms (GBOAs) have traditionally been used to tackle such problems when they are smooth, continuous, and convex. By evaluating the sensitivity of the objective function with respect to the design variables, GBOAs determine descent directions and perform line searches to reduce the objective function efficiently. While this approach provides rapid and precise convergence, it is inherently local and prone to becoming trapped in suboptimal solutions, particularly when the design space is highly nonlinear or involves discrete variables.

Metaheuristic algorithms (MAs), as introduced in previous chapters, offer a complementary perspective. By maintaining a population of candidate solutions distributed across the design space, they iteratively improve solution quality through global exploration and local exploitation [158], [170], [207], [245], [246], [247], [248], [249]. This population-based strategy allows metaheuristics to navigate complex landscapes with multiple local optima, non-convex constraints, or discrete variables. However, despite their versatility, metaheuristics can face limitations in convergence speed, computational cost, and the balance between exploration and exploitation.

Recognizing the complementary strengths and limitations of gradient-based and population-based metaheuristic algorithms, this chapter introduces the Hybrid Approximate Line Search Grey Wolf JAYA Algorithm (HALSGWJA), a novel hybrid approach that effectively combines global search with local gradient guidance. HALSGWJA integrates the Grey Wolf Optimizer (GWO) [12], [171], [172], [211] and the JAYA algorithm [13], [196], [201], enhanced with approximate gradient information and line search. GWO provides structured exploration inspired by grey wolf hunting behaviour, while JAYA guides candidate solutions toward the best and away from the worst, combining survival-of-the-fittest principles with directed population control. By introducing approximate gradient-based line search in the initial trial generation, HALSGWJA directs solutions along descent directions while retaining global exploration, eliminating the need for heuristic acceptance/rejection criteria, and achieving faster, more robust convergence[158], [170], [207].

The algorithm was extensively tested on 20 mechanical engineering problems, including the 19 problems from the CEC2020 suite and the highly nonlinear crashworthiness design problem with 11 variables and 10 constraints. These cases involved static and dynamic optimization, sizing and topology problems, and discrete-variable designs. HALSGWJA consistently outperformed 18 state-of-the-art metaheuristics on the CEC2020 problems and 25 algorithms in the crashworthiness scenario, frequently achieving zero or near-zero standard deviation in the optimized cost. It also ranked first in computational efficiency, requiring fewer analyses than competing methods, and demonstrated robust performance across more than 40 metaheuristics.

By seamlessly integrating local gradient guidance with population-based global search, HALSGWJA exemplifies a new generation of hybrid metaheuristic algorithms capable of effectively addressing the challenges of real-world design problems. In the following sections, Section 2 presents a detailed formulation of HALSGWJA, Section 3 discusses the optimization

test problems and corresponding results, and Section 4 concludes with a summary and perspectives for future research.

6.2 The Hybrid Approximate Line Search Grey Wolf JAYA Algorithm

The Hybrid Approximate Line Search Grey Wolf JAYA (HALSGWJA) extends the FHGWJA framework by explicitly integrating approximate gradient-based guidance into the population-based metaheuristic search. While FHGWJA relies on rank-based refinement, JAYA-based corrections, and mirroring strategies to improve trial solutions, HALSGWJA introduces approximate line search along descent directions for each candidate solution during the initial generation of trial designs. This enables the algorithm to leverage local sensitivity information from the cost function without performing full gradient evaluations, effectively combining global exploration with informed local exploitation.

Specifically, HALSGWJA modifies Step 2 and Step 3 of FHGWJA by:

1. **Gradient-informed preliminary trial generation:** Candidate solutions generated via GWO are adjusted using approximate gradient information, creating trial designs that are more likely to improve the objective function. This reduces reliance on probabilistic acceptance/rejection mechanisms and improves convergence reliability.
2. **Line search along descent directions:** For each trial solution, a line search is performed along an estimated descent vector derived from the approximate gradient, allowing the algorithm to identify a step size that effectively reduces the penalized cost function. This approach directly addresses the limitation of classical GWO and FHGWJA, where trial solutions may fail to improve the current best design.
3. **Enhanced population renewal:** By guiding all trial solutions along descent directions while retaining the global search capabilities of GWO and JAYA, HALSGWJA ensures that each population update has a high likelihood of improvement. The combined effect of approximate gradient guidance and line search reduces stagnation and accelerates convergence without introducing additional control parameters beyond N_{POP} and $N_{MaxIter}$.
4. **Improved balance between exploration and exploitation:** Unlike FHGWJA, where refinement relies on rank-based, JAYA, and mirroring strategies alone, HALSGWJA systematically integrates local gradient cues to enhance exploitation, while population-based search preserves exploration. This hybridization achieves a more robust search, particularly effective for high-dimensional, nonlinear, and constrained problems.

The following presents a detailed description of the enhancements implemented in HALSGWJA.

Step 1 - Initialization

As in FHGWJA, HALSGWJA starts by randomly generating a population of N_{POP}candidate solutions within the prescribed design bounds. Each solution represents a potential design vector to be evolved through subsequent iterations. The initial population is evaluated using

the penalized objective function $W_p(\bar{X}_{it})$, which will later serve as the basis for the computation of approximate gradient information in the refinement stage.

Step 2 - Gradient-Informed Preliminary Trial Generation

Candidate solutions are first generated using the classical Grey Wolf Optimizer (GWO) mechanism. The best three individuals are identified as α, β, and δ wolves, while the remaining population forms the ω wolves. However, the perturbation applied to the preliminary trial design $(\bar{X}_{i,tr})^{prel}$ may sometimes be too small to yield a significant improvement in the best solution $\bar{X}_{best}$ or the current individual $\bar{X}_{it}$.

To address this limitation, HALSGWJA employs approximate gradient information to refine the preliminary trial solutions. Ideally, the gradients of both the cost and constraint functions should be evaluated for each perturbed design $\bar{X}_{it}$. However, such evaluations would be computationally expensive. Instead, HALSGWJA estimates the rate of variation of the penalized cost function with respect to $\bar{X}_{it}$.

The greatest improvement is generally obtained when perturbing the design from $\bar{X}_{it}$ toward $\bar{X}_{best}$. Nevertheless, this may not always provide the fastest improvement when considering the ratio between the penalized cost function variation and the perturbation magnitude:

$$\frac{\Delta W_p(i,k)}{\Delta S(i,k)} = \frac{\| W_p(\bar{X}_{it}) - W_p(\bar{X}_k) \|}{\| \bar{X}_{it} - \bar{X}_k \|} \tag{6.1}$$

where $\bar{X}_k$ represents another search agent ranked higher in the population. If $\bar{X}_k$has a better rank than $\bar{X}_{it}$, the direction

$$\bar{S}_{i,k} = (\bar{X}_k - \bar{X}_{it}) \tag{6.2}$$

defines a descent direction with respect to $\bar{X}_{it}$. Conversely, if $\bar{X}_k$ ranks lower than $\bar{X}_{it}$, perturbing along $(\bar{X}_k - \bar{X}_{it})$ would not lead to improvement. In this case, the descent direction is reversed as

$$\bar{S}_{i,k} = -(\bar{X}_k - \bar{X}_{it}) = (\bar{X}_{it} - \bar{X}_k) \tag{6.3}$$

Based on this principle, HALSGWJA computes an approximate (average) gradient of the penalized cost function with respect to each design $\bar{X}_{it}$ as

$$\gamma(i,k) = \frac{\Delta W_p(i,k)}{\Delta S(i,k)}, \tag{6.4}$$

for all other designs $\bar{X}_k$ in the population.

The preliminary trial design $(\bar{X}_{i,tr})^{prel}$ generated by the GWO scheme is then refined as follows:

$$\bar{X}_{i,it} = (\bar{X}_{i,tr})^{prel} + \tau_{best}(\bar{X}_{best} - \bar{X}_{it}) + \tau_{FST,btt}(\bar{X}_{FST,better} - \bar{X}_{it}) - \\ -\tau_{FST,wrs}(\bar{X}_{FST,worst} - \bar{X}_{it}) \quad (6.5)$$

where $\bar{X}_{FST,better}$ is the higher-ranked design (other than $\bar{X}_{best}$) yielding the largest $\gamma(i,k)$value, and $\bar{X}_{FST,worst}$ is the lower-ranked design producing the largest $\gamma(i,k)$. The coefficients τ_{best}, $\tau_{FST,btt}$, and $\tau_{FST,wrs}$are random numbers uniformly distributed in [0,1].

Equation 6.5 refines the position of $(\bar{X}_{i,tr})^{prel}$ by combining three complementary descent directions that can improve the current design $\bar{X}_{it}$. The direction $(\bar{X}_{best} - \bar{X}_{it})$ drives the solution toward the global best record, $(\bar{X}_{FST,better} - \bar{X}_{it})$ promotes the fastest local improvement toward higher-ranked designs, and the reversed direction $-(\bar{X}_{FST,worst} - \bar{X}_{it})$ helps escape inferior designs. The random weighting factors τ_{best}, $\tau_{FST,btt}$, and $\tau_{FST,wrs}$control the perturbation magnitude along these three paths, ensuring diversity while maintaining descent-oriented progression.

Step 3- Evaluation and Correction of Trial Designs: JAYA Phase and Elitist Strategies

In this step, the set of trial designs $(\bar{X}_{i,tr})$ is evaluated to determine whether they should be retained in the next population.In the classical GWO, a new design $(\bar{X}_{i,tr})$ replaces the old one $(\bar{X}_{it})$ if $W_p(\vec{X}_{i,tr}) < W_p(\bar{X}_{it})$. However, this approach requires evaluating all constraints again. In SHGWJA and FHGWJA an elitist strategy was proposed to reduce this computational effort by comparing the unpenalized cost of the trial design with the penalized cost of the current best design $\bar{X}_{best}$:

- if $W(\bar{X}_{i,tr}) > W_p(\bar{X}_{best}) \Rightarrow$ reject $\bar{X}_{i,tr}$
- if $W(\bar{X}_{i,tr}) < W_p(\bar{X}_{best})$ and $W(\bar{X}_{i,tr}) \leq 1.1\, W(\bar{X}_{best}) \Rightarrow$ provisionally accept $\bar{X}_{i,tr}$

In HALSGWJA, this step is no longer required, since Equation 6.5 already refines each trial design by combining three meaningful descent directions, ensuring a decrease in the cost function relative to the previous design. Therefore, HALSGWJA automatically includes all trial designs $(\bar{X}_{i,tr})$ generated by Equation 6.5 in the new population Π_{tr}.

Unlike classical or improved GWO versions [170], [207], HALSGWJA considers not only the leading wolves (α, β, and δ) but also the influence of the entire population when generating $(\bar{X}_{i,tr})$. This collective update mechanism effectively reduces the risk of stagnation.

HALSGWJA integrates an adaptive JAYA-based refinement strategy to enhance trial designs generated in the GWO phase. HALSGWJA employs a JAYA-based adaptive scheme to further refine trial designs. In SHGWJA and FHGWJA, JAYA operators were used either to exploit

($\bar{X}_{i,tr}$) when it was promising or to re-explore the search space when it was not, according to the condition:

- $W(\bar{X}_{i,tr}) \leq 1.1\, W(\bar{X}_{best})$(exploitation phase)
- $W(\bar{X}_{i,tr}) > 1.1\, W(\bar{X}_{best})$(exploration phase)

The fixed threshold 1.1 was inspired by the acceptance criterion of Simulated Annealing, enabling the algorithm to occasionally accept slightly worse solutions to escape local minima.

HALSGWJA replaces this fixed value with an adaptive threshold ρ_w, dynamically updated at each iteration according to population characteristics and gradient information:

$$\rho_w = \min\left\{\frac{W_p(\bar{X}_\delta)}{W_p(\bar{X}_{best})},\ \frac{W_{\delta,tr}}{W_{best,tr}},\ \frac{\| \bar{\nabla} W_p(\bar{X}_{best}) \|}{\| \bar{\nabla} W(\bar{X}_{best,tr}) \|},\ \frac{\| \bar{\nabla} W(\bar{X}_{best,tr}) \|}{\| \bar{\nabla} W_p(\bar{X}_{best}) \|}\right\} \tag{6.6}$$

This formulation allows HALSGWJA to adaptively adjust the acceptance criterion depending on both cost function progress and gradient magnitudes between consecutive populations.

Approximate gradients are evaluated using finite-difference-like ratios averaged across key descent directions:

$$\| \bar{\nabla} W_p(\bar{X}_{best}) \| = \text{Mean}\left\{\frac{W_p(\bar{X}_{best}) - W_p(\bar{X}_\beta)}{\| \bar{X}_{best} - \bar{X}_\beta \|},\ \frac{W_p(\bar{X}_{best}) - W_p(\bar{X}_\delta)}{\| \bar{X}_{best} - \bar{X}_\delta \|},\ \frac{W_p(\bar{X}_{best}) - W_p(\bar{X}_{FAST})}{\| \bar{X}_{best} - \bar{X}_{FAST} \|}\right\}$$

$$\| \bar{\nabla} W(\bar{X}_{best,tr}) \| = \text{Mean}\left\{\frac{W(\bar{X}_{best,tr}) - W(\bar{X}_{\beta,tr})}{\| \bar{X}_{best,tr} - \bar{X}_{\beta,tr} \|},\ \frac{W(\bar{X}_{best,tr}) - W(\bar{X}_{\delta,tr})}{\| \bar{X}_{best,tr} - \bar{X}_{\delta,tr} \|},\ \frac{W(\bar{X}_{best,tr}) - W(\bar{X}_{FAST,tr})}{\| \bar{X}_{best,tr} - \bar{X}_{FAST,tr} \|}\right\} \tag{6.7}$$

These computations have negligible additional cost since both $W_p$and Wvalues are already available from previous evaluations.

If $W(\bar{X}_{i,tr}) \leq \rho_w\, W(\bar{X}_{best})$, HALSGWJA performs a JAYA-based exploitation: the best and worst designs for the JAYA operator are determined by comparing the two populations:

$$\bar{X}_{best,JAYA} = \begin{cases} \bar{X}_{best,tr}, & \text{if } W_p(\bar{X}_{best,tr}) \leq W_p(\bar{X}_{best}) \\ \bar{X}_{best}, & \text{otherwise} \end{cases} \tag{6.8}$$

$$\bar{X}_{worst,JAYA} = \begin{cases} \bar{X}_{\delta,tr}, & \text{if } W(\bar{X}_{\delta,tr}) \leq W(\bar{X}_\delta) \\ \bar{X}_\delta, & \text{otherwise} \end{cases} \tag{6.9}$$

The refined trial solution is then computed as:

$$\bar{X}'_{i,tr} = \bar{X}_{i,tr} + \bar{\lambda}_1 \cdot (\bar{X}_{best,JAYA} - \bar{X}_{i,tr}) - \bar{\lambda}_2 \cdot (\bar{X}_{worst,JAYA} - \bar{X}_{i,tr}) \tag{6.10}$$

where $\bar{\lambda}_1$ and $\bar{\lambda}_2$ are random vectors with components uniformly distributed in [0,1].

The new design $\bar{X}'_{i,tr}$ replaces $\bar{X}_{i,tr}$in the population if it yields a lower cost: $W(\bar{X}'_{i,tr}) < W(\bar{X}_{i,tr})$.

When $W(\bar{X}'_{i,tr}) > \rho_w\, W(\bar{X}_{best})$, HALSGWJA switches to exploration. The worst design across both populations is identified as:

$$\bar{X}_{worst,JAYA} = \begin{cases} \bar{X}_{worst,tr}, & \text{if } W(\bar{X}_{worst,tr}) \leq \mathrm{W}(\bar{X}_{worst}) \\ \bar{X}_{worst}, & \text{otherwise} \end{cases} \tag{6.11}$$

A new exploratory candidate is then generated:

$$\bar{X}'_{i,tr} = \bar{X}_{i,JAYA} + \bar{\lambda}_1 \cdot (\bar{X}_{best,JAYA} - \bar{X}_{i,JAYA}) - \bar{\lambda}_2 \cdot (\bar{X}_{worst,JAYA} - \bar{X}_{i,JAYA}) \tag{6.12}$$

where:

$$\bar{X}_{i,JAYA} = \begin{cases} \bar{X}_{i,tr}, & \text{if } W(\bar{X}_{i,tr}) \leq \mathrm{W}(\bar{X}_{it}) \\ \bar{X}_{it}, & \text{otherwise} \end{cases} \tag{6.13}$$

This mechanism ensures that each updated individual in $\Pi_{iter-fina}$ is at least as good as its previous version, while maintaining diversity.

If the exploration step also fails, i.e. $W(\bar{X}'_{i,tr}) > \rho_w\, W(\bar{X}_{best})$ and $W(\bar{X}_{i,tr}) > \rho_w\, W(\bar{X}_{best})$, HALSGWJA applies a mirroring strategy to redirect the search along a descent direction with respect to $\bar{X}_{best}$:

$$\bar{X}''^{,mirr}_{i,tr} = \begin{cases} \bar{X}_{best} + \eta_{mirr}\,(\bar{X}_{best} - \bar{X}'_{i,tr}), & \text{if } W(\bar{X}'_{i,tr}) \leq W(\bar{X}_{i,tr}) \\ \bar{X}_{best} + \eta_{mirr}\,(\bar{X}_{best} - \bar{X}_{i,tr}), & \text{otherwise} \end{cases} \tag{6.14}$$

where $\eta_{mirr} \in [0,1]$ is a random scalar controlling the perturbation amplitude. This strategy mirrors the previous unsuccessful direction, effectively avoiding non-descent moves and promoting convergence without generating infeasible designs.

Step 4. Resort population, update α, β, and δ wolves, $\bar{X}_{best}$, and $\bar{X}_{worst}$

Once HALSGWJA has updated the entire population, the N_{POP} individuals are sorted according to the penalized cost function. The three best individuals are assigned as wolves α, β and δ with their corresponding design vectors $\bar{X}_\alpha, \bar{X}_\beta, \bar{X}_\delta$. The best and worst search agents are set as $\bar{X}_{best}$ and $\bar{X}_{worst}$, respectively.

To prevent stagnation, HALSGWJA verifies whether the α,β and δ wolves have been updated during the current iteration. An elitist criterion based on descent directions is applied: $\bar{S}_\beta = \bar{X}_{best} - \bar{X}_\beta$, $\bar{S}_\delta = \bar{X}_{best} - \bar{X}_\delta$. These directions are descent directions for the β and δ wolves, as moving along them improves the design toward the current best solution (α -wolf). Similar to FHGWJA, HALSGWJA perturbs β and δ wolves along $\bar{S}_\beta$ and $\bar{S}_\delta$. The positions of β and δ wolves are first displaced from the center of mass of the three best individuals and then mirrored with respect to $\bar{X}_{best}$:

$$\begin{aligned} \bar{X}_\beta^{mirr} &= \left(\frac{\bar{X}_{best} + \bar{X}_\beta + \bar{X}_\delta}{3} - \bar{X}_\delta\right) + (1 + \eta_{mirr,\beta}) \cdot \bar{X}_{best} - \eta_{mirr,\beta} \cdot \bar{X}_\beta \\ \bar{X}_\delta^{mirr} &= \left(\frac{\bar{X}_{best} + \bar{X}_\beta + \bar{X}_\delta}{3} - \bar{X}_\delta\right) + (1 + \eta_{mirr,\delta}) \cdot \bar{X}_{best} - \eta_{mirr,\delta} \cdot \bar{X}_\delta \end{aligned} \tag{6.15}$$

where the random numbers $\eta_{mirr,\beta}$ and $\eta_{mirr,\delta}$ are extracted from the interval (0,1) to limit the step sizes along $\bar{S}_\beta$ and $\bar{S}_\delta$, reducing the risk of infeasible solutions.

Unlike the original scheme in FHGWJA, which could produce two limit cases:

1. $\eta_{mirr,\beta} = \eta_{mirr,\delta} = 0 \rightarrow \bar{X}_\beta^{mirr}$ and $\bar{X}_\delta^{mirr}$ coincide with $\bar{X}_{best}$
2. $\eta_{mirr,\beta} = \eta_{mirr,\delta} = 1 \rightarrow \bar{X}_\beta^{mirr}$ and $\bar{X}_\delta^{mirr}$ are symmetric to $\bar{X}_\beta$ and $\bar{X}_\delta$ about $\bar{X}_{best}$

HALSGWJA includes the term $\left(\frac{\bar{X}_{best} + \bar{X}_\beta + \bar{X}_\delta}{3} - \bar{X}_\delta\right)$ in Equation 6.15, ensuring that the mirrored wolves are always at least better than the δ -wolf.

The best three search agents among $\bar{X}_\alpha$, $\bar{X}_\beta$, $\bar{X}_\delta$, $\bar{X}_\beta^{mirr}$ and $\bar{X}_\delta^{mirr}$ are stored as the new α, β and δ wolves for the next iteration.

The two worst search agents are compared with the rest of the population and may replace $\bar{X}_{worst}$ and the second worst design. This includes $\bar{X}_\beta$ and $\bar{X}_\delta$ if they did not improve any of the top wolves. Replacing the worst designs improves the average quality of the population, increasing the likelihood of generating high-quality trial solutions in the following iteration.

Step 5. Convergence check

Standard deviations on optimization variables and penalty function values of candidate designs have to decrease as the search process progresses towards the global optimum. For this reason, HALSGWJA normalizes the standard deviation of the design vectors with respect to the average design vector $\bar{X}_{aver} = \left(\sum_{i=1}^{N_{POP}} \bar{X}_{it}\right)/N_{POP}$ of the population and the standard deviation of the penalty function values with respect to the average penalty function value $W_{p,\text{aver}} = \left(\sum_{i=1}^{N_{POP}} W_p(\bar{X}_{it})\right)/N_{POP}$. The termination criterion used by HALSGWJA is:

$$Max\left\{\begin{array}{l}\dfrac{STD\{\|\bar{X}_1-\bar{X}_{aver}\|,\|\bar{X}_2-\bar{X}_{aver}\|,\ldots,\|\bar{X}_{NPOP}-\bar{X}_{aver}\|\}}{\|\bar{X}_{aver}\|};\\ \dfrac{STD\{|W_{p,1}-W_{p,aver}|,|W_{p,2}-W_{p,aver}|,\ldots,|W_{p,NPOP}-W_{p,aver}|\}}{W_{p,aver}}\end{array}\right\}\leq\varepsilon_{conv} \quad (6.16)$$

where the convergence limit ε_{conv} is equal to 10^{-7}. Penalty function values are equal to cost function values for feasible solutions or in unconstrained optimization problems.

Steps 2 through 5 are repeated until HALSGWJA converges to the global optimum.

Step 6. Terminate optimization process

HALSGWJA terminates the optimization process and writes output data (i.e., optimum design and optimized cost function value) in the results file.

Table 6.1. summarizes the enhancements introduced in HALSGWJA, particularly the integration of approximate gradient information and line search, which aim to improve convergence speed, solution quality, and robustness against stagnation. Each algorithmic step is compared, showing where HALSGWJA retains FHGWJA's original mechanisms and where it introduces new strategies.

TABLE 6.1. COMPARATIVE ANALYSIS BETWEEN SHGWJA [170] AND FHGWJA [207].

Step	FHGWJA	HALSGWJA	Key Differences
1. INITIALIZATION	Randomly generate population of wolves.	Same as FHGWJA.	No difference.
2. GWO PHASE: PRELIMINARY TRIAL SOLUTIONS	Classical GWO to generate trial designs; α, β, δ wolves guide others.	Classical GWO for preliminary trials, but gradient approximation available for local guidance.	HALSGWJA adds local gradient knowledge to enhance directional search.
3. RANK-BASED REFINEMENT	Step size μ determined via population ranking and penalized cost function variation; adjust trial solution along vector from current to best solution.	Same rank-based criterion applied; additionally uses approximate gradient to further refine trial solutions.	HALSGWJA introduces gradient-guided adjustment, improving local exploitation.
4. EVALUATION & REPAIR (ELITIST + JAYA)	Elitist pre-selection and JAYA-based refinement: move toward best, away from worst; 1.1 × cost threshold allows limited acceptance.	Elitist + JAYA refinement integrated with gradient descent and line search: trial solutions refined along descent directions; line search determines optimal step size α_LS.	HALSGWJA combines gradient + line search with JAYA, improving convergence speed and solution quality.
MIRRORING STRATEGY	Apply mirroring to α, β, δ wolves if stagnation occurs; use descent directions relative to current best.	Same mirroring strategy applied, but mirrored solutions also incorporate gradient-informed directions and line search steps.	HALSGWJA's mirroring is gradient-enhanced, increasing probability of escaping local minima.
5. POPULATION UPDATE & REORDERING	Sort population; update α, β, δ wolves; discard worst two solutions; reassess elite positions.	Same procedure but update also considers gradient-enhanced trial solutions from previous step.	HALSGWJA integrates gradient-refined solutions into population update for better exploitation-exploration balance.

Step	FHGWJA	HALSGWJA	Key Differences
6. CONVERGENCE CHECK	Standard deviations of positions and penalized cost function compared to thresholds.	Same convergence criteria: gradient-guided step sizes may accelerate convergence.	HALSGWJA potentially converges faster due to guided descent directions.
7. TERMINATION	Store best solution and corresponding cost function value; output results.	Same termination criteria.	No difference in termination, but HALSGWJA may require fewer iterations/function evaluations.

Figure 6.1 shows the flowchart of the HALSGWJA algorithm developed in this study. HALSGWJA is an advanced GWO that updates the population to improve the current best solution or, at least, most of the individuals in the population during each iteration. The integration of approximate gradient information into HALSGWJA enhances the optimization search process. Additionally, the JAYA-based schemes and other elitist strategies implemented in HALSGWJA optimize the sequence of exploration and exploitation phases, increasing population diversity while consistently selecting high-quality trial designs with a limited number of function evaluations. HALSGWJA employs a master-slave structure, where the master algorithm, the GWO, generates trial solutions, and the slave algorithm, JAYA, refines or corrects these solutions. Like SHGWJA and FHGWJA, HALSGWJA does not require any additional internal parameters beyond those of its component algorithms. Therefore, the population size ($NPOP$) and the maximum number of iterations ($Nitermax$) remain the only hyperparameters that the user must define. Notably, the 1.1 heuristic threshold used in SHGWJA and FHGWJA to accept or reject trial designs based on the criterion $W(\bar{X}_{i,tr}) \leq 1.1\, W(\bar{X}_{best})$ is replaced in HALSGWJA by the more robust, flexible, and adaptive criterion $W(\bar{X}_{i,tr}) \leq \rho_w\, W(\bar{X}_{best})$.

Unlike most GWO/JAYA implementations, which require $NPOP \times Nitermax$ function or constraint evaluations (e.g., structural analyses in structural optimization), HALSGWJA computes descent directions without extra constraint analyses. Approximate gradients are derived from existing penalized cost function values, allowing reliable assessment of trial designs at minimal computational cost.

The algorithm consistently generates high-quality designs with a small population, reflecting the natural hunting behaviour of grey wolves, which typically hunt in packs of 10-20. Accordingly, HALSGWJA employs a population of 20 individuals, a choice confirmed effective via sensitivity analysis.

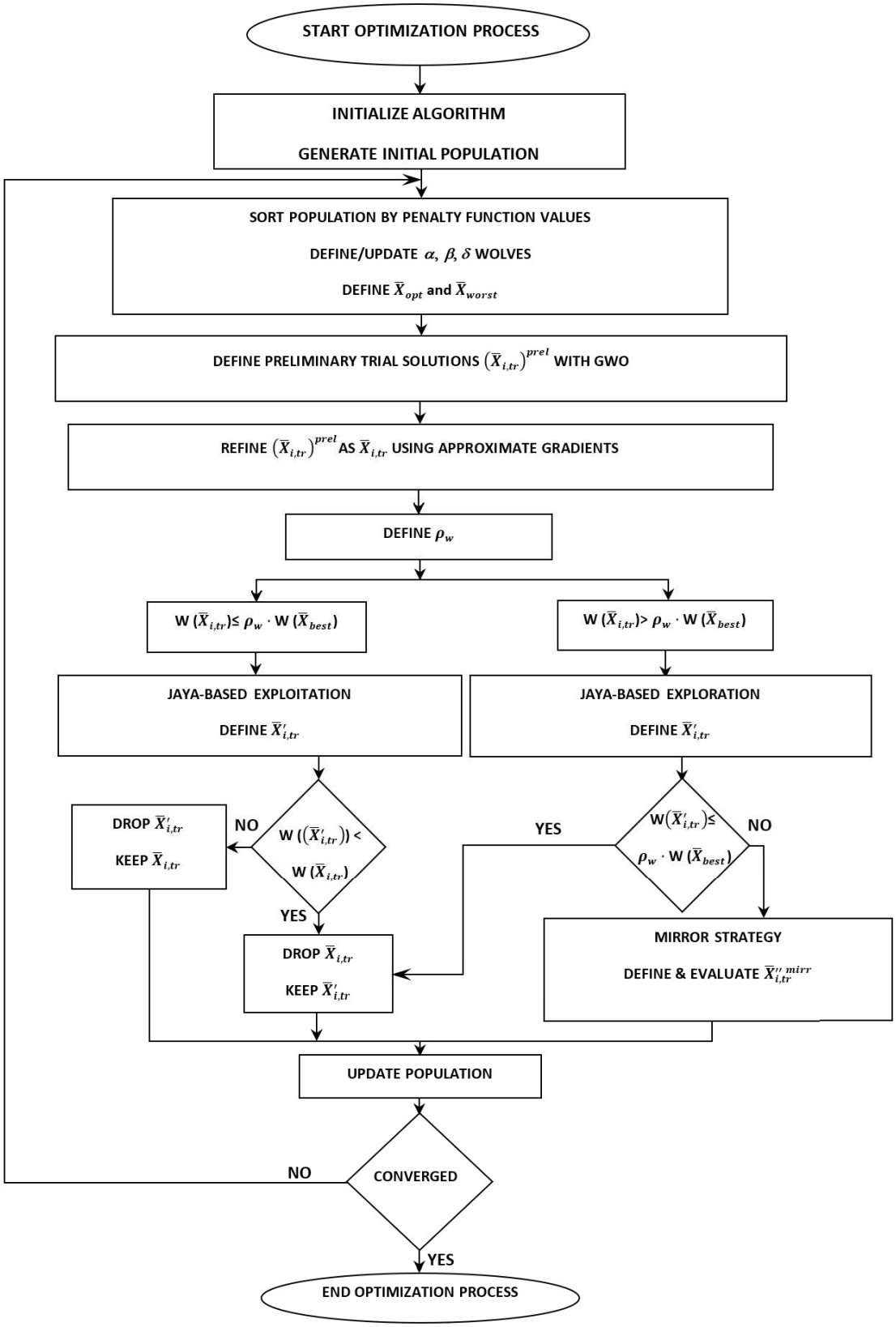


Figure 6.1. Flowchart of the HALSGWJA algorithm

6.3 Test Problems and Optimization Results

The HALSGWJA algorithm was tested on 20 real-world mechanical engineering problems. The first 19 belong to the CEC2020 suite (also referred to as RC15–RC33) [132], [140], [177], [193], [196], [197], [201], [215], [216], [220], [250], [251], [252], [253], [254], [255], [256], [257], [258], [259], [260], [261], [262], [263], [264], [265], [266], [267], [268], [269], [270], [271], [272], [273], [274], [275].

Table 6.2 summarizes their main features, including up to 30 design variables, 86 inequality constraints, and the optimization goals-minimization for all except problem 14, which maximizes rolling bearing capacity.

Problems 4, 8, 12, 16, and 17 include discrete or integer variables, while problems 8, 9, and 17 also involve equality constraints.

Full details of the CEC2020 problems (NDV, G, H) are in Ref. [201] and partially in Refs. [193], [196], [197], [220], [250], [251], [252], [253], [254], [255], [256], [257], [258], [259], [260], [261], [262] for the complete library and Refs. [6], [132], [140], [170], [177], [215], [216], [263], [264], [265], [266], [267], [268], [269], [270], [271], [272], [273], [274], [275], [276] for individual cases.

For brevity, this study presents only problems 2, 12, and 19, which have the largest numbers of variables and constraints (14, 22, 30; 15, 86, 30, respectively), and problem 14, the only maximization case.

Table 6.2. DESCRIPTION OF THE CEC2020 REAL-WORLD MECHANICAL ENGINEERING TEST PROBLEMS.

Problem	Description	Objective	NDV	G	H	Optimum
1	Speed reducer	*Minimize weight*	7	11	0	2.9944244658E+03
2	Industrial refrigeration system design	*Min. fabrication/operation costs*	14	15	0	3.2213000814E−02
3	Tension/compression spring design (1)	*Minimize weight*	3	3	0	1.2665232788E−02
4	Pressure vessel design	*Minimize fabrication cost*	4	4	0	5.8853327736E+03
5	Welded beam design	*Minimize fabrication cost*	4	5	0	1.6702177263E+00
6	Three-bar truss design	*Minimize weight*	2	3	0	2.6389584338E+02
7	Multiple disk clutch brake design	*Minimize mass*	5	7	0	2.3524245790E−01
8	Planetary gear train design optimization	*Minimize max errors in gear ratios*	9	10	1	5.2576870748E−01
9	Step-cone pulley problem	*Minimize weight*	5	8	3	1.6069868725E+01
10	Robot gripper problem	*Minimize difference max/min forces*	7	7	0	2.5287918415E+00
11	Hydro-static thrust bearing design	*Minimize bearing power loss*	4	7	0	1.6161197651E+03
12	Four-stage gear box problem	*Minimize weight*	22	86	0	3.5359231973E+01
13	Ten-bar truss design	*Minimize weight*	10	3	0	5.2445076066E+02
14	Rolling element bearing	*Maximize load carrying capacity*	10	9	0	1.4614135715E+04
15	Gas transmission compressor design	*Minimize cost of gas delivery*	4	1	0	2.9648954173E+06
16	Tension/compression spring design (2)	*Minimize steel wire volume*	3	8	0	2.6138840583E+00
17	Gear train design	*Match target total gear ratio*	4	1	1	0.0000000000E+00
18	Himmelblau's function	*Minimize quadratic model*	5	6	0	−3.0665538672E+04
19	Topology optimization	*Material layout for min compliance*	30	30	0	2.6393464970E+00

Problem 2 aims to minimize fabrication and operation costs of industrial refrigeration systems with constraints on geometry and heat transfer characteristics. The design variables of this highly nonlinear mechanical engineering optimization problem are: evaporator shell diameter

(X_1), condenser shell diameter (X_2), evaporator shell thickness (X_3), condenser shell thickness (X_4), evaporator tubesheet thickness (X_5), condenser tubesheet thickness (X_6), evaporator fluid velocity (X_7), condenser fluid velocity (X_8), evaporator tube passes (X_9), condenser tube passes (X_{10}), evaporator tube length (X_{11}), condenser tube length (X_{12}), condenser effectiveness (X_{13}), and inlet/outlet fluid temperature difference (X_{14}). The optimization problem is stated as:

$$\begin{aligned} Minimize\ W(\bar{X}) = {} & 63098.88 \cdot X_2 \cdot X_4 \cdot X_{12} + 5441.5 \cdot X_2^2 \cdot X_{12} + 115055.5 \cdot X_2^{1.664} \cdot X_6 + 6172.27 \cdot X_2^2 \cdot X_6 + 63098.88 \cdot X_1 \cdot X_3 \cdot X_{11} + 5441.5 \\ & \cdot X_1^2 \cdot X_{11} + 115055.5 \cdot X_1^{1.664} \cdot X_5 + 6172.27 \cdot X_1^2 \cdot X_5 + 140.53 \cdot X_1 \cdot X_{11} + 281.29 \cdot X_1 \cdot X_3 + 70.26 \cdot X_1^2 + 281.29 \\ & \cdot X_3 \cdot X_{11} + 281.29 \cdot X_3^2 + \\ & +14437 \cdot X_1^2 \cdot X_8^{1.8812} \cdot X_{12}^{0.3424} \cdot X_7 \cdot X_9^{-1} \cdot X_{10} \cdot X_{14}^{-1} + 20470.2 \cdot X_1^2 \cdot X_7^{2.893} \cdot X_{11}^{0.316} \end{aligned} \tag{6.17}$$

Subject to $(NCP + NBP) \cdot NOB = 721$ *constraints:*

$$\begin{aligned} & G_1(\bar{X}) = 1.524 \cdot X_7^{-1} - 1 \le 0 \\ & G_2(\bar{X}) = 1.524 \cdot X_8^{-1} - 1 \le 0 \\ & G_3(\bar{X}) = 0.07789 \cdot X_1 - 2 \cdot X_7^{-1} \cdot X_9 - 1 \le 0 \\ & G_4(\bar{X}) = 7.05305 \cdot X_1^2 \cdot X_2^{-1} \cdot X_8^{-1} \cdot X_9^{-1} \cdot X_{10} \cdot X_{14}^{-1} - 1 \le 0 \\ & G_5(\bar{X}) = 0.0833 \cdot X_{13}^{-1} \cdot X_{14} - 1 \le 0 \\ & G_6(\bar{X}) = 47.136 \cdot X_2^{0.333} \cdot X_{10}^{-1} \cdot X_{12} - 1.333 \cdot X_8 \cdot X_{13}^{2.1195} + 62.08 \cdot X_8^{0.2} \cdot X_{10}^{-1} \cdot X_{12}^{-1} \cdot X_{13}^{2.1195} - 1 \le 0 \\ & G_7(\bar{X}) = 0.04771 \cdot X_8^{1.8812} \cdot X_{10} \cdot X_{12}^{0.3424} - 1 \le 0 \\ & G_8(\bar{X}) = 0.0488 \cdot X_7^{1.893} \cdot X_9 \cdot X_{11}^{0.316} - 1 \le 0 \\ & G_9(\bar{X}) = 0.0099 \cdot X_1 \cdot X_3^{-1} - 1 \le 0 \\ & G_{10}(\bar{X}) = 0.0193 \cdot X_2 \cdot X_4^{-1} - 1 \le 0 \\ & G_{11}(\bar{X}) = 0.0298 \cdot X_1 \cdot X_5^{-1} - 1 \le 0 \\ & G_{12}(\bar{X}) = 0.056 \cdot X_2 \cdot X_6^{-1} - 1 \le 0 \\ & G_{13}(\bar{X}) = 2 \cdot X_9^{-1} - 1 \le 0 \\ & G_{14}(\bar{X}) = 2 \cdot X_{10}^{-1} - 1 \le 0 \\ & G_{15}(\bar{X}) = X_{12} \cdot X_{11}^{-1} - 1 \le 0 \end{aligned} \tag{6.18}$$

with the following boundaries:

$0.001 \le X_i \le 5 \qquad for\ i = 1, \dots, 14$

Problem 12 aims to minimize the structural weight of a four-stage gearbox. The design variables correspond to positions of the gears (X_{gi}, Y_{gi}) and pinions (X_{p1}, Y_{p1}), blank thickness (b_i), and number of teeth (N_{gi}, N_{pi}), with i=1, 2, 3, 4. Design constraints refer to the pitch, kinematics, contact ratio, gears strength, assembly and size. The feasible region of search-space is only 0.01% of the total search space thus leading to the presence of many local solutions. The optimization problem is stated as:

$$Minimize\ W(\bar{X}) = \left(\frac{\pi}{1000}\right) \cdot \sum_{i=1}^{4} \frac{b_i \cdot c_i^2 \cdot \left(N_{pi}^2 + N_{gi}^2\right)}{(N_{pi} + N_{gi})^2} \tag{6.17}$$

Subject to :

$$\begin{aligned} & G_1(\bar{X}) = \left(\frac{366000}{\pi \cdot \omega_1} + \frac{2 \cdot c_1 \cdot N_{p1}}{N_{p1} + N_{g1}}\right) \cdot \frac{\left(N_{p1} + N_{g1}\right)^2}{4 \cdot b_1 \cdot c_1^2 \cdot N_{p1}} - \frac{\sigma_N \cdot J_R}{0.0167 \cdot W \cdot K_o \cdot K_M} \le 0 \\ & G_2(\bar{X}) = \left(\frac{366000}{\pi \cdot \omega_1 \cdot N_{p1}} + \frac{2 \cdot c_2 \cdot N_{p2}}{N_{p2} + N_{g2}}\right) \cdot \frac{\left(N_{p2} + N_{g2}\right)^2}{4 \cdot b_2 \cdot c_2^2 \cdot N_{p2}} - \frac{\sigma_N \cdot J_R}{0.0167 \cdot W \cdot K_o \cdot K_M} \le 0 \\ & G_3(\bar{X}) = \left(\frac{366000 \cdot N_{g1} \cdot N_{g2}}{\pi \cdot \omega_1 \cdot N_{p1} \cdot N_{p2}} + \frac{2 \cdot c_3 \cdot N_{p3}}{N_{p3} + N_{g3}}\right) \cdot \frac{\left(N_{p3} + N_{g3}\right)^2}{4 \cdot b_3 \cdot c_3^2 \cdot N_{p3}} - \frac{\sigma_N \cdot J_R}{0.0167 \cdot W \cdot K_o \cdot K_M} \le 0 \\ & G_4(\bar{X}) = \left(\frac{366000 \cdot N_{g1} \cdot N_{g2} \cdot N_{g3}}{\pi \cdot \omega_1 \cdot N_{p1} \cdot N_{p2} \cdot N_{p3}} + \frac{2 \cdot c_4 \cdot N_{p4}}{N_{p4} + N_{g4}}\right) \cdot \frac{\left(N_{p4} + N_{g4}\right)^2}{4 \cdot b_4 \cdot c_4^2 \cdot N_{p4}} - \frac{\sigma_N \cdot J_R}{0.0167 \cdot W \cdot K_o \cdot K_M} \le 0 \\ & G_5(\bar{X}) = \left(\frac{366000}{\pi \cdot \omega_1} + \frac{2 \cdot c_1 \cdot N_{p1}}{N_{p1} + N_{g1}}\right) \cdot \frac{\left(N_{p1} + N_{g1}\right)^3}{4 \cdot b_1 \cdot c_1^2 \cdot N_{g1} \cdot N_{p1}^2} - \left(\frac{\sigma_H}{c_p}\right)^2 \cdot \frac{sin\phi \cdot cos\phi}{0.0334 \cdot W \cdot K_o \cdot K_M} \le 0 \\ & G_6(\bar{X}) = \left(\frac{366000 \cdot N_{g1}}{\pi \cdot \omega_1 \cdot N_{p1}} + \frac{2 \cdot c_2 \cdot N_{p2}}{N_{p2} + N_{g2}}\right) \cdot \frac{\left(N_{p2} + N_{g2}\right)^3}{4 \cdot b_2 \cdot c_2^2 \cdot N_{g2} \cdot N_{p2}^2} - \left(\frac{\sigma_H}{c_p}\right)^2 \cdot \frac{sin\phi \cdot cos\phi}{0.0334 \cdot W \cdot K_o \cdot K_M} \le 0 \\ & G_7(\bar{X}) = \left(\frac{366000 \cdot N_{g1} \cdot N_{g2}}{\pi \cdot \omega_1 \cdot N_{p1} \cdot N_{p2}} + \frac{2 \cdot c_3 \cdot N_{p3}}{N_{p3} + N_{g3}}\right) \cdot \frac{\left(N_{p3} + N_{g3}\right)^3}{4 \cdot b_3 \cdot c_3^2 \cdot N_{g3} \cdot N_{p3}^2} - \left(\frac{\sigma_H}{c_p}\right)^2 \cdot \frac{sin\phi \cdot cos\phi}{0.0334 \cdot W \cdot K_o \cdot K_M} \le 0 \end{aligned}$$

$$G_8(\bar{X}) = \left(\frac{366000 \cdot N_{g1} \cdot N_{g2} \cdot N_{g3}}{\pi \cdot \omega_1 \cdot N_{p1} \cdot N_{p2} \cdot N_{p3}} + \frac{2 \cdot c_4 \cdot N_{p4}}{N_{p4} + N_{g4}}\right) \cdot \frac{\left(N_{p4} + N_{g4}\right)^3}{4 \cdot b_4 \cdot c_4^2 \cdot N_{g4} \cdot N_{p4}^2} - \left(\frac{\sigma_H}{c_p}\right)^2 \cdot \frac{sin\phi \cdot cos\phi}{0.0334 \cdot W \cdot K_o \cdot K_M} \leq 0$$

$$G_{9-12}(\bar{X}) = -N_{pi} \cdot \sqrt{\frac{sin2\phi}{4} - \frac{1}{N_{pi}} + \left(\frac{1}{N_{pi}}\right)^2} + N_{gi} \cdot \sqrt{\frac{sin2\phi}{4} + \frac{1}{N_{gi}} + \left(\frac{1}{N_{gi}}\right)^2} + \frac{sin\phi \cdot \left(N_{pi} + N_{gi}\right)}{2} + C \cdot R_{min} \cdot \pi \cdot cos\phi \leq 0$$

$$G_{13-1}\ (\bar{X}) = d_{min} - \frac{2 \cdot c_i \cdot N_{pi}}{N_{pi} + N_{gi}} \leq 0$$

$$G_{17-20}(\bar{X}) = d_{min} - \frac{2 \cdot c_i \cdot N_{gi}}{N_{pi} + N_{gi}} \leq 0$$

$$G_{21}(\bar{X}) = X_{p1} + \frac{\left(N_{p1} + 2\right) \cdot c_1}{N_{p1} + N_{g1}} - L_{max} \leq 0$$

$$G_{22-24}(\bar{X}) = -L_{max} + \left(\frac{\left(N_{pi} + 2\right) \cdot c_i}{N_{pi} + N_{gi}}\right)_{i=2,3.4} + X_{g(i-1)} \leq 0$$

$$G_{25}(\bar{X}) = -X_{p1} + \left(\frac{\left(N_{p1} + 2\right) \cdot c_1}{N_{p1} + N_{g1}}\right) \leq 0$$

$$G_{26-2}\ (\bar{X}) = \left(\frac{\left(N_{pi} + 2\right) \cdot c_i}{N_{pi} + N_{gi}} - X_{g(i-1)}\right)_{i=2,3,4} \leq 0$$

$$G_{29}(\bar{\mathrm{X}}) = Y_{p1} + \left(\frac{\left(N_{p1} + 2\right) \cdot c_1}{N_{p1} + N_{g1}}\right) - L_{max} \leq 0 \qquad (6.18)$$

$$G_{30-}\ (\bar{\mathrm{X}}) = -L_{max} + \left(\frac{\left(N_{pi} + 2\right) \cdot c_i}{N_{pi} + N_{gi}} + Y_{g(i-1)}\right)_{i=2,3,4} \leq 0$$

$$G_{33}(\bar{\mathrm{X}}) = -Y_{p1} + \left(\frac{\left(N_{p1} + 2\right) \cdot c_1}{N_{p1} + N_{g1}}\right) \leq 0$$

$$G_{34}\ \ (\bar{\mathrm{X}}) = \left(\frac{\left(N_{pi} + 2\right) \cdot c_i}{N_{pi} + N_{gi}} - Y_{g(i-1)}\right)_{i=2,3,4} \leq 0$$

$$G_{37-40}(\bar{\mathrm{X}}) = -L_{max} + \frac{\left(N_{gi} + 2\right) \cdot c_i}{N_{pi} + N_{gi}} + X_{gi} \leq 0$$

$$G_{41}\ \ (\bar{\mathrm{X}}) = -X_{gi} + \frac{\left(N_{gi} + 2\right) \cdot c_i}{N_{pi} + N_{gi}} \leq 0$$

$$G_{45-4}\ (\bar{\mathrm{X}}) = Y_{gi} + \frac{\left(N_{gi} + 2\right) \cdot c_{\mathrm{i}}}{N_{pi} + N_{gi}} - L_{max} \leq 0$$

$$G_{49-52}(\bar{\mathrm{X}}) = -Y_{gi} + \frac{\left(N_{gi} + 2\right) \cdot c_{\mathrm{i}}}{N_{pi} + N_{gi}} \leq 0$$

$$G_{53}\ \ (\bar{\mathrm{X}}) = (b_i - 8.255) \cdot (b_i - 5.715) \cdot (b_i - 12.70) \cdot (0.945c_i - N_{pi} - N_{gi}) \cdot (-1) \leq 0$$

$$G_{57-60}(\bar{\mathrm{X}}) = (b_i - 8.255)(b_i - 3.175)(b_i - 12.70)(0.646c_i - N_{pi} - N_{gi}) \leq 0$$

$$G_{61-64}(\bar{\mathrm{X}}) = (b_i - 5.715)(b_i - 3.175)(b_i - 12.70)(0.504c_i - N_{pi} - N_{gi}) \leq 0$$

$$G_{65-68}(\bar{\mathrm{X}}) = (b_i - 5.715)(b_i - 3.175)(b_i - 8.255)(0c_i - N_{pi} - N_{gi}) \leq 0$$

$$G_{69-72}(\bar{\mathrm{X}}) = (b_i - 8.255)(b_i - 5.715)(b_i - 12.70)(N_{pi} + N_{gi} - 1.812c_i)(-1) \leq 0$$

$$G_{73-7}\ (\bar{\mathrm{X}}) = (b_i - 8.255)(b_i - 3.175)(b_i - 12.70)(N_{pi} + N_{gi} - 0.945c_i) \leq 0$$

$$G_{77-80}(\bar{\mathrm{X}}) = (b_i - 5.715)(b_i - 3.175)(b_i - 12.70)(N_{pi} + N_{gi} - 0.646c_i)(-1) \leq 0$$

$$G_{81-84}(\bar{\mathrm{X}}) = (b_i - 5.715)(b_i - 3.175)(b_i - 8.255)(N_{pi} + N_{gi} - 0.504c_i) \leq 0$$

$$G_{85}(\bar{\mathrm{X}}) = \omega_{MIN} - \frac{\omega_1 \cdot N_{p1} \cdot N_{p2} \cdot N_{p3} \cdot N_{p4}}{N_{g1} \cdot N_{g2} \cdot N_{g3} \cdot N_{g4}} \leq 0$$

$$G_{86}(\bar{\mathrm{X}}) = \frac{\omega_1 \cdot N_{p1} \cdot N_{p2} \cdot N_{p3} \cdot N_{p4}}{N_{g1} \cdot N_{g2} \cdot N_{g3} \cdot N_{g4}} - \omega_{MAX} \leq 0$$

where:

$$\bar{\mathrm{X}} = \{N_{p1}, N_{g1}, N_{p2}, N_{g2}, \ldots, b_1, b_2, \ldots, X_{p1}, X_{g1}, X_{p2}, X_{g2}, Y_{p1}, Y_{g1}, Y_{p2}, Y_{g2}, \ldots, Y_{g4}\}, c_i = \sqrt{\left(Y_{gi} - Y_{pi}\right)^2 + \left(X_{gi} - X_{pi}\right)^2}, \sigma_N = 2090, \sigma_H = 3290,$$

$$J_R = 0.2, c_p = 464, W = 55.9, K_o = 1.5, K_M = 1.6, d_{min} = 25, L_{max} = 127, \phi = 120°, C \cdot R_{min} = 1.4, \omega_1 = 5000\ , \omega_{MIN} = 245, \omega_{MAX} = 255.$$

with the following boundaries:

$$7 \leq N_{pi}, N_{gi} \leq 76\ \in integer\ , b_{\mathrm{i}} \in \{3.175, 12.7, 8.255, 5.715\}, X_{pi}, X_{gi}, Y_{pi}, Y_{gi} \in \{12.7, 38.1, 25.4, 50.8, 76.2, 63.5, 88.9, 114.3, 101.6\}$$

Problem 14 seeks to maximize the dynamic load capacity of a rolling element bearing. The design variables are the pitch diameter (D_m), balls diameter (D_b), number of balls (Z, integer variable), inner raceway curvature coefficient (f_i), outer raceway curvature coefficient (f_o), minimum ball diameter limiter (K_{Dmin}), maximum ball diameter limiter (K_{Dmax}), outer ring strength parameter (ε), mobility condition parameter (e), and bearing width limiter (ζ). The

design constraints are related to geometry, kinematics, structural integrity, and operational feasibility. The optimization problem is stated as:

$$Maximize\ W(\bar{X}) = \begin{cases} if\ D_b \le 25.4\ mm: & f_c \cdot Z^{\frac{2}{3}} \cdot D_b^{1.8} \\ otherwise: & 3.647 \cdot f_c \cdot Z^{\frac{2}{3}} \cdot D_b^{1.4} \end{cases} \tag{6.19}$$

$Subject\ to:$

$$\begin{aligned} &G_1(\bar{X}) = Z - \frac{\phi_0}{2 \cdot arcsin\left(\frac{D_b}{D_m}\right)} - 1 \le 0 \\ &G_2(\bar{X}) = K_{Dmin} \cdot (D-d) - 2D_b \le 0 \\ &G_3(\bar{X}) = 2 \cdot D_b - K_{Dmax} \cdot (D-d) \le 0 \\ &G_4(\bar{X}) = \zeta \cdot B_w - D_b \le 0 \\ &G_5(\bar{X}) = 0.5 \cdot (D+d) - D_m \le 0 \\ &G_6(\bar{X}) = D_m - (0.5+e) \cdot (D+d) \le 0 \\ &G_7(\bar{X}) = \varepsilon \cdot D_b - 0.5 \cdot (D - D_m - D_b) \le 0 \\ &G_8(\bar{X}) = 0.515 - f_i \le 0 \\ &G_9(\bar{X}) = 0.515 - f_o \le 0 \end{aligned} \tag{6.20}$$

where:

$$D = 160, d = 90, B_w = 30, T = D - d - 2 \cdot D_b, \gamma = D_b\ /D_m,$$

$$\phi_0 = 2 \cdot \pi \cdot arccos\left(\frac{\left[\left\{\frac{(D-d)}{2} - 3 \cdot \left(\frac{T}{4}\right)\right\}^2 + \left\{\frac{D}{2} - \frac{T}{4} - D_b\right\}^2 - \left\{\frac{d}{2} + \frac{T}{4}\right\}^2\right]}{2 \cdot \left\{\frac{(D-d)}{2} - 3 \cdot \left(\frac{T}{4}\right)\right\} \cdot \left\{\frac{D}{2} - \frac{T}{4} - D_b\right\}} \right),$$

$$f_c = 37.91 \cdot \left[1 + \left\{1.04 \cdot \left(\frac{1-\gamma}{1+\gamma}\right)^{1.72} \cdot \left(\frac{f_i \cdot (2 \cdot f_o - 1)}{f_o \cdot (2 \cdot f_i - 1)}\right)^{0.41}\right\}^{\frac{10}{3}}\right]^{-0.3} \left[\frac{\gamma^{0.3}(1-\gamma)^{1.39}}{(1+\gamma)^{\frac{1}{3}}}\right] \left[\frac{2f_i}{2f_i - 1}\right]^{0.41},$$

$with\ the\ following\ boundaries$:

$125 \le D_m \le 150$; $10.5 \le D_b \le 31.5$; $4 \le Z \le 50$; $0.515 \le f_i, f_o \le 0.6$; $0.4 \le K_{Dmin} \le 0.5$; $0.6 \le K_{Dmax} \le 0.7$; $0.3 \le \varepsilon \le 0.4$; $0.02 \le e \le 0.1$; $0.6 \le \zeta \le 0.85$.

The objective of Problem 19 is to optimize the material distribution for a prescribed set of loads, within a defined design domain and subject to performance-related constraints. A simply supported bar is discretized into 30 finite elements and subjected to a vertical load F applied at midspan. The optimization seeks to minimize the structural compliance, equivalently, to maximize the stiffness, by iteratively refining the material topology within the design space.

In this formulation, $[K]$ and $\{u\}$ denote the global stiffness matrix and the global nodal displacement vector, respectively, while k_e and u_e represent the stiffness matrix and displacement vector of the e-th element. Two constraints ensure that the total structural volume equals a prescribed target value V_0, and that the nodal forces remain in equilibrium with the applied loads. To achieve this, a density variable X_e is assigned to each element, representing its relative contribution to the overall stiffness or compliance of the structure. This relationship is typically governed by a power-law interpolation scheme. The optimization problem can thus be formulated as follows:

$$Minimimze\ W(\bar{X}) = \{u\}^T[K]\{u\} = \sum_{e=1}^{30}(X_e)^p \cdot u_e^T \cdot k_e \cdot u_e \tag{6.21}$$

$Subject\ to:$

$$H_1(\bar{X}) = \frac{V(\bar{X})}{V_o} - 1 = 0$$
$$H_2(\bar{X}) = [\bar{K}] \cdot \{u\} - \{F\} = 0 \tag{6.22}$$

where:
p=3

$with\ the\ following\ boundaries, for\ e = 1, \dots, 30$:

$0 < X_e \leq 1$

The final mechanical engineering optimization problem concerns the crashworthiness optimization of a vehicle structure. The objective is to minimize the structural weight while ensuring that the vehicle can effectively absorb the energy from a lateral impact caused by a moving barrier. The vehicle's structural configuration must therefore be optimized to guarantee occupant safety, reduce mass, and maintain overall performance.

Figure 6.2 (adapted from Ref. [207]) illustrates the vehicle frame components subject to optimization, along with the moving barrier and the direction of the side impact considered in the analysis.

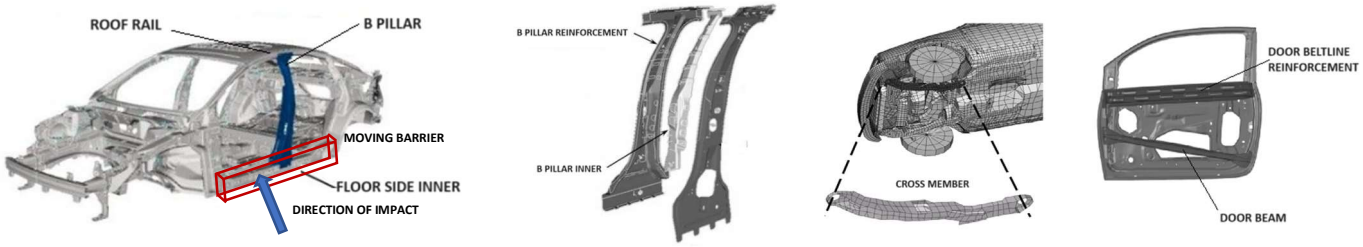


Figure 6.2. Schematic of vehicle parts (adapted from [207]) optimized in the car side impact problem.

The test problem involves 11 design variables, defined as follows: the B-pillar inner thickness (X_1), B-pillar reinforcement thickness (X_2), floor side inner thickness (X_3), cross-member thickness (X_4), door beam thickness (X_5), door beltline reinforcement thickness (X_6), and roof rail thickness (X_7); the shape factor of the B-pillar inner (X_8) and the shape factor of the floor side inner (X_9); the barrier height (X_{10}), and the lateral impact position relative to the vehicle's center of mass (X_{11}).

The mathematical formulation of the optimization is given earlier in Section 5.3.4; the governing reference equations are 5.10 and 5.11.

The objective function $W(\bar{X})$, representing the structural weight of the vehicle components considered for optimization, and the constraint functions $G(\bar{X})$were approximated in [233] using response surface models to address the high nonlinearity of the problem. Further details

on this test case can be found in Refs. [132], [140], [177], [207], [215], [216], [232], [233], [234], [270], [272], [273], [274].

The best solution reported in the literature, obtained using the SHGWJA algorithm [207], corresponds to $\{0.5, 1.21204, 0.5, 0.77908, 0.5, 1.49004, 0.5, 0.345, 0.345, -28.9781, 0.0001\}$, achieving a minimum structural weight of 21.38340 kg.

HALSGWJA was implemented in MATLAB as a standalone optimization code. Optimization runs were performed on a standard portable computer with 16 MB of RAM and a 3.1 MHz AMD processor. To obtain statistically significant information on HALSGWJA's computational efficiency, each test problem was solved through 25 independent optimization runs, each starting from a different initial population. The population size and maximum number of iterations for HALSGWJA were consistently set to 10 and 5000, respectively, except for CEC2020 Problems 2 and 11, where the maximum number of iterations was increased to 10000.

Following the approach used for SHGWJA and FHGWJA [170], [207], initial populations always included candidate solutions with up to 1000% constraint violation, to evaluate the algorithm's ability to rapidly guide the search toward the optimal region. It should be noted that HALSGWJA consistently required far fewer analyses than the theoretical computational budget of $N_{POP} \times N_{itermax}$ evaluations. In fact, constraint evaluations performed for updating the population in each iteration accounted for more than 90% of the total computational cost, confirming that the fast line search implemented by the algorithm is computationally inexpensive.

HALSGWJA handled optimization constraints using the static penalty function (SPF) approach described in Equations 4.10 and 4.11. In SPF, penalty factors are independent of the iteration number and remain constant throughout the search process (see the classical surveys [277], [278]). For comparison, competitors of HALSGWJA employed several other constraint-handling techniques, including:

1. Dynamic penalty function (DPF) methods, where the penalty term increases over iterations.
2. DPF variants where penalties for infeasible solutions increase when solutions move away from the feasible region and decrease as the search approaches feasibility.
3. Hybrid augmented Lagrangian with interior-point penalty, where the penalty is small far from constraints and tends to infinity near constraint boundaries.
4. Linear combinations of the objective and constraint functions, with or without varying penalty factors.
5. Adaptive penalty function (APF), where individuals violating constraints are reinitialized.
6. Death penalty approaches, where infeasible solutions are eliminated or assigned an extreme objective value to remove them from the search.

7. Hierarchical sorting (HS), prioritizing solutions with lower constraint violations regardless of objective value, always selecting the lowest-cost feasible solution as the current best.
8. Dynamic constraint tolerance, where the tolerance εgradually decreases during the optimization: early iterations use relaxed constraints to maintain diversity, while later iterations enforce stricter feasibility ($G(\bar{X}) \leq \varepsilon$, with ε decreasing from 0.01 initially to 0.001 at the final iteration).

Following SHGWJA [170], a sensitivity analysis was performed on HALSGWJA by varying the penalty factor p in Equation 4.10 from 10^0 to 10^6, simulating a dynamic scenario where penalties depend on the distance from constraint boundaries. Large penalty values amplify constraint violations, forcing the optimizer to generate high-quality designs that satisfy constraints. Remarkably, the maximum deviation from the best optimized cost obtained for varying penalty factors never exceeded 0.001%, confirming HALSGWJA's insensitivity to the choice of constraint-handling parameters.

6.3.1. Results of CEC 2020 problems

The HALSGWJA algorithm developed in this study was compared with 18 state-of-the-art metaheuristic optimizers. HALSGWJA's competitors can be grouped into three categories: (a) CEC2020 winners; (b) advanced variants of GWO and JAYA that have already demonstrated strong performance comparable to the CEC2020 winners; (c) physics-based and nature-inspired algorithms released between 2023 and 2025, shown to outperform high-performing or well-established metaheuristic optimizers.

Specifically, the following algorithms were included in the comparison:

1. EnMODE- Enhanced Multi-Operator Differential Evolution [250].
2. COLSHADE- Linearly Reduced Population Size Success History-based Adaptive Differential Evolution for Constrained Optimization [251].
3. En(L)SHADE- Success History-based Adaptive Differential Evolution with a gradient-based repair strategy [193].
4. IMODE-KG- Improved Multi-Operator Differential Evolution with a Knowledge-Guided Information Sharing strategy [252].
5. DESS- Differential Evolution with Self-Adaptive Spherical Search [253].
6. Self-Adaptive Spherical Search [196].
7. SDDS-SABC- Split-Detect-Discard-Shrink combined with Sophisticated Artificial Bee Colony, including modifications in initialization and search strategies [197].
8. IYDSE- Improved Young's Double-Slit Experiment [254].
9. KOA- Kepler Optimization Algorithm [264].

10. CGKOA- Improved Kepler Optimization Algorithm with adaptive function strategy, sinusoidal chaotic gravity strategy, lateral crossover strategy, and elite gold rush strategy [255].
11. MSFPSO- Multi-Strategy Fusion Enhanced Particle Swarm Optimization [256].
12. ESAO- Enhanced Snow Ablation Optimizer [257].
13. IDMO- Improved Dwarf Mongoose Optimization [258].
14. LMWOAGWO- Hybrid algorithm combining Lévy flight with modified Whale Optimization Algorithm (WOA) and Grey Wolf Optimizer (GWO) [220].
15. CSELGWO- Contact List Subpopulation Mixed Evolution Grey Wolf Optimizer [259]. This multi-strategy enhanced GWO variant obtains high-quality local information, generates a subpopulation updated with the main population via mixed evolution, significantly improving diversity and convergence. Levy flight with archives and activation mechanisms aids in escaping local optima.
16. MPSGO- Marine Predators Social Group Optimization [260].
17. MAHA- Modified Artificial Hummingbird Algorithm [261].
18. EJAYA- Enhanced JAYA [177], which updates design variables based on the best and worst solutions, the average solution, and a historical population of candidate solutions. Local exploitation is guided by weighted averages of best/worst and average solutions, while global exploration leverages historical population information.

The selection of these competitor algorithms makes the evaluation of HALSGWJA's performance particularly meaningful. In fact, EnMODE [250], COLSHADE [251], and SASS [196] were winners of the CEC2020 competition and thus represent the top-ranked challengers on real-world engineering problems, as reported in [193], [197], [220], [253], [259], [260], [261]. As reported in [193], En(L)SHADE achieved a better normalized score (0.2849 vs. 0.3371 and 0.7162, respectively) and a higher feasibility rate (98.25% vs. 97.89% and 74.39%, respectively) than SASS and COLSHADE across all independent optimization runs for the entire CEC2020 library of real-world engineering problems. The normalized score was calculated by weighting the problem dimensionality with the normalized adjusted objective function values of the best, mean, and median solutions. SDDS-ABC [197] outperformed COLSHADE specifically on the nineteen CEC2020 mechanical engineering problems considered in this study, achieving a Friedman's test score of 2.32 compared to 2.37 for COLSHADE (see Ref. [197]). IMODE-KG [252] surpassed twelve other metaheuristic optimizers, including the CEC2020 winners EnMODE and SASS. Its Friedman's test score was 2.2, compared to 3.2 and 4.3667 for EnMODE and SASS, respectively. Notably, these results were obtained with a computational budget of only 10,000 analyses, much smaller than the typical CEC2020 budget (100000 analyses for $NDV \leq 10$ and 200000 for $10 < NDV \leq 30$). Under the same limited budget of 10000 analyses, the advanced JAYA variant EJAYA [177] was reported in [252] to be highly competitive with EnMODE, SASS, and IMODE-KG, achieving a Friedman's test score of 3.7, approximately in the middle of the 2.2-4.3667 range observed for these three algorithms. It is worth noting that Ref. [262] (2024) presented

numerical experiments on CEC2020 real-world mechanical engineering problems, reporting the exact number of analyses required by COLSHADE, SASS, and EnMODE in comparison to the Piranha Predation Optimization Algorithm (PPOA) developed in the same study. Although the theoretical computational budget was set to 1000000 analyses, COLSHADE, SASS, and EnMODE completed their optimizations in some problems with fewer than 10000 analyses on average: COLSHADE required 220-8999 analyses (problems 6, 7, 15, 17, 18, 19); SASS required 360-7605 analyses (problems 9, 17, 19); and EnMODE required 215-9948 analyses (problems 1, 2, 6, 17). However, the success rate of these algorithms in finding the target solutions was 0 in five problems (4, 8, 10, 14, 16), which contrasts with other studies that re-ran COLSHADE, SASS, and EnMODE on benchmark problems. This discrepancy may be attributed to the specific implementation details used in Ref. [262]. DESS [253] outperformed 18 other metaheuristic optimizers, including the top-ranked CEC2020 solvers EnMODE, SASS, and COLSHADE. DESS achieved a normalized score of 0.1230, compared to 0.1378, 0.2427, and 0.4514 for EnMODE, SASS, and COLSHADE, respectively. MPSGO [260] and MAHA [261], two nature-inspired algorithms released in 2024 that model animal behaviours and, in some cases, integrate aspects of human social behaviour, were also highly competitive with the CEC2020 winners. Specifically, MPSGO matched the best designs obtained by EnMODE and COLSHADE in 17 out of 19 CEC2020 mechanical engineering problems, while MAHA ranked second overall, behind SASS but ahead of EnMODE and COLSHADE across the complete set of real-world engineering problems. The hybrid LMWOAGWO algorithm [220], combining GWO and WOA, and the advanced GWO variant CSELGWO [259] also performed competitively, ranking immediately after EnMODE and COLSHADE in 17 of the 19 CEC2020 mechanical engineering problems. Reported scores were 3.23 for LMWOAGWO (vs. 2.14-2.45 for the top algorithms) and 2.54 for CSELGWO (vs. 2.41-2.5). IYDSE [254], KOA [264], GKOA [255], and ESAO [257] are four physics-based metaheuristic algorithms released between 2023 and 2025 that ranked first among 8-10 other metaheuristic optimizers, including variants of PSO, GWO, WOA, the Starling Murmuration Optimizer (SMO), Artificial Hummingbird Algorithm (AHA), and Dwarf Mongoose Optimization (DMO). IDMO [258], an improved variant of DMO released in 2023, outperformed 12 other metaheuristics, including highly cited methods such as GWO and WOA. MSFPSO [256], released in 2025, surpassed 10 other metaheuristic algorithms, including five additional PSO variants. Table 6.3.a-b compares the optimization results obtained by HALSGWJA on the CEC2020 mechanical engineering problems with those reported in the literature for the selected competitors. For each test problem, the table reports the best (B), average (A), and worst (W) optimized cost values, the required number of analyses, and the corresponding standard deviations (STD) over 30 independent runs. Statistical data on the computational cost of HALSGWJA's competitors (i.e., number of analyses and STD) are included where available. For brevity, the average number of analyses and standard deviation are reported in the same row and indicated as (A/D). Additionally, the A/B/W/STD and A/D nomenclature is explained only for HALSGWJA and the first test problem. Data for EnMODE, COLSHADE, SASS, IYDSE, KOA, and EJAYA were aggregated from their original studies and subsequent investigations; the best solutions for each test case are reported in Table 6.3.a-b.

Table 6.3.a. COMPARISON OF OPTIMIZATION RESULTS OBTAINED BY HALSGWJA AND ITS COMPETITORS IN THE CEC2020 MECHANICAL ENGINEERING PROBLEMS.

Problem	HALSGWJA Present	EnMODE [250, 220, 261, 262♥]	COLSHADE [251, 220, 261, 262♥]	En(L)SHADE [193]	IMODE-KG [252]	DESS [253]	SASS [196, 197, 261, 262♥]	SDDS-SABC [197]	IYDSE [254,252]
1 SPEED REDUCER	(*B*) 2994.425	2994.4	2994.4	$2.99\cdot10^{3}$	2994.425	$2.89\cdot10^{3}$	2994.425	2963.911 [1]	$2.99\cdot10^{3}$
	(*A*) 2994.425	2994.4	2994.4	$2.99\cdot10^{3}$	2994.425	$2.99\cdot10^{3}$	2994.425	2975.081 [1]	$2.99\cdot10^{3}$
	(*W*) 2994.425	2994.4	2994.4	$2.99\cdot10^{3}$	2994.425	N/A	2994.425	2982.083 [1]	$3.10\cdot10^{3}$
	(*STD*) 0	$4.6412\cdot10^{-13}$	$4.5475\cdot10^{-13}$	0	$1.1003\cdot10^{-7}$	N/A	$4.641\cdot10^{-13}$	1.32691	0.216
	(*B*) 2326								
	(*A/D*) 2538 ± 116	100000	100000	100000	10000	100000	100000	20000	30000
	(*W*) 2728	{2277}♥	{12930}♥				{51430}♥		
2 INDUSTRIAL REFRIGERATION SYSTEM	0.0322130	0.032213	0.032213	$3.22\cdot10^{-2}$	N/A	$3.22\cdot10^{-2}$	0.032213	0.0321969 [1]	$3.29\cdot10^{-2}$
	0.0322132	0.032213	0.032213	$3.22\cdot10^{-2}$	N/A	$3.22\cdot10^{-2}$	0.032213	0.0334143	$4.99\cdot10^{-2}$
	0.0322160	0.032213	0.032213	$3.22\cdot10^{-2}$	N /A	N/A	0.032213	0.0335463	0.115
	$6.7082\cdot10^{-7}$	$3.1672\cdot10^{-18}$	0	0	N/A	N/A	$1.416\cdot10^{-17}$	0.26439	0.0233
	4695								
	5478± 1003	200000	200000	200000	N/A	200000	200000	20000	30000
	6258	{9948}♥	{37190}♥				{174700}♥		
3 TENSION/ COMPRESSION SPRING (CASE 1)	0.012665	0.012665	0.012665	$1.27\cdot10^{-2}$	0.0126742	$1.27\cdot10^{-2}$	0.012665	0.0112423 [1]	$1.27\cdot10^{-2}$
	0.012668	0.012710	0.012665	$1.27\cdot10^{-2}$	0.0132706	$1.27\cdot10^{-2}$	0.012665	0.0121864 [1]	$1.27\cdot10^{-2}$
	0.012670	0.012719	0.012665	$1.27\cdot10^{-2}$	0.0149735	N/A	0.012665	0.0136756	$1.27\cdot10^{-2}$
	$2.1152\cdot10^{-6}$	$2.0138\cdot10^{-5}$	$1.0625\cdot10^{-7}$	0	$4.7759\cdot10^{-4}$	N/A	0	$1.589\cdot10^{-3}$	$1.34\cdot10^{-7}$
	2103								
	3316 ± 1137	100000	100000	100000	10000	100000	100000	20000	30000
	4097	{47560}♥	{14630}♥				{43890}♥		
4 PRESSURE VESSEL	6059.714 (*5885.331*)	6059.7	6059.7	$6.06\cdot10^{3}$	6059.714	$6.06\cdot10^{3}$	6059.714	5979.985	6103.368
	6059.714 (*5885.331*)	6059.7	6062.2	$6.06\cdot10^{3}$	6256.737	$6.06\cdot10^{3}$	6059.714	5981.044	6385.287
	6059.714 (*5885.331*)	6059.7	6090.5	$6.09\cdot10^{3}$	6410.087	N/A	6059.714	5985.026	6810.762
	0 (*0*)	$9.2825\cdot10^{-13}$	8.3591	11.5	155.130	N/A	$3.713\cdot10^{-12}$	2.96641	185.361
	3852 (*3510*)								
	5104 ± 1374	100000	100000	100000	10000	100000	100000	20000	10000
	(*4672 ±1203*)	{N/A}♥	{N/A}♥				{N/A}♥		
	6193 (*5713*)								
5 WELDED BEAM	1.670218	1.6702	1.6702	1.67	1.67022	1.67022	1.670218	1.452753 [1]	1.67
	1.670218	1.6702	1.6702	1.67	1.67022	1.67022	1.670218	1.580213 [1]	1.67
	1.670218	1.6702	1.6702	1.67	1.67022	N/A	1.670218	1.690036	1.67
	0	0	0	0	$1.2677\cdot10^{-9}$	N/A	$2.266\cdot10^{-16}$	0.96852	$1.16\cdot10^{-5}$
	2975								
	3224 ± 285	100000	100000	100000	10000	100000	100000	20000	30000
	3691	{22930}♥	{16060}♥				{67660}♥		
6 THREE-BAR TRUSS	263.8915	263.90	263.90	$2.64\cdot10^{2}$	N/A	$2.64\cdot10^{2}$	263.8958	263.6053 [1]	$2.64\cdot10^{2}$
	263.8915	263.90	263.90	$2.64\cdot10^{2}$	N/A	$2.64\cdot10^{2}$	263.8958	263.6724 [1]	$2.64\cdot10^{2}$
	263.8915	263.90	263.90	$2.64\cdot10^{2}$	N/A	N/A	263.8958	264.2198	$2.64\cdot10^{2}$
	0	0	0	0	N/A	N/A	$5.802\cdot10^{-14}$	$1.275\cdot10^{-3}$	$4.12\cdot10^{-14}$
	705								
	961 ± 168	100000	100000	100000	N/A	100000	100000	20000	30000
	1224	{8581}♥	{5594}♥				{12760}♥		
7 MULTIPLE DISK CLUTCH BRAKE	0.235243	0.23524	0.23524	0.235	0.23524	0.235	0.235242	0.218881 [1]	0.235
	0.235243	0.23524	0.23524	0.235	0.23524	0.235	0.235242	0.236467	0.235
	0.235243	0.23524	0.23524	0.235	0.23524	N/A	0.235242	0.251975	0.235
	0	$1.1331\cdot10^{-16}$	0	0	$1.4021\cdot10^{-16}$	N/A	$2.833\cdot10^{-17}$	$2.656\cdot10^{-2}$	$3.37\cdot10^{-11}$
	792								
	900 ± 85	100000	100000	100000	10000	100000	100000	20000	30000
	1193	{13040}♥	{4858}♥				{12910}♥		
8 PLANETARY GEAR TRAIN	0.523250	0.52577	0.52577	0.526	0.526581	0.526	0.525967	0.523274	0.526
	0.525714	0.52691	0.54103	0.633	0.558675	0.532	1.001524	0.532176	0.528
	0.529188	0.53121	0.74667	1.300	0.674039	N/A	3.521656	0.538756	0.537
	$1.2440\cdot10^{-3}$	$1.4402\cdot10^{-3}$	0.042573	0.231	0.036539	N/A	0.725184	$5.416\cdot10^{-4}$	$2.81\cdot10^{-3}$
	2095								
	3260 ± 1039	100000	100000	100000	10000	100000	100000	20000	30000
	4364	{N/A}♥	{N/A}♥				{N/A}♥		
9 STEP-CONE PULLEY	16.03601	16.070	16.070	16.1	N/A	16.1	16.06987	12.30213 [1]	16.1
	16.04033	16.070	16.070	16.1	N/A	16.1	16.06987	15.96217 [1]	16.1
	16.04529	16.070	16.070	16.1	N/A	N/A	16.06987	17.10746	16.1
	$4.6437\cdot10^{-3}$	$3.3335\cdot10^{-14}$	0	0	N/A	N/A	$3.626\cdot10^{-15}$	2.01784	$4.30\cdot10^{-6}$
	4278								
	4424 ± 322	100000	100000	100000	N/A	100000	100000	20000	30000
	5000	{38090}♥	{14090}♥				{4826}♥		

10 ROBOT GRIPPER	2.5437852	2.5438	2.5438	2.54	N/A	2.54	2.543786	2.673169	2.70
	2.5437854	2.5438	2.5438	2.54	N/A	2.54	2.543786	2.678953	3.14
	2.5437857	2.5438	2.5438	2.54	N/A	N/A	2.543786	2.681085	3.47
	$2.8263 \cdot 10^{-7}$	$1.3501 \cdot 10^{-12}$	0	0	N/A	N/A	0	0.264387	0.218
	3988								
	4645 ± 787	100000	100000	100000	N/A	100000	100000	20000	30000
	5302	{N/A}♥	{N/A}♥				{N/A}♥		
11 HYDRO-STATIC THRUST BEARING	1616.120	1616.1	1616.1	$1.62 \cdot 10^{3}$	1783.954	$1.62 \cdot 10^{3}$	1616.120	1839.641	2483.159
	1616.120	1616.1	1639.0	$1.62 \cdot 10^{3}$	1918.389	$1.62 \cdot 10^{3}$	1616.120	1841.907	3201.317
	1616.120	1616.1	2129.1	$1.62 \cdot 10^{3}$	2242.035	N/A	1616.123	1845.063	4174.048
	0	$1.7751 \cdot 10^{-11}$	1007.3	0	115.871	N/A	$9.425 \cdot 10^{-4}$	1.42788	469.274
	7400								
	8624 ± 724	100000	100000	100000	10000	100000	100000	20000	10000
	9527	{43930}♥	{66620}♥				{64410}♥		
12 FOUR-STAGE GEAR BOX	35.29269	35.359	35.359	36.2	N/A	36.2	36.2504	43.14295	14.9^{2}
	35.41012	35.728	36.611	39.6	N/A	39.5	38.5141	44.01259	33.9^{2}
	35.65141	37.268	40.931	44.1	N/A	N/A	45.5064	45.36397	67.9
	0.1687	0.59911	1.3677	2.0	N/A	N/A	2.11482	0.49725	14.9
	591								
	1021 ± 318	200000	200000	200000	N/A	200000	200000	20000	30000
	1531	{139000}♥	{111700}♥				{61630}♥		
13 TEN-BAR TRUSS	524.5888	524.45	524.45	$5.24 \cdot 10^{2}$	524.480	$5.24 \cdot 10^{2}$	524.4576	522.9644^{1}	535.9481
	524.6004	524.45	524.45	$5.24 \cdot 10^{2}$	525.075	$5.24 \cdot 10^{2}$	524.4692	523.0724^{1}	545.6267
	524.6111	524.45	524.45	$5.24 \cdot 10^{2}$	528.814	N/A	524.4820	525.0326	559.1333
	0.0113519	$3.7602 \cdot 10^{-7}$	0	$2.01 \cdot 10^{-9}$	0.96481	N/A	$6.620 \cdot 10^{-3}$	$3.572 \cdot 10^{-2}$	5.8317
	1911								
	3321 ± 1092	100000	100000	100000	10000	100000	100000	20000	10000
	4801	{27750}♥	{19660}♥				{N/A}♥		
14 ROLLING ELEMENT BEARING	16958.202	16958	16958	$1.70 \cdot 10^{4}$	16958.202	$1.44 \cdot 10^{4}$	14614.136	14811.085	16981.126^{2}
	16958.202	16958	16958	$1.70 \cdot 10^{4}$	16958.204	$1.58 \cdot 10^{4}$	14614.136	14835.501	17045.036^{2}
	16958.202	16958	16958	$1.70 \cdot 10^{4}$	16958.238	N/A	14614.136	14842.886	17203.713^{2}
	0	$3.7130 \cdot 10^{-12}$	0	0	$6.6535 \cdot 10^{-3}$	N/A	0	3.07322	49.492
	2492								
	3117 ± 385	100000	100000	100000	10000	100000	100000	20000	10000
	3681	{17970}♥	{N/A}♥				{52340}♥		
15 GAS TRANSMISSION COMPRESSOR	2964893.937	$2.9649 \cdot 10^{6}$	$2.9649 \cdot 10^{6}$	$2.96 \cdot 10^{6}$	2964895.42	$2.96 \cdot 10^{6}$	2964895.4	2851884.4^{1}	2965825.50
	2964893.937	$2.9649 \cdot 10^{6}$	$2.9649 \cdot 10^{6}$	$2.96 \cdot 10^{6}$	2964895.42	$2.96 \cdot 10^{6}$	2964895.4	2975324.9	2972596.89
	2964893.937	$2.9649 \cdot 10^{6}$	$2.9649 \cdot 10^{6}$	$2.96 \cdot 10^{6}$	2964895.42	N/A	2964895.4	2992643.5	3011188.39
	0	$1.4258 \cdot 10^{-9}$	0	$1.43 \cdot 10^{-9}$	$9.2584 \cdot 10^{-6}$	N/A	$4.753 \cdot 10^{-10}$	5.08416	7938.995
	4377								
	5661 ± 932	100000	100000	100000	10000	100000	100000	20000	10000
	6587	{11630}♥	{8999}♥				{58880}♥		
16 TENSION/ COMPRESSION SPRING (CASE 2)	2.658722	2.6586	2.6586	2.66	2.658559	2.66	2.658559	2.684243	2.65
	2.659550	2.8149	2.6618	2.66	2.658559	2.66	2.658559	2.694280	2.77
	2.660999	3.6359	2.6995	2.66	2.658559	N/A	2.658559	2.710875	2.93
	$9.4671 \cdot 10^{-4}$	0.3657	0.011105	0	$4.5168 \cdot 10^{-16}$	N/A	$4.532 \cdot 10^{-16}$	$4.179 \cdot 10^{-2}$	0.103
	4223								
	4705 ± 325	100000	100000	100000	10000	100000	100000	20000	30000
	4996	{N/A}♥	{N/A}♥				{N/A}♥		
17 GEAR, TRAIN	$2.3078 \cdot 10^{-21}$	0	0	0	$7.7037 \cdot 10^{-34}$	$6.71 \cdot 10^{-27}$	0	$1.6712 \cdot 10^{-3}$	$1.75 \cdot 10^{-21}$
	$3.5365 \cdot 10^{-21}$	0	$1.8807 \cdot 10^{-16}$	0	$7.9172 \cdot 10^{-14}$	$8.22 \cdot 10^{-16}$	$1.8106 \cdot 10^{-18}$	$1.8316 \cdot 10^{-3}$	$3.25 \cdot 10^{-15}$
	$4.9608 \cdot 10^{-21}$	0	$1.2074 \cdot 10^{-15}$	0	$6.5771 \cdot 10^{-13}$	N/A	$4.4913 \cdot 10^{-17}$	$2.6419 \cdot 10^{-3}$	$4.38 \cdot 10^{-14}$
	$1.1169 \cdot 10^{-21}$	0	$3.8137 \cdot 10^{-16}$	0	$1.4824 \cdot 10^{-13}$	N/A	$8.9800 \cdot 10^{-18}$	$4.2583 \cdot 10^{-3}$	$9.83 \cdot 10^{-15}$
	860								
	1170 ± 456	100000	100000	100000	10000	100000	100000	20000	30000
	1870	{215}♥	{220}♥				{360}♥		
18 HIMMELBLAU	−30664.873	−30666	−30666	$-3.07 \cdot 10^{4}$	−30665.539	$-3.07 \cdot 10^{4}$	−30665.539	−30665.834	−30660.325
	−30664.848	−30666	−30666	$-3.07 \cdot 10^{4}$	−30665.539	$-3.07 \cdot 10^{4}$	−30665.539	−30653.100	−30619.973
	−30664.834	−30666	−30666	$-3.07 \cdot 10^{4}$	−30665.539	N/A	−30665.539	−30698.769	−30561.855
	0.0195687	$3.7130 \cdot 10^{-12}$	0	0	$6.2543 \cdot 10^{-5}$	N/A	$7.426 \cdot 10^{-12}$	0.268765	25.447
	4494								
	4694 ± 219	100000	100000	100000	10000	100000	100000	20000	10000
	5001	{21190}♥	{2267}♥				{46340}♥		
19 TOPOLOGY OPTIMIZATION	2.63935	2.6393	2.6393	2.64	2.639347	2.64	2.639347	2.6378241	2.64
	2.63935	2.6393	2.6393	2.64	2.639359	2.64	2.639347	2.6381981	2.64
	2.63935	2.6393	2.6393	2.64	2.639449	N/A	2.639347	2.639520	2.64
	0	$1.0175 \cdot 10^{-15}$	0	0	$2.8581 \cdot 10^{-3}$	N/A	$4.532 \cdot 10^{-16}$	$3.153 \cdot 10^{-4}$	$1.34 \cdot 10^{-15}$
	2613								
	3021 ± 237	200000	200000	200000	10000	200000	200000	20000	30000
	3526	{31740}♥	{6496}♥				{7605}♥		

Table 6.3.b. CONTINUATION OF TABLE 6.3.A: COMPARISON OF OPTIMIZATION RESULTS OBTAINED BY HALSGWJA AND ITS COMPETITORS IN THE CEC2020 MECHANICAL ENGINEERING PROBLEMS.

Problem	HALSGWJA Present	KOA-CGKOA [264*,252+] [255♣]	MSFPSO [256]	ESAO [257]	IDMO [258]	LMWOA GWO [220]	CSELGWO [259]	MPSGO [260]	MAHA [261]	EJAYA [177,252]
1 SPEED REDUCER	(B) 2994.425 (A) 2994.425 (W) 2994.425 (STD) 0 (B) 2326 (A/D) 2538 ± 116 (W) 2728	2995.351+ 2996.772 2998.994 0.942086 10000	2994.42 2994.42 2994.42 $1.40 \cdot 10^{-12}$ N/A	2994.4 2994.4 N/A $9.3445 \cdot 10^{-6}$ 15000	2994.4 2994.4 N/A $6.8768 \cdot 10^{-5}$ 50000	N/A $3.00 \cdot 10^{3}$ N/A 0.940 100000	N/A 2994 N/A $4.623 \cdot 10^{-13}$ 100000	2994.4 2994.4 2994.4 $5.8987 \cdot 10^{-5}$ 100000	2994.425 2994.425 2994.425 $4.5475 \cdot 10^{-13}$ 100000	2994.430 2994.459 2994.526 0.0250189 10000
2 INDUSTRIAL REFRIGERATION SYSTEM	0.0322130 0.0322132 0.0322160 $6.7082 \cdot 10^{-7}$ 4695 5478 ± 1003 6258	0.032213♣ 0.76148 2.949 1.2958 50000	0.032 484.562 9690.477 2166.848 N/A	0.032213 $4.6809 \cdot 10^{13}$ N/A $2.0934 \cdot 10^{14}$ 15000	0.033960 0.037929 N/A $3.5133 \cdot 10^{-3}$ 50000	N/A $4.01 \cdot 10^{-2}$ N/A $4.43 \cdot 10^{-3}$ 200000	N/A 0.03256 N/A $1.868 \cdot 10^{-3}$ 200000	0.032213 0.032217 0.032226 $3.3268 \cdot 10^{-6}$ 200000	0.032213 0.032213 0.032213 $2.0812 \cdot 10^{-17}$ 200000	N/A N/A N/A N/A N/A
3 TENSION/ COMPRESSION SPRING (CASE 1)	0.012665 0.012668 0.012670 $2.1152 \cdot 10^{-6}$ 2103 3316 ± 1137 4097	0.012665♣ 0.012665 0.012665 0 50000	0.01267 0.01270 0.01272 $2.64 \cdot 10^{-5}$ N/A	0.012672 0.012689 N/A $1.1829 \cdot 10^{-5}$ 15000	0.012665 0.012665 N/A $2.1898 \cdot 10^{-8}$ 50000	N/A $1.27 \cdot 10^{-2}$ N/A $5.32 \cdot 10^{-5}$ 100000	N/A 0.01267 N/A $6.232 \cdot 10^{-9}$ 100000	0.012665 0.012665 0.012665 $4.6867 \cdot 10^{-9}$ 100000	0.0126652 0.0126653 0.0126662 0.16248 100000	0.012665 0.012668 0.012687 $4.6331 \cdot 10^{-6}$ 15000
4 PRESSURE VESSEL	6059.714 (*5885.331*) 6059.714 (*5885.331*) 6059.714 (*5885.331*) 0 (*0*) 3852 (*3510*) 5104 ± 1374 (*4672 ±1203*) 6193 (*5713*)	5885.434* 5885.434 N/A $1.265 \cdot 10^{-8}$ 50000	6032.55 6052.11 6090.53 20.94 N/A	6059.8 6060.6 N/A 0.63725 15000	6059.7 6063.8 N/A 10.474 50000	N/A $6.58 \cdot 10^{3}$ N/A 458 100000	N/A 6059 N/A 5.534 100000	6059.7 6059.7 6059.7 $2.6584 \cdot 10^{-5}$ 100000	5885.354 5885.414 5885.854 0.16248 100000	5885.333 5885.886 5894.777 1.734 16000
5 WELDED BEAM	1.670218 1.670218 1.670218 0 2975 3224 ± 285 3691	1.6702♣ 1.6702 1.6702 $7.10 \cdot 10^{-16}$ 50000	1.67022 1.67022 1.67022 $9.11 \cdot 10^{-16}$ N/A	1.6709 1.6715 N/A $4.9309 \cdot 10^{-4}$ 15000	1.6702 1.6702 N/A $5.8316 \cdot 10^{-9}$ 50000	N/A 1.67 N/A $4.03 \cdot 10^{-4}$ 100000	N/A 1.670 N/A $3.93 \cdot 10^{-16}$ 100000	1.6702 1.6702 1.6702 $1.1292 \cdot 10^{-7}$ 100000	1.670218 1.670218 1.670218 $2.2205 \cdot 10^{-16}$ 100000	1.670397 1. 670742 1.671558 $3.1593 \cdot 10^{-4}$ 10000
6 THREE-BAR TRUSS	263.8915 263.8915 263.8915 0 705 961 ± 168 1224	263.8958* 263.8958 263.8958 0 50000	263.8915 263.8915 263.8915 0 N/A	263.90 263.90 N/A $1.2012 \cdot 10^{-9}$ 15000	263.90 263.90 N/A $1.7053 \cdot 10^{-13}$ 50000	N/A $2.64 \cdot 10^{2}$ N/A $1.02 \cdot 10^{-4}$ 100000	N/A 263.9 N/A $2.076 \cdot 10^{-14}$ 100000	263.90 263.90 263.90 $1.0469 \cdot 10^{-4}$ 100000	263.8958 263.8958 263.8958 $1.7053 \cdot 10^{-13}$ 100000	N/A N/A N/A N/A N/A
7 MULTIPLE DISK CLUTCH BRAKE	0.235243 0.235243 0.235243 0 792 900 ± 85 1193	0.23524♣ 0.23524 0.23524 $1.14 \cdot 10^{-16}$ 50000	0.23524 0.23524 0.23524 $1.14 \cdot 10^{-16}$ N/A	0.23524 0.23606 N/A $2.2368 \cdot 10^{-3}$ 15000	0.23524 0.23524 N/A $2.8715 \cdot 10^{-16}$ 50000	N/A 0.235 N/A $1.68 \cdot 10^{-7}$ 100000	N/A 0.2352 N/A $2.411 \cdot 10^{-16}$ 100000	0.23524 0.23524 0.23524 $1.1117 \cdot 10^{-16}$ 100000	0.235243 0.235243 0.235243 0 100000	0.235243 0.235243 0.235243 $1.952 \cdot 10^{-12}$ 10000
8 PLANETARY GEAR TRAIN	0.523250 0.525714 0.529188 $1.2440 \cdot 10^{-3}$ 2095 3260 ± 1039 4064	0.52577♣ 0.52696 0.53319 0.0015832 50000	N/A N/A N/A N/A N/A	0.53571 0.56443 N/A 0.043832 15000	0.52524 0.52759 N/A $2.8802 \cdot 10^{-3}$ 50000	N/A 0.526 N/A $5.60 \cdot 10^{-4}$ 100000	N/A 0.5299 N/A $4.480 \cdot 10^{-3}$ 100000	0.52577 0.52815 0.53320 $1.9494 \cdot 10^{-3}$ 100000	0.525967 0.530809 0.543846 $4.2605 \cdot 10^{-3}$ 100000	0.527347 0.542460 0.673032 0.0256885 10000
9 STEP-CONE PULLEY	16.03601 16.04033 16.04529 $4.6437 \cdot 10^{-3}$ 4278	8.5633♣ 3 8.5633 3 8.5633 3 $9.27 \cdot 10^{-15}$	16.09027 16.59500 17.13636 0.52157	16.116 16.763 N/A 0.31569	16.090 16.090 N/A $3.8548 \cdot 10^{-8}$	N/A 16.5 N/A 0.209	N/A 16.2 N/A 0.2608	16.070 16.070 16.070 $4.6376 \cdot 10^{-7}$	16.0699 16.0699 16.0699 $3.5527 \cdot 10^{-15}$	N/A N/A N/A N/A

	4424 ± 322 5000	50000	N/A	15000	50000	100000	100000	100000	100000	N/A
10 ROBOT GRIPPER	2.5437852 2.5437854 2.5437857 $2.8263 \cdot 10^{-7}$ 3988 4645 ± 787 5302	2.5455♣ 2.6382 2.7991 0.071785 50000	2.54379 2.68250 3.11950 0.23285 N/A	2.5929 2.8768 N/A 0.25476 15000	2.5528 2.6979 N/A 0.12596 50000	N/A N/A N/A N/A N/A	N/A N/A N/A N/A N/A	2.5442 2.5496 2.5966 0.010519 100000	2.638958 2.638958 2.638958 $4.4409 \cdot 10^{-16}$ 100000	2.601873 3.135123 3.883629 0.429461 10000
11 HYDRO-STATIC THRUST BEARING	1616.120 1616.120 1616.120 0 7400 8624 ± 724 9527	2403.717+ 3168.443 4551.426 514.898 10000	137.08396 [4] 143.78383 [4] 270.73097 [4] 29.8803 N/A	1918.4 3224.5 N/A 876.2 15000	1640.9 1744.5 N/A 51.215 50000	N/A $1.83 \cdot 10^{3}$ N/A 55.2 100000	N/A $1.616 \cdot 10^{3}$ N/A 12.57 100000	1643.1 2350.3 2475.1 2.5456 100000	1624.990 1625.431 1625.450 0.0901429 100000	1625.443 1631.510 1767.661 26.272 150000
12 FOUR-STAGE GEAR BOX	35.29269 35.41012 35.65141 0.1687 591 1021 ± 318 1531	36.6012♣ 43.8296 63.54 7.3255 50000	98.69 115448.92 411255.69 126054.20 N/A	41.697 $4.8979 \cdot 10^{14}$ N/A $9.372 \cdot 10^{14}$ 15000	37.461 $2.7239 \cdot 10^{15}$ N/A $6.3419 \cdot 10^{15}$ 50000	N/A N/A N/A N/A N/A	N/A N/A N/A N/A N/A	35.359 35.699 37.258 0.56923 200000	34.63987 [5] 39.19013 48.81480 3.48167 200000	N/A N/A N/A N/A N/A
13 10-BAR TRUSS	524.5888 524.6004 524.6111 0.0113519 1911 3321 ± 1092 4801	529.4834+ 534.2848 542.7236 3.62097 10000	524.50833 527.44007 532.35484 3.02564 N/A	525.98 532.62 N/A 3.5857 15000	524.45 527.07 N/A 2.9632 50000	N/A $5.29 \cdot 10^{2}$ N/A 2.32 100000	N/A 524.4 N/A $4.989 \cdot 10^{-3}$ 100000	524.45 524.45 524.45 $2.9397 \cdot 10^{-3}$ 100000	524.4095 524.4591 524.5082 0.0257801 100000	525.0392 528.3298 535.6228 2.55928 10000
14 ROLLING ELEMENT BEARING	16958.202 16958.202 16958.202 0 2492 3117 ± 385 3681	16960.674+ 16970.531 16986.506 7.07606 10000	16958.20 16958.20 16958.20 0 N/A	16958 16958 N/A 0.013976 15000	16958 16958 N/A $6.6483 \cdot 10^{-8}$ 50000	N/A $1.70 \cdot 10^{4}$ N/A 9.29 100000	N/A $1.461 \cdot 10^{4}$ N/A 15.57 100000	16958 16958 16958 0 100000	14614.136 14614.136 14614.136 $1.8190 \cdot 10^{-12}$ 100000	16958.202 16958.203 16958.206 $7.0516 \cdot 10^{-4}$ 10000
15 GAS TRANSMISSIO N COMPRESSOR	2964893.937 2964893.937 2964893.937 0 4377 5661 ± 932 6587	2965375.3+ 2967464.3 2971180.3 1506.404 10000	1227929.15 [4] 1227929.15 [4] 1227929.15 [4] $2.39 \cdot 10^{-10}$ N/A	$2.9649 \cdot 10^{6}$ $2.9790 \cdot 10^{6}$ N/A 16072 15000	$2.9649 \cdot 10^{6}$ $2.9790 \cdot 10^{6}$ N/A $8.33 \cdot 10^{-10}$ 50000	N/A $2.96 \cdot 10^{6}$ N/A 6.15 100000	N/A $2.965 \cdot 10^{6}$ N/A $1.125 \cdot 10^{-9}$ 100000	$2.9649 \cdot 10^{6}$ $2.9649 \cdot 10^{6}$ $2.9649 \cdot 10^{6}$ $2.5505 \cdot 10^{-9}$ 100000	2964895.059 2964895.340 2964895.945 0.183209 100000	2964896.4 2964916.3 2965063.0 34.9261 10000
16 TENSION/ COMPRESSION SPRING (CASE 2)	2.658722 2.659550 2.660999 $9.4671 \cdot 10^{-4}$ 4223 4705 ± 325 4996	2.6586♣ 2.6606 2.6995 $9.1532 \cdot 10^{-3}$ 50000	2.65856 2.66879 2.69949 0.01819 N/A	2.6586 2.9313 N/A 0.2426 15000	2.6586 2.6586 N/A $9.982 \cdot 10^{-11}$ 50000	N/A 2.71 N/A 0.0948 100000	N/A 2.666 N/A 0.01644 100000	2.6586 2.6586 2.6586 $1.6402 \cdot 10^{-12}$ 100000	2.658558 2.658559 2.658559 $2.7129 \cdot 10^{-7}$ 100000	2.658560 2.662352 2.700421 0.0103828 10000
17 GEAR TRAIN	$2.3078 \cdot 10^{-21}$ $3.5365 \cdot 10^{-21}$ $4.9608 \cdot 10^{-21}$ $1.1169 \cdot 10^{-21}$ 860 1170 ± 456 1870	0♣ $6.57 \cdot 10^{-22}$ $6.90 \cdot 10^{-21}$ $2.02 \cdot 10^{-21}$ 50000	0 0 0 0 N/A	0 0 N/A 0 15000	$6.7660 \cdot 10^{-19}$ $2.3015 \cdot 10^{-16}$ N/A $3.7492 \cdot 10^{-16}$ 50000	N/A $9.53 \cdot 10^{-18}$ N/A $1.54 \cdot 10^{-17}$ 100000	N/A $7.917 \cdot 10^{-15}$ N/A $3.838 \cdot 10^{-14}$ 100000	0 0 0 0 100000	0 0 0 0 100000	$1.350 \cdot 10^{-18}$ $1.315 \cdot 10^{-13}$ $1.444 \cdot 10^{-12}$ $3.559 \cdot 10^{-13}$ 10000
18 HIMMELBLAU	−30664.873 −30664.848 −30664.834 0.0195687 4494 4694 ± 219 5001	−30663.305+ −30652.862 −30625.231 8.63379 10000	−30665.54 −30665.54 −30665.54 0 N/A	−30666 −30666 N/A $9.9915 \cdot 10^{-5}$ 15000	−30666 −30666 N/A $5.4957 \cdot 10^{-8}$ 50000	N/A $-3.07 \cdot 10^{4}$ N/A 0.318 100000	N/A $-3.067 \cdot 10^{4}$ N/A $1.109 \cdot 10^{-11}$ 100000	−30666 −30666 −30666 $5.8638 \cdot 10^{-8}$ 100000	−30665.539 −30665.539 −30665.539 $1.4552 \cdot 10^{-11}$ 100000	−30665.54 −30665.54 −30665.53 $9.6829 \cdot 10^{-4}$ 10000
19 TOPOLOGY OPTIMIZATIO N	2.63935 2.63935 2.63935 0 2613 3021 ± 237	2.6393♣ 2.6393 2.6393 $1.39 \cdot 10^{-15}$ 50000	2.63935 2.63935 2.63935 $9.11 \cdot 10^{-16}$ N/A	2.6393 2.6393 N/A $2.8925 \cdot 10^{-10}$ 15000	2.6393 2.6583 N/A $5.4519 \cdot 10^{-2}$ 50000	N/A 2.64 N/A $1.36 \cdot 10^{-15}$ 200000	N/A 2.639 N/A $3.392 \cdot 10^{-15}$ 200000	2.6393 2.6393 2.6393 0 200000	2.639347 2.639347 2.639347 $4.4409 \cdot 10^{-16}$ 200000	2.639364 2.661795 2.747870 0.0281439 10000

3526

[1] The optimized design corresponding to this solution was not shown in Ref. [197].

[2] The optimized design corresponding to this solution was not shown in Ref. [254].

[3] The optimized design corresponding to this solution was not shown in Ref. [255].

[4] The optimized design corresponding to this solution was not shown in Ref. [256].

[5] The optimized design corresponding to this solution was not shown in Ref. [261].

• Average number of analyses for the successful optimization runs of Ref. [262].

It can be seen from Table 6.3.a-b that HALSGWJA was the best optimizer overall in test problems 1 (speed reducer design), 2 (industrial refrigeration system), 3 (tension-compression spring design case 1), 4 (pressure vessel design), 5 (welded beam design), 6 (three-bar truss design), 7 (multiple disk clutch brake design), 8 (planetary gear train design), 9 (step-cone pulley design) and 11 (hydrostatic thrust bearing design) converging to the target optima of these problems and requiring less analyses than its 18 competitors to complete the optimization process. The target optimum costs indicated in Table 6.2 for the CEC2020 real-world mechanical engineering problems were either reached or slightly improved by HALSGWJA also in test cases 12 (four-stage gear box design), 14 (rolling element bearing design), 15 (gas transmission compressor design) and 19 (topology optimization), again requiring less analyses than its 18 competitors to complete the optimizations.

It should be noted that the pressure vessel design problem (Problem 4) presents two variants, one with continuous and the other with mixed optimization variables. The corresponding optimization results are reported in Table 6.3.a-b, where "italics" values denote HALSGWJA's results for the continuous-variable case. The vessel (see chapter 4.3.2.4) consists of a cylindrical segment closed at both ends by spherical caps, and is characterized by four design variables: thickness of the cylindrical segment (X_1), thickness of the spherical caps (X_2), radius of curvature of the spherical caps (X_3), and length of the cylindrical segment (X_4). The optimum cost of 5885.333 listed in Table 6.2 refers to the continuous-variable variant, while in the mixed-variable version, $X_1$and X_2 are integer multiples of the available rolled steel plate thicknesses (see Ref. [170]), yielding a corresponding optimum cost of 6059.714. The mixed-variable version was solved by all algorithms compared in Table 6.3.a-b, except for CGKOA [255], MAHA [261], and EJAYA [252], [177], which addressed the continuous-variable case. Regardless of the problem variant, HALSGWJA consistently converged to the target solution, requiring only 3510- 6193 analyses, compared to at least 10000 analyses needed by its competitors.

A further noteworthy case is the rolling element bearing design problem (Problem 14), where the objective is to maximize the load-bearing capacity. The target optimum cost of 14614.136 listed in Table 6.2 corresponds to the design vector $\bar{X}_{opt,1} = \{131.2,18,4,0.6,0.6,0.4,0.6,0.3,0.025,0.6\}$. The third design variable, the number of balls Z, must take an integer value and was rounded during the optimization from 4.476 to 4. Another solution reported in the literature for two nature-inspired algorithms, MENGO [266] and COLMA [267], yielded an optimized cost of 16958.2, with Z rounded from 4.51 to 5, and the optimal design $\bar{X}_{opt,2} = \{131.2,18,5,0.6,0.6,0.4071,0.6437,0.3,0.0632,0.6\}$. Although $\bar{X}_{opt,1}$ and $\bar{X}_{opt,2}$ share the same values for X_1, X_2, X_4,and X_5, their optimized costs differ because the objective function W (defined by Equation 6.19) depends explicitly on the rounded value of

$X_3 = Z$. Both $\bar{X}_{opt,1}$and $\bar{X}_{opt,2}$can thus be considered valid target solutions: the former ($W = 14614.136$) was achieved by SASS [196], [197], [253], DESS [253], SDDS-SABC [197], and MAHA [261], while the latter ($W = 16958.2$) was obtained by HALSGWJA and the remaining algorithms. The computational efficiency of HALSGWJA was confirmed by its ability to converge to the 16958.2 cost in all independent runs. Additional solutions for Problem 14 have also been reported in the literature. For instance, Ref. [265] reports, for the Energy Valley Optimizer (EVO), an optimized design $\bar{X}_{opt,3} = 125.7191, 21.4256, 11, 0.515, 0.515, 0.4632,$

0.6999,0.3,0.06343,0.6042} yielding a cost of 81859.8. Similarly, the Teaching-Learning-Based Optimization (TLBO) method [6] achieved a comparable result ($W = 81859.74$) with $\bar{X}_{opt,4} =$ {125.7191,21.42559,11,0.515,0.515,0.424266,0.633948,0.3,0.068858,0.799498}. Ref. [273] reported for the Artificial Lemming Algorithm an optimized cost of 85549.2 with $\bar{X}_{opt,5} = \{125.7191,21.4256,11,0.515,0.515,0.4014,0.6138,0.3,0.0307,0.6001\}$, and the same value was later reported in Ref. [177] for EJAYA, with $\bar{X}_{opt,6} = \{125.7191, 21.4256, 11,0.515, 0.515, 0.4009, 0.6228,$

0.3,0.1,0.6023}. However, since $\bar{X}_{opt,3}$, $\bar{X}_{opt,4}$, $\bar{X}_{opt,5}$, and $\bar{X}_{opt,6}$share identical values for X_1, X_2, X_3, X_4, and X_5, their corresponding optimized costs should coincide. It can thus be proven that the correct optimized cost is 81859.8, not 85549.2. Interestingly, HALSGWJA consistently converged to $\bar{X}_{opt,3}$ and $\bar{X}_{opt,4}$, achieving the correct optimized cost of 81 859.8 with zero standard deviation, simply by restarting the search from initial populations obtained by perturbing $\bar{X}_{opt,2}$ under the side constraint $X_3 = Z > 5$. This cascade optimization process, involving an initial convergence toward $\bar{X}_{opt,2}$ followed by perturbations toward $\bar{X}_{opt,3}$and $\bar{X}_{opt,4}$, was always completed within 8000 analyses, significantly fewer than the 10000-20000 analyses required by the algorithms in Refs. [6],[177], [265], [273].

In the robot gripper design problem (Problem 10), HALSGWJA achieved a best optimized cost of 2.5437852, slightly higher than the target value of 2.528792 in Table 6.2 but in perfect agreement with the best solution (2.5438) reported for the CEC2020 winner algorithms. Similarly, for Problem 16 (tension/compression spring design case 2), HALSGWJA achieved a minimum steel wire volume of 2.658722, differing slightly from the target value of 2.61388 in Table 6.2 but fully consistent with published results ranging from 2.658558 to 2.684243 (see Table 6.3.a-b). Remarkably, HALSGWJA was also the fastest optimizer in both test cases.

In test problems 13 (ten-bar truss design) and 18 (Himmelblau's function), HALSGWJA achieved best optimized costs of 524.5885 and -30664.873, respectively, compared to the corresponding target values of 524.4508 and -30665.539, corresponding to only 0.0263% and 0.00217% cost penalties. Once again, HALSGWJA ranked first in computational speed in both problems.

In summary, the proposed algorithm practically never missed the target optimal design in any of the test problems. Comparable performance was achieved only by EnMODE [250],[220], [261] and COLSHADE [251], [220], [261], remarkably, two of the CEC2020 real-world mechanical engineering competition winners. Conversely, En(L)SHADE [193], IMODE-KG

[252], DESS [253], SASS [196], [197], [261], and MSFPSO [256] failed to reach the target optimum only in problem 12 (four-stage gearbox weight minimization), with best optimized weights ranging between 36.2504 and 98.69, compared to the 35.359 target. MPSGO [260], MAHA [261], and EJAYA [177],[252] also missed the target optimum in only one test case: MPSGO in problem 11 (optimized cost 1643.1 vs. 1616.120 target), and MAHA and EJAYA in problem 10, obtaining 2.638958 and 2.601873, respectively, compared to the 2.5438 reference cost reported in the CEC2020 benchmark. SDDS-SABC [197] missed the target optimum in two test problems: problem 4 (5979.985 vs. 5885.333 target) and problem 12 (43.14295 vs. 35.359 target), with a maximum cost penalty of 22.02%. KOA/CGKOA [252], [255] failed to match the target optimum in three problems: 11 (2403.717 vs. 1616.120), 12 (36.6012 vs. 35.359), and 13 (529.4834 vs. 524.4508), corresponding to a maximum penalty of 48.73%. IDMO [258] also missed the target in three test problems: 2 (0.033960 vs. 0.032213), 11 (1640.9 vs. 1616.120), and 12 (37.461 vs. 35.359), with a maximum penalty of 5.945%. IYDSE [254], [252] failed in four test problems, 2 (3.29×10^{-2} vs. 0.032213), 10 (2.70 vs. 2.5438), 11 (2483.159 vs. 1616.120), and 13 (535.9481 vs. 524.4508), with a maximum penalty of 53.65%. Similarly, ESAO [257] missed the target in four test problems: 8 (0.53571 vs. 0.52577), 10 (2.5929 vs. 2.5438), 11 (1918.4 vs. 1616.120), and 12 (41.697 vs. 35.359), with a maximum penalty of 18.7%. CSELGWO [259] missed the target average solution in test problems 2 and 8, with a maximum average cost penalty of 1.077%, while LMWOAGWO [220] obtained worse average costs than the targets in problems 2, 4, 9, 11, 13, and 16, with a maximum penalty of 24.5%. (Note: best and worst optimization data were not reported in [220], [259].)

It should also be noted that SDDS-SABC was reported in [197] to produce, in several test problems, significantly better best-run results than the target values listed in Table 6.2. The most relevant improvements were claimed for: (i) problem 9 (step-cone pulley design): 12.30213 vs. 16.07 (-25.139%), (ii) problem 5 (welded beam design): 1.452753 vs. 1.670218 (-13.02%), (iii) problem 3 (tension/compression spring design, case 1): 0.0112423 vs. 0.012665 (-11.233%), (iv) problem 7 (multiple disk clutch brake design): 0.21881 vs. 0.23524 (-6.954%), (v) problem 15 (gas transmission compressor design): 2851,884.4 vs. 2964,895.4 (-3.812%), and (vi) problem 1 (speed reducer design): 2963.911 vs. 2994.425 (-1.019%). However, the corresponding optimized designs were not provided, and their feasibility was not verified in [197]. Moreover, the worst solutions of SDDS-SABC for problems 3, 5, 7, 9, and 15 were worse than those obtained by HALSGWJA, and its average optimized designs in problems 7 and 15 were also inferior, indicating lower robustness compared to HALSGWJA. Furthermore, IYDSE [252], [254] was reported to yield an apparently better solution than the target in the four-stage gearbox problem (problem 12), with an optimized cost of 14.9 vs. 35.359. However, no information was provided regarding constraint satisfaction or feasibility checks. In addition, its largest optimized cost and standard deviation across independent runs were significantly worse than those of HALSGWJA. IYDSE was also reported to obtain a slightly better cost than the target in the rolling element bearing problem (problem 14): 16981.126 vs. 16958.2 (also achieved by HALSGWJA). While the optimized cost of 16981.126 reported in [252] represents a major improvement over the suboptimal 5.60×10^{3} initially presented in [254], no evidence of the feasibility of this solution was provided. Moreover, the average and worst optimized costs of 17045.036 and 17203.713 reported in [252] are inconsistent, as problem 14 is a maximization problem.

CGKOA [255] was reported to achieve a much lower structural weight than the target solution in the step-cone pulley design problem 9, only 8.5633 compared to 16.0699. However, the corresponding optimal design of CGKOA was not provided in [255]. A deeper investigation into the optimized solutions of problem 9 reveals that the best feasible design ever reported in the literature has a minimum weight of 8.18 and was obtained in 2024 by the Secretary Bird Optimization Algorithm (SBOA), as developed in Ref. [271]. The design vector for this solution was d_1 = 17 mm, d_2 = 28.3 mm, d_3 = 50.8 mm, d_4 = 84.5 mm, and w = 90 mm, where $d_{1–4}$ and w respectively denote the pulley diameters and the pulley width. The best design obtained by HALSGWJA was d_1 = 38.477 mm, d_2 = 53.101 mm, d_3 = 95.346 mm, d_4 = 158.332 mm, and w = 47.913 mm, weighing 16.036. This is much closer to the classical solution (d_1 = 38.4 mm, d_2 = 52.9 mm, d_3 = 70.5 mm, d_4 = 84.5 mm, w = 90 mm) yielding the target pulley structural weight of 16.097.

In fact, since the cost function of problem 9 is expressed as $W = 7200 \cdot w \cdot \{d_1^2 \cdot [1 + (750/350)^2] +$

$+d_2^2 \cdot [1 + (450/350)^2] + d_3^2 \cdot [1 + (250/350)^2] + d_4^2 \cdot [1 + (150/350)^2]\}$, that is $W = 7200 \cdot w \cdot \{5.5918 \cdot d_1^2 + 2.6531 \cdot d_2^2 + 1.5102 \cdot d_3^2 + 1.1837 \cdot d_4^2\}$, the cost function is mostly sensitive to variations in d_1 and d_2. Interestingly, by removing the side constraints $d_{1–4}$ > 35 mm originally included in the HALSGWJA optimizations, the present algorithm was able to converge to the 8.18 minimum-weight design reported in Ref. [271]. This was achieved within approximately 10000 analyses, significantly faster than the computational budgets of 30000 to 50000 analyses indicated in [271], [255], [197] for the structural weights of 8.18, 8.5633, and 12.30213, respectively obtained by SBOA, CGKOA, and SDDS-SABC. The best optimized costs found by MSFPSO [256] in test problems 11 (hydrostatic thrust bearing design) and 15 (gas compressor transmission design) appear significantly lower than the corresponding target values reported in Table 6.2, namely, 137.08396 vs. 1616.120 and 1227, 929.15 vs. 2964,895.4, respectively. MSFPSO also obtained a slightly better optimized cost than the target value (6032.55 vs. 6059.714) in the pressure vessel design problem 4 variant with mixed variables. However, the corresponding optimized designs and constraint margins for problems 11, 15, and 4 were not indicated in [256], making it impossible to verify the feasibility of these designs. Furthermore, in problem 4, MSFPSO obtained a higher cost in its worst optimization run and a larger standard deviation on the optimized cost than the present algorithm.

MAHA [261] slightly improved the target optimum cost of the four-stage gearbox problem, achieving a minimum weight of 34.63987 compared to the 35.359 target; however, the corresponding optimal design and constraint margin information were not provided in Ref. [261]. Moreover, the average and worst optimized weights found by MAHA were approximately 10.7% and 36.9% higher than those of HALSGWJA, respectively, indicating a significantly lower robustness of MAHA compared to the present algorithm.

The relevance of the present results is underscored by the fact that the 18 competitors selected for comparison with HALSGWJA were highly efficient metaheuristic algorithms that either: (i) won the CEC2020 competition on real-world mechanical engineering problems (i.e., EnMODE, COLSHADE, SASS); (ii) implemented improved or hybrid variants of HALSGWJA's component algorithms GWO and JAYA (i.e., LMWOAGWO, CSELGWO,

EJAYA), which proved competitive with the CEC2020 winners; (iii) were not based on GWO/JAYA but nonetheless rivalled the CEC2020 winners (i.e., En(L)SHADE, IMODE-KG, DESS, SDDS-ABC, MPSGO, MAHA); (iv) outperformed highly cited algorithms such as PSO, GWO, and WOA (i.e., IYDSE, KOA, GKOA, MSFPSO, ESAO, IDMO).

Each selected competitor of HALSGWJA was, in turn, proven to outperform between 4 and 21 other metaheuristic algorithms.

For brevity, detailed comparisons between HALSGWJA and other GWO/JAYA variants or hybrid GWO-JAYA formulations that solved only a subset of the CEC2020 problems are not included in this study. However, it is worth noting that HALSGWJA significantly outperformed the Simple Hybrid Grey Wolf JAYA (SHGWJA) algorithm originally developed in 2024 by hybridizing GWO and JAYA [170]. SHGWJA served as the foundational base for HALSGWJA and was proven in Ref. [170] to be highly competitive (and in some cases superior) to state-of-the-art metaheuristic algorithms, including CEC2020 winners and advanced variants of GWO and JAYA, the component algorithms used in both studies to develop hybrid metaheuristic formulations.

SHGWJA was tested on seven engineering problems, including four representative CEC2020 test problems: (i) industrial refrigeration system design (problem 2), (ii) tension–compression spring design (case 1, problem 3), (iii) pressure vessel design (problem 4, both continuous and mixed-variable variants), and (iv) welded beam design (problem 5).

The following points summarize the main findings:

- Problem 2: Both HALSGWJA and SHGWJA converged to the target optimum cost of 0.032213. However, the present algorithm completed the 20 independent optimization runs in only 4695- 6258 analyses (average 5478 ± 1003), whereas SHGWJA required an average of 8517 ± 2370 analyses (fastest run: 5165 analyses). Furthermore, HALSGWJA achieved better average, worst, and standard deviation of optimized costs: 0.0322132 vs. 0.032215, 0.032216 vs. 0.032219, and 6.7082×10^{-7} vs. 3.671×10^{-6}, respectively.
- Problem 3: HALSGWJA converged to the target optimum cost of 0.012665, while SHGWJA reached a slightly higher cost of 0.0126665. Again, HALSGWJA recorded better average, worst, and standard deviation values: 0.012668 vs. 0.012682, 0.012670 vs. 0.012692, and 2.1152×10^{-6} vs. 1.11×10^{-5}, respectively. Moreover, HALSGWJA required fewer analyses (fastest run: 2103 vs. 2247; average: 3316 ± 1137 vs. 6347 ± 2246).
- Problem 4: In both continuous and mixed-variable variants, HALSGWJA and SHGWJA converged to the target optimum costs (5885.331 and 6059.714, respectively), both with zero standard deviation. Nevertheless, HALSGWJA required fewer analyses: (i) Continuous variant: fastest 3510 vs. 6732; average 4672 ± 1203 vs. 9773 ± 2561. (ii) Mixed variant: fastest 3852 vs. 7604; average 5104 ± 1374 vs. 9712 ± 2730. Notably, even the slowest HALSGWJA runs required fewer analyses than the fastest SHGWJA runs (5713 vs. 6732 for continuous; 6193 vs. 7604 for mixed).
- Problem 5: Both algorithms converged to the target optimum cost of 1.670218 with zero standard deviation, yet HALSGWJA was more efficient (fastest run: 2975 vs.

3691; average: 3224 ± 285 vs. 4438 ± 424). Remarkably, the slowest HALSGWJA run required fewer analyses than the fastest SHGWJA run (3691 vs. 3791).

Although the data presented in Table 6.3.a-b are highly composite, resulting from the diverse parameter settings adopted for the optimization runs of the 18 metaheuristic methods compared with HALSGWJA, the proposed algorithm is clearly superior to its competitors in terms of its ability to converge to the optimal solutions of the CEC2020 real-world mechanical engineering problems. This superiority becomes even more evident when considering the remarkably small number of analyses required by HALSGWJA during the optimization process.

In fact, Table 6.3.a-b also lists the number of analyses required by the fastest optimization run of HALSGWJA that converged to the best optimized cost for each test problem. The proposed algorithm consistently completed the optimization process within only 591 analyses (for the four-stage gearbox design problem 12) to 7400 analyses (for the hydrostatic thrust bearing design problem 11), whereas even the fastest competitors, such as IMODE-KG [252], IYDSE [254], [252], KOA [264], [252], and EJAYA [177], [252], required at least 10000 analyses.

Interestingly, IMODE-KG converged in 10000 analyses to designs inferior to those obtained by HALSGWJA in: (i) problem 3 (tension/compression spring design, case 1), with an optimized cost of 0.0126742 vs. the target optimum of 0.012665 reached by HALSGWJA in only 2013 analyses; (ii) problem 8 (planetary gear-train design), with an optimized cost of 0.526581 vs. the 0.523250 optimum cost reached by HALSGWJA in only 2095 analyses; (iii) problem 11 (hydrostatic thrust bearing design), with an optimized cost of 1783.954 vs. the 1616.120 optimum reached by HALSGWJA within only 7400 analyses.

Similarly, IYDSE also converged within 10000 analyses to worse designs than HALSGWJA in: (i) problem 4 (pressure vessel design, mixed-variable variant), with an optimized cost of 6103.368 vs. the 6059.714 target optimum achieved by HALSGWJA in only 3852 analyses; (ii) problem 11, with an optimized cost of 2483.159 vs. the 1616.120 optimum cost achieved by HALSGWJA in only 7400 analyses; (iii) problem 13, with an optimized cost of 535.5885 vs. the 524.5888-524.6111 optimized costs obtained by HALSGWJA within 1911 to 4801 analyses; (iv) problem 18, with an optimized cost of -−30660.325 vs. the −30664.873 optimum cost achieved by HALSGWJA within only 4694 analyses.

KOA also converged within 10000 analyses to worse designs than HALSGWJA in: (i) problem 1 (speed reducer design), with an optimized cost of 2995.351 vs. the 2994.425 target optimum achieved by HALSGWJA within only 2326 analyses; (ii) problem 11, with an optimized cost of 2403.717 vs. the 1616.120 optimum cost reached by HALSGWJA within only 7400 analyses; (iii) problem 13, with an optimized cost of 529.4834 vs. the 524.5888-524.6111 optimized costs obtained by HALSGWJA within 1911 to 4801 analyses; (iv) problem 15, with an optimized cost of 2965,375.3 vs. the 2964,893.937 optimum cost achieved by HALSGWJA within only 4377 analyses; (v) problem 18, with an optimized cost of −30663.305 vs. the −30664.873 optimum cost reached by HALSGWJA within only 4694 analyses.

Likewise, EJAYA (a very advanced JAYA variant) converged within 10000 analyses to worse designs than HALSGWJA in: (i) problem 8, with an optimized cost of 0.527347 vs. the 0.523250 optimum reached by HALSGWJA within only 2095 analyses; (ii) problem 10, with an optimized cost of 2.601873 vs. the 2.5437852 optimum cost achieved by HALSGWJA

within only 2095 analyses; (iii) problem 13, with an optimized cost of 525.0392 vs. the 524.5888-524.6111 optimized costs obtained by HALSGWJA within 1911 to 4801 analyses.

Another strong indication of HALSGWJA's computational efficiency is that the optimization data listed in Table 6.3 for the CEC2020 top-performing algorithms, namely, EnMODE [250], [220], [261], COLSHADE [251], [220], 261], and SASS [196], [197], [261], refer to the classical computational budgets of: (i) 100000 analyses for test problems 1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, and 18; or (ii) 200000 analyses for test problems 2, 12, and 19.

In summary, the CEC2020 winners required at least one order of magnitude more analyses than the present algorithm to complete their optimizations.

It should be emphasized that Table 6.3.a-b reports the computational costs of the best and worst optimization runs, together with the average and standard deviation of the number of analyses performed by HALSGWJA. Comparable data were unavailable for competing algorithms, except for the average number of analyses required by the CEC2020 winners EnMODE, COLSHADE, and SASS, as reported in Ref. [262]. The data in [262] were originally provided to support the comparison of the Piranha Predation Optimization Algorithm (PPOA) proposed in that study.

According to [262], PPOA converged to feasible designs in only 11 out of 19 test problems despite a theoretical computational budget of 1000000 analyses, requiring on average at least 26900 analyses (except for problems 1, 7, 17, and 19). PPOA was faster than HALSGWJA only in problem 17. EnMODE was on average faster than HALSGWJA in problems 1 (2277 vs. 2538 analyses) and 17 (215 vs. 1170 analyses). COLSHADE outperformed HALSGWJA in problems 17 and 18 (220 vs. 1170 and 2267 vs. 4694 analyses, respectively), whereas SASS showed a lower computational cost only in problem 17 (360 vs. 1170 analyses). However, since Ref. [262] did not specify the best and worst computational runs or the standard deviations for EnMODE, COLSHADE, and SASS, a direct comparison of their computational costs with those of HALSGWJA is not possible.

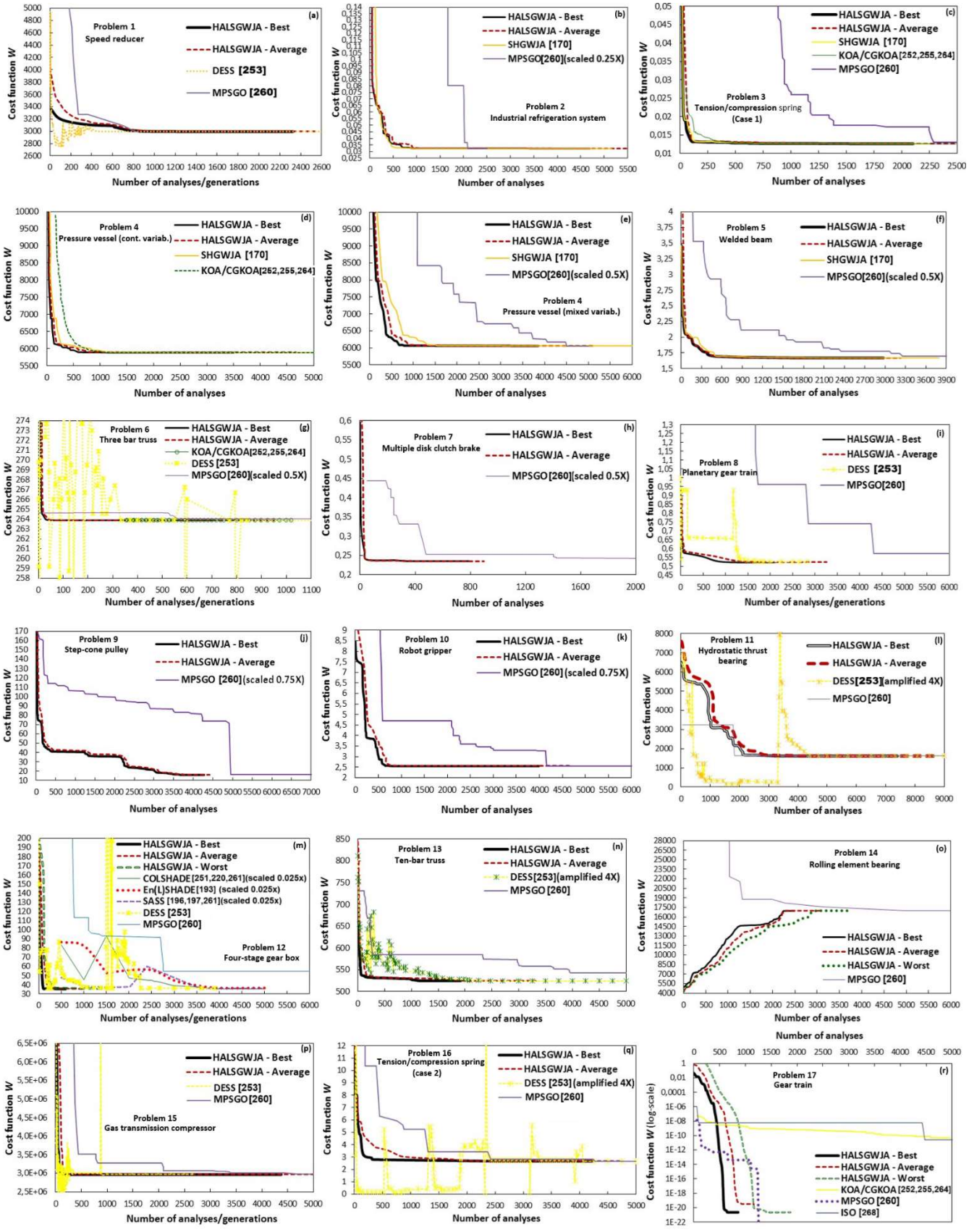
In summary, Ref. [262] highlights the following points: (i) the CEC2020 winners converged to target solutions in only 14-15 test problems, which contrasts with the results reported for their original implementations; (ii) performance gaps between HALSGWJA and each CEC2020 winner occurred in only one or two of the 14-15 successfully solved problems; (iii) the largest performance gap was 11% (problem 1); and (iv) the average number of analyses reported for the CEC2020 winners was of the same order of magnitude as that observed for HALSGWJA.

HALSGWJA also demonstrated high robustness in both optimized cost values and computational effort. Specifically, it achieved zero standard deviation on the optimized cost for test problems 1, 4, 5, 6, 7, 11, 14, 15, and 19. For all other cases, the ratio between the standard deviation and the average optimized cost did not exceed 0.476%. Notably, even in test problem 12, where this maximum ratio occurred, HALSGWJA achieved the lowest standard deviation among all compared algorithms, confirming its consistency despite the problem's strong nonlinearity. In this case, HALSGWJA yielded a standard deviation of only 0.1687 (optimized costs between 35.29269 and 35.65141, using 531-1591 analyses), compared with 0.49725 for SDDS-ABC [197] (43.14925-45.36397 within 20000 analyses) and 0.56923 for MPSGO [260] (35.359-37.258 within 200000 analyses). The CEC2020 winners EnMODE

[250], [220], [261], COLSHADE [251], [220], [261], and SASS [196], [197], [261] achieved standard deviations of 0.59911 (1.677% of an average optimized cost of 35.728), 1.3677 (3.736% of 36.611), and 2.11484 (5.491% of 38.5141), respectively. In contrast, MSFPSO [256], ESAO [257], and IDMO [258] exhibited much larger deviations in problem 12, namely 126054.2, 9.372×10^{14} (within 15000 analyses), and 6.3419×10^{15} (within 50000 analyses). These results confirm the difficulty faced by most metaheuristics in converging to the target solution of this highly nonlinear problem, irrespective of the initial population, while HALSGWJA achieved superior stability with a significantly lower computational budget.

Regarding robustness in computational cost, the ratio between the standard deviation and the mean number of analyses required by HALSGWJA never exceeded 38.97%, with an average of 18.07% and a median of 16.7%. Figure 6.3 illustrates the convergence histories obtained for all CEC2020 problems, clearly confirming HALSGWJA's stable convergence behaviour. On average, the convergence curves of its best and mean optimization run overlapped after approximately 49.2% of the best run's optimization history. This overlap occurred after 75% of the optimization history only in four problems, planetary gear train design (8, 76.4%), step-cone pulley (9, 81.8%), ten-bar truss (13, 78.5%), and gear train (17, 87.2%, reaching cost values below the 10^{-15} double-precision limit), but appeared before 25% of the history in four others: industrial refrigeration system (2, 23.4%), three-bar truss (6, 8.51%), four-stage gearbox (12, 16.9%), and gas transmission compressor (15, 11.4%). Figure 6.3 also highlights HALSGWJA's superior convergence relative to its competitors. Convergence curves for the entire CEC2020 real-world mechanical engineering problem set were available only for DESS [253] and MPSGO [260]. However, several curves for DESS in Ref. [253] were either out of scale or of insufficient resolution and were therefore omitted for clarity. Since DESS and MPSGO reported optimization histories in terms of generations (iterations) rather than analyses, the data from [253], [260] were converted accordingly.

DESS, a differential evolution, based method, reduces its population size linearly from an initial value of $6\times NDV^2$ while distributing search agents across four sub-populations. Hence, each generation performs at least four new analyses, assuming one search agent per sub-population. In test problems 1, 6, 8, 12, and 15 (see Fig. 6.3.a, g, i, m, p), HALSGWJA reached the target optima in fewer analyses than the number of generations required by DESS. Thus, even in the ideal case where DESS performed a single analysis per generation, it would still require more evaluations than HALSGWJA. For problems 11, 13, 16, and 18, the DESS convergence curves in Fig. 6.3.l, n, q, s were amplified by a factor of four ("amplified 4×") to ensure a more realistic comparison. Consequently, in Figure 3 the horizontal axis is labelled "Number of analyses/generations" when actual DESS generations are considered and "Number of analyses" when amplified data are used.


(a)
Problem 1
Speed reducer
HALSGWJA - Best
HALSGWJA - Average
DESS [253]
MPSGO [260]
Cost function W
Number of analyses/generations
(b)
HALSGWJA - Best
HALSGWJA - Average
SHGWJA [170]
MPSGO[260](scaled 0.25X)
Problem 2
Industrial refrigeration system
Number of analyses
(c)
HALSGWJA - Best
HALSGWJA - Average
SHGWJA [170]
KOA/CGKOA[252,255,264]
MPSGO[260]
Problem 3
Tension/compression spring
(Case 1)
Number of analyses
(d)
Problem 4
Pressure vessel (cont. variab.)
HALSGWJA - Best
HALSGWJA - Average
SHGWJA [170]
KOA/CGKOA[252,255,264]
Number of analyses
(e)
HALSGWJA - Best
HALSGWJA - Average
SHGWJA [170]
MPSGO[260](scaled 0.5X)
Problem 4
Pressure vessel (mixed variab.)
Number of analyses
(f)
Problem 5
Welded beam
HALSGWJA - Best
HALSGWJA - Average
SHGWJA [170]
MPSGO[260](scaled 0.5X)
Number of analyses
(g)
Problem 6
Three bar truss
HALSGWJA - Best
HALSGWJA - Average
KOA/CGKOA[252,255,264]
DESS [253]
MPSGO[260](scaled 0.5X)
Number of analyses/generations
(h)
Problem 7
Multiple disk clutch brake
HALSGWJA - Best
HALSGWJA - Average
MPSGO[260](scaled 0.5X)
Number of analyses
(i)
Problem 8
Planetary gear train
HALSGWJA - Best
HALSGWJA - Average
DESS [253]
MPSGO[260]
Number of analyses/generations
(j)
Problem 9
Step-cone pulley
HALSGWJA - Best
HALSGWJA - Average
MPSGO [260](scaled 0.75X)
Number of analyses
(k)
Problem 10
Robot gripper
HALSGWJA - Best
HALSGWJA - Average
MPSGO [260](scaled 0.75X)
Number of analyses
(l)
Problem 11
Hydrostatic thrust bearing
HALSGWJA - Best
HALSGWJA - Average
DESS[253](amplified 4X)
MPSGO[260]
Number of analyses
(m)
HALSGWJA - Best
HALSGWJA - Average
HALSGWJA - Worst
COLSHADE[251,220,261](scaled 0.025x)
En(L)SHADE[193] (scaled 0.025x)
SASS [196,197,261](scaled 0.025x)
DESS [253]
MPSGO[260]
Problem 12
Four-stage gear box
Number of analyses/generations
(n)
Problem 13
Ten-bar truss
HALSGWJA - Best
HALSGWJA - Average
DESS[253](amplified 4X)
MPSGO[260]
Number of analyses
(o)
Problem 14
Rolling element bearing
HALSGWJA - Best
HALSGWJA - Average
HALSGWJA - Worst
MPSGO [260]
Number of analyses
(p)
HALSGWJA - Best
HALSGWJA - Average
DESS [253]
MPSGO[260]
Problem 15
Gas transmission compressor
Number of analyses/generations
(q)
Problem 16
Tension/compression spring
(case 2)
HALSGWJA - Best
HALSGWJA - Average
DESS [253](amplified 4X)
MPSGO[260]
Number of analyses
(r)
Problem 17
Gear train
Cost function W (log-scale)
HALSGWJA - Best
HALSGWJA - Average
HALSGWJA - Worst
KOA/CGKOA[252,255,264]
MPSGO [260]
ISO [268]

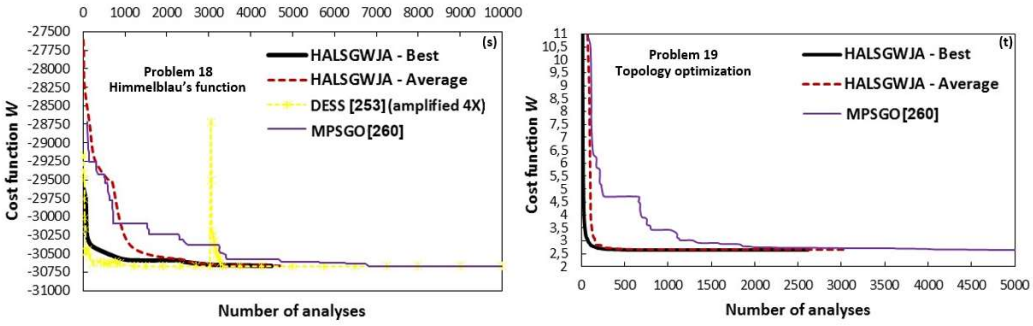


Figure 6.3. Convergence curves obtained for HALSGWJA and its competing algorithms in the CEC2020 real-world mechanical engineering benchmark problems. In the vertical axes, the comma (",") is used as the decimal separator.

The amplification factor of 4 adopted for DESS generations was considerably smaller than the initial population of $6 \times NDV^2$search agents used in its first generation: 96 (6×4^2), 600 (6×10^2), 54 (6×3^2), and 150 (6×5^2) for test problems 11, 13, 16, and 18, respectively. As shown in Figure 6.3.l, n, q, s, DESS converged to the optimal solutions of these problems within 1100, 555, 1000, and 800 generations. By linearly decreasing the population from 96, 600, 54, and 150, the computational cost would correspond to approximately 63850, 75270, 59230, and 60050 analyses, respectively, significantly higher than the 7400, 1911, 4223, and 4494 analyses required by HALSGWJA.

Unlike HALSGWJA, which monotonically converged to the optimum in all test problems, DESS exhibited pronounced oscillations and often generated infeasible intermediate designs. This behaviour stems from its spherical search operator, which explores a spherical region around each solution and uses gradient repair to guide infeasible candidates toward the feasible space. In contrast, HALSGWJA employs approximate gradient information associated with the best candidates in the population. Its JAYA-based perturbation and mirror strategies promote the generation of high-quality trial designs and steer the search away from perturbations that increase the cost function.

The number of analyses for the MPSGO algorithm [260] was estimated by multiplying the population size $N_{POP} = 50$ in Ref. [260] by the number of optimization iterations, as both component algorithms, the Marine Predators Algorithm (MPA) and Social Group Optimization (SGO), evaluate all agents per iteration. MPSGO was reported in [260] to converge to the optimal solutions of the CEC2020 real-world mechanical engineering problems within 100-200 iterations, corresponding to 5000-10000 analyses in Figure 6.3. For clarity, the MPSGO convergence curves were scaled by (i) 0.25 for test problem 2 (Figure 6.3.b); (ii) 0.5 for test problems 4 (mixed-variable variant, Figure 6.3.e), 5 (Figure 6.3.f), 6 (Figure 6.3.g), 7 (Figure 6.3.h), and 9 (Figure 6.3.j); and (iii) 0.75 for test problem 10 (Figure 6.3.k). These were labelled as "scaled 0.25X," "scaled 0.5X," and "scaled 0.75X" in Figure 6.3.

Even after scaling, MPSGO remained substantially slower than HALSGWJA in all test cases. Its optimization trajectories exhibited multiple plateaus with marginal improvements throughout the process. This can be attributed to the five sequential phases of MPSGO: (i) the inspiration phase (SGO-dominated imitation of the best design), (ii) Brownian exploration (MPA-dominated), (iii) self-awareness (modified SGO shifting from exploration to exploitation), (iv) combined exploration-exploitation (Lévy and Brownian random walks), and

(v) high exploitation (local search near the global optimum). Each phase occupies one-fifth of the total search, following the traditional "explore first, exploit later" paradigm. In contrast, HALSGWJA dynamically balances exploration and exploitation according to the optimization trend, ensuring smooth and monotonic convergence. Its line search mechanisms further accelerate movement toward regions of the design space likely containing the global optimum.

Although Figure 6.3.l might suggest that MPSGO converged faster than HALSGWJA in the hydrostatic thrust bearing problem (test 11), MPSGO reached only a suboptimal cost of 1643.1, compared to the target optimum of 1616.120 achieved by HALSGWJA. HALSGWJA reduced the initial cost gap (approximately 6520 vs. 3251) within the first 1100 analyses and smoothly approached the optimum by around 3000 analyses. Conversely, MPSGO showed little improvement for the first 1815 analyses, followed by a sharp reduction between 1816-1836 analyses, after which it became trapped in the local minimum at 1643.1. In the rolling element bearing problem (test 14, Figure 6.3.o), MPSGO converged to the target optimum cost of 16958 by steadily reducing the objective, whereas HALSGWJA increased it toward the target, consistent with the problem's maximization nature. Notably, MPSGO never produced intermediate solutions superior to those generated in HALSGWJA's least successful run.

Representative convergence data in Figure 6.3 were also available for (i) COLSHADE [251], [220], [261], En(L)SHADE [193], and SASS [196], [197], [261] in the four-stage gearbox problem 12 (Ref. [193]); (ii) KOA [252], [264] and CGKOA [255] in test problems 3 (tension/compression spring case 1), 4 (pressure vessel design with continuous variables), 6 (three-bar truss design), and 17 (gear train design); (iii) SHGWJA [170] in test problems 2 (industrial refrigeration system), 3 (tension/compression spring case 1), and 4 (pressure vessel design with continuous or mixed variables); and (iv) the Attributes-based Information-learning Search Optimization (ISO) algorithm [268], which emulates human psychological assessment and adaptive learning, in the gear train design problem 17.

Figure 6.3.m presents the convergence comparison for the four-stage gearbox problem (test 12), the CEC2020 benchmark case featuring the largest number of design variables (22) and nonlinear constraints (86). The optimization histories of COLSHADE [251], [220], [261], En(L)SHADE [193], and SASS [196], [197], [261] were scaled by a factor of 0.025 to fit their computational budget of 200000 analyses into the plot of Figure 6.3.m. The CEC2020 winner COLSHADE [251], [220], [261] reached the target optimum cost of 35.359 within 3990 scaled analyses, whereas the best optimization run of HALSGWJA achieved the overall best cost of 35.294 in only 591 analyses. Moreover, even the worst HALSGWJA run required only about 470 analyses to obtain a cost of 36.265, comparable to the final optimized costs achieved by DESS (36.2 within ~2265 generations), En(L)SHADE (36.2 within 3990 scaled analyses), and SASS (36.25 within 3575 scaled analyses). Notably, DESS repeatedly reached the 36.2 cost level after roughly 200, 245, 275, 1495, 1540, 1610, and 1630 generations but displayed strong oscillations and frequent transitions between feasible and infeasible designs. These results highlight the superior convergence stability of HALSGWJA, as COLSHADE, En(L)SHADE, and SASS never produced intermediate designs better than those of the present algorithm.

The best optimization runs of KOA [252], [264] and CGKOA [255] for problems 3, 4 (continuous variables), and 6 (Figure 6.3.c,d,g) were also outperformed by the *average* runs of HALSGWJA when initialized from comparable or better starting points. In the gear train

design problem 17 (Figure 6.3.r), all HALSGWJA runs began from much poorer initial designs than those used for KOA/CGKOA (initial cost values from 0.0464 to 2.046 versus 1.4×10^{-8} for KOA/CGKOA). Similarly, the ISO algorithm [268] started from a very low initial cost (10^{-6}). Nevertheless, HALSGWJA required only 435-900 analyses to close these large initial gaps and generate intermediate solutions superior to those of KOA, CGKOA, and ISO. The various mechanisms in KOA and CGKOA, aimed at balancing exploration and exploitation, enhancing population diversity, removing poor solutions, accelerating high-quality output, and analyzing elite groups from multiple perspectives, were outperformed by HALSGWJA's intrinsic ability to generate new trial designs consistently improving the current best record.

In test problem 17, ISO [268] was originally reported to outperform twelve state-of-the-art optimizers, including HALSGWJA's component algorithm (GWO), key components of CEC2020 winners (differential evolution), widely cited methods (particle swarm optimization, whale optimization algorithm, genetic algorithms, harmony search), and other metaheuristics (bee swarm, equilibrium optimizer, etc.). ISO was chosen for comparison because its structure resembles GWO: three leading individuals hierarchically share information with the rest of the population, analogous to the α, β, and δ wolves guiding the pack toward the prey in GWO. However, the incorporation of gradient information and line search in HALSGWJA proved more effective than ISO at preventing stagnation, smoothing large step transitions in the optimization history, and repairing trial designs that failed to significantly improve, or only marginally improved, the current best solution $\bar{X}_{best}$.

Interestingly, Figure 6.3. b–f shows that SHGWJA [170] was relatively competitive with HALSGWJA in test problems 2, 3, 4, and 5. Its convergence curves were consistently closer to those of HALSGWJA than to those of MPSGO or KOA/CGKOA. This behaviour results from the similar hybridization strategies of SHGWJA and HALSGWJA, both combining GWO and JAYA with elitist acceptance, rejection schemes for trial designs. Such consistency reinforces the efficiency of generating new solutions that maximize the likelihood of improving the current best record.

6.3.2. Comparative Performance Evaluation of HALSGWJA and Competing Algorithms on the CEC2020 Benchmark Problems

HALSGWJA and its 18 competitors were evaluated using five performance indicators: (i) best, (ii) average, and (iii) worst optimized cost values across independent optimization runs, (iv) the corresponding standard deviation of the optimized cost (STD), and (v) the number of analyses required for convergence. Table 6.4 summarizes the performance scores of HALSGWJA across the 19 CEC2020 real-world mechanical engineering problems considered in this study. In each category, a score of "1" indicates the best performance and "19" the worst. The table also reports the deviation of HALSGWJA from its closest competitors in cases where it did not rank within the top quartile (i.e., outside the best 21% of algorithms, corresponding to ranks higher than 4 out of 19).

As shown in Table 6.3, HALSGWJA achieved the highest overall ranking among all algorithms tested. Specifically, it ranked first in:

- 11 out of 19 problems for the best optimized cost,

- 10 out of 19 for the average optimized cost,
- 12 out of 19 for the worst optimized cost,
- 10 out of 19 for the standard deviation of optimized cost, and
- all 19 problems for the number of analyses required.

In addition, HALSGWJA ranked second in 4, 3, and 1 problems for the best, average, and worst optimized cost, respectively, and fourth once for the standard deviation of the optimized cost.

Table 6.4. HALSGWJA ranks obtained in the CEC2020 mechanical engineering problems.

Test problem	Best	Average	Worst	STD	No. analyses
SPEED REDUCER	1	1	1	1	1
INDUSTRIAL REFRIGERATION SYSTEM DESIGN	1	7 (0.0322132 vs. 0.032213)	5 (0.0322160 vs. 0.032213)	6 ($6.7082 \cdot 10^{-7}$ vs. 0)	1
TENSION/ COMPRESSION SPRING DESIGN (CASE 1)	1	7 (0.012668 vs. 0.012665)	5 (0.012670 vs. 0.012665)	8 ($2.1152 \cdot 10^{-6}$ vs. 0)	1
PRESSURE VESSEL DESIGN	1	1	1	1	1
WELDED BEAM DESIGN	1	1	1	1	1
THREE-BAR TRUSS DESIGN	1	1	1	1	1
MULTIPLE DISK CLUTCH BRAKE DESIGN	1	1	1	1	1
PLANETARY GEAR TRAIN DESIGN OPTIMIZATION	1	1	1	4	1
STEP-CONE PULLEY PROBLEM	1	1	1	10 ($4.6437 \cdot 10^{-3}$ vs. 0)	1
ROBOT GRIPPER PROBLEM	1	1	1	6 ($2.8263 \cdot 10^{-7}$ vs. 0)	1
HYDRO-STATIC THRUST BEARING DESIGN	1	1	1	1	1
FOUR-STAGE GEAR BOX PROBLEM	2	2	1	1	1
TEN-BAR TRUSS DESIGN	12 (524.5885 vs. 522.9644)	10 (524.6004 vs. 524.45)	7 (524.6111 vs. 524.45)	7 (0.0113519 vs. 0)	1
ROLLING ELEMENT BEARING	2	2	2	1	1
GAS TRANSMISSION COMPRESSOR DESIGN	2	1	1	1	1
TENSION/ COMPRESSION SPRING DESIGN (CASE 2)	12 (2.658722 vs. 2.658558)	6 (2.659550 vs. 2.658559)	5 (2.660999 vs. 2.658559)	7 ($9.4671 \cdot 10^{-4}$ vs. 0)	1
GEAR TRAIN DESIGN	13 ($2.3078 \cdot 10^{-21}$ vs. 0)	8 ($3.5365 \cdot 10^{-21}$ vs. 0)	6 ($4.9608 \cdot 10^{-21}$ vs. 0)	7 ($1.1169 \cdot 10^{-20}$ vs. 0)	1
HIMMELBLAU'S FUNCTION	14 (-30664.873 vs. -30665.54)	15 (-30664.848 vs. -30665.54)	10 (-30664.834 vs. -30665.53)	13 (0.0195687 vs. 0)	1
TOPOLOGY OPTIMIZATION	2	2	1	1	1

In summary, HALSGWJA demonstrated excellent performance across the CEC2020 real-world mechanical engineering test suite, ranking within the 1st quartile 15, 13, 13, and 11 times for best, average, worst, and standard deviation of optimized cost, respectively, and consistently achieving the fastest convergence in all test problems. Even though HALSGWJA ranked below 4th place in certain categories, its performance remained highly satisfactory. For instance, in terms of best optimized cost, HALSGWJA ranked 12th in problems 13 (10-bar truss) and 16 (tension/compression spring case 2), 13th in problem 17 (gear train), and 14th in problem 18 (Himmelblau's function). However, the relative deviations from the top solutions were negligible: 0.311% (problem 13), 0.00617% (problem 16), 0.00218% (problem 18), while the best cost in problem 17 (2.3078×10^{-21}) remained well below the double precision limit of 10^{-15}.

Similarly, for average optimized cost, HALSGWJA ranked between 6th and 15th in problems 2, 3, 13, 16, and 18, yet the performance gap relative to top algorithms ranged only from 6.209×10^{-4}% to 0.0373%, with the average cost in problem 17 again far below 10^{-15} (3.5635×10^{-21}). For worst optimized cost, HALSGWJA ranked between 5th and 10th in problems 2, 3, 13, 16, 17, and 18, with deviations between 2.270×10^{-3}% and 0.0918%; the worst cost in problem 17 (4.9608×10^{-21}) remained well below double precision limits. Standard deviation rankings ranged from 6th to 13th in problems 2, 3, 9, 10, 13, 16, 17, and 18, yet the ratio of standard deviation to target optimum cost varied only from 1.11×10^{-5}% to 0.0167%, with the value in problem 17 again below 10^{-15}.

Table 6.6 further confirms the robustness of HALSGWJA by reporting the number of first-place rankings achieved across the selected performance indicators for all CEC2020 problems. The "B+A+W" column sums scores for best, average, and worst costs; "B+A+W+STD" includes standard deviation; and "TOTAL" incorporates all indicators, including the number of analyses. HALSGWJA achieved a total score of 62, ranking first overall, ahead of En(L)SHADE [193] with 60 points, which has been reported as highly competitive with CEC2020 winners. It also outperformed CEC2020-winning algorithms such as EnMODE [250], [220], [261], COLSHADE [251], [220], [261], and SASS [196], [197], [261] due to its consistently higher computational speed, enabling convergence to target optima or very close approximations. Even when accounting for lower analysis counts reported in Ref. [262] for EnMODE, COLSHADE, and SASS in problems 17 and 18, HALSGWJA's total score would remain 60, jointly ranking it best alongside En(L)SHADE.

Excluding computational speed, HALSGWJA ranked 7th for the "B+A+W" score based solely on optimized cost values, achieving 33 points compared to 36-46 points for En(L)SHADE, EnMODE, COLSHADE, MPSGO, SASS, and MAHA. When including standard deviation ("B+A+W+STD"), HALSGWJA's rank improved to 4th with 43 points, following En(L)SHADE, COLSHADE, and EnMODE (44-60 points).

It should be noted that, while the performance data for EnMODE and COLSHADE were aggregated from multiple studies, HALSGWJA's evaluation is based solely on data collected in the present study. The higher "B+A+W" and "B+A+W+STD" scores of some competitors largely reflect their substantially larger computational budgets. For instance, in problem 2, En(L)SHADE, EnMODE, COLSHADE, MPSGO, SASS, and MAHA completed their runs

within 200000 analyses, whereas HALSGWJA completed all optimizations with only 4695-6258 analyses.

To further assess HALSGWJA's computational efficiency, additional trials were performed by restarting runs that had previously missed the target solution of 0.0322130, using initial populations near suboptimal solutions. HALSGWJA consistently converged to the target optimum (achieving zero standard deviation) within 9000 analyses for all repaired cases, still well below the analyses required by En(L)SHADE, EnMODE [262], COLSHADE, MPSGO, SASS, and MAHA. Consequently, HALSGWJA should be considered first-ranked also for average/worst costs and standard deviation, increasing its "B+A+W" and "B+A+W+STD" scores by 2 and 3 points, respectively.

Similarly, in problem 3, HALSGWJA completed primary runs in 2103-4097 analyses, and all suboptimal runs converged to the target cost of 0.012665 within 6000 analyses, again below competitors' computational cost. Accounting for these improvements, HALSGWJA's "B+A+W" and "B+A+W+STD" scores would rise to 37 and 49, respectively, corresponding to 5th and 3rd ranks overall, and the total score including computational efficiency would increase to 68 (or 66 considering Ref. [262]), outperforming En(L)SHADE by 8 (or 6) points.

Table 6.6. Numbers of 1st ranks achieved by HALSGWJA and its competitors in the CEC2020 real-world mechanical engineering problems.

Algorithm	Best	Average	Worst	STD	No. analyses	B+A+W	B+A+W+STD	TOTAL
HALSGWJA - Present	11	10	12	10	19	33	43	62
EnMODE [250],[220], [261]	14	13	14	3	0	41	44	44
COLSHADE[251],[220],[261]	13	12	13	11	0	38	49	49
En(L)SHADE [193]	16	14	16	14	0	46	60	60
IMODE-KG [252]	10	8	9	0	2	27	27	29
DESS [253]	14	13	N/A	N/A	0	27	27	27
SASS [196],[197],[261]	12	13	12	3	0	37	40	40
SDDS-SABC [197]	8	6	1	0	0	15	15	15
IYDSE [254],[252]	10	5	5	0	0	20	20	20
KOA [264],[252]-CGKOA [255]	10	8	6	2	0	24	26	26
MSFPSO [256]	12	9	7	4	0	28	32	32
ESAO [257]	11	5	N/A	1	0	16	17	17
IDMO [258]	12	7	N/A	0	0	19	19	19
LMWOAGWO [220]	N/A	8	N/A	0	0	8	8	8
CSELGWO [259]	N/A	9	N/A	0	0	9	9	9
MPSGO [260]	13	13	12	3	0	38	41	41

MAHA [261]	14	12	10	2	0	36	38	38
EJAYA [177],[252]	10	5	3	0	1	18	18	19

6.3.3. Results of the car side impact problem

The optimization results of the car side impact problem obtained with the HALSGWJA algorithm were compared against 25 state-of-the-art metaheuristic optimizers. These competitors were classified into several groups: (a) hybrid GWO-JAYA algorithms incorporating elitist strategies, such as FHGWJA [207]; (b) advanced GWO variants including multi-strategy fusion improved Grey Wolf Optimization (IGWO) [215], chaotic GWO, and adaptive dynamic self-learning grey wolf optimization (ASGWO) [264]; (c) advanced JAYA variants, e.g., EJAYA [177]; (d) particle swarm optimization (PSO) [125] and comprehensive learning PSO (CLPSO) [279]; (e) efficient animal-behaviour-inspired methods, including cuckoo search optimization (CSO) [138], tunicate swarm algorithm (TSA) [280], starling murmuration optimizer (SMO) [140], improved continuous ant colony optimization (LIACOR) [281], krill herd algorithm (KH) [282], best variant of artificial bee colony (ABC) [283], whale optimization algorithm (WOA) [143], horned lizard optimizer algorithm (HLOA) [284], artificial lemming algorithm (ALA) [273], Harris hawks optimizer (HHO) [208], dwarf mongoose optimization (DMO) [285], starfish optimization algorithm (SFOA) [274], and firefly algorithm (FA) [132]; (f) human-behavior-inspired methods such as the attributes-based information-learning search optimization (ISO) [268] and dream optimization algorithm (DOA) [272]; (g) mathematics-inspired algorithms, including the exponential-trigonometric optimization algorithm (ETO) [270] and arithmetic optimization algorithm (AOA) [114]; and (h) high-performance hybrid methods, e.g., LSHADE-SPACMA [185], which combines history-based adaptive differential evolution with linear population size reduction and covariance matrix adaptation evolution strategy.

Table 6.7 reports the optimized designs and corresponding structural weights obtained by HALSGWJA and its competitors. Solutions are clustered according to the literature: IGWO, chaotic GWO, CSO, and TSA [215]; SMO, LIACOR, KH, best ABC, CLPSO, and WOA [140]; DOA and HLOA [272]; ALA, HHO, AOA, DMO, and LSHADE-SPACMA [273]; FA and PSO [132].

HALSGWJA achieved the lowest structural weight of 21.37798 kg, outperforming FHGWJA [207] (21.38340 kg) and IGWO [215] (21.39473 kg). Notably, HALSGWJA and FHGWJA converged to the discrete design values of 0.345 for variables X_8 and X_9, whereas IGWO and chaotic GWO failed to satisfy the prescribed discrete constraints, rendering their solutions infeasible and violating constraints $G_6 - G_9$ by 13.9-23.7%.

Chaotic GWO, CSO, and TSA yielded competitive weights ranging from 21.46164 to 22.70296 kg, though without satisfying the discrete variable constraints. Most other competitors produced weights between 22.84238 and 22.88605 kg (ALA, HHO, EJAYA, SMO, FA, LIACOR, ISO, PSO, AOA, DMO, LSHADE-SPACMA, ASGWO, KH, and best ABC), with several failing to meet the discrete requirements for X_8 and X_9. The remaining six algorithms, CLPSO, WOA, ETO, SFOA, HLOA, and DOA, obtained weights between 23.06244 and 23.96820 kg, with ETO and DOA also missing the target discrete values.

The optimized weights obtained by the 26 algorithms in Table 6.7 ranged by 12%, from the lightest design of 21.37798 kg achieved by HALSGWJA to the heaviest of 23.9682 kg obtained

by DOA. The high nonlinearity of this test problem and its specific formulation explain the large number of local optima identified by the different algorithms. Notably, the cost function $W(\bar{\boldsymbol{X}})$ defined in Equation 6.8 depends only on six optimization variables (X_1, X_2, X_3, X_4, X_5, and X_7), while only X_{10} appears in all constraint functions (Equation 6.9). Consequently, the search space includes many competitive solutions with significantly different variable values.

Despite a modest 12% variation in optimized cost, the first nine design variables in Table 6.7 varied up to 30.4%, while X_{10} and X_{11} varied by 1446.1% and 762.9%, respectively, to satisfy all constraints. The variables included in the cost function, X_1, X_2, X_3, X_4, X_5, and X_7, changed by 8.71%, 19.61%, 6.03%, 15.93%, 13.05%, and 5.88%, respectively, yielding an average dispersion of 11.54%. This variation aligns with the 12% difference in optimized cost, consistent with the linear response surface used to model the cost function.

The success of HALSGWJA in this context can be attributed to its ability to generate high-quality trial designs that consistently improve the current best solution, thereby effectively avoiding local minima.

Table 6.7. Optimization results obtained by HALSGWJA and its competitors in the car side impact problem.

Algorithms	**X1** (mm)	**X2** (mm)	**X3** (mm)	**X4** (mm)	**X5** (mm)	**X6** (mm)	**X7** (mm)	**X8** (mm)	**X9** (mm)	**X10** (mm)	**X11** (mm)	**Structural weight** (kg)
HALSGWJA Present	0.50000	1.21301	0.50000	0.77611	0.50000	1.48975	0.50000	0.345	0.345	−29.1035	0.00011	21.37798
FHGWJA [207]	0.50000	1.21204	0.50000	0.77908	0.50000	1.49004	0.50000	0.345	0.345	−28.9781	0.00010	21.38340
EJAYA [177]	0.50000	1.11631	0.50000	1.30228	0.50000	1.50000	0.50000	0.345	0.32680	−19.5709	0.00838	22.84297
IGWO [215]	0.50000	0.88641	0.50000	1.25781	0.64809	0.91372	0.50000	1.00000	0.52483	1.94790	15.3719	21.39473
Chaotic GWO	0.84252	0.50000	0.50000	1.36186	0.83852	0.86445	0.57679	0.94066	0.24622	4.55062	12.3084	21.46164
CSO [138,215]	0.78544	0.56882	0.50000	1.34514	0.82107	0.86975	0.74586	0.89302	0.39281	0.89510	1.17997	22.00444
TSA [280,215]	0.50315	0.89471	0.50000	1.40533	0.86396	0.84245	0.59580	0.86533	0.06579	0.12364	2.06338	22.70296
ASGWO [264]	0.50004	1.13454	0.50009	1.27905	0.50020	1.49996	0.50005	0.34496	0.33248	−16.3332	−2.14912	22.87188
SMO [140]	0.5000	1.11634	0.5000	1.30224	0.5000	1.50000	0.5000	0.345	0.345	−19.566	0.000001	22.84298
LIACO$_R$ [281,140]	0.5000	1.11593	0.5000	1.30293	0.5000	1.50000	0.5000	0.192	0.345	−19.640	−0.000003	22.84299
KH [282,140]	0.5000	1.14747	0.5000	1.26118	0.5000	1.5000	0.5000	0.345	0.345	−13.998	−0.8984	22.88596
Best ABC [283,140]	0.5000	1.30539	0.5000	1.10312	0.5000	0.50000	0.5000	0.345	0.345	14.213	20.3306	22.88605
CLPSO [279,140]	0.5061	1.17379	0.5013	1.24706	0.5037	1.49560	0.5000	0.345	0.345	−9.5985	3.3627	23.06244
WOA [143,140]	0.5000	1.09276	0.5000	1.41233	0.5000	1.45497	0.5000	0.345	0.192	−24.038	−3.1789	23.12717
ISO [268]	0.5000	1.11642	0.5000	1.30211	0.5000	1.5000	0.5000	0.345	0.345	−19,5525	−0.00152663	22.843
ETO [270]	0.50282	1.2414	0.51604	1.2201	0.60334	1.3878	0.5000	0.74832	0.06747	2.2526	−7.2818	23.2574
DOA [272]	0.5081	1.2021	0.5318	1.3052	0.5719	1.4954	0.5557	0.3030	0.2585	−24.8171	3.4047	23.9682
HLOA [284,272]	0.5000	1.0669	0.8016	1.0704	0.5040	1.4873	0.5000	0.192	0.192	−29.9786	3.2119	23.6956
ALA [273]	0.5000	1.1163	0.5000	1.3020	0.5000	1.5000	0.5000	1.0000	0.2391	−19.5747	−0.0022	22.84238
HHO [208,273]	0.5000	1.1137	0.5000	1.3065	0.5000	1.5000	0.5000	1.0000	0.5997	−20.0430	0.0010	22.84277
AOA [114,273]	0.5000	1.1179	0.5000	1.3005	0.5000	1.5000	0.5000	0.8525	0.1021	−19.2807	−1.3098	22.84756
DMO [285,273]	0.5000	1.1077	0.5000	1.3175	0.5000	1.5000	0.5000	0.6647	0.5054	−21.1276	−0.2616	22.84771

LSHADE-SPACMA [185,273]	0.5000	1.1059	0.5000	1.3213	0.5000	1.5000	0.5000	0.7355	0.9547	−21.4276	0.4018	22.84992
SFOA [274]	0.5000	1.2340	0.5000	1.1870	0.8750	0.8920	0.4000	0.345	0.192	1.5000	0.5720	23.56158
FA [132]	0.50000	1.36000	0.50000	1.20200	0.50000	1.12000	0.50000	0.345	0.192	8,87307	−18.9981	22.84298
PSO [132,125]	0.50000	1.11670	0.50000	1.30208	0.50000	1.50000	0.50000	0.345	0.192	−19.5494	−0.00431	22.84474

Table 6.8 presents a comparison of the best ("B-W"), average ("A-W"), and worst ("W-W") optimized weights, as well as the standard deviation on optimized weight ("SD-W"), obtained by HALSGWJA and its competitors across multiple runs. The table also reports the corresponding number of structural analyses required for the optimization process, indicated as "B-NSA," "A-NSA," "W-NSA," and "SD-NSA." Statistical data on optimized weight were available only for FHGWJA [207], EJAYA [177], ISO [268], DOA and HLOA [272], SFOA [274], and FA and PSO [132]. Data on computational cost were available solely for FHGWJA [207].

Table 6.8. Comparison of statistical data on optimized weight and computational cost for HALSGWJA and its competitors in the car side impact problem.

Algorithms	***B–W*** (kg)	***A–W*** (kg)	***W–W*** (kg)	***STD–W*** (kg)	***B–NSA***	***A–NSA***	***W–NSA***	***STD–NSA***
HALSGWJA Present	21.37798	21.37798	21.37798	0	1240	1291	1387	85
FHGWJA [7]	21.38340	21.3905	21.4145	0.007125	1367	1309	1606/1110	343
EJAYA [70]	22.84297	22.94398	23.26191	0.17098	N/A	27000	N/A	N/A
IGWO [72]	21.39473	N/A	N/A	N/A	N/A	30000	N/A	N/A
Chaotic GWO	21.46164	N/A	N/A	N/A	N/A	30000	N/A	N/A
CSO [95,72]	22.00444	N/A	N/A	N/A	N/A	30000	N/A	N/A
TSA [96,72]	22.70296	N/A	N/A	N/A	N/A	30000	N/A	N/A
ASGWO [73]	22.87188	N/A	N/A	N/A	N/A	15000	N/A	N/A
SMO [76]	22.84298	N/A	N/A	N/A	N/A	30000	N/A	N/A
LIACO$_R$ [97,76]	22.84299	N/A	N/A	N/A	N/A	30000	N/A	N/A
KH [98,76]	22.88596	N/A	N/A	N/A	N/A	30000	N/A	N/A
Best ABC [99,76]	22.88605	N/A	N/A	N/A	N/A	30000	N/A	N/A
CLPSO [94,76]	23.06244	N/A	N/A	N/A	N/A	30000	N/A	N/A
WOA [100,76]	23.12717	N/A	N/A	N/A	N/A	30000	N/A	N/A
ISO [79]	22.843	22.8569	23.2612	0.0764	N/A	15000	N/A	N/A
ETO [81]	23.2574	N/A	N/A	N/A	N/A	30000	N/A	N/A
DOA [83]	23.9682	25.0673	25.9506	0.5111	N/A	2000	N/A	N/A
HLOA [101,83]	23.6956	28.4803	34.5482	2.5047	N/A	2000	N/A	N/A
ALA [84]	22.84238	N/A	N/A	N/A	N/A	15000	N/A	N/A

HHO [102,84]	22.84277	N/A	N/A	N/A	N/A	15000	N/A	N/A
AOA [104,84]	22.84756	N/A	N/A	N/A	N/A	15000	N/A	N/A
DMO [103,84]	22.84771	N/A	N/A	N/A	N/A	15000	N/A	N/A
LSHADE-SPACMA [105,84]	22.84992	N/A	N/A	N/A	N/A	15000	N/A	N/A
SFOA [85]	23.56158	23.56160	23.56203	$8.23{\cdot}10^{-5}$	N/A	50000	N/A	N/A
FA [87]	22.84298	22.89376	24.06623	0.16667	N/A	20000	N/A	N/A
PSO [87,93]	22.84474	22.89429	23.21354	0.15017	N/A	20000	N/A	N/A

Table 6.8 shows that HALSGWJA consistently converged to the global optimum weight of 21.37798 kg across all optimizations runs, completing the search with the lowest number of structural analyses. The algorithm required between 1240 and 1387 analyses, with an average computational cost of only 1291 analyses. FHGWJA [207] was the second fastest optimizer, completing its best run in 1367 analyses, approximately 10% more than HALSGWJA. Interestingly, HALSGWJA was slightly faster on average than FHGWJA (1291 vs. 1309 analyses), while always converging to the global optimum, whereas FHGWJA converged twice to its worst optimized weight of 21.4145 kg, requiring 1110 and 1606 analyses for these runs. This variation explains why FHGWJA's average computational cost was lower than that of its best run. Unlike FHGWJA, which used elitist strategies to refine trial solutions, HALSGWJA employed line search augmented with gradient information. The gradients calculated by HALSGWJA, $\Delta W_p(i,j)/\Delta S(i,j)$, coincide with the actual gradients of the linear cost function when the design vector is feasible, a property particularly advantageous for the linear car side impact problem.

DOA and HLOA ranked third in computational speed, completing their optimizations within 2000 analyses but producing the worst designs overall, with structural weights ranging from 23.6956 to 23.9682 kg [272]. All other algorithms required between 15000 and 50000 analyses.

Besides achieving zero standard deviation in optimized cost, HALSGWJA demonstrated high robustness in computational efficiency. The ratio between the standard deviation and average number of analyses was only 6.58%, consistent with the best values observed in the CEC2020 mechanical engineering test problems. Figure 6.4 confirms the excellent convergence behavior of HALSGWJA by comparing its optimization histories in the best, average, and worst runs with those of FHGWJA [207], ISO [268], and ETO [270]. ISO and ETO required 15,000 (NPOP=30 × 500 iterations) and 30,000 (NPOP=30 × 1000 iterations) analyses, respectively; their histories in Figure 6.4 were scaled by 0.1 and 0.05, with "(scaled 0.1X)" and "(scaled 0.05X)" denoting these scaled optimization histories.

The superiority of HALSGWJA is notable given its conservative starting designs: the initial structural weight of HALSGWJA's designs was about 40 kg, 1.5 times heavier than ISO's (~24 kg). Despite this, HALSGWJA closed the gap within 20-35 analyses, independent of the run type. ETO started at ~24 kg but reduced weight slowly. Moreover, HALSGWJA required fewer than 110 analyses to reach a feasible intermediate design weighing ~21.6 kg (just 1% above the global optimum), 17 analyses to reach below DOA's optimized weight of 23.9682 kg, and

18 analyses to reach below HLOA's weight of 23.6956 kg. The convergence curves for HALSGWJA's best and average runs practically overlapped after only 1000 analyses (~80% of the best run). Remarkably, HALSGWJA's worst run always produced better intermediate designs than the best run of FHGWJA, the closest competitor in convergence speed.

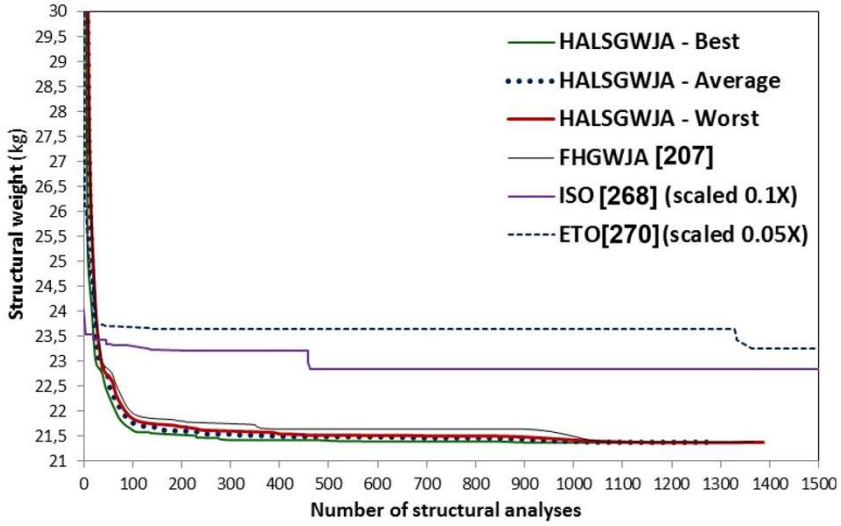


Figure 4. Convergence curves recorded for HLSGWJA and its competitors in the car side impact problem. In the vertical axis, the "," notation indicates the decimal signs.

As observed for the CEC2020 real-world mechanical engineering problems, HALSGWJA outperformed its competitors by consistently converging to the global optimum in a fast and robust manner.

6.4. Conclusions: Towards the Global Optimum from SHGWJA, FHGWJA to HALSGWJA

This chapter introduced HALSGWJA, a hybrid metaheuristic that combines the Grey Wolf Optimizer (GWO) and the JAYA algorithm, enhanced through an approximate line search mechanism. By incorporating approximate gradient information, HALSGWJA generates trial designs that consistently improve upon the current best solution while maintaining a low computational burden. The selection of trial designs relies mainly on cost function evaluations rather than expensive constraint handling, allowing the algorithm to operate efficiently even in highly nonlinear optimization landscapes.

HALSGWJA was rigorously assessed across twenty real-world mechanical engineering problems, including nineteen benchmark cases from the CEC2020 library and a challenging crashworthiness optimization problem. The algorithm demonstrated superior convergence, accuracy, and stability, frequently achieving target optima with negligible or zero standard deviation. Moreover, it required fewer function evaluations than competing metaheuristics, confirming its remarkable efficiency and precision.

As illustrated in Table 6.9, the conceptual and structural evolution from SHGWJA to FHGWJA, and ultimately HALSGWJA, reflects a coherent and deliberate methodological progression.

TABLE 6.9- COMPARATIVE STRUCTURAL AND ALGORITHMIC ANALYSIS ANALYSIS BETWEEN SHGWJA [170], FHGWJA [207], HALSGWJA

Aspect	SHGWJA	FHGWJA	HALSGWJA
CONCEPTUAL AIM	Demonstrate that high optimization performance can be achieved through a minimal and easily implementable hybrid of GWO and JAYA.	Develop a fast, robust hybrid algorithm improving convergence control, balance of exploration/exploitation, and computational efficiency.	Transition from proof-of-concept simplicity to a fully engineered optimization algorithm.
ALGORITHMIC STRUCTURE	Two main stages: 1) GWO-based trial generation. 2) JAYA-based elitist improvement.	Seven structured stages: initialization, preliminary GWO phase, rank-based refinement, elitist evaluation and repair, population reordering, convergence check, termination.	Multi-stage architecture offering fine-grained control of search dynamics.
EXPLORATION–EXPLOITATION BALANCE	Implicitly governed by GWO's α–β–δ hierarchy and JAYA perturbations.	Explicitly managed through rank-proportional updates, adaptive leader repositioning, and mirroring strategies.	Dynamic balance replacing static control, preventing stagnation and local trapping.
RANK-BASED REFINEMENT	Not implemented; perturbations depend solely on GWO and JAYA equations.	Introduced a rank-proportional variation mechanism to adjust step size according to individual quality.	Provides controlled perturbations and preserves population diversity.
ELITIST STRATEGY	Simple elitist rule: reject obviously inferior solutions before constraint evaluation.	Hierarchical elitism: two-level filtering using raw cost thresholds (e.g., 1.1 × W_best), followed by selective JAYA refinement.	Substantial reduction of redundant function and constraint evaluations.
REPAIR MECHANISM	None; inferior solutions are discarded.	Introduces repair operators based on JAYA updating when solutions are promising but suboptimal.	Converts borderline solutions into potential improvements, increasing convergence probability.
MIRRORING STRATEGY	Absent.	Novel mirroring-based descent direction used when all trial solutions fail to improve best record.	Restores convergence direction and avoids stagnation in flat or deceptive search spaces.
CONSTRAINT HANDLING	Basic penalty-based approach.	Deferred constraint evaluation: constraints are checked only after potential improvement is confirmed.	Drastically lowers computational cost in highly constrained problems.
INTERNAL PARAMETERS	None besides population size and iteration limit.	Same parameter-free nature preserved; dynamic convergence makes iteration limit optional.	Maintains simplicity despite added algorithmic complexity.
POPULATION UPDATE	Standard sorting after each iteration; α, β, δ updated based on best costs.	Enhanced reordering with elitist correction if leaders stagnate, using descent-direction perturbation.	Ensures continuous leadership evolution and avoids premature convergence.
COMPUTATIONAL EFFICIENCY	Low computational cost due to minimal structure, but limited exploitation depth.	High efficiency achieved through selective evaluation and repair; faster convergence across tests.	Better trade-off between cost and precision.
APPLICATION SCOPE	Engineering and mathematical optimization with up to 14 variables and 721 constraints.	Extended to complex real-world problems with up to 29 variables and 1200 nonlinear constraints.	Demonstrates scalability and robustness on larger, more nonlinear design spaces.
PERFORMANCE METRICS	Achieved global or near-global optima with negligible deviation; competitive speed.	Consistently reached global optima with zero or near-zero standard deviation; ranked 1st–2nd in speed and robustness.	Demonstrates superior reliability and computational stability.
OVERALL CHARACTERIZATION	Minimalist, pedagogical hybrid emphasizing simplicity and ease of implementation.	Advanced, performance-oriented hybrid integrating adaptive mechanisms for real engineering applications.	Represents a methodological and practical evolution of the previous algorithm.

SHGWJA [170], the initial hybrid formulation, embodies a minimalist design that merges GWO exploration with JAYA exploitation within a parameter-free structure. Its simplicity ensures robustness and ease of implementation, yet its purely stochastic search limits exploitation in complex or multimodal landscapes. FHGWJA [207] builds upon this foundation by introducing a structured multi-stage process incorporating rank-based elitism, repair operators, and deferred constraint evaluation. These enhancements explicitly manage the exploration–exploitation balance and significantly reduce redundant function evaluations, resulting in faster and more stable convergence. Finally, HALSGWJA further extends this framework by integrating gradient-based refinement and adaptive line search, creating a semi-deterministic component that accelerates convergence, avoids stagnation, and maintains solution diversity. Its adaptive elitism and gradient-informed correction mechanisms provide fine-grained control over search dynamics, ensuring efficiency even under tight nonlinear constraints.

The functional comparison in Table 6.10 highlights how HALSGWJA transitions from proof-of-concept simplicity to a fully engineered optimization algorithm. The introduction of approximate gradients and adaptive line search replaces static exploration–exploitation control with a dynamic balance, automatically guided by descent directions. This innovation leads to a substantial improvement in convergence speed and reliability, outperforming over forty contemporary metaheuristic algorithms on the CEC2020 benchmark suite.

TABLE 6.10. Functional and Performance-Based Comparison among SHGWJA, FHGWJA, and HALSGWJA

Feature	SHGWJA	FHGWJA	HALSGWJA (Hybrid Approximate Line Search Grey Wolf JAYA)
KEY INNOVATION	Simple combination of GWO and JAYA	Elitist and repair strategies for improved convergence	Integration of approximate line search and gradient-based refinement
UPDATE MECHANISM	GWO exploration followed by JAYA-based exploitation	GWO and JAYA sequentially combined with ranking and repair steps	GWO and JAYA enhanced with approximate gradients and adaptive line search
EXPLORATION–EXPLOITATION BALANCE	Controlled through simple elitism	Optimized via dynamic balancing of exploration/exploitation	Automatically tuned using gradient-derived descent directions
MAIN COMPUTATIONAL STEPS	6	7	6
GRADIENT USE	None	None	Yes (approximate gradients)
ELITIST STRATEGY	Simple, direct improvement of the best record	Advanced, rank-based elitist repair and mirroring	Adaptive elitism integrated with gradient-based correction
RELATIVE PERFORMANCE	Competitive with state-of-the-art metaheuristics	Superior or comparable to best-performing MHAs	Outperformed 40+ algorithms on the CEC2020 benchmark
APPLICATION DOMAINS	Civil, mechanical, and robotic engineering	Hydraulic, mechanical, and structural design	Real-world mechanical design and crashworthiness optimization

A critical synthesis of strengths and weaknesses, summarized in Table 6.11, further underscores the algorithmic maturation. SHGWJA stands out for its elegance and robustness despite limited exploitation capacity; FHGWJA introduces structural sophistication and adaptive elitism at moderate complexity; HALSGWJA achieves the most favorable trade-off between computational cost and optimization precision through gradient-informed intelligence. While the latter entails higher per-iteration computational cost, the total evaluation count is significantly reduced, yielding faster overall convergence.

TABLE 6.11. Strengths (PRO) and Weaknesses (CONS) of the SHGWJA, FHGWJA, HALSGWJA

Algorithm	PRO	CONS
SHGWJA	Extremely simple formulation. No algorithm-specific parameters. High robustness and reliable convergence. Competitive even for large-scale problems.	Limited exploitation capability for complex landscapes. No directional or gradient-based guidance.
FHGWJA	Enhanced exploration exploitation balance. Use of rank-based elitism and repair mechanisms. High precision and stability. Parameter-free formulation.	Higher algorithmic complexity than SHGWJA. Requires careful tuning of elitist and repair mechanisms.
HALSGWJA	Superior overall performance. Integration of approximate gradient information. Avoids stagnation and accelerates convergence. Substantially reduces number of evaluations.	More complex implementation. Requires computation of approximate gradients. Higher per-iteration computational cost.

In summary, the progression from SHGWJA to FHGWJA and finally HALSGWJA represents a methodologically coherent and empirically validated advancement in hybrid metaheuristic design. SHGWJA establishes a robust and elegant baseline; FHGWJA introduces structured adaptivity and enhanced precision; HALSGWJA culminates the sequence with gradient-guided intelligence, achieving state-of-the-art performance across diverse engineering optimization problems. Current research efforts are focused on extending HALSGWJA to large-scale applications involving hundreds of variables and thousands of nonlinear constraints, further exploring its potential as a powerful and scalable tool for industrial-scale engineering design.

In the next chapters, the investigation will be extended toward practical implementations within the "Industry 4.0" paradigm. In particular, attention will focus on two novel advanced algorithms derived from the hybrid metaheuristic design philosophy established in this work: Smart Explorer 1.0, a novel optimization algorithm for autonomous path planning and adaptive decision-making, and TSPGWJO (Travelling Salesman Problem Grey Wolf-JAYA Optimizer), a discrete variant of SHGWJA specifically developed for combinatorial and logistics optimization. These developments aim to illustrate how hybrid metaheuristics can be seamlessly integrated into intelligent, data-driven industrial environments, effectively bridging the gap between theoretical algorithmic innovation and real-world operational intelligence.

CHAPTER VII: SMART EXPLORER 1.0: INTELLIGENT ROBOTIC PATH PLANNING FOR INDUSTRY 4.0 APPLICATIONS

Along the shifting boundary between its own space and the unknown, the intelligent explorer senses subtle cues, navigating obstacles and currents, adapting with intuition and anticipating from past experiences. Flowing with the rhythms of the environment, it advances toward its goal, uncovering hidden paths and possibilities previously invisible.

7.1 INTRODUCTION TO PATH PLANNING IN ROBOTICS: SMART EXPLORER 1.0 AND CLASSICAL ALGORITHMS

Smart Explorer 1.0 (SE1.0) [286] is a novel path planning algorithm, specifically designed for Industry 4.0 contexts characterized by complex and dynamic environments. The algorithm was developed in collaboration with the Applied Mechanics research group at the Polytechnic University of Bari (POLIBA), under the scientific supervision of Prof. Luciano Lamberti (Full Professor of Mechanical Design), Prof. Mario Massimo Foglia (Associate Professor of Applied Mechanics) and Prof. Giulio Reina (Associate Professor of Applied Mechanics). The research team also included and Dr. Angelo Ugenti (Postdoctoral Researcher in Applied Mechanics). Leveraging Digital Terrain Models (DTMs), SE1.0 performs fine-grained environmental analysis by integrating terrain features (e.g., slope, stability, soil properties, obstacles) with dynamic factors (e.g., heat sources, electromagnetic fields). The managerial advantages of SE1.0 lie in its potential to increase decision-making transparency, reduce downtime, and support predictive planning. Nevertheless, its adoption may also raise concerns related to computational complexity, system interoperability, and initial capital investment. To demonstrate its practical applicability, SE1.0 is systematically evaluated in this chapter against both traditional paths planning algorithms, including A* and APF, as well as more advanced methods, such as RRT*.

To contextualize the contributions of SE1.0, it is important to first consider the role of autonomous navigation in mobile robotics. Autonomous navigation represents a critical capability, as it significantly reduces dependence on human intervention while enabling efficient and flexible deployment in complex environments. Path planning involves determining the optimal route for a robot to travel from its current position to a desired destination while ensuring efficiency and safety. Typically, the optimal path is defined as one that minimizes an objective function, such as travel time, energy consumption, or collision risk. These objectives are particularly crucial in high-stakes missions, such as search-and-rescue operations [287], where timely intervention can be life-saving, or planetary exploration [288], where energy resources are extremely limited and must be carefully managed [289]. Nevertheless, environmental characteristics and robot locomotion constraints impose limits on feasible paths. Complex and dynamic terrains, characterized by obstacles, variable slopes, and other environmental factors, further complicate the planning process. Despite their relevance, classical methods such as A* [290], [291], [292], [293], [294] and APF [295], [296], [297], [298], [299] show limitations when applied to irregular or dynamic environments. A*, although efficient, with complexity $O(E + VlogV)$, where E and V denote the number of edges and vertices, cannot adequately model physical factors such as slope variation or terrain hazards. Similarly, APF overlooks essential aspects of terrain morphology. To address these shortcomings, metaheuristic approaches were later introduced, including Rapidly Exploring Random Trees (RRT) and their optimized version RRT* [300], [301], [302], [303], which extend applicability to high-dimensional or partially known spaces. However, these methods remain computationally expensive and highly sensitive to parameter tuning.

The SE1.0 aims to overcome these limitations by performing a detailed terrain analysis and incorporating critical factors into a refined "alpha matrix," which evaluates traversability based on environmental parameters, such as slope thresholds, soil properties, and the presence of heat

sources or electromagnetic fields, as well as parameters that consider the vehicle's geometric characteristics to prevent risks such as overturning and collisions [304].

The development of path planning algorithms began with Dijkstra [305], [306] in 1959, who introduced a method for computing shortest paths in graphs. Hart et al. [290] extended this in 1968 with A*, adding heuristics to guide the search more efficiently. Li [295] proposed an interesting variant of the Artificial Potential Field (APF) introduced by Khatib in 1986 [307], where attractive and repulsive forces guide robots, though local minima remain a limitation. Terrain-specific adaptations followed: Ishigami et al. [289] (2011) introduced the Dynamic Mobility Index (DMI) for planetary rovers, incorporating slope, soil, speed, wheel slippage, and energy. Hybrid methods emerged for complex, dynamic contexts; Alenezi and Almeshal [287] combined Probabilistic Roadmaps (PRM) [308] with Spiral Dynamic Algorithm (SDA) [309] for UGVs in hazardous environments, enhancing safety and adaptability. In 2021, Sánchez-Ibáñez et al.[310] provided a comprehensive classification of path planning algorithms for autonomous mobile robots, focusing on ground vehicles and extendable to other surface robots, while addressing environmental models, robot dynamics, and their applicability.
Recent enhancements to classical algorithms include Yu et al. [293], who adapted A* for AGVs with kinematic constraints; Zhang et al.[291], who introduced a Variable Neighbourhood Search (VNS)-based A*; and Kabir et al. [294], who achieved significant computational improvements via dynamic weighting and heuristic optimization. APF methods have also evolved: Huang et al.[297] improved UAV navigation with real-time obstacle avoidance and environmental adaptation, while Ni and Zhou [298] enhanced AUV APF with sonar-based dynamic force adjustments.

Building on these algorithmic improvements, path planning methods can be categorized as global, local, or learning-based [311] [312]. Global methods, including A* and RRT-type algorithms [300], [301], [302], assume complete environmental knowledge, whereas local methods (APF) adapt trajectories in real-time [313], often relying on LiDAR, vision, or sensor fusion. Learning-based methods, including deep reinforcement learning (DRL), optimize trajectories through reward-based exploration but remain computationally intensive and require careful design to balance exploration and exploitation.

This section has provided an overview of the key developments and classifications in path planning, establishing the context for the remainder of the chapter. Sections 7.2–7.3 describe the architecture of SE1.0; Section 7.4 presents comparative results; Section 7.5 illustrates an industrial case study; Section 7.6 discusses managerial and technical implications, emphasizing strengths and limitations; concludes with future research directions and implementation perspectives.

7.2 DESIGN AND PRINCIPLES OF THE SMARTEXPLORER1.0 (SE1.0) ALGORITHM

The SE1.0 has the potential to leverage Digital Terrain Models (DTMs) (see Figure 7.1) and integrate geometric, environmental, and vehicle-specific parameters, together with sensory data, to dynamically adapt to complex and variable terrains. DTMs represent the spatial distribution of terrain elevations and support interpretation and decision-making across diverse applications [314], [315]. They are commonly used in geospatial analysis. for testing

algorithms under extreme scenarios, such as steep gradients, uneven surfaces, and high navigational variability.

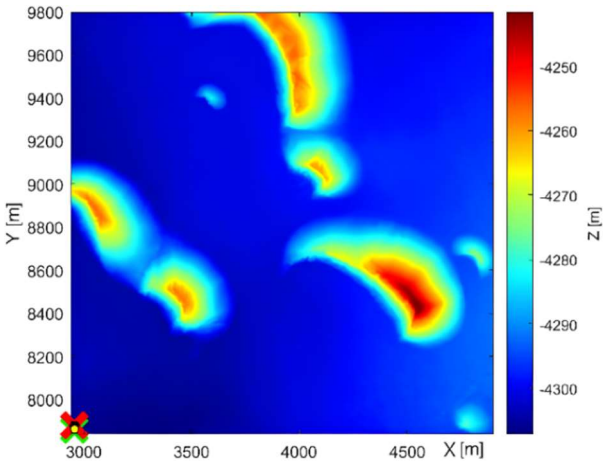


Figure 7.1- TYPICAL CONTOUR MAP OF A DIGITAL TERRAIN MODEL (DTM)

The core feature of SE1.0 is the introduction of the Alpha matrix $\alpha(i,j)$, a "traversability map" based on environmental parameters (K-type) and vehicle characteristics. This approach is inspired by the human cognitive system, which interprets environmental information based on individual perception to navigate and overcome external challenges. Similarly, SE1.0 replicates this adaptive capacity by integrating sensory inputs with advanced computational models, with the green-red areas defined by thresholds used to interpret the environment. In the absence of specific data, the values of K, on which $\alpha(i,j)$ is based, are set to 1 and assigned different weights based on empirical criteria related to the risk associated with the specific phenomenon being measured. Following this logic, additional parameters related to other phenomena can be incorporated to further refine the navigation capability without compromising the vehicle's integrity. The main phenomena that compromise vehicle stability are overturning and pitching (see Figure 7.2)

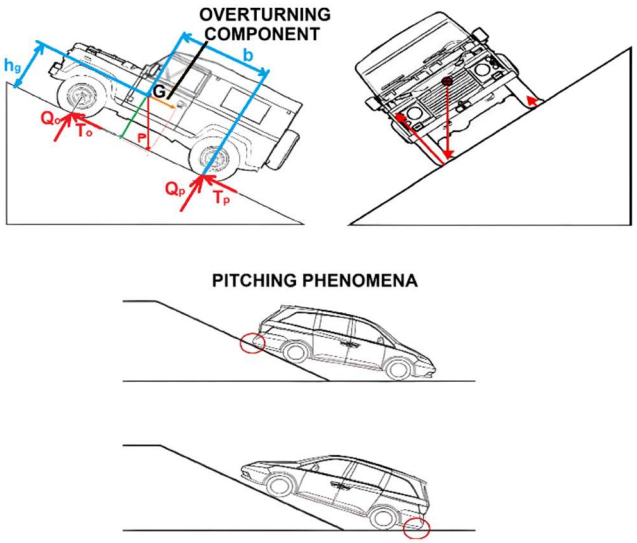


Figure 7.2- PRIMARY PHENOMENA OF STABILITY LOSS: OVERTURNING AND PITCHING.

Overturning occurs when lateral forces tip the vehicle over, typically when the center of gravity exceeds a critical point, causing loss of lateral stability. Pitching refers to forward or backward rotation around the vehicle's transverse axis, often caused by acceleration changes or terrain variations, affecting longitudinal stability [316], [317], [318], [319]. These phenomena are prevented through the introduction of threshold parameters, which can be managed differently in the XY, XZ and YZ planes and depend on the vehicle's turning radius and load constraints, as detailed below.

The first step in testing SE1.0 involved loading a Digital Terrain Model (DTM) of Martian soil to simulate a highly complex environment, with a start point (A) and a target point (B) defined. The start and target points do not need to coincide with grid nodes, as the algorithm can identify the nearest point within the map data. If either point lies outside the map's bounds, an error message is displayed, prompting the user to select a different map.

After this phase, SE1.0 requires the definition of key configuration parameters, such as the "maximum number of iterations" (Maxiter), the "number of candidates" (candidate_number), the prediction level, and the thresholds that define critical limits for terrain slope and slope variations. The prediction level, which determines the algorithm's forward-looking planning scope, is dynamically adjusted according to the complexity of the environment. This "scouting phase" forms the foundation for effective route optimization by proactively addressing terrain and environmental challenges.

At each iteration, SE1.0 assesses multiple candidate moves from the current position, represented at the center of a 3×3 matrix as shown in the Figure 7.3. This approach enables comprehensive exploration of the surrounding terrain and allows the vehicle to backtrack if stagnation or local optima are encountered. The process also includes mechanisms to prevent revisiting previously traversed areas and the avoidance of critical obstacles.

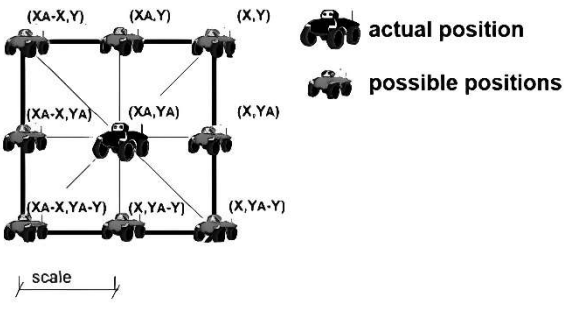


Figure 7.3- CANDIDATE POSITIONS REPRESENTATION: THE CURRENT POSITION IS AT THE CENTER, SURROUNDED BY EIGHT POTENTIAL CANDIDATE POSITIONS.

The mathematical approach solves a nonlinear, multimodal optimization problem (see Equation 7.1). For each candidate position, P(i,j), the algorithm calculates a cost function, incorporating multiple factors, such as proximity to the destination, obstacle avoidance, and

minimization of elevation changes, allowing it to quantify the relative advantage of each potential move. If stagnation occurs or the maximum number of iterations is exceeded, SE1.0 adapts by increasing the prediction level or backtracking to previously visited positions, ensuring continuous progress toward the goal while effectively managing terrain and environmental challenges.

The objective function, $FO_{tot}(i,j)$, is computed using the Euclidean distance formula, which accounts for horizontal (x-axis) and vertical movements (y-axis) on the XY plane of the map, as well as the elevation of each point (z-axis):

$$FO_{tot}(i,j) = \sqrt{(x(i,j) - x_B)^2 + (y(i,j) - y_B)^2 + (z(i,j) - z_B)^2} \tag{7.1}$$

The optimization problem consists in minimizing this function. To prevent revisiting previously traversed points or areas too close to the starting path, penalties are applied in two ways:

1. History penalty: If a position already exists in the history of visited points, a high penalty (1e4) is assigned.
2. Blacklist penalty: If a candidate position lies in a "blacklist," it is treated as an obstacle and given the maximum penalty (1e6).

The function explicitly ignores the current position, which has already been visited. After evaluating all candidate positions, SE1.0 selects the position with the minimum objective function value, and the coordinates of the final optimal position are returned.

For each position on the map, the Alpha matrix is calculated, integrating multiple parameters to deliver an adaptive evaluation of navigational challenges. The calculation is performed as follows:

$$\alpha(i,j) = K_s(i,j) \cdot K_g(i,j) \cdot K_t(i,j) \cdot K_e(i,j) \cdot \big(\beta(i,j) + \delta(i,j)\big) \cdot \gamma(i,j) \tag{7.2}$$

The values of the various parameters K are determined empirically and, respectively, represent:

K_s: Soil Stiffness Coefficient, that is the stiffness or rigidity of the soil (elastic properties). $K_s = 1$ for rigid soil or $K_s > 1$ for deformable soil (e.g., $K_s = 10^5$ for gas).

K_g: Granular Composition Coefficient, that is the grain size distribution and composition of the soil (compaction and stability). $K_g = 1$ for low rock or $K_g > 1$ for sand, silt, clay.

K_t: Temperature Sources Coefficient or Atmospheric perturbations. $K_t = 1$ for standard conditions or $K_t > 1$ for dangerous conditions.

K_e: Electromagnetic Sources coefficient. $K_e = 1$ for standard conditions or $K_e > 1$ for dangerous conditions.

The parameters β, δ, and γ (Figure 7.4) represent specific penalties that affect the overall "cost" or navigability of a given terrain cell and are also linked to the phenomena of vehicle rollover and pitching.

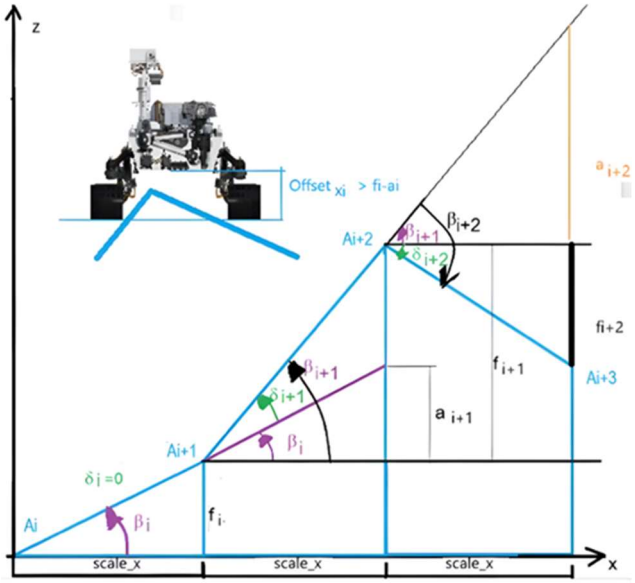


Figure 7.4- REPRESENTATION OF ALPHA MATRIX'S CHARACTERISTIC ANGLES AND INTERFACE PARAMETERS FOR VEHICLE STABILITY ASSESSMENT (OFFSET).

Their roles can be interpreted as follows:

β is the slope of the soil and denotes a "Penalty for Elevation Changes". Higher β values correspond to areas where elevation changes significantly, making movement more energy-intensive or technically challenging for the vehicle or agent. The β parameter influences the cost of moving through terrain with steep inclines or descents. For example, if a position has a steep slope, $\alpha(i,j)$ will increase proportionally to the β weighted elevation change.

δ is the slope variation of the soil and represents an "Obstacle Penalty". If a terrain cell is obstructed (e.g., by rocks, trees, or other physical barriers), $\alpha(i,j)$ is significantly increased by the δvalue to discourage pathfinding through that cell. This ensures that the algorithm strongly avoids regions that are physically impassable, effectively directing the path toward alternative routes.

γ is a prediction factor and indicates a "Directional Penalty" or "Cumulative Slope". It sums penalties in a specific direction, accounting for the aggregated difficulty of traversing adjacent cells in a particular orientation. For instance, γ sums the $\alpha(i,j)$ values of cells along the x- or y-axes, evaluating directional trends such as sustained inclines, rugged terrain, or terrain-specific penalties. By incorporating γ, SE1.0 penalizes the paths that involve prolonged movement through unfavourable conditions, guiding the trajectory toward smoother or more efficient routes. The larger the accumulated gamma, the higher the penalty and, consequently, the less favourable that position becomes in the pathfinding process.

$$\gamma = \sum_{step=1}^{prediction\ level} \alpha_{candidate} \cdot (i + step, j + prediction\ level - 1) \tag{7.3}$$

Thus, to refine position scoring, the parameter γ is determined by summing the α values in specific directional subdomains: positive x-direction: contributions from leftward positions, negative x-direction: contributions from rightward positions, positive y-direction: contributions from upper positions, negative y-direction: contributions from lower positions.

Figure 7.5 illustrates the prediction strategy, which is set to a default value of 4. This value has been empirically determined to be optimal for typical operating conditions, and it dynamically adjusts, ranging from 1 to 100, depending on the complexity of the environment.

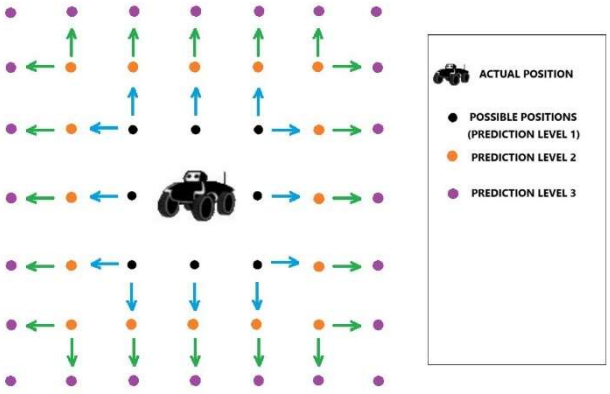


Figure 7.5- FRAMEWORK OF A SYSTEMATIC APPROACH FOR THE PREDICTION STRATEGY.

The computed $\gamma$is then compared against a series of thresholds tied to the prediction hierarchy:

if $\gamma \geq 10^6$, the penalty assigned to $FO_{tot}(i,j)$ depends on the prediction level:

First prediction level: $FO_{tot}(i,j) = 10^6$.

Second prediction level: $FO_{tot}(i,j) = 10^5$.

Third prediction level: $FO_{tot}(i,j) = 10^4$.

Fourth prediction level: $FO_{tot}(i,j) = 10^3$.

This hierarchy enables SE1.0 to impose stricter penalties as predictions become more immediate, thereby discouraging paths with high directional penalties.

Moreover, for each terrain cell (i,j), the elevation difference between the current point and the next point along the x-axis, f_x, and the corresponding slope, β_x, and similarly, f_y, along the y-axis, are computed as follows:

$$f_x(i,j) = z(i,j+1) - z(i,j) \tag{7.4}$$

$$f_y(i,j) = z(i+1,j) - z(i,j) \tag{7.5}$$

which represents the gradient of elevation along the horizontal direction in the XZ and YZ planes.

Correspondingly, the slopes β and slope variations δ are calculated in the YZ and XZ planes:

$$\beta_x(i,j) = \arctan(f_x(i,j)scale_x + \varepsilon) \tag{7.6}$$

$$\beta_y(i,j) = \arctan\big(f_y(i,j)scale_y + \varepsilon\big) \tag{7.7}$$

where $scale_x$ and $scale_y$ define the resolution or step size of the DTM's grid and ε is a small numerical constant (set to 1e-6 in the code) introduced to prevent division by zero during computations.

Referring to Figure 7.4, the parameter δ_x is calculated by considering the difference between the current slope and the predicted slope. The angular component is then combined with the sine of the previous slope:

$$\delta_x(i,j) = \arcsin\left(\frac{f_x(i,j) - a_x(i,j)}{c_x(i,j)} \cdot \sin\left(\frac{\pi}{2} + \beta_x(i,j-1)\right)\right) \tag{7.8}$$

In this formula:

$$\frac{f_x(i,j) - a_x(i,j)}{c_x(i,j)} \tag{7.9}$$

represents the normalized slope variation

$$\sin\left(\frac{\pi}{2} + \beta_x(i,j-1)\right) \tag{7.10}$$

introduces the orientation of the previous slope to make the calculation dependent on the terrain's history.

$$a_x(i,j) = scale_x \cdot \tan\big(\beta_x(i,j-1)\big) \tag{7.11}$$

estimates the expected elevation based on the previous slope $\beta_x(i, j-1)$ to highlight variations relative to this prediction.

$$c_x(i,j) = \sqrt{scale_x^2 + f_x^2(i,j)} \tag{7.12}$$

calculates the diagonal length of the triangle formed between two consecutive points along the x-direction.

Similar calculations are also performed for the y-axis and then it is combined x and y slopes into a composite slope β and slope variation δ as follows:

$$\beta(i,j) = \sqrt{\beta_x^2(i,j) + \beta_y^2(i,j)} \tag{7.13}$$

$$\delta(i,j) = \sqrt{\delta_x^2(i,j) + \delta_y^2(i,j)} \tag{7.14}$$

Once the alpha matrix is calculated, the vehicle's external environment is fully interpreted, allowing for the accurate identification of areas that can be safely navigated. This process ensures the vehicle can move without compromising safety or integrity, while staying within the constraints defined by the matrix.

Figures 7.6 and 7.7 represent the visual output obtained with the alpha matrix criteria applied on the DTM's map and clearly highlights navigable regions (green) and obstacles (red), providing operators with intuitive and actionable insights. It is evident that the vehicle's ability to handle steeper slopes introduces greater complexity into the map. This added layer of complexity enhances the vehicle's capability to navigate through more challenging and rugged terrains, effectively enabling it to traverse difficult and intricate environments.

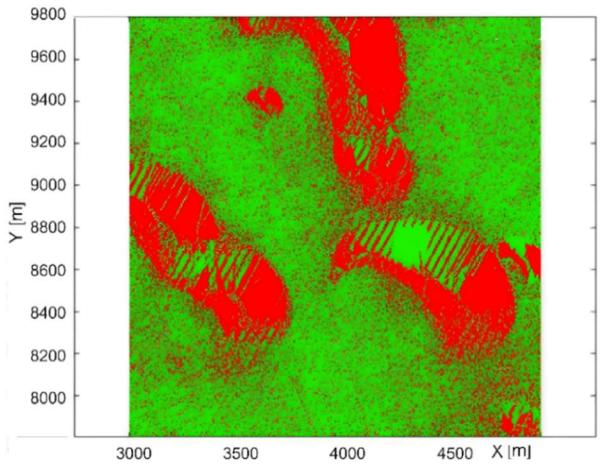


Figure 7.6- VISUAL MAP OF ACCESSIBLE AREAS (GREEN) AND NON-ACCESSIBLE AREAS (RED), AS DETERMINED BY THE CONSTRAINTS OF THE A MATRIX, FOR THRESHOLDS SET FOR 10% SLOPES.

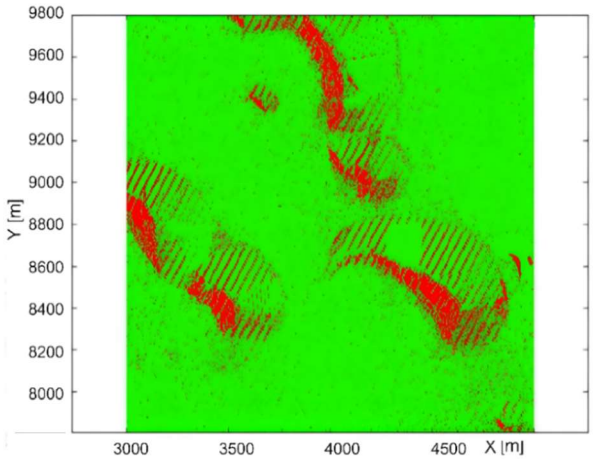


Figure 7.7- VISUAL MAP OF ACCESSIBLE AREAS (GREEN) AND NON-ACCESSIBLE AREAS (RED), AS DETERMINED BY THE CONSTRAINTS OF THE A MATRIX, FOR THRESHOLDS SET FOR 20% SLOPES.

The computational complexity of SE1.0 is influenced by three main factors: (i)the number of points in the map grid $N_{pt,map}$, which determine the size of the Alpha matrix. (ii)the number of iterations actually performed during the path search act_{iter}, which may be lower than the Maxiter. (iii)the maximum number of locations $N_{loc,iter}$, that can be visited in each iteration.

The number of grid points is determined by the square of the ratio between the map size and the map resolution. For instance, for a square region of side 1000 m with a resolution of 2m, the total number of points is given by $N_{pt,map} = \left(\frac{1000}{2}\right)^2$. The Alpha matrix is computed for the entire set of $N_{pt,map}$ grid points. Moreover, as illustrated in Figure 7.3, the maximum number of locations that SE1.0 can visit in a given iteration corresponds to the number of candidate positions in the parameter $candidate_{number}$, which is set to 8 in the present case. N summery the computational complexity of SE1.0 can be expressed as: $O(N_{pt,map+candidat\ _{number}} \cdot act_{iter})$. Despite these considerations, SE1.0 requires only basic computations and moderate memory, mainly determined by the grid size $N_{pt,map}$. The algorithm's flow chart is shown in Figure 7.8.

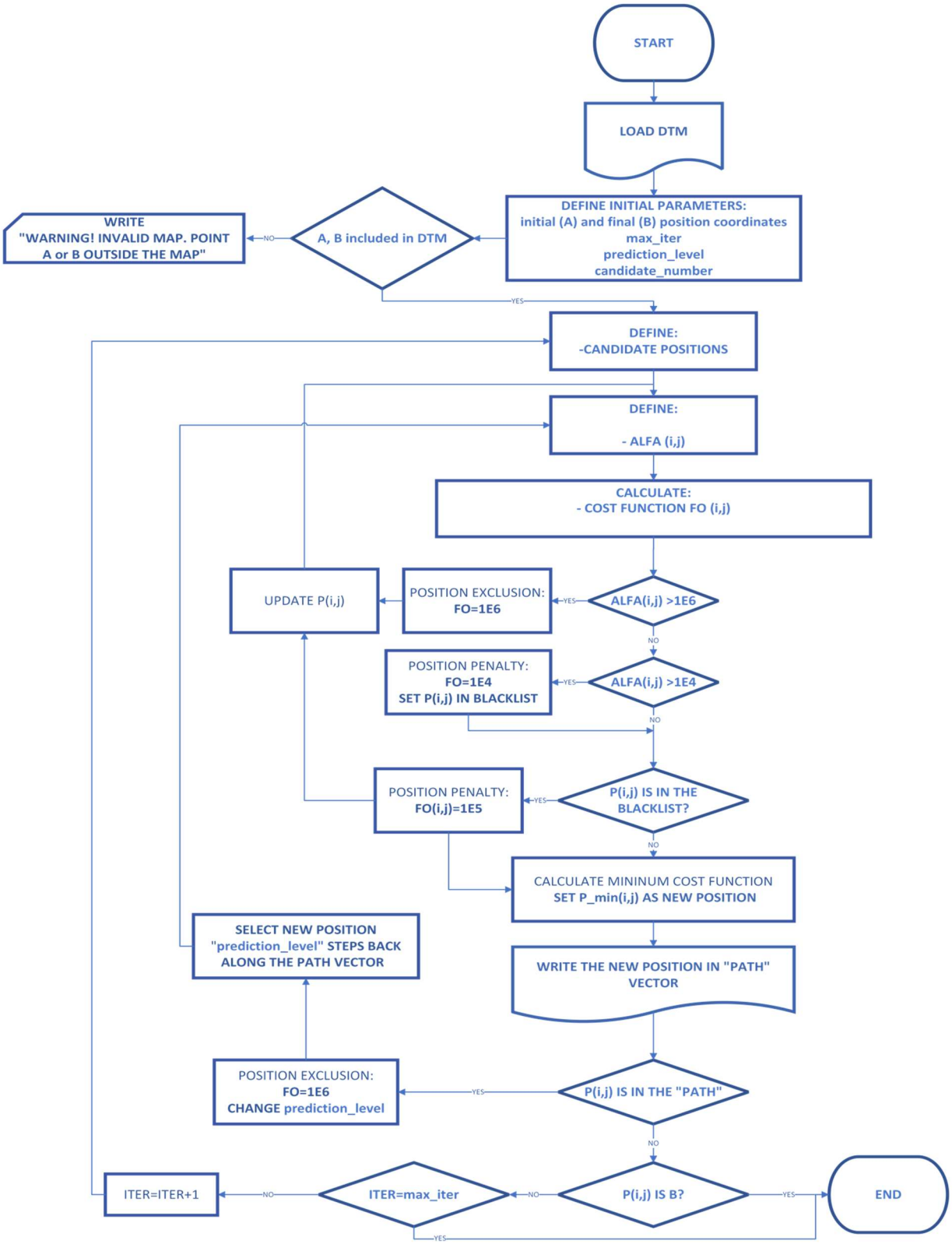


Figure 7.8- FLOWCHART OF THE SE1.0 ALGORITHM

7.3 CHARACTERIZATION OF ENVIRONMENTAL FACTORS AND PARAMETERS FOR SE1.0

To assess its effectiveness, SE1.0 was benchmarked against the A*, APF, and RRT* methods through a set of numerical simulations. A consistent comparison was ensured by employing a common static map, derived from the DTM using the Alpha matrix criterion described in the previous section. All K-type parameters were fixed to 1; as a result, sensor-based inputs were excluded, and the analysis relied solely on the topographic information encoded in the DTM.

Unlike conventional approaches, SE1.0 is inherently capable of handling DTMs of varying resolution and quality, since the Alpha matrix is defined directly from the model characteristics. Real-world resolution effects are incorporated through the $scale$ parameter.

For the present study, a high-resolution Martian surface DTM was selected from the HiRISE repository, owing to the complexity of its topography:

https://www.uahirise.org/dtm/ESP_035588_2565

The dataset features a grid step size of 0.98 m in both x- and y-directions; accordingly, the $scale$ parameter was set to 0.98. Once this parameter is defined, SE1.0 performs trajectory search consistently with the map resolution.

The parameters aimed at preventing overturning and pitching were defined as follows:

Anti-rollover thresholds:

$\beta_{xmin} = \beta_{ymin} = -0.245\, rad$: minimum slope threshold in radians for a 20% descent manageable by the vehicle.

$\beta_{xmax} = \beta_{ymax} = 0.245\, rad$: maximum slope threshold in radians for a 20% ascent manageable by the vehicle.

Anti-pitching thresholds:

$\delta_{xmin} = \delta_{ymin} = -0.245\, rad$: minimum slope variation threshold in radians for a 20% descent manageable by the vehicle.

$\delta_{xmax} = \delta_{ymax} = 0.245\, rad$: maximum slope variation threshold in radians for a 20% ascent manageable by the vehicle.

To evaluate SE1.0, simulations were carried out in environments featuring both high and low obstacle densities. The goal was to test the vehicle's ability to escape dead ends and concave areas, while also identifying the optimal path and determining the most appropriate default value for the prediction factor.

Accessible areas (green) and non-accessible areas (red) are shown in Figure 7.9. These maps were generated using a 20% slope threshold, which ensures that the vehicle operates within

safe limits. Areas exceeding this slope are considered unsafe for navigation. Thresholds lower than 20% were not used, as they would prevent meaningful comparisons with the A* and APF algorithms, which perform poorly on highly constrained or complex paths.

The DTM map used to demonstrate the SE1.0 algorithm is shown in Figure 7.1. The map has a resolution of 0.98 meters per grid cell.

The initial evaluation focused on the algorithm's ability to resolve vehicle entrapment within a small non-convex region. The DTM was processed using the Alpha-matrix method with a 20% slope threshold. The planned trajectory extended from a starting point inside the non-convex zone, A (2956, 7879) m, to a target point located outside the concavity, B (2956, 7860) m.

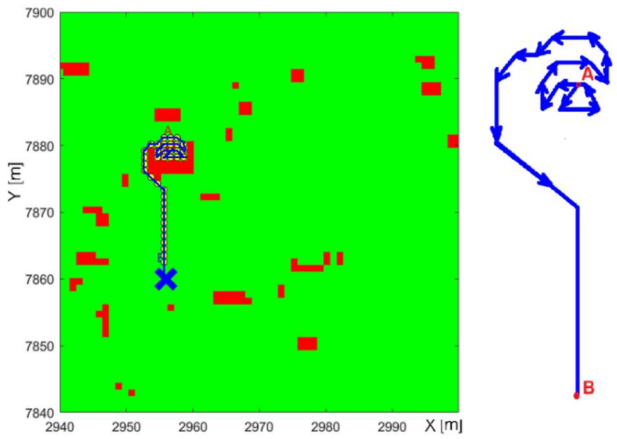


Figure 7.9- DETAIL OF THE MAP WITH TRAJECTORY POSITIONS (LEFT) AND SEQUENCE OF MOVEMENTS (RIGHT) AS OBTAINED BY THE BASELINE SE1.0 IN SCENARIO A. 20% SLOPE THRESHOLD.

Figure 7.9 shows the green-red map of traversable and non-traversable areas generated with the Alpha-matrix method (20% slope threshold) and the path obtained via a one-step solution. SE1.0 initially explores multiple positions within the concavity of the non-convex region, producing paths with several sharp turns. While these avoid prohibited zones, they can increase traversal time and reduce efficiency in dynamic environments.

To address this, an isoline-based pathfinding method was introduced, with weights (w) used to influence path selection. By following elevation contours, SE1.0 favours smoother trajectories, prioritizing points along the same or closely related iso-levels, while weights penalize certain directions to guide the path toward more efficient routes.

The objective function in Equation 7.1 was modified to introduce penalty factors, as described below:

$$FO_{tot}(i,j) = \sqrt{w_x \cdot (x(i,j) - x_B)^2 + w_y \cdot (y(i,j) - y_B)^2 + w_z \cdot (z(i,j) - z_B)^2} \quad (7.15)$$

where w_x, w_y , w_z represent the weights assigned to the horizontal, vertical, and altitude directions, respectively, used to prioritize one direction over the others.

A higher value for a specific weight increases the penalty associated with that direction, thereby discouraging movement along that axis relative to others.

Various simulations were conducted to determine the optimal penalty factors configuration, aimed at enhancing the quality of the generated path.

Figures 7.10 and 7.11 present the results of two specific case studies, representing the best and the worst, respectively:

Case 1: $w_x = w_y = 1, w_z = 10$.

Case 2: $w_x = w_y = 1, w_z = 100$.

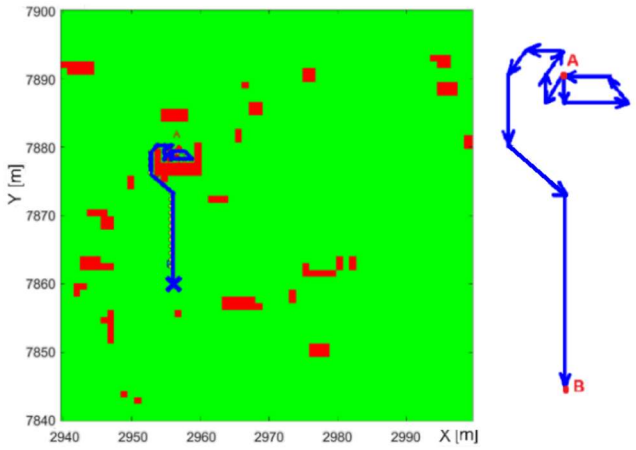


Figure 7.10- SE1.0 - CASE 1: $w_x = 1, w_y = 1, w_z = 10$, 20% SLOPE THRESHOLD. DETAIL OF THE MAP WITH TRAJECTORY POSITIONS (LEFT) AND SEQUENCE OF MOVEMENTS (RIGHT).

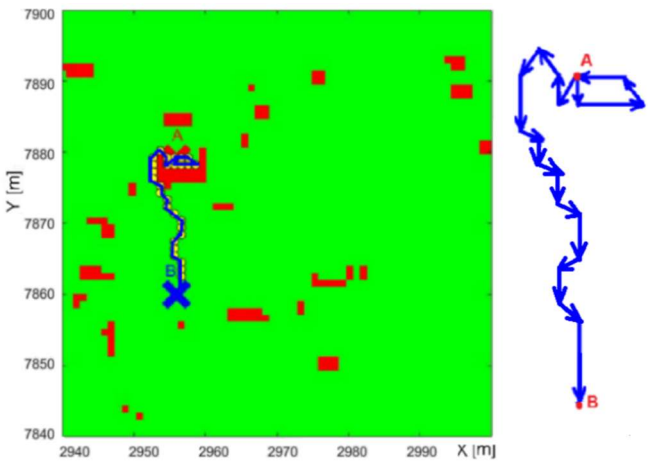


Figure 7.11- SE1.0 - CASE 1: $w_x = 1, w_y = 1, w_z = 100$, 20% SLOPE THRESHOLD. DETAILS OF THE MAP WITH TRAJECTORY POSITIONS (LEFT) AND SEQUENCE OF MOVEMENTS (RIGHT).

Using a low horizontal weight and moderately higher vertical weight, SE1.0 effectively prioritizes slope-following movements, minimizing unnecessary deviations while adhering closely to contour lines.

Minor deviations from iso-levels are permitted, whereas larger deviations are penalized to maintain path fidelity.

When the vertical weight is set too high, the algorithm excessively avoids elevation changes, producing trajectories with greater deviations and amplified horizontal fluctuations.

This refinement simplifies the trajectory, reducing overall path length and improving energy efficiency, consistent with prior studies [320], [321] showing that streamlined paths lower energy consumption and mechanical wear in complex, non-convex environments.

Performance is further enhanced by incorporating a predictive strategy using the prediction factor γ. Optimal results, shown in Figure 7.12, occur at γ = 4, which balances distance minimization and energy efficiency, providing an effective and sustainable framework for autonomous navigation across diverse terrain scenarios.

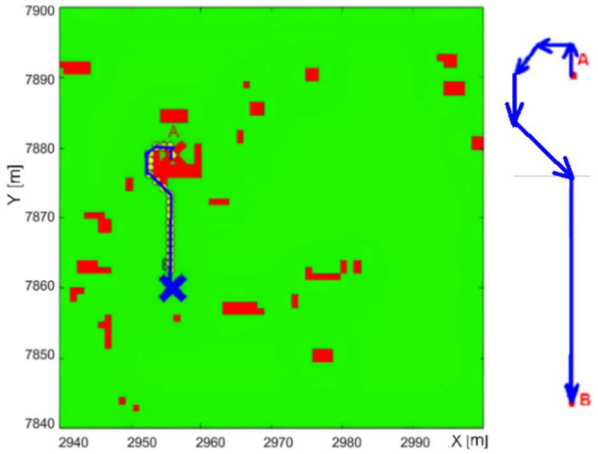


Figure 7.12- SE1.0 - CASE 1: $w_x = 1, w_y = 1, w_z = 10$, 20% SLOPE THRESHOLD, WITH PREDICTION FACTOR $\gamma = 4$. DETAILS OF THE MAP WITH TRAJECTORY POSITIONS.

Simulations confirmed that the γ parameter is central to SE1.0's performance, balancing exploration of the search space with exploitation of promising paths. A value of γ = 4 provides a robust compromise, yielding efficient navigation without unnecessary detours.

Further tests showed that γ is sensitive to environmental conditions, and adaptive adjustment based on real-time feedback can further optimize performance.

This modulation enables the algorithm to maintain shorter trajectories and minimize energy use, supporting more efficient and reliable operation in complex terrains.

7.4 RESULTS OF SE1.0: NAVIGATION, PATH PLANNING, AND PREDICTIVE PERFORMANCE

A comparative analysis was conducted to evaluate SE1.0 in relation to conventional path planning methods (A* and APF) and RRT*, a modern metaheuristic algorithm. The study examined three scenarios (A, B, and C) described in Table 7.1, using path length, computational time, and number of iterations as performance metrics. All simulations were performed on an Apple M1 Pro, equipped with 8 cores, 16 GB RAM, and a 3228 MHz clock speed.

TABLE 7.1- SCENARIOS USED FOR THE COMPARISON BETWEEN SE1.0, A, APF AND RRT*.*

Scenario	Initial Point	Final Point	Slope threshold
A	(2956, 7879)	(2956, 7860)	20%
B	(3200, 8000)	(4600, 9400),	20%
C	(3200, 8000)	(4600, 9400),	40%

A*, APF, and RRT* employ distinct path-finding strategies suited to different environments. A* efficiently finds shortest paths in grid-based maps with convex obstacles using heuristic-guided structured search. APF relies on potential fields but often fails in non-convex regions due to local minima, which can trap the vehicle [307], [322], [323], [324], [325], [326]. RRT*, a sampling-based algorithm with asymptotic optimality, performs well in high-dimensional spaces but requires many samples and significant memory to approach the global optimum.

Comparisons with SE1.0, particularly in Scenario A (Figure 7.12), highlight these strengths and limitations. As shown in Figure 7.13, APF cannot escape concave obstacle regions effectively, whereas SE1.0 successfully navigates complex topologies.

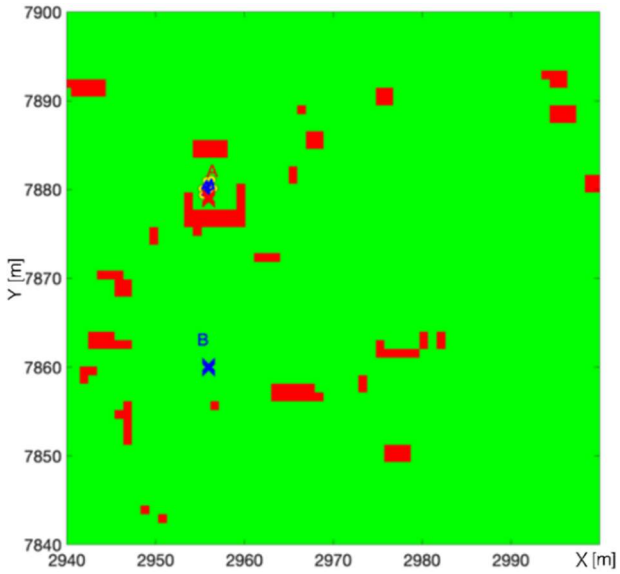


Figure 7.13- APF-SCENARIO A, 20% SLOPE THRESHOLD. DETAILS OF THE MAP WITH TRAJECTORY POSITIONS. ENTRAPMENT INTO THE NON-CONVEX REGION AFTER 11 ITERATIONS.

As shown in the contour plot of the potential function in Figure 7.14, the vehicle becomes trapped in local minima. This limitation reflects the method's failure to represent the environment's global structure accurately, resulting in incomplete or erroneous potential fields that substantially degrade navigation performance.

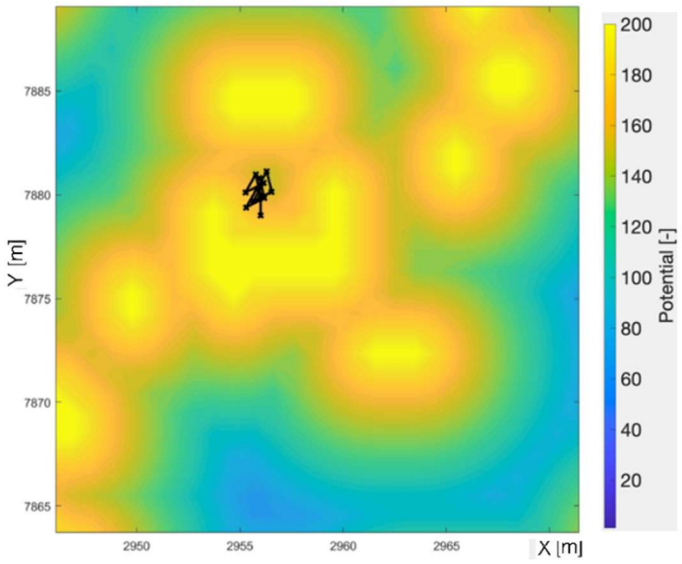


Figure 7.14- APF- SCENARIO A, 20% SLOPE THRESHOLD. POTENTIAL FIELD'S ZOOM IN WITH TRAJECTORY POSITIONS. ENTRAPMENT INTO THE NON-CONVEX REGION AFTER 11 ITERATIONS.

Rimon and Koditschek [323] first highlighted that APF methods struggle in non-convex regions, a limitation further confirmed by Kim and Khosla [324], who showed that local

minima often prevent optimal solutions. Zhu and Latombe [325] suggested hybrid strategies combining APF with global planning to mitigate these issues.

In this study, the classical APF formulation was used without modifications such as random walks, which are challenging to implement due to problem-dependent step sizes and the undefined topology of candidate points near local minima.

In contrast, A* employs an informed heuristic that evaluates paths based on actual cost and estimated distance to the goal, allowing it to favour routes closer to the global optimum and avoid entrapment in suboptimal regions, as shown in Figure 7.15.

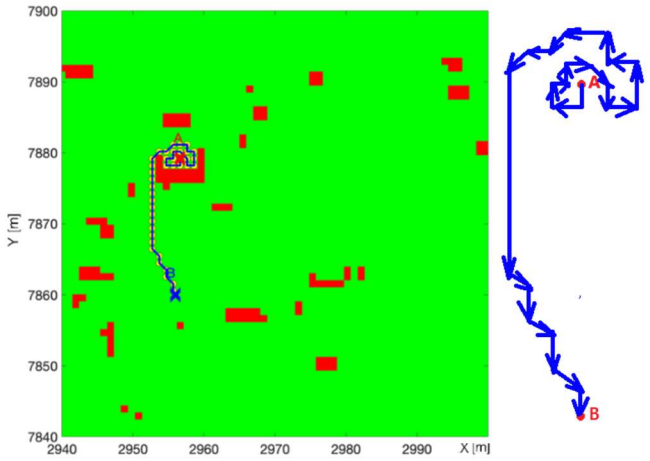


Figure 7.15- A- SCENARIO A, 20% SLOPE THRESHOLD. DETAILS OF THE MAP WITH TRAJECTORY POSITIONS.*

The structured search strategy of A* is well-documented in the literature. Hart et al. [290] demonstrated that A* effectively balances exploration and exploitation by prioritizing promising regions, avoiding unnecessary computations and suboptimal local minima. By expanding nodes toward the goal, the algorithm ensures global optimality and favors lower-cost paths.

A*'s adherence to the principle of optimality guarantees that once a path is selected, it will not be abandoned unless a better alternative is found. This makes A* particularly robust in complex environments with non-convex obstacles, outperforming the limitations of APF.

Figure 7.16 shows the trajectory generated by the RRT* (Rapidly-exploring Random Tree Star) algorithm in Scenario A.

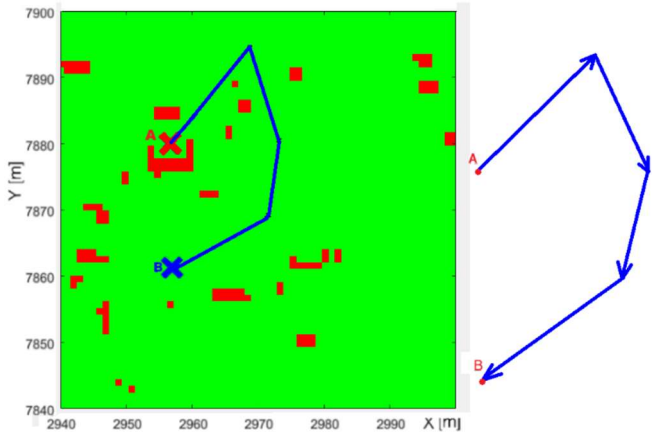


Figure 7.16- RRT- SCENARIO A, 20% SLOPE THRESHOLD. DETAILS OF THE MAP WITH TRAJECTORY POSITIONS.*

RRT* generated a feasible path, though generally less optimal than those produced by A* and SE1.0, due to its probabilistic, incremental nature. The diagram on the right illustrates the sequence of motion segments and direction changes. While RRT* avoids obstacles, the trajectory is non-linear and includes multiple directional adjustments, reflecting the algorithm's adaptive response to environmental constraints.

Table 7.2 presents a comparison of SE1.0, APF, A*, and RRT* for Scenario A (non-convex problem). Cumulative travelled distances and path completion are reported, highlighting the strengths and limitations of each method in this small-scale scenario.

TABLE 7.2- COMPARISON OF PATH PLANNING METHODS IN SCENARIO A, INITIAL POINT (2956, 7879), FINAL POINT (2956, 7860), 20% SLOPE THRESHOLD.

Method	Total Path Length [m]	N iterations up to the stop	Computational Time [s]	Goal reached
SE1.0	25.14	25	0.090-0.113	YES
APF	11.00(*)	11(*)	0.301-0.401(*)	NO
A*	39.75	38	0.024-0.047	YES
RRT*	64.11	18136	4.395-4.622	YES

(*) Value up to stagnation in local minima without reaching the goal.

In non-convex zones, APF fails to reach the goal after only 11 iterations due to stagnation in local minima, highlighting its limited effectiveness in complex scenarios. A* successfully reaches the goal despite the longest total path length (39.75 m), demonstrating robustness against local minima and an effective balance between computational effort and path optimality. RRT* produces the longest trajectory (64.11 m) and requires 18136 iterations, with a computational time of 4.395-4.622 s. Its incremental, sampling-based strategy avoids obstacles but generates indirect, meandering paths, particularly in environments with dispersed obstacles. SE1.0 achieves a moderate path length of 25.14 m, reaching the goal efficiently while avoiding unnecessary detours. The algorithm consistently converges to the same

optimized path, with a standard deviation of total distance never exceeding 0.1%, demonstrating both robustness and strong potential for optimality. Overall, SE1.0 balances computational efficiency and path quality more effectively than APF, A*, or RRT* in non-convex zones.

Building on these results, the study extended the analysis to a more complex setting (Scenario B), encompassing a larger area with both convex and non-convex regions and varying obstacle densities, including high- and low-density zones. Such complex environments better reflect real-world challenges, where non-convex regions and uneven obstacle distributions can hinder algorithm performance. While A* and APF perform well in simple scenarios, they often struggle to adapt to these more intricate conditions.

In Scenario B, the trajectory was defined from Point A (3200, 8000) to Point B (4600, 9400) on the initial DTM map, covering 3200–4600 m along the x-axis and 8000-9400 m along the y-axis, with a resolution of 0.98 m to ensure spatial accuracy.

The classical APF method failed to find a path due to the high obstacle density, which created numerous local minima and trapped the agent. This limitation stems from APF's reliance on gradient-based potential fields and its sensitivity to tuning parameters, such as the weighting of repulsive and attractive forces. Poorly calibrated weights can lead to collisions or endless oscillations between obstacles, highlighting the need to combine APF with global planning strategies.

Figure 7.17 shows that A* performs better than APF in Scenario B but still experiences stagnation in areas of high obstacle density.

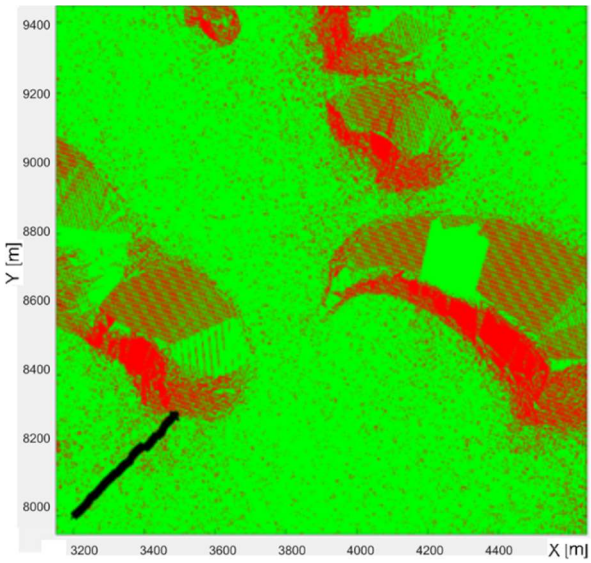


Figure 7.17- A-SCENARIO B, 20% SLOPE THRESHOLD. MAP WITH TRAJECTORY POSITIONS AND STAGNATION PHENOMENA.*

The performance of A* is highly dependent on its heuristic function. In densely cluttered regions, the heuristic may fail to clearly distinguish optimal from suboptimal paths, causing

excessive computation in local areas with little progress. High obstacle density further intensifies the trade-off between path optimality and computational efficiency, as A* may favour seemingly promising paths that struggle to escape congested zones, resulting in stagnation or inefficient trajectories. This illustrates the limitations of classical path-planning algorithms in environments with non-uniform obstacle distributions.

Figure 7.18 shows the trajectory generated by RRT* in Scenario B. The algorithm successfully finds a collision-free path through the complex, non-convex landscape, demonstrating adaptability where APF and A* may fail due to local minima or grid resolution limitations. While the resulting path is not always optimal in length or smoothness, RRT*'s random sampling and tree expansion strategy enables reliable global pathfinding in large, cluttered environments. The increased planning time reflects the scenario's complexity but is offset by the algorithm's robustness in challenging terrains.

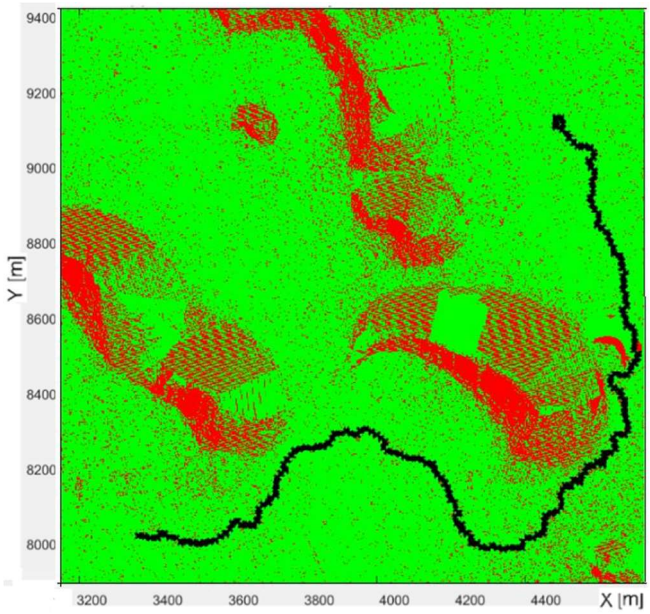


Figure 7.18- RRT- SCENARIO B, 20% SLOPE THRESHOLD. MAP SHOWING TRAJECTORY POSITIONS LEADING TO THE GOAL.*

Figure 7.19 demonstrates that Smart Explorer 1.0 (SE1.0) reliably reaches the goal even in extended scenarios with heterogeneous obstacle densities, outperforming both APF and A*. This success stems from SE1.0's dynamic balance between exploration and exploitation, guided by its adaptive prediction factor, which modulates search behaviour according to local environmental conditions. Unlike APF, which relies on local gradients and struggles in dense or non-convex regions, SE1.0 efficiently navigates varying obstacle configurations.

Compared to A*, SE1.0 combines global path-planning strategies with real-time adaptability, preventing stagnation in high-density areas. The algorithm can explore alternative routes when bottlenecks occur, maintaining consistent progress toward the goal without excessive computational overhead.

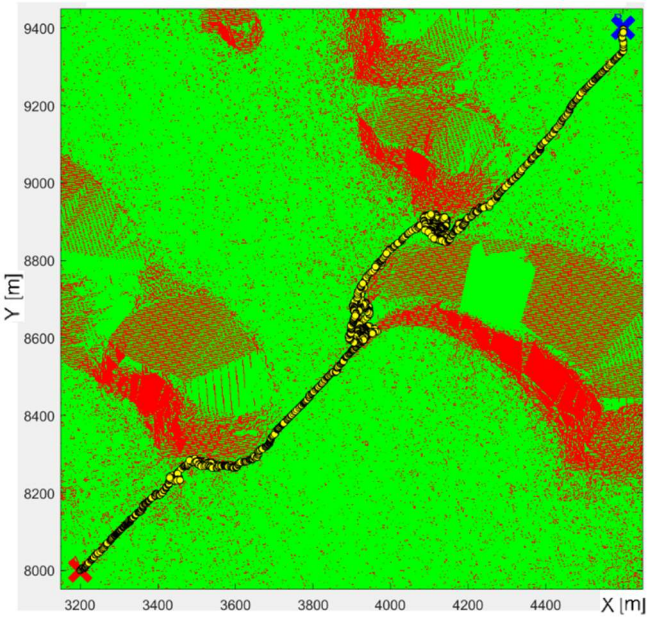


Figure 7.19- SE1.0- SCENARIO B, 20% SLOPE THRESHOLD. MAP SHOWING TRAJECTORY POSITIONS LEADING TO THE GOAL.

A comparison of the paths generated by SE1.0 in shouting mode (Figure 7.19) and RRT* (Figure 7.18) reveals that SE1.0 produces a significantly more direct and smoother trajectory. While both algorithms avoid high-slope regions (red zones), SE1.0 navigates with greater precision, exploiting terrain geometry more effectively and maintaining tight margins along critical areas without violating slope constraints.

The SE1.0 trajectory closely follows a globally optimal course, minimizing detours and curvature. This demonstrates the algorithm's ability to balance slope limitations with path optimality, leveraging environmental structure for a streamlined trajectory. Its structured, deterministic search strategy ensures local continuity and cost-aware exploration.

Table 7.3 summarizes the relative performance of SE1.0, APF, A*, and RRT* in Scenario B, highlighting the advantages of SE1.0 in complex environments.

TABLE 7.3- COMPARISON OF PATH PLANNING METHODS IN SCENARIO B, INITIAL POINT (3200, 8000), FINAL POINT (4600, 9400), 20% SLOPE THRESHOLD.

Method	Total Path Length [m]	N iterations up to the stop	Computational Time [s]	Goal reached
SE1.0	17012.23	8616	11.305-12.056	YES
APF	12.00 (*)	12(*)	0.290-0.354(*)	NO
A*	503.68 (*)	387(*)	0.026-0.027(*)	NO
RRT*	3868.72	71261	91.137-95.798	YES

(*) Value up to stagnation in local minima without reaching the goal.

In Scenario B, SE1.0 successfully reached the goal, covering a total path length of 17012.23 m. Although the straight-line distance between start and goal is only 1980 m, the extended trajectory reflects SE1.0's exhaustive exploration, which visits numerous nodes to ensure successful navigation in complex terrain. An improved SE version under development aims to smooth the trajectory by identifying "tangle" regions, clusters of adjacent nodes with repeated x or y coordinates and progressively shrinking path branches between TS and TE points. Preliminary results suggest this approach could reduce the cumulative travel distance by at least 60%.

In contrast, APF fails after only 12 m due to entrapment in local minima, while A* does not reach the goal, covering 503.68 m and stagnating in high-density regions. SE1.0 completed 8616 iterations within 11.3-12.1 s. Despite the higher computational cost, its adaptive exploration strategy enables goal attainment, highlighting its advantages over classical algorithms in complex, constrained environments.

Table 7.3 highlights the trade-off between computational efficiency and problem-solving capability. SE1.0 requires greater computational resources but consistently reaches the goal, effectively overcoming local minima and maintaining global trajectory optimization.

RRT* also reaches the goal with a shorter, more direct path (3868.72 m) and without tangle regions. However, it requires 71261 iterations and 91.137-95.798 seconds, reflecting the high computational cost of its stochastic, incremental sampling approach. Its trajectory is less smooth and exhibits local irregularities, demonstrating that, while RRT* can handle complex terrain, it lacks the structured determinism of SE1.0. Consequently, SE1.0 offers a superior balance of path feasibility, goal attainment, and potential for optimization, particularly when smoothing procedures are applied, making it suitable for large-scale, non-convex environments.

To further evaluate performance, Scenario C was considered, with terrain slopes set to 40% in the alpha-matrix, producing a red-green map with lower obstacle density and simulating demanding stability conditions. Start and goal points were the same as in Scenario B.

APF again fails after only 11 m and 11 iterations, indicating that it requires near-flat terrain or highly stable vehicles to function effectively. Figure 7.20 shows that, under these extreme slope conditions, A* remains effective, demonstrating its relative robustness compared to APF.

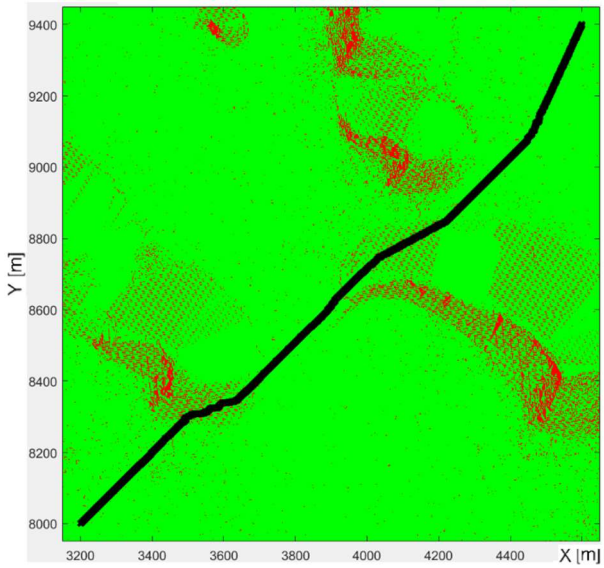


Figure 7.20- A- SCENARIO C, 40% SLOPE THRESHOLD. MAP SHOWING TRAJECTORY POSITIONS LEADING TO THE GOAL.*

In this challenging scenario, where the vehicle can handle slopes up to 40%, the A* algorithm proves effective. Unlike APF, A* avoids entrapment in zero-net-force regions, enabling reliable navigation across steep terrain. Figure 7.21 shows the RRT* results under the same conditions. While RRT* successfully reaches the goal, its trajectory is suboptimal, with multiple abrupt directional changes. In contrast, the path generated by A* remains the most efficient for this scenario, demonstrating superior performance in terms of smoothness, directness, and goal attainment.

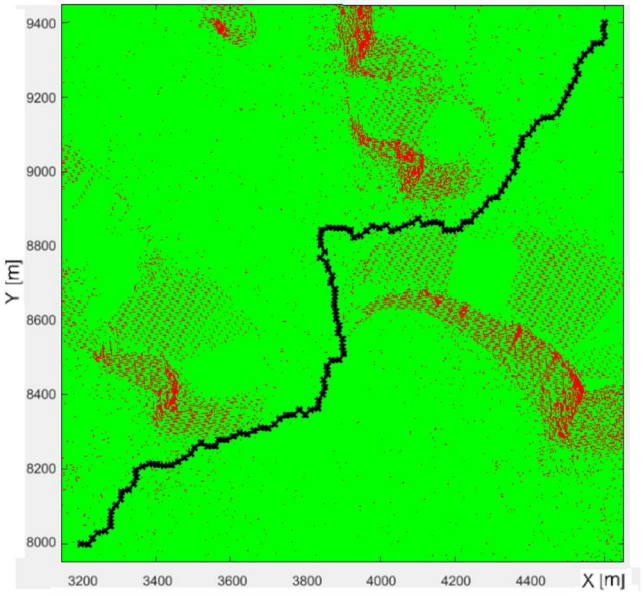


Figure 7.21- RRT - SCENARIO C, 40% SLOPE THRESHOLD. MAP SHOWING TRAJECTORY POSITIONS LEADING TO THE GOAL.*

Under these improved conditions, the novel Smart Explorer 1.0 (SE1.0) algorithm successfully reaches the goal once again, exhibiting reduced difficulty compared to the previously investigated scenario, as illustrated in Figure 7.22.

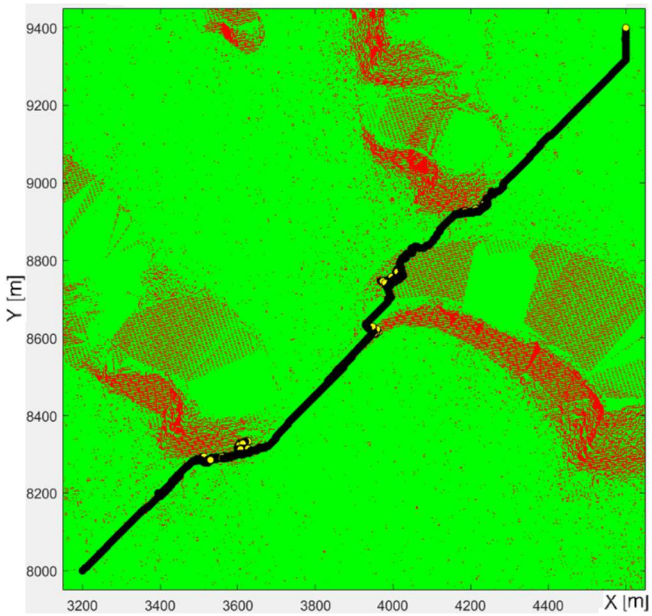


Figure 7.22- SE1.0 - SCENARIO C, 40% SLOPE THRESHOLD. MAP SHOWING TRAJECTORY POSITIONS LEADING TO THE GOAL.

Table 7.4 presents the path-planning performance of the four algorithms, SE1.0, APF, A*, and RRT*, for Scenario C.

TABLE 7.4- COMPARISON OF PATH PLANNING METHODS IN SCENARIO C, INITIAL POINT (3200, 8000), FINAL POINT (4600, 9400), 40% SLOPE THRESHOLD

Method	Total Path Length [m]	N iterations up to the stop	Computational Time [s]	Goal reached
SE1.0	5404.67	1798	2.761-2.847	YES
APF	11.00 (*)	11(*)	0.294-0.309 (*)	NO
A*	2132.58	1653	0.042-0.058	YES
RRT*	2548.06	5527	2.341-2.473	YES

(*) Value up to stagnation in local minima without reaching the goal.

RRT* achieves a shorter path and lower computational time than SE1.0 but requires significantly more iterations. SE1.0's reported path includes all segments explored in "scouting mode," whereas RRT* considers only the final optimized trajectory, highlighting differences in exploration overhead. Despite this, SE1.0 navigates robustly, covering 5404.67 m between start and goal. Preliminary improvements targeting tangle regions are expected to reduce this distance by at least 50%.

APF fails to reach the goal, stagnating after only 11 m due to local minima, confirming its limitations in complex, steep terrains. A* successfully reaches the goal with 2132.58 m, completing 1653 iterations in 0.042-0.058 s. SE1.0 requires 1798 iterations and 2.761-2.847 s, demonstrating reliable performance under challenging conditions.

SE1.0 was further validated on the real "DUNE" robot (Figure 7.23.a).

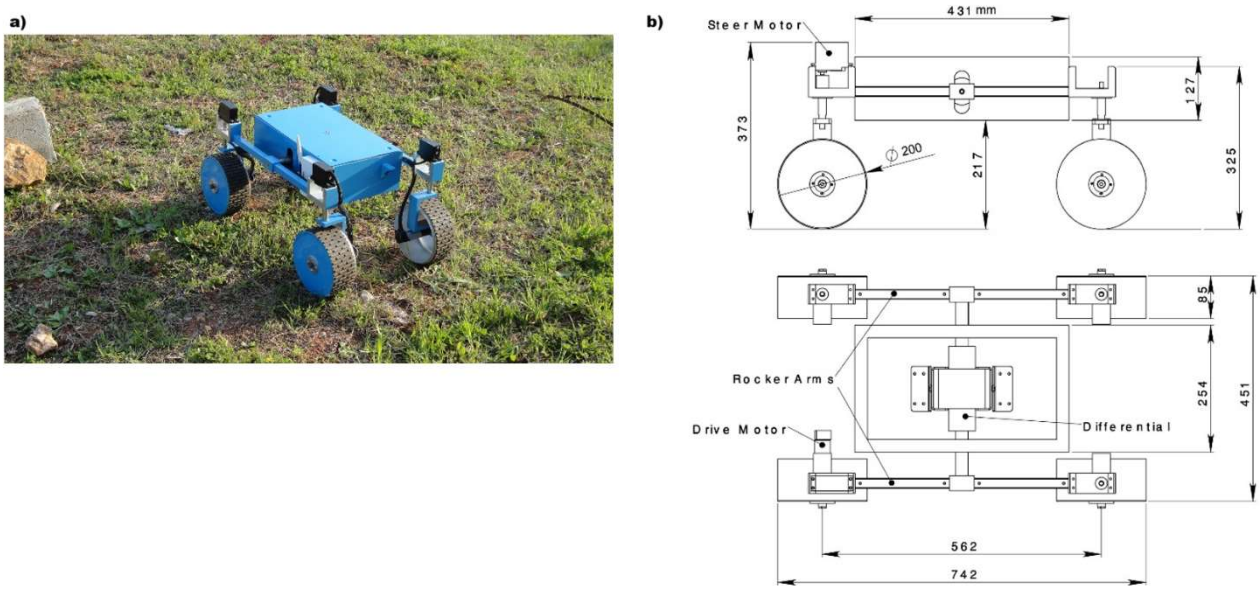


Figure 7.23- REAL ROBOT DUNE USED IN THIS STUDY (a) AND ITS GEOMETRICAL DIMENSIONS (b).

DUNE is a kinematically equivalent replica of a planetary exploration rover, custom-built at the Robotic Mobility Lab of the Polytechnic University of Bari (Italy). Its main geometric dimensions are shown in Figure 23.b. The robot features an independent four-wheel drive and steering system, coupled with a rocker-type passive suspension, and has an overturning threshold of 26.1° (49.03%).

Figures 7.24, 7.25, 7.26 and 7.27 illustrate the performance of the four path-planning algorithms evaluated in this study. Figure 7.24 highlights the failure of the traditional APF method to reach the goal. In contrast, Figures 7.25–7.27 demonstrate that SE1.0, A*, and RRT* successfully guide the DUNE robot to the target, confirming the practical applicability and reliability of these algorithms in real-world rover navigation.

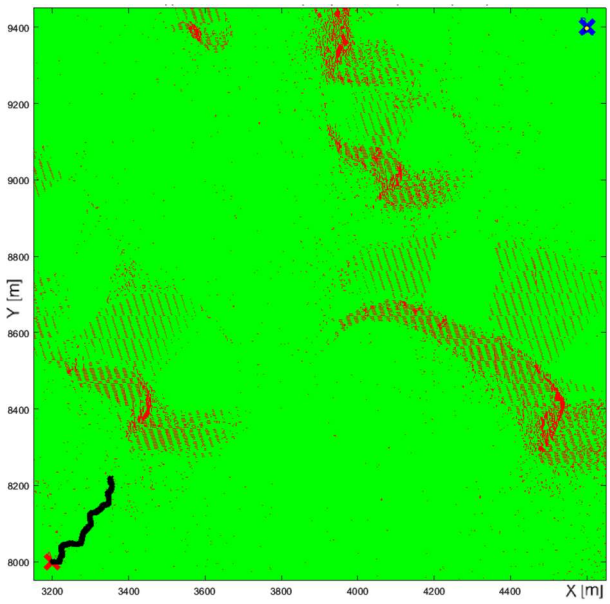


Figure 7.24- APF - SCENARIO "REAL WORLD APPLICATION" ON THE "DUNE" ROBOT, 49.03% SLOPE THRESHOLD. MAP DISPLAYING TRAJECTORY POSITIONS WITHOUT REACHING THE GOAL.

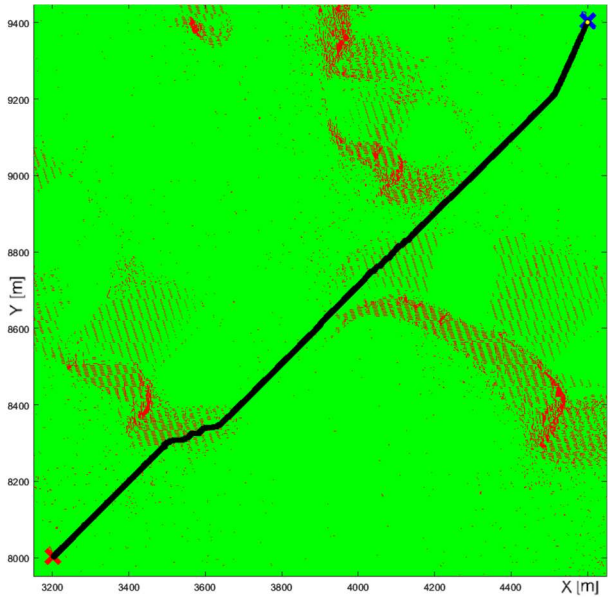


Figure 7.25- A - SCENARIO "REAL WORLD APPLICATION" ON THE "DUNE" ROBOT, 49.03% SLOPE THRESHOLD. MAP DISPLAYING TRAJECTORY POSITIONS THAT LEAD TO THE GOAL.*

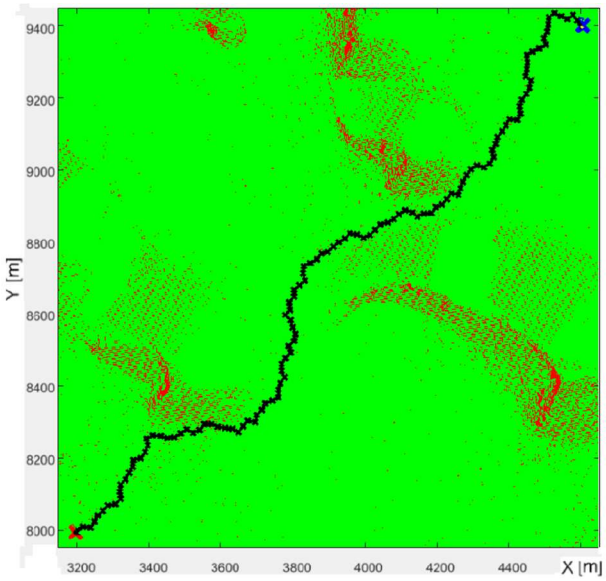


Figure 7.26- RRT - SCENARIO "REAL WORLD APPLICATION" ON THE "DUNE" ROBOT, 49.03% SLOPE THRESHOLD. MAP DISPLAYING TRAJECTORY POSITIONS THAT LEAD TO THE GOAL.*

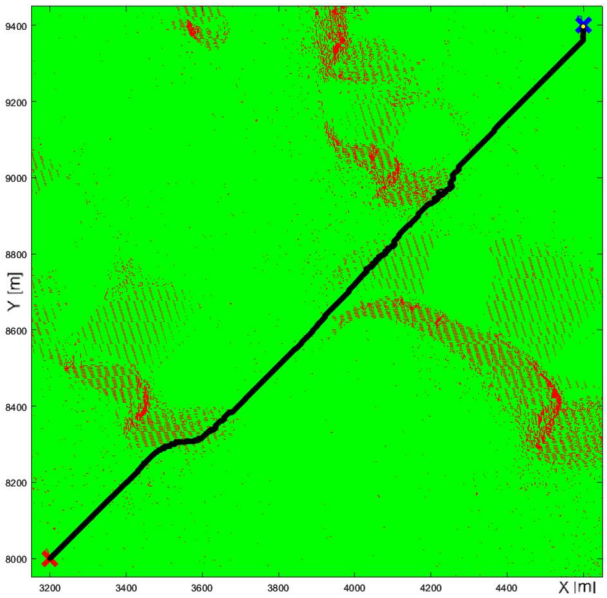


Figure 7.27- SE1.0 - SCENARIO "REAL WORLD APPLICATION" ON THE "DUNE" ROBOT, 49.03% SLOPE THRESHOLD. MAP DISPLAYING TRAJECTORY POSITIONS REACHING THE GOAL.

Table 7.5 presents a comparative analysis of the four path planning methods - SE1.0, Artificial Potential Field (APF), A* and RRT* - applied to the DUNE real robot navigation scenario.

TABLE 7.5- COMPARISON OF PATH PLANNING METHODS IN A REAL-WORLD APPLICATION (DUNE ROBOT). INITIAL POINT (3200, 8000), FINAL POINT (4600, 9400), 49.03% SLOPE THRESHOLD.

Method	Total Path Length [m]	N iterations up to the stop	Computational Time [s]	Goal reached
SE1.0	2231.20	1713	1.187	YES
APF	373 (*)	373(*)	2.51(*)	NO
A*	2079.70	1565	0.093	YES
RRT*	2502.83	3965	2.125	YES

(*) Value up to stagnation in local minima without reaching the goal.

The scenario in real world application involves traversal from an initial point (3200, 8000) m to a final point (4600, 9400) m on the same Martian DTM used for SE1.0 calibration, with a slope threshold of 49.03%, consistent with the rover's terrain capabilities. Performance was evaluated using four metrics: total path length [m], number of iterations, computational time [s], and goal attainment.

SE1.0 successfully reaches the goal, demonstrating resilience in complex terrain, albeit with a slightly longer path (2231.2 m) and moderate computational cost (1.187 s) compared to A*. While SE1.0 prioritizes goal attainment over path length, preliminary improvements suggest tangle regions can be removed to reduce distance by at least 5%, narrowing the gap with A* (2079.7 m).

APF fails due to stagnation in local minimum. RRT* reaches the goal with a final path of 2502.83 m over 3695 iterations and 2.125 s; however, SE1.0's distance includes the full exploration phase (scouting mode), making direct comparisons nontrivial. Despite this, SE1.0 achieves faster computation and robust navigation.

7.5 EXPERIMENTAL VALIDATION: SE1.0 NAVIGATION PERFORMANCE OF DUNE ROVER ON UNSTABLE TERRAIN

In the study of terrestrial mobility, particularly for robotic exploration on soft or unstable terrains, two critical phenomena, slippage and sinkage, strongly affect vehicle performance and efficiency [327], [328]. These factors are especially relevant for wheeled, tracked, or legged systems, such as rovers, navigating substrates like sand, snow, mud, or extraterrestrial surfaces.

Slippage occurs when a vehicle's traction element (wheel or track) rotates faster than the forward motion due to insufficient friction, leading to propulsion loss, increased energy consumption, and component wear [329], [330]. It is typically quantified via the slippage ratio, indicating traction efficiency. High slippage can immobilize the vehicle in soft soils.

Sinkage refers to the vertical penetration of wheels or tracks into the terrain when applied pressure exceeds the soil's bearing capacity, influenced by vehicle weight, contact area, and soil properties [331]. Excessive sinkage increases rolling resistance and often exacerbates slippage, forming a feedback loop that impairs mobility.

Understanding these interactions is essential for planetary exploration, where rovers must traverse uncharted and unstable terrains. To investigate these effects, the DUNE rover was simulated on a Martian DTM map using APF, A*, and SE1.0, with a focus on slippage and sinkage. Two highly unstable zones were incorporated as impassable areas by assigning elevated K_g values, defined as circular regions: C1 (3325 m, 8125 m), R1 = 50 m; C2 (3825 m, 8525 m), R2 = 100 m.

SE1.0 successfully detects and avoids these hazardous zones, optimizing its trajectory (Figure 7.31), whereas APF and A* fail to reach the target (Figures 7.28, 7.29). Notably, A* is diverted toward the second unstable area, becoming trapped, highlighting the limitations of classical methods in navigating challenging, unstable terrains.

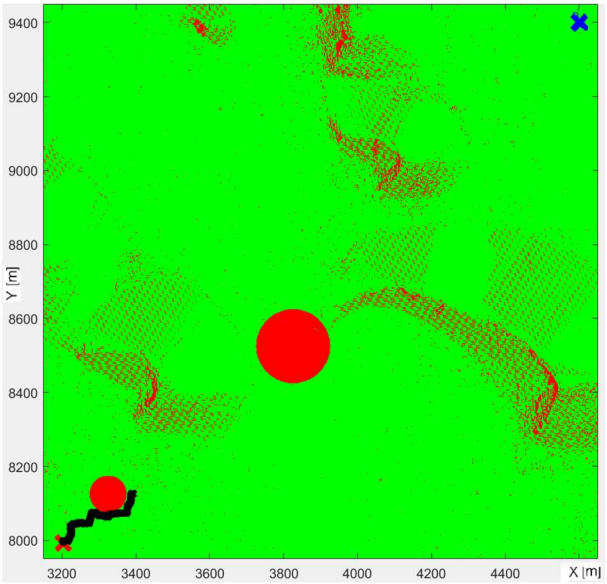


Figure 7.28- APF - SCENARIO "UNSTABLE TERRAIN" (TWO CIRCULAR AREAS) ON THE "DUNE" ROBOT, 49.03% SLOPE THRESHOLD. MAP DISPLAYING TRAJECTORY POSITIONS WITHOUT REACHING THE GOAL.

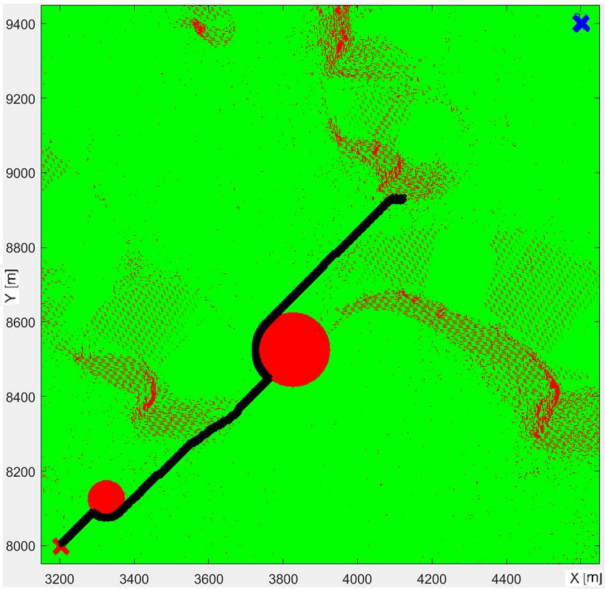


Figure 7.29- A - SCENARIO "UNSTABLE TERRAIN" (TWO CIRCULAR AREAS) ON THE "DUNE" ROBOT, 49.03% SLOPE THRESHOLD. MAP DISPLAYING TRAJECTORY POSITIONS WITHOUT REACHING THE GOAL.*

An interesting observation arises with the RRT* algorithm, which successfully reaches the goal but generates a more complex and less optimized trajectory compared to SE1.0 (Figure 7.30). RRT* fails to identify the narrow passage near coordinates (4900, 8200), a corridor constrained by large obstacles and densely populated with smaller ones. Consequently, the final path length is 3523.81 m, significantly longer than the 2528.15 m achieved by SE1.0's scouting mode. This outcome highlights a key limitation of RRT*: navigating highly constrained environments often requires precise manoeuvring, which the algorithm's stochastic sampling strategy may not reliably capture.

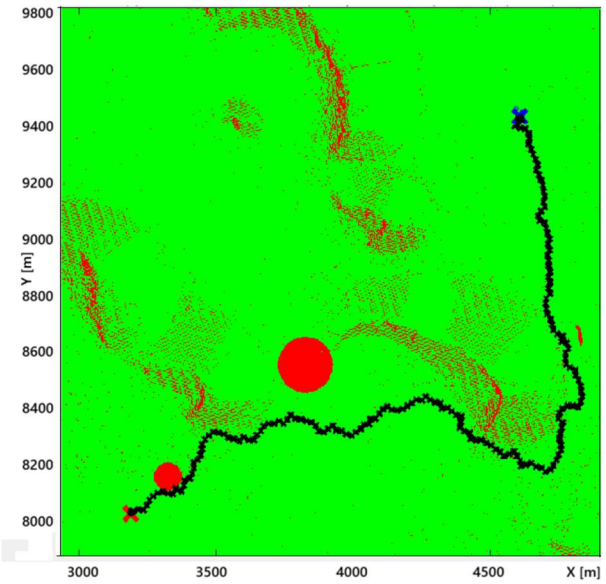


Figure 7.30- RRT - SCENARIO "UNSTABLE TERRAIN" (TWO CIRCULAR AREAS) ON THE "DUNE" ROBOT, 49.03% SLOPE THRESHOLD. MAP DISPLAYING TRAJECTORY POSITIONS WITHOUT REACHING THE GOAL.*

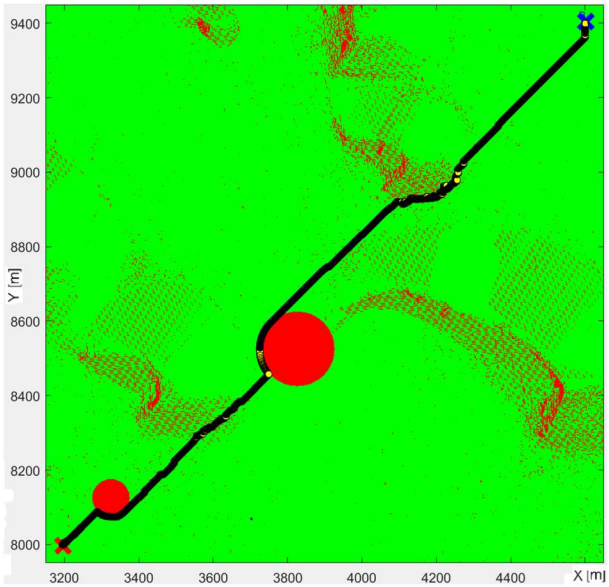


Figure 7.31- SE1.0 - SCENARIO "UNSTABLE TERRAIN" (TWO CIRCULAR AREAS) ON THE "DUNE" ROBOT, 49.03% SLOPE THRESHOLD. MAP DISPLAYING TRAJECTORY POSITIONS REACHING THE GOAL.

Table 7.6 presents a comparative analysis of the four path planning methods - SE1.0, APF, A* and RRT* - applied to navigation on unstable terrain, with two hazardous circular areas and a slope threshold of 49.03%.

TABLE 7.6- COMPARISON OF PATH PLANNING METHODS ON UNSTABLE TERRAIN. FIRST CIRCULAR AREA (CENTER C1 (3325 M, 8125 M), RADIUS R1 = 50 M), SECOND CIRCULAR AREA (CENTER C2 (3825 M, 8525 M), RADIUS R2 = 100 M), 49.03% SLOPE THRESHOLD.

Method	Total Path Length [m]	N iterations up to the stop	Computational Time [s]	Goal reached
SE1.0	2528.15	1852	1.232	YES
APF	350 (*)	350 (*)	2.86 (*)	NO
A*	1620.70 (*)	1250 (*)	0.643 (*)	NO
RRT*	3523.81	5900	2.228	YES

(*) Value up to stagnation in local minima without reaching the goal.

SE1.0 successfully reaches the goal with the shortest path directly obtained in scouting mode and demonstrates the fastest computational performance (1.232 s), outperforming RRT* (2.228 s). APF fails after 350 iterations due to stagnation in a local minimum, covering only 350 m in 2.86 s, while A* also becomes trapped after 1250 iterations, failing to complete the scenario.

7.6 CONCLUSIONS AND FUTURE PERSPECTIVES ON SE 1.0

The Smart Explorer 1.0 is a one-step metaheuristic path planning algorithm developed to minimize the Euclidean distance to the target while ensuring robust hazard avoidance.

SE1.0 demonstrates an effective equilibrium between exploration and exploitation, the two fundamental and complementary phases characterizing metaheuristic algorithms. Exploration promotes diversification by probing distinct regions of the search space, thereby mitigating the risk of local entrapment. This phase is guided by a dynamic prediction factor, which adapts to terrain complexity and obstacle density, and by the selection of candidate positions with minimal penalties, in a manner analogous to reward-based learning. Exploitation, by contrast, intensifies the search in promising areas through structured generation of candidate positions, thereby improving convergence precision.

The principal innovation of SE1.0 is the Alpha matrix, a dynamic framework that encodes global terrain attributes, including slope, slope variation, soil properties, and environmental hazards. This representation enables comprehensive evaluation of navigational difficulty and is continuously updated to reflect changing conditions. It simultaneously optimizes horizontal and vertical motion, penalizes high-risk zones, discourages revisiting previously explored nodes, and permits adaptive weighting of terrain-dependent movements.

In Table 7.7, an objective and methodologically grounded assessment of SE1.0 is presented in comparison with the notable algorithms A*, APF, and RRT*, thereby elucidating the distinctive strengths and limitations inherent to each approach.

TABLE 7.7 - PROS AND CONS OF SE1.0, A, APF AND RRT**

	PROS	CONS
SE1.0	Detailed terrain analysis (slopes, slope variations, obstacles). Dynamic risk prevention (overturning, collisions). Suitable for real-world, complex terrains (autonomous navigation).	Higher algorithmic complexity. Requires detailed input data (e.g., Digital Terrain Map).
A*	Efficient algorithm for graph-based pathfinding (O (E + V log V)). Flexible heuristics for diverse applications.	Does not account for physical constraints like slopes or collisions. Heuristic effectiveness depends on proper calibration.
APF	Simple and extremely fast to implement. Useful for scenarios with few obstacles and minimal constraints.	Inadequate for complex terrains or significant obstacles. Ignores physical terrain constraints and realistic penalties.
RRT*	Asymptotically optimal solution with increasing samples. Capable of handling high-dimensional and continuous spaces.	High computational cost. Performance may degrade in densely cluttered or narrow spaces. Requires careful parameter tuning.

	Well-suited for exploration in unknown or partially known environments.

In Table 7.7, SE1.0 emerges as the most versatile path-planning solution for complex terrains with dynamic variations, slopes, and obstacles. Unlike competing approaches, it delivers consistent performance across all test scenarios, including variable topography, unstable ground, and real rover trials, without relying on assumptions or predefined conditions. By contrast, A* is effective primarily in structured environments, APF is restricted to simple terrains, and RRT* achieves good performance but at the cost of high computational demand and sensitivity to parameter settings.

SE1.0 incorporates critical features for terrain-aware navigation, such as approach-angle and anti-rollover thresholds, enhancing stability and safety, particularly in UAV applications. **It can integrate non-geometric obstacle data from onboard sensors** (e.g., soft terrain, gas detection [332], dynamic objects), enabling real-time adaptability and greater reliability in uncertain environments. Its predictive component is aligned with the Velocity Obstacle (VO) framework [333]**,** supporting robust motion planning for nonholonomic and underactuated systems through velocity constraints and dynamic obstacle avoidance.

The motion planning strategy of SE1.0 prioritizes paths that minimize slope exposure, directly linking energy consumption to distance travelled. A "tangle removal" strategy currently under development is expected to further reduce path length and improve efficiency. Additional work focuses on optimizing both upstream and downstream paths throughout navigation, allowing dynamic adaptation to slope, obstacles, and terrain variability while maintaining computational efficiency.

Overall, SE1.0 represents a robust hybrid metaheuristic algorithm. In planetary exploration, it optimizes energy use while maintaining reliability in unpredictable terrains with limited prior information. Furthermore, it shows potential for deployment in environments affected by electromagnetic interference or thermal hazards, such as industrial and hazardous sites, where advanced resilience is required.

Ongoing research explores the integration of population-based metaheuristic optimization algorithms (MAs) [12], [85], [88], [126], [146], [166], [174], [207], [334], including Grey Wolf Optimizer, JAYA, Particle Swarm Optimization, Genetic Algorithms, Differential Evolution, Giant Trevally Optimizer, and Marine Predators Algorithm. These methods can help overcome local minima stagnation, foster global exploration, and refine trajectories in highly variable terrains.

To ensure enhanced robustness in real-time navigation under rapid and unpredictable changes, several strategies are under investigation:

i. Real-Time Data Integration and Adaptive Control: continuous updates from sensors (LiDAR, cameras, IMUs) refine terrain models on the fly, enabling immediate responses to evolving conditions.
ii. Predictive Modelling: machine learning and statistical methods anticipate terrain deformations or obstacle motion, allowing proactive trajectory adjustments.

iii. Rigorous Simulation Testing: diverse dynamic environments with moving obstacles, varying terrain, and sudden disturbances provide a platform for iterative validation and refinement.

Future development is oriented toward SE2.0, which will extend SE1.0 with enhanced real-time adaptability and population-based optimization strategies, ensuring higher robustness, efficiency, and mission-specific flexibility.

CHAPTER VIII: TSPGWJO: TOWARDS HYBRID INTELLIGENCE FOR OPTIMAL NAVIGATION OF AUTONOMOUS ROBOTS

Upon reaching one target, the explorer realizes the journey is never solitary. Beyond each accomplishment, countless destinations await and knowing them from the outset allows it to weave a path of harmony. Guided by intuition like a wolf sensing the pack, refined by strategies that mirror success and correct missteps, it balances exploration and exploitation, learning from each step to reveal the most efficient sequence of possibilities, and returning to its origin enriched by the entirety of the voyage.

8.1 MOTIVATIONS AND BACKGROUND FOR HYBRID METAHEURISTIC APPROACHES TO THE TSP UNDER KINEMATIC CONSTRAINTS

In precision agriculture, autonomous field robots must navigate highly unstructured and heterogeneous environments characterized by spatially dispersed and irregularly clustered weed infestations. The routing challenge differs markedly from conventional formulations, as the target locations are grouped into clusters with inter-point distances often smaller than the vehicle's minimum turning radius. This spatial configuration significantly intensifies the complexity of path planning, demanding strategies capable of coordinating both inter-cluster transitions and intra-cluster manoeuvres. Moreover, effective operation under these conditions must ensure strict compliance with the vehicle's kinematic and manoeuvrability constraints, thereby linking high-level route optimization with dynamically feasible motion execution.

Traditional formulations of the Travelling Salesman Problem (TSP), predominantly combinatorial in nature, often neglect the vehicle's dynamic and nonholonomic limitations, producing solutions that are mathematically optimal yet physically infeasible. To bridge this gap, this chapter presents a comparative investigation of two complementary approaches tailored for kinematically constrained routing in autonomous agricultural robotics.

The first method adopts a classical cascade framework, wherein a discrete TSP solution, derived via the JAYA metaheuristic, is refined through the Simple Hybrid Grey Wolf-JAYA Algorithm (SHGWJA) [170]. This two-stage pipeline integrates kinematic feasibility in the continuous domain, transforming the optimal visit sequence into a dynamically admissible trajectory.

The second method, termed the Travelling Salesman Problem Wolf JAYA Optimizer (TSPWJO), reformulates the classical TSP within a hybrid metaheuristic-probabilistic framework. It introduces two core innovations:

- Mixed-population strategy: a coupling of discrete and continuous optimization processes that enables a smooth transition from combinatorial sequencing to probabilistic modelling. This mechanism enhances global exploration, local exploitation, and population diversity while accommodating uncertainty and spatial variability [335], [336], [337].
- Intermediate-point spline strategy: a dynamic trajectory-smoothing process based on cubic spline interpolation [170], enriched by adaptive intermediate-point insertion. This guarantees curvature continuity, compliance with turning-radius limits, and smooth transitions between waypoints [338].

Together, these mechanisms establish a unified routing and motion-optimization paradigm, seamlessly integrating combinatorial planning with kinematic realism. Simulation experiments on benchmark agricultural scenarios confirm that TSPWJO significantly outperforms classical and hybrid cascade methods in terms of convergence speed, trajectory smoothness, and kinematic feasibility [335], [338], [339], [340], [341], [342].

The remainder of this paper is structured as follows. Section 8.2 reviews conventional metaheuristic approaches to the TSP and their known limitations in constrained domains with

a kinematic reformulation of the TSP grounded in the simplified bicycle model, explicitly incorporating nonholonomic constraints. Section 8.3 presents both the cascade framework and the proposed TSPWJO algorithm. Section 8.4 reports simulation results and comparative evaluations. Section 8.5 discusses algorithmic robustness, adaptability, and scalability. Finally, Section 8.6 concludes the study and outlines future research directions in multi-agent cooperation, dynamic target adaptation, and integration with low-level motion controllers.

By embedding vehicle kinematics directly into the combinatorial core of the TSP, this work advances a novel paradigm in autonomous robotic path planning, one that reconciles theoretical optimality with the practical demands of real-world agricultural operations.

8.2 CLASSICAL AND NOVEL METAHEURISTICS FOR THE TRAVELLING SALESMAN PROBLEM UNDER KINEMATIC CONSTRAINTS

The Traveling Salesman Problem (TSP) is a canonical NP-hard combinatorial optimization problem that aims to determine the shortest Hamiltonian cycle visiting each node exactly once before returning to the origin. A broad range of classical metaheuristics has been developed to tackle the TSP, including Genetic Algorithms (GA) [266], Simulated Annealing (SA) [267], Tabu Search (TS) [268], and Ant Colony Optimization (ACO) [269]. These approaches differ in their balance between exploration and exploitation, leveraging mechanisms such as adaptive neighbourhoods, probabilistic acceptance, and decentralized decision-making. Among them, the Lin-Kernighan heuristic [270], based on variable-depth local search, remains one of the most effective for symmetric TSPs, dynamically adjusting edge exchanges to enhance convergence and maintain solution diversity. Similarly, Tabu Search exploits adaptive memory structures to avoid cycling and escape local optima, proving effective in rugged search landscapes.

In contrast to static formulations, dynamic TSP (DTSP) models have been proposed to better suit autonomous robotic contexts, where operational conditions evolve over time [6, 15]. These models incorporate real-time environmental variations—such as dynamic obstacles, mission updates, and terrain changes—thus requiring continuous route re-optimization through adaptive planning mechanisms.

A notable advancement was presented by Str et al.[343], who introduced a Discrete Particle Swarm Optimization (DPSO) algorithm employing heterogeneous parameter configurations to solve the DTSP. In this work, the DTSP was decomposed into successive TSP sub-problems, where pheromone-based information from prior solutions was retained to guide subsequent optimization phases. The algorithm features an automatic parameter tuning scheme, enabling self-adaptive behaviour independent of problem size or computational time. The use of non-uniform parameter values across particles increases population entropy, improving robustness and exploratory performance. Comparative analyses demonstrated that heterogeneous-parameter DPSO surpasses the standard DPSO and performs comparably to advanced ACO variants, though its high computational demand limits its suitability for real-time applications.

Given the exponential growth of computational complexity with the number of nodes, exact TSP solvers become impractical for large instances. Consequently, alternative paradigms have

emerged. Zhang et al. [272] introduced a deep reinforcement learning (DRL) framework capable of producing near-optimal solutions with markedly reduced computational cost. DRL-based methods have successfully addressed DTSP instances involving up to 40 customers among more than 100 locations, generating solutions in near real time. Parallelization further enhances computational efficiency: Wu et al. [273] demonstrated GPU-based implementations of DPSO, reducing runtime without hardware modifications and enabling scalable, real-time execution.

In applied contexts such as autonomous agricultural weeding [274], robotic platforms must navigate complex environments and perform precise, controlled path reversals [275, 276]. Vehicle navigation kinematics depend strongly on wheelbase configuration and wheel type [277], influencing steering angle, turning radius, and stability. These effects become pronounced when transitioning from simplified kinematic models (e.g., unicycle [278], bicycle, or differential drive) to more realistic, constrained geometries such as Ackermann steering [279] (Figure 8.1). Systems with fewer wheels (unicycles, bicycles) require active stabilization and exhibit instability at low speeds due to diminished gyroscopic effects and reduced self-stabilizing dynamics. In contrast, multi-wheeled platforms, particularly four- or six-wheeled, offer superior stability and adaptability across varied terrains. The increased number of contact points improves load distribution and reduces tipping risk, while enabling advanced suspension architectures that enhance traction and manoeuvrability on uneven surfaces.

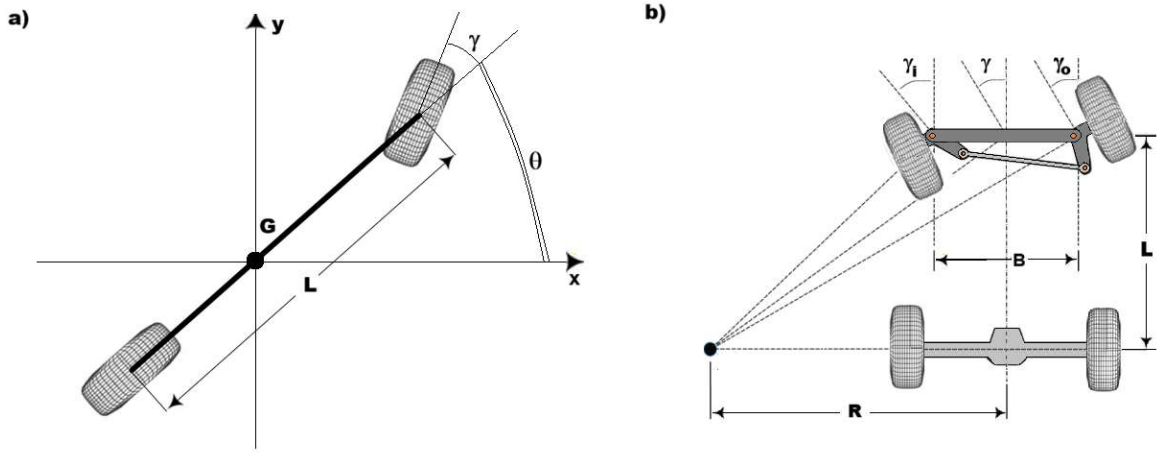


Figure 8.2. a) BICYCLE GEOMETRY; b) ACKERMANN STEERING GEOMETRY.

To explore the impact of kinematic constraints on trajectory planning, a simplified bicycle kinematic model is employed, as shown in Figure 8.2. This model adheres to a predefined trajectory [344], whose optimization is achieved through the modern hybrid metaheuristic algorithm *Simple Hybrid Grey Wolf–JAYA Optimizer* [170].

Let x and y represent the position of the centre of mass, v the linear velocity, and γ the steering angle. The system's motion dynamics are described by the classical differential equation governing uniform motion.

$$\dot{x} = v \cdot cos\gamma \tag{8.1}$$
$$\dot{y} = v \cdot sin\gamma \tag{8.2}$$

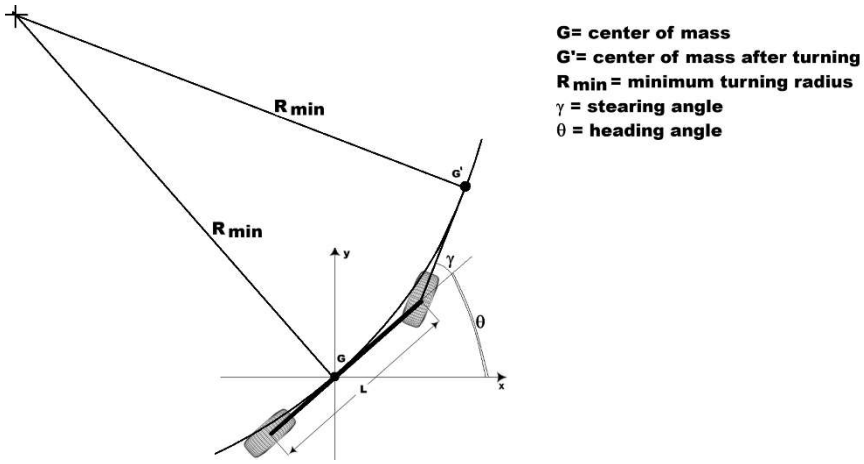


Figure 7.3. SIMPLIFIED KINEMATIC MODEL OF A BICYCLE-TYPE VEHICLE.

The steering angle γ denotes the deviation between the front wheel direction and the vehicle's longitudinal axis; larger values yield smaller turning radius. The minimum turning radius R_{min} defines a primary kinematic constraint, representing the smallest curvature permitted by the vehicle's geometry.

In any turning manoeuvre, the trajectory radius must be no less than the minimum turning radius, which can be expressed mathematically as:

$$R \geq R_{min} \tag{8.3}$$

where $R_{min} = \frac{L}{\tan \gamma_{max}}$ and L is the vehicle's wheelbase (the distance between the front and the rear axles).

The heading angle θ defines the vehicle's orientation relative to the global x-axis, while the steering angle γ controls front-wheel rotation and local path curvature. Their relationship in a bicycle model is given by the differential equation:

$$\dot{\theta} = \frac{v}{L} \cdot \tan \gamma \tag{8.4}$$

For a discretized trajectory (x, y), the corresponding kinematic variables for nonholonomic vehicle control are computed as follows:

$$\theta_i = atan\frac{y_{i+1} - y_i}{x_{i+1} - x_i} \tag{8.5}$$

$$v_i = \frac{\sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}}{\Delta t} \tag{8.6}$$

$$\gamma_i = \text{atan}\frac{L \cdot (\theta_{i+1} - \theta_i)}{v_i \cdot \Delta t} \tag{8.7}$$

To ensure accurate path tracking in dynamic or constrained environments, angular velocities can be calculated using a dedicated control function, as follows:

$$\omega_i = \frac{\theta_{i+1} - \theta_i}{\Delta t} \tag{8.8}$$

The autonomous rover can reverse without changing orientation, with linear velocity dynamically adjusted based on the heading error between its current orientation θ and the desired heading θ_d [267].

If the absolute heading error exceeds $\frac{\pi}{2}$, that means $|\theta_d - \theta| > \frac{\pi}{2}$, indicating the target is behind the vehicle, the control logic reverses the motion direction as follows:

$$\theta_d \leftarrow normalizeAngle(\theta_d + \pi) \tag{8.9}$$

$$v = -v_{max} \cdot \left(1 - 0.5 \cdot \frac{|\theta_d - \theta|}{\pi}\right) \tag{8.10}$$

$$\omega = sat(\omega_{MAX} \cdot normalizeAngle(\theta_d - \theta)) \tag{8.11}$$

To constrain the vehicle's motion, the function $sat(\cdot)$ limits values to $[-\omega_{MAX}, \omega_{MAX}]$ [345], while $normalizeAngle(\cdot)$ [346] keeps angles within $[-\pi, \pi]$. In Equation 8.9 the factor 0.5 balances reactivity and stability during reversals; higher values (0.7-1) more strongly reduce backward speed when heading errors are large, improving stability and smoothing sharp manoeuvres.

8.3. TSPWJO: A HYBRID MULTI-STRATEGY ALGORITHM FOR TSP USING MIXED POPULATIONS AND INTERMEDIATE-POINT STRATEGIES

A common approach to solving the TSP, shown in Figure 8.3, is a two-stage cascade: first, the optimal sequence of target locations is determined, then a path-planning algorithm connects them while satisfying the vehicle's kinematic or dynamic constraints [347], [348], [349], [350], [351], [352].

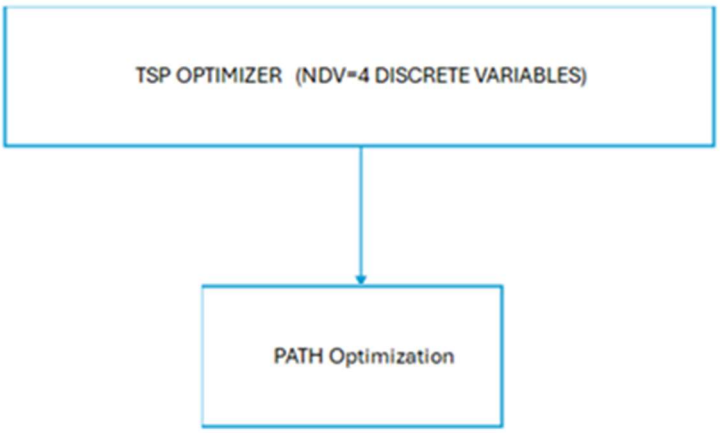


Figure 7.4. CASCADE RESOLUTION OF THE TSP FOLLOWED BY TRAJECTORY OPTIMIZATION FOR 4 LOCATIONS.

Practical constraints such as limited turning radius and bounded acceleration are not considered in standard TSP formulations, yet they critically impact manoeuvrability in real-world scenarios. These kinematic limitations increase computational complexity, particularly when metaheuristics must jointly optimize target sequencing and path feasibility. To address this, the Simple Hybrid Grey Wolf-Jaya Algorithm (SHGWJA) [170] is adopted for its robust performance across diverse optimization problems and its balanced exploration-exploitation capabilities, which ensure effective global search and local refinement. Its simple structure also facilitates performance evaluation and hyperparameter sensitivity analysis.

The original SHGWJA is enhanced with two key strategies: a mixed-population approach to improve exploration, and a combination of cubic spline interpolation with an intermediate-point strategy [276] to refine continuous paths and better satisfy vehicle kinematics. These improvements form the Traveling Salesman Problem Wolf-Jaya Optimizer (TSPWJO), an effective algorithm for kinematically constrained TSP variants.

In this approach, the TSP is transformed from a strictly combinatorial problem into a probabilistic one. The number of decision variables (NDV) is defined as NDV $= N + N \cdot p \cdot c$, where N is the number of locations, p is the number of intermediate points between consecutive locations, and c is the dimensionality of each point. For 2D terrestrial vehicles. $c = 2$ (coordinates x and y). For instance, with $N = 4$ locations each augmented with $p = 2$ intermediate points, NDV becomes NDV $= 4 + 4 \cdot 2 \cdot 2 = 20$ as shown in Figure 8.4.



Figure 8.5. TSPWJO WITH MIXED POPULATION STRATEGY FOR 4 LOCATIONS AND 2 INTERMEDIATE POINTS FOR EACH SEQUENCE.

The motivation for this approach arises from the inherent intractability of finding exact TSP solutions as the number of locations increases. The total number of possible sequences grows

factorially with the number of locations ($N_{\text{location}}!$), making exhaustive search computationally infeasible even for moderate instances. For example, 10 locations yield over 3.6×10^6routes, while 100 locations exceed 10^{157}. This combinatorial explosion highlights the limitations of exact methods and motivates the use of stochastic or metaheuristic algorithms for large-scale TSPs.

The TSPWJO algorithm manages these challenges through a mixed-population strategy, partitioning decision variables into discrete and continuous domains. The first Ndiscrete variables encode the visiting sequence of target locations, ensuring each is visited once, with the starting point either user-defined or randomly selected. Continuous variables correspond to intermediate points inserted between locations [260], allowing cubic splines to generate smooth paths while respecting the vehicle's minimum turning radius ($R_{\min}$).

The robot's motion is thus represented by a piecewise cubic polynomial of a spatial parameter s(e.g., arc length):

$$f_i(s) = a_3 \cdot s^3 + a_2 \cdot s^2 + a_1 \cdot s + a_0, s_i \leq s \leq s_{i+1} \tag{8.12}$$

Cubic splines are preferred due to their mathematical properties: unlike high-degree polynomials, they avoid instability and oscillations (Runge's phenomenon) [276], and unlike unconstrained B-splines, they explicitly pass through specified points [277].

Abrupt changes in direction, especially involving forward and reverse motion, can introduce discontinuities in the velocity $f_i'(s)$. Using a single global polynomial for multiple points may also cause oscillations. To mitigate this, the trajectory is segmented, with each interval represented by a separate cubic spline. For forward-only motion, segments span the entire path through all waypoints and intermediate points. When reverse motion is possible, segments are defined between successive locations and their associated intermediate points, as illustrated in Figure 8.5.

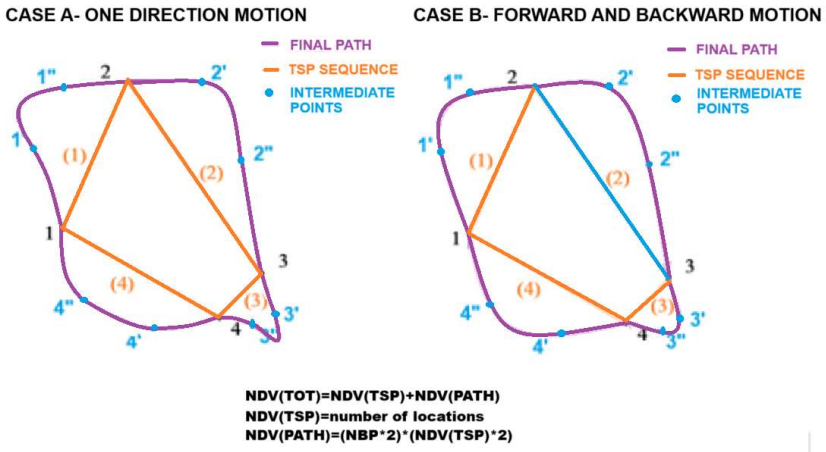


Figure 8.6. TRAJECTORY REPRESENTATION USING CUBIC SPLINES WITH INTERMEDIATE-POINTS STRATEGY FOR MONODIRECTIONAL OR BIDIRECTIONAL VEHICLES WITHOUT KINEMATIC CONSTRAINTS.

In scenarios allowing bidirectional motion, the velocity $f_i'(t)$ may be discontinuous, enabling the robot to stop and immediately reverse direction while maintaining smooth translation. Mathematically, position continuity is preserved:

$$s = \lim_{t \to t_i^-} f(t) = \lim_{t \to t_i^+} f(t) \tag{8.13}$$

while velocity discontinuity occurs:

$$\lim_{t \to t_i^-} f'(t) \neq \lim_{t \to t_i^+} f'(t) \tag{8.14}$$

allowing instantaneous stops and reversals.

For forward-only vehicles, the robot must rotate in place to align with the next location, making heading continuity critical. Sharp orientation changes cannot be mitigated by reversing, potentially increasing transition times and manoeuvring complexity, especially in cluttered or narrow environments.

Velocity discontinuities correspond to theoretically infinite acceleration:

$$a = \lim_{t \to t_i^-} f''(t) \to \infty \quad or \quad a = \lim_{t \to t_i^+} f''(t) \to \infty \tag{8.15}$$

which imposes extreme mechanical stress in real systems due to motor limits, material fatigue, and wheel slip [278].

Various initialization strategies have been explored to define the search space for intermediate points between target locations (Figure 8.6). Using a single intermediate point proved to be the most effective strategy, yielding faster convergence and more consistent, smoother trajectories.

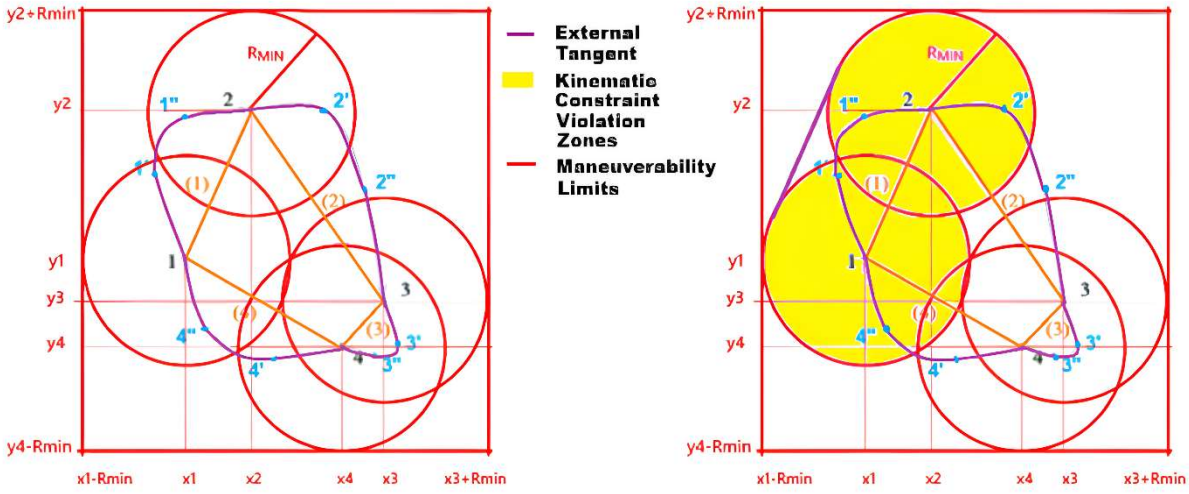


Figure 8.7. PRELIMINARY ANALYSIS OF KINEMATIC LIMITATIONS AND SEARCH SPACE STRATEGIES FOR TRAJECTORY OPTIMIZATION

The first strategy, Turning Radius-Based Initialization (TRBI), sets bounds for each variable based on the points' minima and maxima, expanded by $R_{\min}$to ensure feasible path curvature:

$$lb_x = min(x) - R_{min} \quad (8.16.a)$$
$$lb_y = min(y) - R_{min} \quad (8.16.b)$$
$$ub_x = max(x) + R_{min} \quad (8.16.c)$$
$$ub_y = max(y) + R_{min} \quad (8.16.d)$$

These bounds confine intermediate points to a feasible region, respecting the vehicle's minimum turning radius and ensuring physically achievable trajectories.

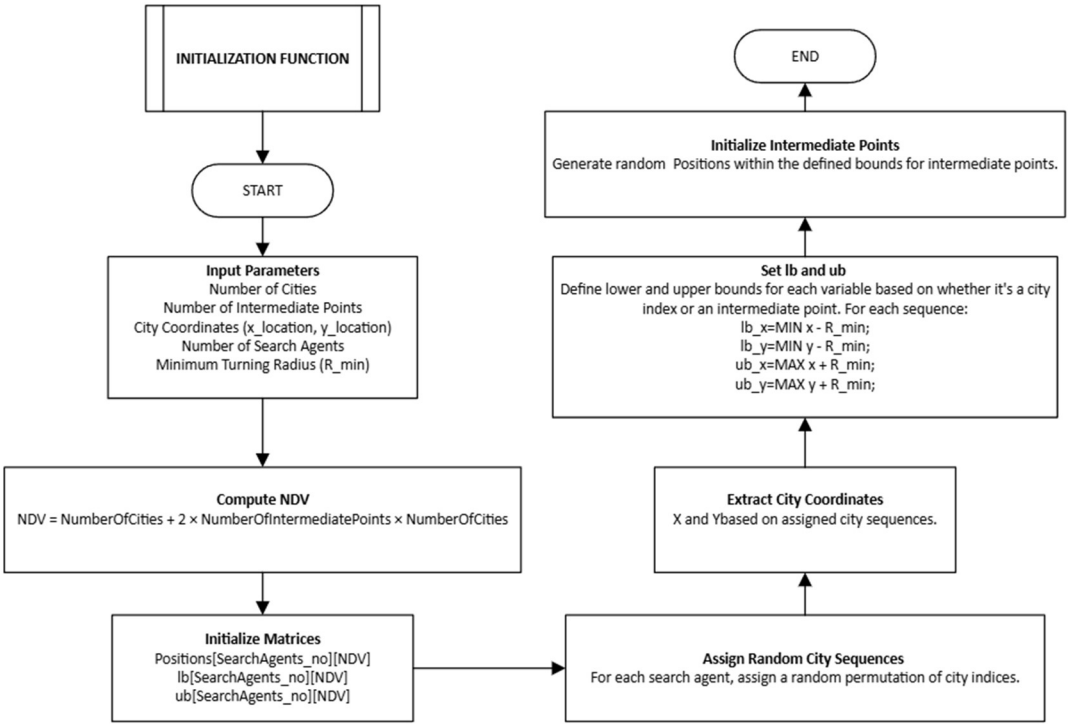


Figure 8.8. FLOW CHART OF THE TURNING RADIUS-BASED INITIALIZATION (TRBI) STRATEGY

The second strategy, Tangent-Guided Intermediate Initialization (TGII), differs from TRBI in how the lower and upper bounds (lb and ub) are defined. TGII employs a geometry-driven approach, based on tangents to circles of radius $R_{\min}$ centered at waypoints, which steer the optimization toward kinematically feasible and manoeuvrable trajectories. These tangents define preferred directions that satisfy the vehicle's minimum curvature constraints, so bounds are guided by the problem's geometry rather than assigned arbitrarily. Figure 8.7 presents the flowchart of the TRBI strategy. The goal is to bias the initial population toward feasible and promising regions of the search space from the start, improving convergence speed and increasing the likelihood of generating solutions compatible with the vehicle's motion constraints. Formally, let N be the number of cities, IP the number of intermediate points per sequence, $R_{\min}$ the minimum turning radius, and $(x_i, y_i) \in \mathbb{R}^2$the coordinates of the i-th city.

Let $T_1^k, T_2^k \in \mathbb{R}^2$ be external tangent points between cities k and k+1 (modulo N), $j \in \{1, \dots, M\}$ the index of intermediate points.

For each search agent $a \in \{1, \dots, A\}$and discrete variable $d \in \{1, \dots, N\}$, the boundaries are defined as:

$$lb_a[d] = 1 \quad (8.17.a)$$
$$ub_a[d] = N \quad (8.17.b)$$

Next, the continuous variable bounds are then computed for each segment $k \in \{1, \dots, N\}$ with M intermediate points, yielding $2 \cdot N \cdot M$ variables. For each point j on segment k, the limits are defined as:

$$lb_a[x_{k,j}] = \min\left(T_{1,x}^{(k)}, T_{2,x}^{(k)}\right) \quad (8.18)$$
$$ub_a[x_{k,j}] = \max\left(T_{1,x}^{(k)}, T_{2,x}^{(k)}\right) \quad (8.19)$$
$$lb_a[y_{k,j}] = \min\left(T_{1,y}^{(k)}, T_{2,y}^{(k)}\right) \quad (8.20)$$
$$ub_a[y_{k,j}] = \max\left(T_{1,y}^{(k)}, T_{2,y}^{(k)}\right) \quad (8.21)$$

If the tangent points are undefined (i.e., the distance between cities $D_k < 2 \cdot R_{min}$ then:

$$T_1^{(k)} = (x_k, y_k) \quad (8.22)$$

$$T_2^k = (x_{k+1}, y_{k+1}) \quad (8.23)$$

and the boundaries are set using the same logic.

Figure 8.9 shows the geometric definition of the search space bounds for a single intermediate point between two consecutive waypoints, based on the TRBI and TGII initialization strategies.

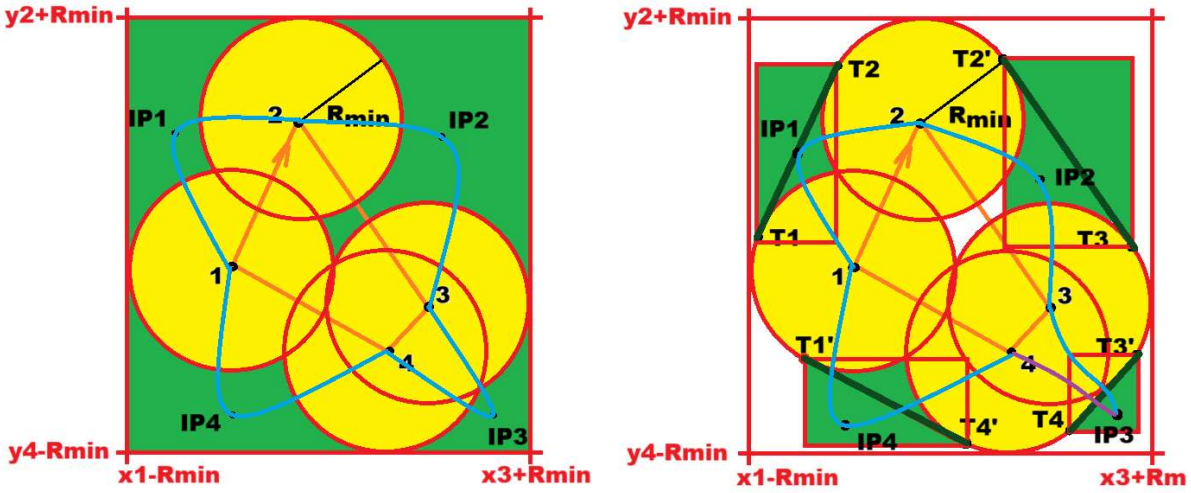


Figure 8.9. GEOMETRIC DEFINITION OF FEASIBLE SEARCH SPACE USING 1 INTERMEDIATE POINT VIA: a) TURNING RADIUS-BASED INITIALIZATION (TRBI) ; b) TANGENT-GUIDED INTERMEDIATE INITIALIZATION (TGII).

Figure *8.10* shows the flowchart of the TGII strategy.

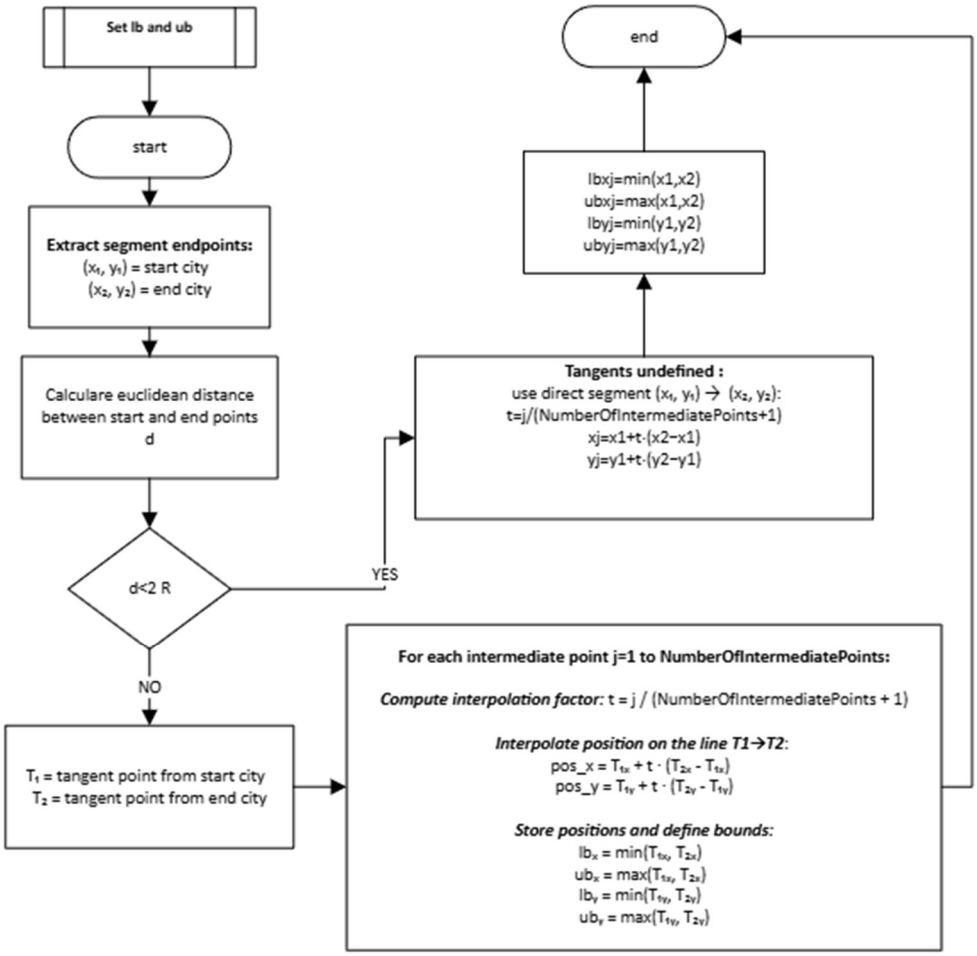


Figure 8.10. FLOW CHART OF THE TGII STRATEGY.

The TSPWJO framework integrates two complementary paradigms. First, the Grey Wolf Optimizer (GWO) identifies a reference point in the solution space, derived from the cooperative dynamics of α, β, and δ wolves. This point acts as a virtual hub that organizes the search. Next, the JAYA algorithm refines exploration, pulling solutions toward the best and pushing them away from the worst, without relying on user-defined parameters.

Stopping criteria are adapted to the sensitivity of the decision variables. For highly sensitive cases, stricter thresholds are applied to ensure stability and meaningful convergence [170].

Additional mechanisms reinforce this hybrid approach: mixed discrete–continuous populations, elitist selection, cost filtering, mirroring for diversity [170], and spline-based constraints to ensure feasibility. In this scheme, GWO plays a formative "Big Bang" role [279], while JAYA drives the "Big Crunch" convergence [279].

Both TRBI and TGII confirm that initialization detail and update mechanisms are decisive for efficient optimization.

8.4. COST FUNCTION AND CONSTRAINTS IN TSPWJO

In conventional TSP formulations, the distance matrix is typically defined by pairwise Euclidean distances between locations. Although computationally straightforward, this approach neglects vehicle-specific kinematic and curvature constraints.

The TSPWJO framework instead embeds trajectory optimization within the construction of cubic splines incorporating intermediate control points. These splines are generated under boundary conditions that enforce the vehicle's minimum turning radius, with steering limits expressed via a curvature constraint. Candidate control points are filtered by geometric feasibility, ensuring trajectories remain continuous, smooth, and kinematically admissible.

Once trajectory-based costs populate the distance matrix, continuity between adjacent path segments is imposed by matching terminal and initial states (position and orientation). This coupling enables the joint optimization of discrete sequencing and continuous trajectory generation within a unified formulation.

Vehicle motion capabilities further influence spline structure. For forward-only motion, the spline must interpolate all locations smoothly while maintaining curvature-limited derivative continuity (Fig. 8.10.a). For bidirectional motion, locally optimal splines are generated for each waypoint pair, allowing direction reversals while preserving curvature and derivative continuity (Fig. 8.10.b).

In both cases, the spline framework enforces vehicle-specific motion limits and integrates seamlessly with population-based optimization methods. The principal distinction lies in the admissible trajectory geometry and turning behaviour, which directly affect the optimization search space and convergence dynamics.

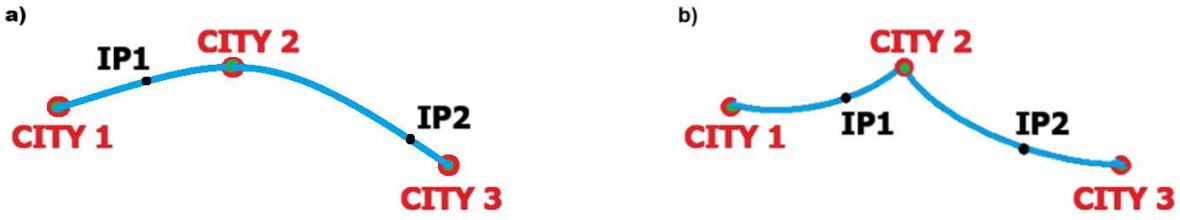


Figure 8.11. ILLUSTRATION OF CUBIC SPLINE INTERPOLATION FOR VEHICLE TRAJECTORY PLANNING UNDER DIFFERENT MOTION CONSTRAINTS: a) SPLINE WITH UNIDIRECTIONAL (FORWARD-ONLY) MOVEMENT; b) SPLINE WITH BIDIRECTIONAL MOVEMENT ALLOWING BOTH FORWARD AND REVERSE MOTIONS.

The cost function introduces two defining parameters, among which the NBP (Number of Breaking Points) quantifies the control points determining the spline segment between consecutive waypoints.

$$NBP = 2 \cdot NumberOfIntermediatePoints + (NumberOfCities - 1) \tag{8.24}$$

The number of intermediate points and cities is defined according to the vehicle's motion constraints, which may differ under unidirectional or bidirectional operation. The Number of Points in the Trajectory (NPNT) denotes the discretization level of the interpolated spline between two consecutive waypoints. In the initial configuration, NPNT $= 100$. Increasing this value improves trajectory accuracy at the cost of higher computational effort.

Once the NBP and NPNT are established, the TSPWJO algorithm randomly samples candidate configurations to identify the optimal spline structure. The optimization is subject to a closed-path constraint, ensuring the final point coincides with the initial one.

Continuity of the first and second derivatives is imposed at all spline junctions, as expressed by the following conditions:

for i = 0, ..., NDV-1

$$x'_i = x'_{i+1} \tag{8.25.a}$$

$$y'_i = y'_{i+1}$$

$$x''_i = x''_{i+1} \tag{8.25.b}$$

$$y''_i = y''_{i+1}$$

In these formulas the Number of Decision Variables (NDV) represents the continuous parameters defining the spline-based trajectory. These variables correspond to the coordinates of intermediate control points that determine the path's geometry and smoothness. Continuity of the first and second derivatives is enforced along the entire route, except at the point where the trajectory closes at the starting city.

The objective function incorporates a penalty when the local curvature radius R drops below a specified minimum threshold $R_{\min}$. For a planar trajectory described by the parametric curve $r(t) = \big(x(t), y(t)\big)$, the curvature $\kappa(t)$is calculated as:

$$\kappa(t) = \frac{\dot{x(t)}\ddot{y}(t) - \dot{y(t)}\ddot{x(t)}}{\big(\dot{x}^2(t) + \dot{y}^2(t)\big)^{\frac{3}{2}}} \tag{8.26}$$

The corresponding curvature radius $R(t)$ is given by:

$$R(t) = \frac{1}{|\kappa(t)|} \tag{8.27}$$

When $R(t) < R_{min}$ a penalty term is introduced in the fitness evaluation to discourage the generation of trajectories that violate the vehicle's kinematic constraints, ensuring that the optimized paths remain physically realizable.

To construct a robust constraint function, several preliminary considerations were addressed. Excessive steering angles can cause traction loss, uneven tire wear, and reduced manoeuvrability in confined environments. When steering exceeds feasible limits, the vehicle may experience understeer, where the front wheels fail to follow the commanded path, or oversteer, where the rear wheels lose grip, potentially resulting in a spin. Moreover, large steering angles accelerate tire degradation thereby compromising both safety and efficiency. Restricting the steering angle thus enhances controllability and enables safe operation in constrained spaces such as parking areas or narrow urban roads.

Building on these principles, the CONSTRAINTS function evaluates and aggregates a set of nonlinear constraints applied to the parametric spline trajectory defined by the control points $x_{bp}$and y_{bp}. These constraints ensure kinematic feasibility, trajectory smoothness, and, when required, obstacle avoidance. Specifically, the following conditions are imposed:

1) Proportionality Constraints Between Consecutive Control Points: to ensure shape consistency, the function enforces proportionality between pairs of x and y coordinates for consecutive decision variables:

$$for\ k = NumberOfCities + 1, \dots, NDV$$

$$G_i = \frac{x_k}{x_{k+2}} - 1 \leq 0 \tag{8.28.a}$$

$$G_{i+1} = \frac{y_k}{y_{k+2}} - 1 \leq 0 \tag{8.28.b}$$

2) Constant Velocity Constraint: assuming unit parametrization of the spline by x, the function imposes that the instantaneous velocity magnitude along the spline trajectory remains close to a constant target value $v_{constant} = 2\frac{m}{s}$:

$$G_i = \frac{|v(x)|}{v_{constant}} - 1 \sim 0 \tag{8.29}$$

3) Maximum velocity constraint: to prevent excessive speeds along the trajectory, the velocity magnitude must not exceed a maximum threshold $v_{max} = 2.5$ m/s:

$$G_i = \frac{|v(x)|}{v_{max}} - 1 \leq 0 \tag{8.30}$$

4) Minimum Curvature Radius Constraint: to ensure smoothness and avoid sharp turns, the trajectory must maintain a minimum radius of curvature, as described above. The constraint is formulated as:

$$G_i = \frac{R_{min}}{R(x)} - 1 \leq 0 \tag{8.31}$$

Then, all constraint violations are aggregated into a penalty score which represents the total violation magnitude and is intended to be minimized or driven to zero:

$$TOTVNS = \sum_{i \in \{G_i > 0\}} G_i^2 \tag{8.32}$$

8.5 COMPARATIVE PERFORMANCE ANALYSIS: CLASSICAL APPROACH VS TSPWJO

The first methodology integrates the JAYA metaheuristic algorithm for sequence optimization and the Simple Hybrid Grey Wolf and JAYA Optimizer (SHGWJO) algorithm for trajectory generation. The framework is designed for autonomous vehicle routing and accommodates both unidirectional and bidirectional motion capabilities.

Table 8.5 reports the performance of the traditional JAYA metaheuristic applied to the Traveling Salesman Problem (TSP) over 20 independent episodes (runs).

Table 8.5. PERFORMANCE OF THE TRADITIONAL JAYA TSP SOLVER ACROSS 20 EPISODES

EPISODE	SEQUENCE	PATH LENGTH [m]
1	[1 8 9 3 10 5 2 7 6 4]	389.802

2	[1 9 8 7 6 4 2 3 5 10]	333.673
3	[1 10 3 5 2 4 6 7 8 9]	333.580
4	[1 9 7 6 4 3 2 5 10 8]	390.392
5	[1 8 7 6 4 9 3 2 5 10]	382.026
6	[1 8 7 6 4 2 3 10 5 9]	331.042
7	[1 8 7 6 4 5 10 2 3 9]	378.353
8	[1 9 8 7 6 4 10 5 3 2]	383.284
9	[1 9 8 7 6 4 2 10 3 5]	356.694
10	[1 7 4 6 2 5 3 10 9 8]	372.664
11	[1 9 7 6 4 2 10 5 3 8]	361.623
12	**[1 8 7 6 4 2 5 3 10 9]**	**319.990**
13	[1 8 7 4 6 10 5 3 2 9]	381.692
14	[1 9 8 7 4 6 2 10 5 3]	376.590
15	[1 8 9 10 3 5 2 4 6 7]	344.861
16	[1 8 9 7 6 4 2 5 3 10]	339.488
17	[1 9 7 6 4 2 3 10 5 8]	361.870
18	[1 8 9 7 4 6 2 10 5 3]	382.497
19	[1 9 3 2 5 10 4 6 7 8]	371.938
20	[1 8 4 6 7 2 10 5 3 9]	382.639

The evaluation focuses on the stability, consistency, and solution quality of the JAYA algorithm under randomized initial conditions. Each episode constitutes an independent run, using a population of 30 and a maximum of 10000 iterations, applied to permutations of 10 locations with City 1 fixed as the start. Across the 20 episodes, the shortest path achieved is 319.990 m (Episode 12), whereas the longest is 390.392 m (Episode 4), resulting in a mean path length of approximately 363.735 m with a standard deviation of 21.75 m. Several episodes (e.g., 2, 3, 6, 15, 16) yield sub-340 m paths, demonstrating the algorithm's capacity to consistently identify high-quality solutions, while others (e.g., 1, 4) exceed 389 m, highlighting the potential for premature convergence or entrapment in local minima. The best performing sequences are generally observed within the first 20 runs.

Figure 8.11.a illustrates the optimal city sequence obtained by the classical JAYA algorithm, with each city annotated by its index and visiting order, and the trajectory displayed as a piecewise-linear path connecting the points. The resulting sequence is subsequently used as input for trajectory-level optimization. Figure 8.11.b shows the SHGWJO-optimized trajectory [170] for the JAYA-generated sequence, accommodating bidirectional motion: blue curves indicate forward segments, while magenta curves denote reverse motion.

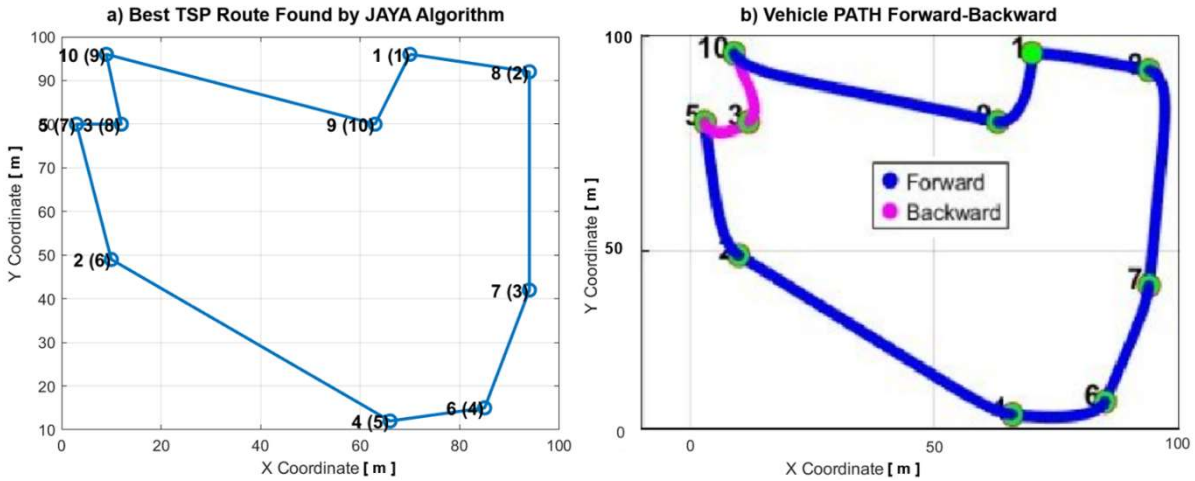


Figure 8.12. a) OPTIMAL SEQUENCING VIA JAYA METAHEURISTIC; b) BIDIRECTIONAL VEHICLE PATH PLANNING OPTIMIZATION VIA SHGWJO.

The trajectory satisfies kinematic and dynamic feasibility, including non-holonomic constraints and a minimum turning radius. Allowing reverse motion improves manoeuvrability in tight or high-curvature areas and enhances the planner's ability to generate smooth, collision-free paths, particularly for differential or omnidirectional vehicles (see Figure 8.12). The bottom-left subplot illustrates the unidirectional motion scenario, where the vehicle is constrained to forward movement. Trajectories are generated via cubic spline interpolation of the waypoints, ensuring second-order geometric continuity (C^2) to maintain smooth curvature and prevent abrupt directional changes that violate physical constraints. Within the SHGWJO framework, spline interpolation is optimized both locally and globally to balance path optimality, trajectory smoothness, and kinematic feasibility. The resulting absence of discontinuities or sharp curvature variations demonstrates the optimizer's precision and its ability to respect motion constraints without sacrificing path quality.

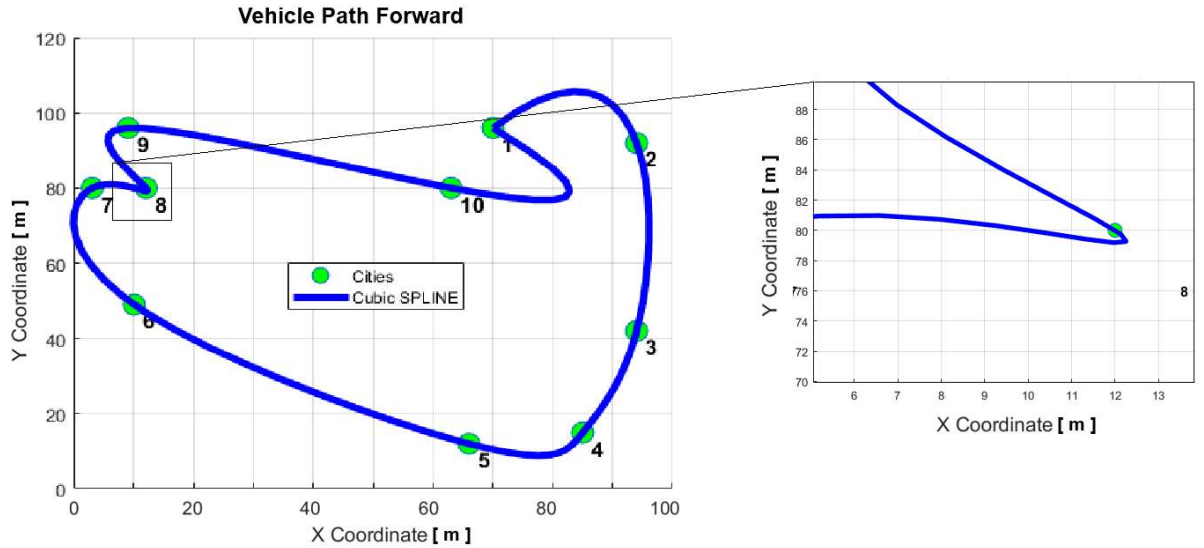


Figure 8.13. UNIDIRECTIONAL TRAJECTORY OPTIMIZATION WITH CUBIC SPLINE INTERPOLATION AND ZOOM-IN ANALYSIS IN THE CITY N. 8.

Figure 8.14 provides the kinematic output for control of a vehicle constrained to forward motion and a vehicle capable of both forward and backward motion.

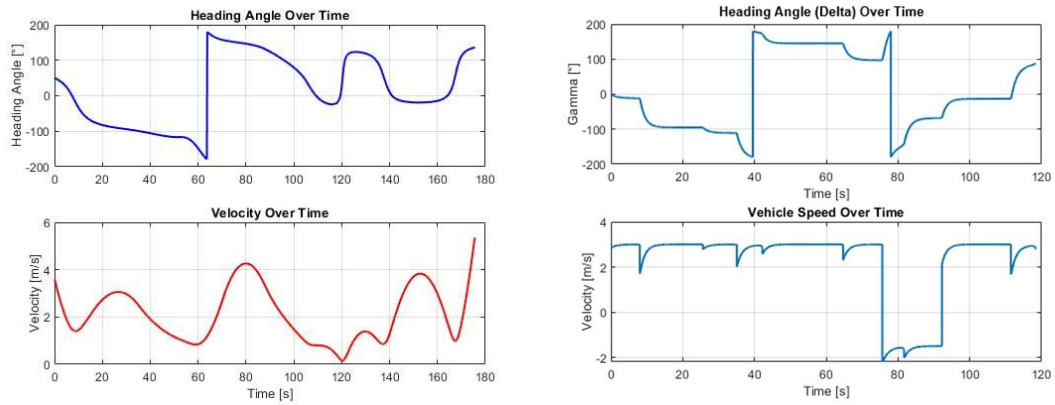


Figure 8.14. KINEMATIC RESULTS FOR A MONODIRECTIONAL VEHICLE WITH FORWARD MOTION (ON THE LEFT) AND BIDIRECTIONAL VEHICLE WITH FORWARD AND BACKWARD MOTION (ON THE RIGHT).

Heading evolution is shown in the upper plots. The unidirectional vehicle (left) exhibits continuous variations, with a rollover near 65 s from +180° to −180°. This reflects nonholonomic constraints under positive velocity, where directional changes rely on high-curvature manoeuvres. In contrast, the bidirectional vehicle (right) demonstrates smoother transitions, with controlled switches between forward and reverse motion. Apparent jumps around 40 s and 80 s result from angular rollover, confirming trajectory feasibility in both cases.

Velocity profiles appear in the lower plots. Frequent oscillations occur for the unidirectional vehicle due to repeated acceleration and deceleration at turns, highlighting inefficiencies of forward-only motion. The bidirectional vehicle maintains more stable velocities, including negative segments (e.g., 78 s and 93 s) corresponding to reverse motion. This flexibility enhances path tracking, reduces heading adjustments, and supports smoother operation compatible with model-based control.

Figure 8.14 shows the optimal vehicle trajectory under a kinematic constraint enforcing a constant forward velocity of $v = 2$ m/s, designed to satisfy the minimum turning radius throughout the cubic spline path. This ensures a continuous and smooth trajectory for a unidirectional vehicle, with a total path length of 325.00 m. Maintaining uniform velocity is crucial for motion control and time-parametrized trajectory analysis in real-time robotic navigation.

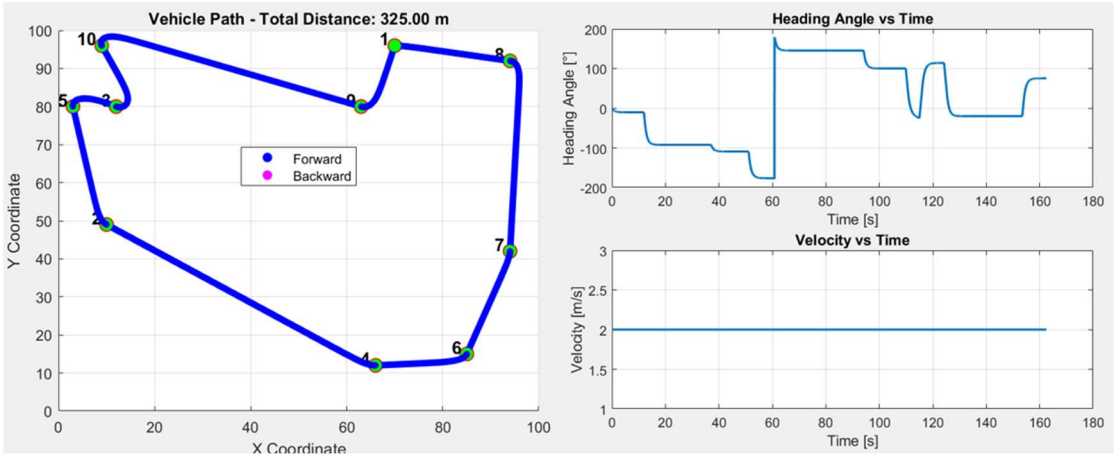


Figure 8.15. UNIDIRECTIONAL TRAJECTORY OPTIMIZATION WITH CUBIC SPLINES AT CONSTANT VELOCITY v = 2 m/s.

The top-right plot depicts heading angle evolution, revealing continuous variations with an apparent jump near City 3 due to angular rollover from +180° to −180°, rather than a physical discontinuity. At this location, a substantial steering manoeuvre is required to respect the minimum turning radius ($R_{\min} = 1.6$ m) while maintaining non-holonomic constraints and continuity of position and orientation.

Building on the literature [353], further investigations consider additional solution strategies for the TSP to preserve kinematic feasibility. These include manoeuvres such as reverse driving segments or multi-point turns, emphasizing the need to integrate trajectory planning with manoeuvre generation to achieve feasible paths in constrained environments.

The results obtained using the Turning Radius-Based Initialization (TRBI) strategy are presented in Figure 8.15.

The Turning Radius-Based Initialization (TRBI) strategy aims to generate initial path configurations that are both geometrically valid and kinematically feasible, particularly under nonholonomic constraints where the minimum turning radius governs manoeuvrability (Figure 8.14). Figure 8.15 presents the results of cubic spline interpolation: through two locations and an intermediate point (Figure 8.15.a), and through all locations with corresponding intermediate points (Figure 8.15.b). Smoothness and curvature continuity are evident in both cases, although sharp transitions appear in cornering regions, notably between Cities 1-2 and 3-4. These transitions enable the vehicle to comply with curvature constraints imposed by the minimum turning radius, highlighting potential kinematic infeasibility for systems such as autonomous ground vehicles or UAVs if such manoeuvres were not properly accounted for.

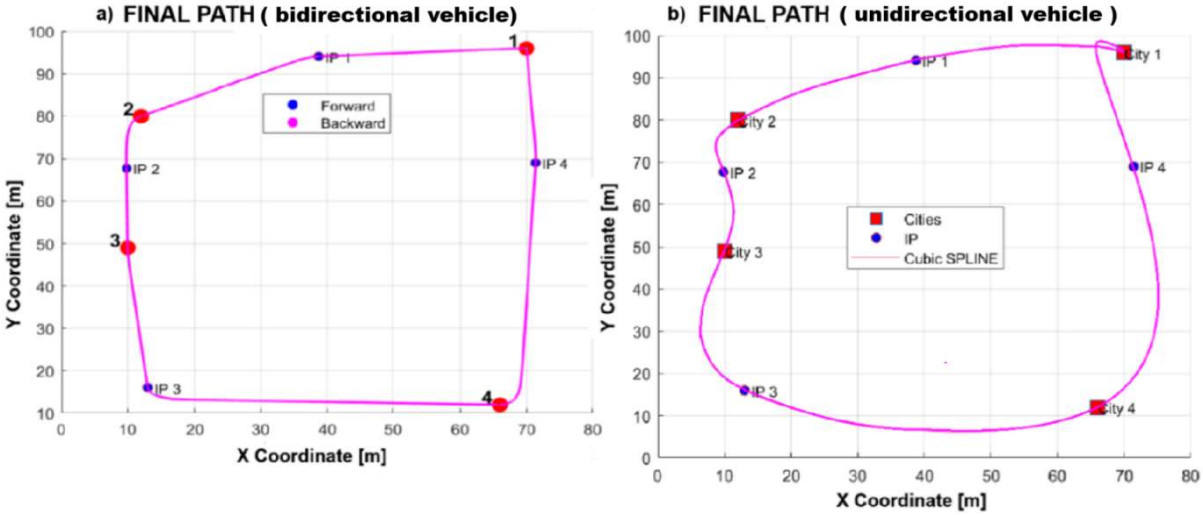


Figure 8.16. TURNING RADIUS-BASED INITIALIZATION (TRBI) FOR BIDIRECTIONAL (A) AND UNIDIRECTIONAL (B) VEHICLE PATH OPTIMIZATION.

Results from the Tangent-Guided Intermediate Initialization (TGII) strategy are shown in Figure 8.16 for both forward-backward (Figure 8.16.a) and forward-only (Figure 8.16.b) vehicle configurations. By leveraging tangent continuity at waypoints, TGII guides the placement of intermediate points, producing smoother transitions between path segments while preserving vehicle orientation, dynamic feasibility, and trajectory optimality.

A simpler alternative has also been developed, which restricts the search space to circles of radius $R_{\min}$ centered at each location, thereby reducing computational effort while maintaining compliance with minimum turning constraints.

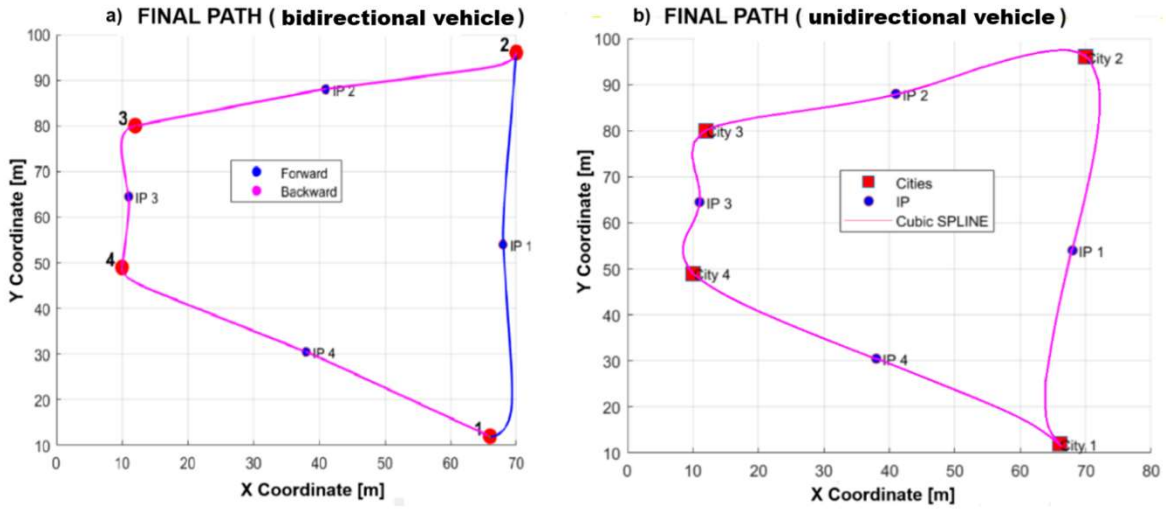


Figure 8.17. TANGENT-GUIDED INTERMEDIATE INITIALIZATION (TGII) FOR BIDIRECTIONAL (a) AND UNIDIRECTIONAL (b) VEHICLE PATH OPTIMIZATION.

The outcomes of this strategy are shown in Figure 8.17, highlighting its preference for straight-line segments over cubic splines when kinematic constraints are met. Trajectory length is markedly reduced while maintaining kinematic feasibility. Despite these benefits, the method

increases algorithmic complexity, warranting further refinement and more extensive investigation.

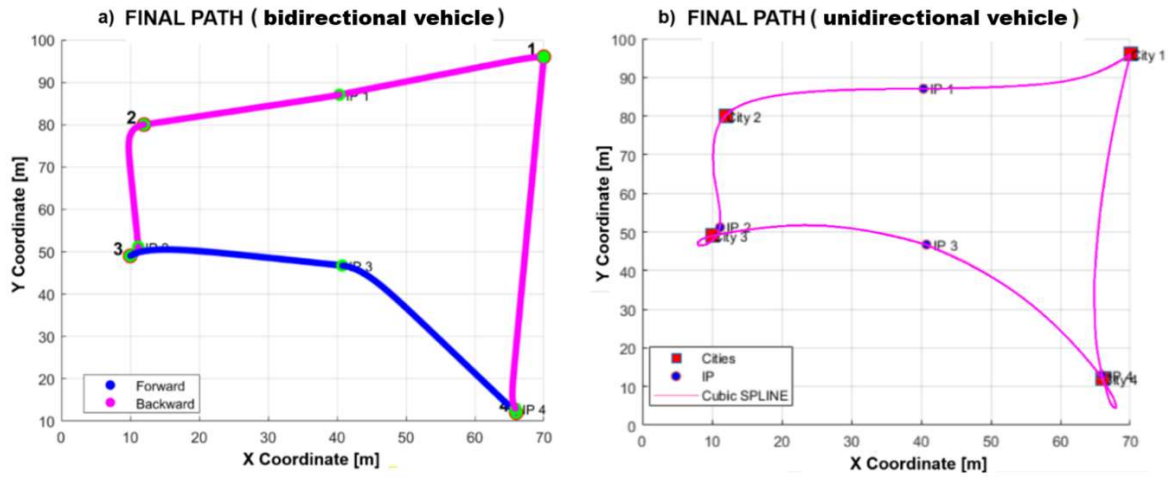


Figure 8.18. INITIALIZATION WITHOUT ZONES INSIDE CIRCLES OF R_MIN AND CENTRED IN LOCATIONS FOR BIDIRECTIONAL (a) AND UNIDIRECTIONAL (b) VEHICLE PATH OPTIMIZATION.

Considerations of the two strategies TGII and TRBI are summarized in Table .

Table 8.2. COMPARISON BETWEEN TANGENT WITH TURNING RADIUS-BASED INITIALIZATION (TRBI) STRATEGIES.

	TGII	TRBI
Initialization Principle	Tangent-aligned IP generation	Minimum curvature-constrained IP generation
Spline Type	Cubic splines guided by local tangents	Cubic splines constrained by feasible turning radius
Path Continuity	Tangent continuity prioritized	Curvature feasibility prioritized
Vehicle Model Compatibility	Emphasizes heading alignment (nonholonomic)	Emphasizes bounded curvature (kinematic)
Directional Strategy	Explicit support for both forward-only and forward-backward modes	Also supports both modes, but with turning radius as primary constraint
Flexibility in tight manoeuvres	High with forward-backward mode due to tangent reversibility	Moderate, depends on the turning radius margin
Smoothness	High (tangent-aligned transitions)	Moderate to high, depending on curvature blending

A more geometrically informed initialization is provided by the TGII strategy compared to TRBI. By placing intermediate points along feasible tangents rather than solely enforcing curvature limits, trajectories become both dynamically feasible and smoother, while better conforming to vehicle orientation constraints. This proves especially advantageous in cluttered environments or for manoeuvre-limited vehicles, where heading discontinuities and curvature violations may cause failures.

Spline interpolation is used by both TGII and TRBI; however, TGII stands out through its local tangent-aware control, enabling high-fidelity trajectories that satisfy nonholonomic constraints, particularly when bidirectional motion is permitted. Following a detailed evaluation, the TSPWJO algorithm was developed incorporating TGII for initialization and candidate solution updates.

Results obtained using the Hybrid TSPWJO with TGII are shown in Figure 8.18, with a population of 20, 100 iterations, and a penalty factor of 10^3. Optimal path lengths are 329.92 m for monodirectional and 312.90 m for bidirectional motion, achieved within only 100 iterations.

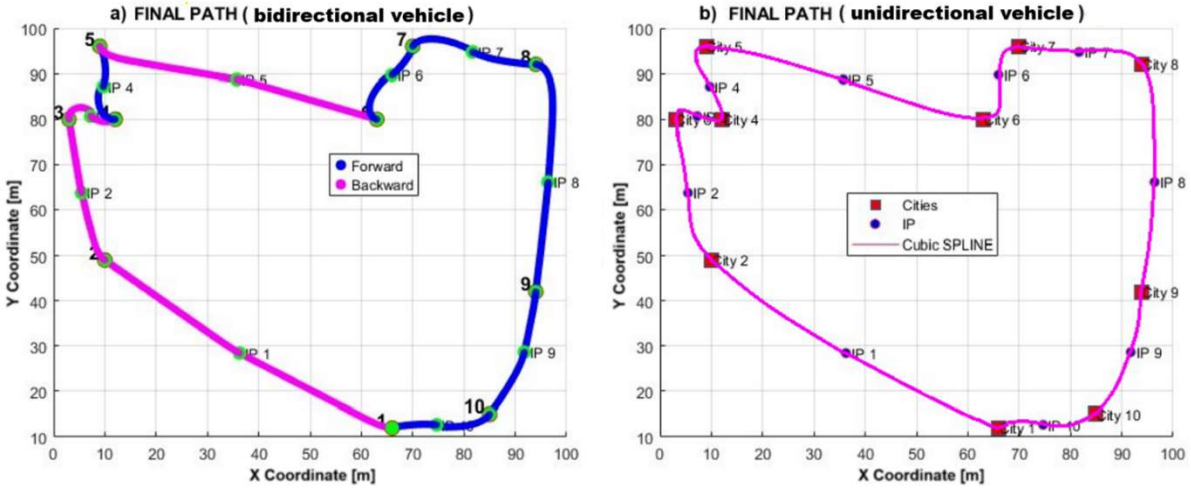


Figure 8.19. TSPWJO RESULTS WITH TGII STRATEGY FOR BIDIRECTIONAL VEHICLE (ON THE LEFT) AND UNIDIRECTIONAL VEHICLE (ON THE RIGHT) AFTER 100 ITERATIONS.

Figure 8.20 provides the TSPWJO kinematic output for control of a vehicle constrained to both forward and backward motion and a vehicle capable of only forward motion.

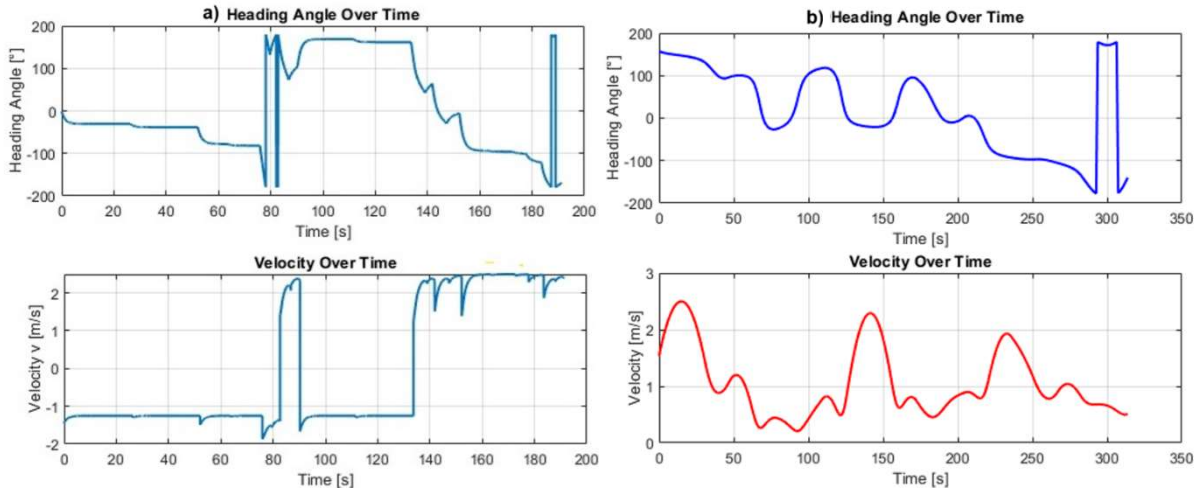


Figure 8.20. TSPWJO KINEMATIC RESULTS FOR A BIDIRECTIONAL VEHICLE WITH FORWARD AND BACKWARD MOTION (ON THE LEFT) AND MONODIRECTIONAL VEHICLE WITH FORWARD MOTION (ON THE RIGHT).

Heading angle evolution (yaw orientation) is shown in the upper subplots for two vehicle configurations: bidirectional (Figure 8.19.a) and monodirectional (Figure 8.19.b). For the monodirectional vehicle, variations appear smooth but include apparent discontinuities caused by angular rollover between +180° and −180°, rather than actual trajectory jumps. Significant reorientations occur at sharp turns or when the vehicle attempts directional changes under

positive-velocity constraints, typical of nonholonomic systems requiring high-curvature manoeuvres or near-180° rotations.

The bidirectional vehicle demonstrates a more continuous and kinematically efficient heading profile. Sharp transitions, visible around $t \approx 80$ s and $t \approx 190$ s, correspond to deliberate direction changes aligned with velocity reversals. Reverse capability reduces orientation deviations and minimizes spatial or energetic demands, offering advantages in constrained environments.

Velocity profiles are depicted in the lower subplots. The monodirectional vehicle maintains strictly positive speeds, not exceeding 2.5 m/s, with frequent oscillations including near-zero decelerations, reflecting repeated acceleration phases to respect curvature constraints and indicating reduced temporal efficiency and higher energy use. In contrast, the bidirectional vehicle maintains smoother, more symmetric velocities, including negative segments (e.g., $t \approx 70$ s and $t \approx 130$ s) representing controlled reverse motion. This flexibility reduces acceleration events, lowers control effort, and improves traversal efficiency, while enabling smoother inputs compatible with predictive control such as Model Predictive Control (MPC).

Performance results for the proposed TSPWJO solver are summarized in Table 8.3. Each of the 20 independent runs produces a visiting sequence and associated path length under Only Forward and Backward-Forward traversal modes. The sequence column lists the visiting order, while path length columns report total distance in meters, with the bidirectional mode exploiting reverse motion to shorten overall traversal compared with unidirectional movement.

Table 8.3. PERFORMANCE OF THE NOVEL TSPWJO ALGORITHM OVER 20 RUNS.

RUNS	SEQUENCE	PATH LENGTH [m] ONLY FORWARD	PATH LENGTH [m] BACKWARD-FORWARD
1	[8 1 9 3 10 2 7 6 5 4]	565.35	539.78
2	[4 2 3 10 8 9 1 7 6 5]	548.86	521.75
3	[10 5 9 1 8 7 6 4 3 2]	398.32	371.14
4	[4 10 2 3 9 7 8 1 6 5]	643.17	619.70
5	[3 5 10 9 8 7 6 1 4 2]	471.56	447.48
6	[9 10 8 4 3 2 7 1 6 5]	750.16	720.79
7	[7 6 1 9 3 2 10 8 5 4]	587.24	555.80
8	[2 5 7 4 1 9 6 8 3 10]	586.64	553.68
9	[9 2 10 3 7 6 8 1 4 5]	599.82	570.37
10	[10 5 9 1 7 6 4 8 3 2]	466.50	437.16
11	[2 5 3 10 7 6 4 9 8 1]	421.58	399.07
12	[6 4 3 10 2 5 9 1 8 7]	392.86	368.12
13	[4 9 2 10 5 3 1 8 7 6]	392.91	371.04
14	**[4 2 5 3 10 9 1 8 7 6]**	**329.92**	**312.90**
15	[2 5 1 8 9 10 3 7 4 6]	478.47	451.04
16	[6 3 2 5 10 1 9 8 4 7]	458.42	430.87
17	[6 4 3 2 5 10 9 8 1 7]	400.14	371.62
18	[1 9 3 5 10 2 7 6 4 8]	402.91	372.39
19	[4 2 5 3 10 9 1 8 6 7]	381.04	360.50

20	[4 3 2 5 10 9 1 8 7 6]	372.04	349.53

Analysis of the 20 runs shows that the Backward-Forward strategy consistently produces shorter path lengths than the Only Forward mode, highlighting the efficiency gains from incorporating bidirectional traversal. Differences in path length vary across runs, reflecting the solver's adaptive exploitation of reverse motion depending on the problem instance.

The minimal path length is achieved in Run 14, where Backward-Forward traversal results in 312.90 m, indicating an optimal or near-optimal configuration. In contrast, run 6 records the highest path cost in both modes, suggesting a more complex or less efficient node sequence.

These findings confirm the TSPWJO solver's effectiveness in generating competitive solutions across multiple runs, with consistent performance improvements observed under the Backward-Forward traversal scheme.

8.6 RESULTS DECODED AND CONCLUSIONS

This chapter investigated the integration of the classical Traveling Salesman Problem (TSP) with vehicle kinematic constraints. It addressed kinematically constrained TSPs by combining sequence optimization, trajectory planning, and hybrid metaheuristic strategies. Inspired by established models such as the Dubins TSP [276] and Reeds-Shepp TSP [277], vehicles were modelled either as forward-only or capable of bidirectional motion, capturing nonholonomic turning limitations while enabling more flexible trajectory planning, albeit with increased computational demand.

The analysis demonstrated that allowing bidirectional motion improved convergence and produced shorter, higher-quality trajectories. While traditional methods separated sequence and trajectory optimization, employing JAYA for sequencing and SHGWJO for refinement, the Traveling Salesman Problem Wolf JAYA Optimizer (TSPWJO) integrated discrete sequence variables with continuous intermediate points, generating smoother cubic spline paths and enhanced manoeuvrability.

Comparative performance analysis (Tables 8.4 and 8.5) revealed that, in unidirectional scenarios, TSPWJO reduced path length by 20% (329.92 m vs 368.49 m) and computation time by over 50%. In bidirectional motion, it achieved even shorter paths (312.90 m vs 323.34 m) with lower computational effort. Enforcing constant velocity slightly increased path lengths and computation times in the cascade approach, highlighting the influence of kinematic constraints on trajectory optimization.

Table 8.4. PERFORMANCE COMPARISON BETWEEN CASCADE-BASED JAYA+SHGWJO FRAMEWORK AND TSPWJO FOR UNIDIRECTIONAL VEHICLE MOTION (ONLY FORWARD ADVANCE).

EPISOD	BEST SEQUENCE	BEST PATH LENGHT [m] without kinematic constraints	BEST PATH LENGTH [m] with kinematic constraints	Computation time [s]

Cascade 2 stages: optimization Jaya +SHGWJO	[1 3 10 5 2 8 7 6 4 9]	319.990	368.490	18.796
TSPWJO	[4 2 5 3 10 9 1 8 7 6]	-	329.920	9.126

Table 8.5. PERFORMANCE COMPARISON BETWEEN CASCADE-BASED JAYA+SHGWJO FRAMEWORK AND TSPWJO FOR BIDIRECTIONAL VEHICLE MOTION (FORWARD-BACKWARD ADVANCE).

EPISOD	BEST SEQUENCE	BEST PATH LENGHT [m] without kinematic constraints	BEST PATH LENGTH [m] with kinematic constraints	Computation time [s]
Cascade 2 stages: optimization Jaya-SHGWJO	[1 8 7 6 4 2 5 3 10 9]	319.990	323.340 325.000(*)	30.496 30.642(*)
TSPWJO	[4 2 5 3 10 9 1 8 7 6]	-	312.90	11.884

(*) imposing v=costant

Further investigation examined the role of initialization strategies. TRBI enforced geometric turning constraints before optimization, while TGII aligned intermediate points with feasible tangents, producing smoother and kinematically compliant trajectories. These approaches narrowed the search space, improved convergence, and demonstrated that vehicle kinematics directly shaped feasible solutions.

Overall, the research confirmed that hybrid metaheuristic strategies combining combinatorial sequence optimization and continuous trajectory generation, supported by TRBI or TGII initialization, outperformed classical cascaded methods. The work produced efficient, smooth, and feasible paths for both unidirectional and bidirectional vehicles, providing clear evidence of the effectiveness of integrated optimization frameworks and highlighting avenues for future studies, including adaptive population management, multi-objective extensions, and real-time cooperative path planning with environmental interactions.

FINAL CONCLUSIONS AND FUTURE DIRECTIONS

Optimization had long propelled advancements in engineering, applied sciences, and industrial innovation, shaping performance, cutting costs, and enhancing system reliability. In complex domains, from structural design and robotics to energy management and safety-critical simulations, the ability to identify optimal solutions efficiently had proved crucial. Yet, real-world problems were often nonlinear, multi-modal, and heavily constrained, rendering classical methods insufficient. Metaheuristic algorithms had emerged as powerful, nature-inspired strategies, capable of balancing broad exploration of the solution space with intensive exploitation of promising solutions.

This thesis provided a comprehensive investigation of nature-inspired metaheuristics, their hybridizations, and their applications to challenging engineering, robotics, and industrial problems, with a particular emphasis on Industry 4.0 and the emerging field of AI-driven industrial optimization. Structured across eight chapters, the work addressed key aspects of hybrid metaheuristic design, AI-based control strategies, and advanced industrial applications, demonstrating how intelligent, adaptive algorithms had met the demands of modern, automated, and data-rich industrial environments.

Chapter I laid the theoretical and methodological foundation by presenting an in-depth study of the JAYA algorithm and the Grey Wolf Optimizer (GWO). Subsections 1.3 to 1.8 covered the classical JAYA algorithm and its variants, including the Self-Adaptive JAYA, Quasi-Oppositional JAYA (QO-JAYA), and Self-Adaptive Multi-Population JAYA, demonstrating their effectiveness through applications such as dam design optimization. Sections 1.9 to 1.11 introduced the GWO algorithm, including its application to benchmark functions like Ackley, along with code implementation and discussion of performance results. Chapter I concluded with a summary of key insights, highlighting the complementary strengths of JAYA and GWO in balancing exploration and exploitation, which laid the groundwork for subsequent hybridizations.

Chapter II extended metaheuristic optimization into AI-based control strategies, focusing on multi-agent systems such as cart-pole mechanisms. The thesis discussed classical and modern control system strategies, the dynamics of single and dual cart-pole setups, and the application of the Alternating Direction Method of Multipliers (ADMM) combined with some strategies of the reinforcement learning for coordination tasks. The results highlighted the potential of integrating metaheuristic optimization with AI control for real-time, dynamic systems.

Chapter III investigated the role of metaheuristic optimization in advanced design and industrial processes, including human-robot interaction (Alpha Mini), augmented and virtual reality systems, and reverse engineering with additive manufacturing. The chapter emphasized the synergy between metaheuristics and intelligent design tools, providing possible future pathways toward integrated, data-driven optimization in complex industrial environments.

Chapters IV, V, and VI focused on the core contributions of the thesis: the development and analysis of three novel hybrid GWO-JAYA algorithms.

Chapter IV presented SHGWJA, detailing the algorithm's design, mathematical formulation, and implementation. Extensive applications were provided across engineering optimization problems, robotics path planning, and structural characterization of composite panels, supported by statistical analyses and comparative evaluations.

Chapter V introduced FHGWJA, a fast hybrid variant emphasizing rapid convergence through mirroring and elitism. The chapter highlighted design optimization applications, demonstrating improved efficiency and solution quality across benchmark and real-world engineering problems.

Chapter VI detailed HALSGWJA, which integrated approximate gradient-based line search with adaptive thresholds to enhance search direction quality. Structural optimization examples illustrated its superior performance, reduced computational cost, and potential for high-precision industrial applications.

Chapters VII and VIII explored the application of hybrid metaheuristics to autonomous robotic systems, with a particular focus on navigation and path planning in challenging environments. Chapter VII introduced Smart Explorer 1.0 (SE1.0), a novel intelligent robotic path planning algorithm, specifically developed to address navigation in complex and hazardous terrains. SE1.0 was rigorously validated using high-resolution Digital Terrain Models (DTMs) of Martian landscapes, demonstrating its capability to navigate unpredictable, extraterrestrial environments. This work not only advanced robotic autonomy in extreme conditions but also opened promising avenues for space exploration applications.

Chapter VIII presented TSPGWJO, a hybrid metaheuristic solution to the Travelling Salesman Problem under kinematic constraints, designed to tackle highly complex agricultural challenges. The algorithm was specifically engineered for optimal planning of autonomous robotic operations in environments inaccessible to humans, such as steep alpine regions, with the objective of efficiently locating and extracting invasive plant species. By combining classical and novel metaheuristic strategies, TSPGWJO enabled precise, adaptive, and efficient coverage of irregular and hazardous terrains, highlighting the potential of hybrid optimization for real-world environmental management and autonomous agricultural robotics.

Together, these studies demonstrated the versatility of hybrid metaheuristics in pushing the boundaries of autonomous systems, from space exploration to high-risk terrestrial applications, bridging advanced optimization techniques with practical AI-driven robotics in both extreme and ecologically sensitive environments.

Particularly noteworthy was the development of three novel hybrid GWO-JAYA algorithms, all implemented by the author: SHGWJA (Simple Hybrid Grey Wolf JAYA Algorithm), FHGWJA (Fast Hybrid Grey Wolf JAYA Algorithm), and HALSGWJA (Hybrid Approximate Line Search Grey Wolf JAYA Algorithm). Each algorithm employed distinct elitism strategies, update mechanisms, and, in the case of HALSGWJA, gradient-informed guidance, reflecting different approaches to balancing exploration and exploitation. Analysing these algorithms within a unified framework allowed for a clear understanding of their operational principles,

highlighted inherent trade-offs, and demonstrated their tangible applicability to both Industry 4.0 and AI-driven systems.

The SHGWJA algorithm emphasized simplicity and repeatability, implementing an elitist trial acceptance criterion ($W(\bar{X}) \leq W(\bar{X}_{best})$), dual JAYA modes for exploration and exploitation, and a mirroring mechanism to prevent stagnation. This approach ensured controlled evaluations, minimized computational overhead, and delivered robust performance, making it suitable for industrial applications where efficiency, reliability, and predictable results were essential.

The FHGWJA algorithm prioritized rapid convergence. It followed a GWO → refinement → JAYA-repair workflow, coupled with a temporary acceptance threshold and a mirroring strategy that transformed non-descent directions into effective descent paths. This mechanism reduced constraint evaluations, maintained low the Number of Functions Evaluation (NFE), and achieved accelerated convergence on complex real-world problems, proving advantageous in time-sensitive industrial optimization scenarios.

The HALSGWJA algorithm integrated approximate gradient estimation and an approximate line search to guide perturbations. By replacing the fixed threshold with an adaptive threshold, ρ_w, calculated from cost variations and averaged gradients, HALSGWJA enhanced search direction quality while minimizing additional evaluations. It achieved superior performance on CEC2020 benchmark problems and crashworthiness simulations, illustrating the potential of gradient-guided hybrid metaheuristics for high-precision, adaptive optimization in AI-enabled Industry 4.0 contexts.

Collectively, these algorithms demonstrated that careful hybridization could significantly enhance solution quality, convergence speed, and robustness, bridging the gap between heuristic efficiency and intelligent adaptivity. Their comparative characteristics are summarized in the following Final Table:

*Final Table- COMPARISON BETWEEN **FHGWJA, SHGWJA, HALSGWJA***

ITEM	FHGWJA (FAST HYBRID GWO–JAYA)	SHGWJA (SIMPLE HYBRID GWO-JAYA)	HALSGWJA (HYBRID APPROX. LINE SEARCH GWO-JAYA)
PRINCIPLE	GWO+JAYA hybrid with selection/elitism strategies and mirroring-based repair.	Minimal GWO+JAYA hybrid introducing simple elitist strategies to enhance exploitation.	GWO+JAYA hybrid integrating approximate line search via averaged gradients.
TRIAL GENERATION	Preliminary GWO → rank-based refinement → JAYA-based repair; mirroring applied if no improvement occurs.	Preliminary GWO → evaluation via elitist criterion 1.1 → JAYA explore/exploit	Preliminary GWO → refinement using approximate gradients → adaptive criterion ϱ_w for explore/exploit.
ELITISM / THRESHOLD	Temporary acceptance threshold 1.1·W_best; explicit elitism and repair (including worst elimination).	Fixed threshold 1.1·W_best for temporary acceptance; simple elitist strategies.	Adaptive threshold ϱ_w (replacing 1.1) computed from averaged gradients and population quality.

ITEM	FHGWJA (FAST HYBRID GWO–JAYA)	SHGWJA (SIMPLE HYBRID GWO-JAYA)	HALSGWJA (HYBRID APPROX. LINE SEARCH GWO-JAYA)
MIRRORING / REPAIR	Mirroring combines non-descent directions to generate candidates along descent directions.	Mirroring to prevent stagnation of $\alpha/\beta/\delta$ wolves, plus simpler JAYA repair.	Mirroring retained as fallback, prioritized along directions guided by approximate gradients.
USE OF GRADIENTS / LINE SEARCH	No	No	Yes: calculation of averaged gradients and approximate line search to direct perturbations
CONSTRAINT EVALUATION	Evaluated only after cost-based checks to reduce NFE; initial decisions based on W().	Same philosophy	Same, but averaged gradients computed without additional constraint evaluations.
COMPLEXITY / NFE	Low–medium	Very low	Moderate: gradient computation adds slight overhead.
ROBUSTNESS / RESULTS	High robustness; rapid convergence	Very good on benchmarks; low standard deviations	Excellent on CEC2020; improved exploration/exploitation balance, reduced NFE
TYPICAL APPLICATIONS	Engineering optimization: structures, robotics, hydraulics	Engineering problems and benchmarks	Complex, multi-constrained problems (CEC2020, crashworthiness)

Finally, the appendices provided extensive support materials, including MATLAB implementations of JAYA and GWO algorithms, ADMM and LQR control codes for cart-pole systems, data for robotics and industrial case studies, and a catalogue of all studied optimization problems across mathematics, mechanics, robotics, hydraulics, and civil engineering. These resources ensure reproducibility, transparency, and practical applicability of the proposed methods.

Overall, this thesis demonstrates how hybrid metaheuristic algorithms, intelligent control strategies, and AI-driven optimization can be seamlessly integrated to tackle complex, multi-constrained problems in modern industrial environments. By delivering efficient, adaptive, and robust solutions, the study contributes directly to Industry 4.0, supporting automation, decision-making, and system performance across engineering, robotics, and manufacturing applications.

Future research should focus on extending these hybrid algorithms to large-scale, multi-objective problems, exploring adaptive mechanisms for dynamic thresholding and elitism, and integrating machine learning techniques to intelligently guide the search process. Such advancements would further enhance the efficiency, robustness, and generality of hybrid metaheuristics, enabling them to tackle increasingly complex engineering and industrial optimization challenges.

A particularly promising line of development concerns the evolution of Smart Explorer 1.0 (SE1.0). Its next-generation version could integrate advanced sensor technologies, such as

LiDAR, stereo vision, and environmental monitoring systems, to achieve real-time perception, terrain assessment, and adaptive navigation in dynamic and uncertain environments. This sensor fusion would transform SE1.0 from an algorithmic framework into a fully autonomous exploration system capable of perceiving, reasoning, and acting intelligently, making it ideally suited for planetary exploration or operations in hazardous terrestrial contexts.

In parallel, the TSPGWJO framework could be enhanced through integration with the upgraded SE1.0 architecture. This synergy would enable real-time adaptation to environmental feedback while optimizing paths under complex kinematic constraints. Such an approach could give rise to a new generation of intelligent robotic platforms capable of performing large-scale operations, such as invasive species removal or environmental mapping in alpine and inaccessible areas, with unprecedented autonomy, precision, and environmental awareness.

Collectively, these advancements would not only refine algorithmic performance but also redefine the boundaries of AI-driven robotic exploration, bridging the gap between theoretical optimization and intelligent, context-aware action at the core of Industry 4.0 and the emerging era of autonomous intelligence.

REFERENCES

[1] Yu-Jun Zheng, «Water wave optimization: A new nature-inspired metaheuristic», *Computers & Operations Research*, vol. 55, pp. 1–11, 2015, doi: https://doi.org/10.1016/j.cor.2014.10.008.
[2] Haktanirlar Ulutas, Berna e AU - Kulturel-Konak, Sadan, «A review of clonal selection algorithm and its applications», *Artificial Intelligence Review*, 2011. [Online]. Disponibile su: https://doi.org/10.1007/s10462-011-9206-1
[3] Taylor, Connor J. *et al.*, «A Brief Introduction to Chemical Reaction Optimization», *American Chemical Society*, 2023. [Online]. Disponibile su: https://doi.org/10.1021/acs.chemrev.2c00798
[4] Mahima Dubey, Vijay Kumar, Manjit Kaur, Thanh-Phong Dao, «A Systematic Review on Harmony Search Algorithm: Theory, Literature, and Applications», in *Mathematical Problems in Engineering*, 2021. [Online]. Disponibile su: https://doi.org/10.1155/2021/5594267
[5] Seyedali Mirjalili, «A Sine Cosine Algorithm for solving optimization problems», vol. 96, pp. 120–133, 2016, doi: https://doi.org/10.1016/j.knosys.2015.12.022.
[6] R.V. Rao, V.J. Savsani, D.P. Vakharia, «Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems», *Computer-Aided Design*, vol. 43, pp. 303–315, 2011, doi: https://doi.org/10.1016/j.cad.2010.12.015.
[7] Moayedi, H.; Tien Bui, D.; Dounis, A.; Ngo, P.T.T., «A Novel Application of League Championship Optimization (LCA): Hybridizing Fuzzy Logic for Soil Compression Coe cient Analysis», *Applied Sciences*, 2020, doi: doi:10.3390/app10010067.
[8] Milani, A., «Evolutionary Algorithms in Intelligent Systems», *Mathematics*, vol. 8, p. 1733, 2020, doi: https://doi.org/10.3390/math8101733.
[9] Alhijawi, B., Awajan, A., «Evolutionary Intelligence», *Genetic algorithms: theory, genetic operators, solutions, and applications*, vol. 17, pp. 1245–1256, 2024, doi: https://doi.org/10.1007/s12065-023-00822-6.
[10] Sengupta, S.; Basak, S.; Peters, R.A., «Particle Swarm Optimization: A Survey of Historical and Recent Developments with Hybridization Perspectives», *Machine Learning and Knowledge Extraction*, vol. 1, pp. 157–191, 2019, doi: https://doi.org/10.3390/make1010010.
[11] Raed Abu Zita Mohammed Azmi Al-Betar,Mohammed A. Awadallah,Iyad Abu Doush,Khaled Assaleh, «An Intensive and Comprehensive Overview of JAYA Algorithm, its Versions and Applications», *Archives of Computational Methods in Engineering*, vol. 29, pp. 763–792, 2022, doi: https://doi.org/10.1007/s11831-021-09585-8.
[12] S. Mirjalili, S. M. Mirjalili, e A. Lewis, «Grey Wolf Optimizer», *Advances in Engineering Software*, vol. 69, pp. 46–61, mar. 2014, doi: 10.1016/j.advengsoft.2013.12.007.
[13] R. Venkata Rao, «Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems», *International Journal of Industrial Engineering Computations*, vol. 7, pp. 19–34, 2016, doi: 10.5267/j.ijiec.2015.8.004.
[14] Wilken, D.; Mercker, M.; Fischer, P.; Vött, A.; Erkul, E.; Corradini, E.; Pickartz, N., «Artificial Bee Colony Algorithm with Adaptive Parameter Space Dimension: A Promising Tool for Geophysical Electromagnetic Induction Inversion», *Remote Sensing*, 2024, doi: https://doi.org/10.3390/rs16030470.
[15] Abualigah L, Diabat A, Geem ZW, «A Comprehensive Survey of the Harmony Search Algorithm in Clustering Applications», *Applied Sciences*, vol. 10, p. 3827, 2020.
[16] R. Venkata Rao, Ankit Saroj, «A self-adaptive multi-population based Jaya algorithm for engineering optimization», *Swarm and Evolutionary Computation*, vol. 37, 2017, doi: https://doi.org/10.1016/j.swevo.2017.04.008.
[17] Rajesh.D, S.Thenappan, Prachi Juyal, Thiyagarajan V.S., D.M.Kalai Selvi, J.Rajeswari, M.Hema Kumar, V.Saravanan, «Quasi Oppositional Jaya Algorithm with Computer Vision based Deep Learning Model for Emotion Recognition on Autonomous Vehicle Drivers», *Journal of*

Intelligent Systems and Internet of Things, vol. 14, pp. 141–153, 2025, doi: https://doi.org/10.54216/JISIoT.140111.

[18] R. Venkata Rao, Ankit Saroj, «A self-adaptive multi-population based Jaya algorithm for engineering optimization», *Swarm and Evolutionary Computation*, vol. 37, pp. 1–26, 2017, doi: https://doi.org/10.1016/j.swevo.2017.04.008.

[19] C. Muro, R. Escobedo, L. Spector, R.P. Coppinger, «Wolf-pack (Canis lupus) hunting strategies emerge from simple rules in computational simulations», *Behavioural Processes*, vol. 88, pp. 192–197, 2011, doi: https://doi.org/10.1016/j.beproc.2011.09.006.

[20] David H. Ackley, *A Connectionist Machine for Genetic Hillclimbing*, vol. 28. in SECS, vol. 28. Springer, 1987. [Online]. Disponibile su: https://doi.org/10.1007/978-1-4613-1997-9

[21] Marcin Molga, Czesław Smutnicki, «Test Functions for Optimization Needs», Poznan University of Technology, 2005. [Online]. Disponibile su: https://robertmarks.org/Classes/ENGR5358/Papers/functions.pdf

[22] Jungtaek Kim, «Benchmark Functions for Bayesian Optimization». 2020.

[23] Karl Johan Åström e Richard M. Murray, *Feedback Systems An Introduction for Scientists and Engineers*, Second Edition. Princeton University Press, 2021. [Online]. Disponibile su: https://press.princeton.edu/books/hardcover/9780691193984/feedback-systems

[24] Richard S. Sutton and Andrew G. Barto, *Reinforcement Learning: An Introduction*, Second Edition. The MIT Press, 2018.

[25] Boyd, S., Parikh, N., Chu, E., Peleato, B. and Eckstein, J., «Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers», *Foundations and Trends in Machine Learning*, vol. 3, pp. 1–122, 2011, doi: https://doi.org/10.1561/2200000016.

[26] Mark W. Spong, «Swing up control of the acrobot using partial feedback linearization*», *IFAC Proceedings Volumes*, vol. 27, pp. 833–838, doi: https://doi.org/10.1016/S1474-6670(17)47404-0.

[27] B. Kiumarsi, K. G. Vamvoudakis, H. Modares and F. L. Lewis, «Optimal and Autonomous Control Using Reinforcement Learning: A Survey», *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, pp. 2042–2062, 2018, doi: 10.1109/TNNLS.2017.2773458.

[28] Karl Johan Åström, Tore Hägglund, *Advanced PID Control*. ISA - The Instrumentation, Systems and Automation Society, 2006.

[29] Kwakernaak, H., & Sivan, R., *Linear Optimal Control Systems*. Wiley, 1972. [Online]. Disponibile su: https://archive.org/details/linearoptimalcon0000kwak

[30] Kemin Zhou, John Comstock Doyle, Keith Glover, *Robust and Optimal Control*. Prentice Hall, 1996.

[31] Isabelle Fantoni , Rogelio Lozano, *Non-linear Control for Underactuated Mechanical Systems*. Springer, 2002.

[32] John Baillieul, Tariq Samad, *Encyclopedia of Systems and Control*, Second Edition. Springer, 2021.

[33] Rios-Martinez, J., Spalanzani, A. & Laugier, C., «From Proxemics Theory to Socially-Aware Navigation: A Survey.», *International Journal of Social Robotics*, vol. 7, pp. 137–153, 2015, doi: https://doi.org/10.1007/s12369-014-0251-1.

[34] Xu, Y., Li, Q., Xu, X., Yang, J. & Chen, Y., «Research Progress of Nature-Inspired Metaheuristic Algorithms in Mobile Robot Path Planning», *Electronics*, vol. 12, 2023, doi: https://doi.org/10.3390/ electronics12153263.

[35] Sercan Ö. Arık, Gregory Diamos, Andrew Gibiansky, John Mille, Kainan Peng, Wei Ping, Jonathan Raiman, Yanqi Zhou, «Deep voice 2: multi-speaker neural text-to-speech», presentato al NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems December 2017, 2017, pp. 2966–2974. [Online]. Disponibile su: https://arxiv.org/abs/1705.08947

[36] Mohamed Salah Salhi e Ali Hamdan Alenezi, «Heliyon», *Voice recognition enhancement by genetic algorithm*, vol. 10, 2024, doi: https://doi.org/10.1016/j.heliyon.2024.e38845.
[37] Yara Pérez Maldonado, Santiago Omar Caballero Morales & Roberto Omar Cruz Ortega, «GA Approaches to HMM Optimization for Automatic Speech Recognition», in *Pattern Recognition (MCPR 2012)*, 2012, pp. 313-322,. doi: https://doi.org/10.1007/978-3-642-31149-9_32.
[38] James Bergstra e Yoshua Bengio, «Random Search for Hyper-Parameter Optimization», *Journal of Machine Learning Research*, vol. 13, pp. 281–305, 2012.
[39] Marina Sokolova, Guy Lapalme, «A systematic analysis of performance measures for classification tasks», *Information Processing & Management*, vol. 45, pp. 427–437, 2009, doi: https://doi.org/10.1016/j.ipm.2009.03.002.
[40] Deng, J.; Hou, R.; Deng, Y.; Long, Y.; Wu, N., «High-Quality Text-to-Speech Implementation via Active Shallow Diffusion Mechanism», *Sensors*, vol. 25, 2025, doi: https://doi.org/10.3390/s25030833.
[41] Kane, J.; Johnstone, M.N.; Szewczyk, P., «Voice Synthesis Improvement by Machine Learning of Natural Prosody», *Sensors*, vol. 24, 2024, doi: https://doi.org/10.3390/s24051624.
[42] Ngiam, Jiquan and Khosla, Aditya and Kim, Mingyu and Nam, Juhan and Lee, Honglak and Ng, Andrew Y., «Multimodal deep learning», presentato al Proceedings of the 28th International Conference on International Conference on Machine Learning, Omnipress, 2011, pp. 689–696. [Online]. Disponibile su: https://dl.acm.org/doi/10.5555/3104482.3104569
[43] Tadas Baltrušaitis, Chaitanya Ahuja, Louis-Philippe Morency, «Multimodal Machine Learning: A Survey and Taxonomy», *Computer Science*, 2017, doi: https://doi.org/10.48550/arXiv.1705.09406 Focus to learn more.
[44] Xiong, J., Hsiang, EL., He, Z. et al., «Augmented reality and virtual reality displays: emerging technologies and future perspectives», *Light Science and Applications*, vol. 10, p. 216, 2021, doi: https://doi.org/10.1038/s41377-021-00658-8.
[45] Gendreau, M.; Potvin, J-Y, *Handbook of Metaheuristics*, Third Ediction. Springer, 2010. [Online]. Disponibile su: https://link.springer.com/book/10.1007/978-3-319-91086-4
[46] Omidi, M.; Van de Perre, G.; Kumar Hota, R.; Cao, H.-L.; Saldien, J.; Vanderborght, B.; El Makrini, I., «Improving Postural Ergonomics during Human–Robot Collaboration Using Particle Swarm Optimization: A Study in Virtual Environment.», *Applied Sciences*, vol. 13, p. 5385, 2023, doi: https://doi.org/10.3390/app13095385.
[47] Hansen, P., Mladenović, N. & Moreno Pérez, J.A., «Variable neighbourhood search: methods and applications», *Annals of Operations Research*, vol. 175, pp. 367–407, 2010, doi: https://doi.org/10.1007/s10479-009-0657-6.
[48] Suárez, P., Gálvez, A., Iglesias, A., «Autonomous Coordinated Navigation of Virtual Swarm Bots in Dynamic Indoor Environments by Bat Algorithm», presentato al Advances in Swarm Intelligence. ICSI 2017, Springer, Cham, 2017. doi: https://doi.org/10.1007/978-3-319-61833-3_19.
[49] Richard Szeliski, *Computer Vision: Algorithms and Applications*. 2010.
[50] Ramkumar Thirunavukarasu, Evans Kotei, «A comprehensive review on transformer network for natural and medical image analysis», *Computer Science Review*, vol. 53, 2024, doi: https://doi.org/10.1016/j.cosrev.2024.100648.
[51] Novac, O.-C.; Chirodea, M.C.; Novac, C.M.; Bizon, N.; Oproescu, M.; Stan, O.P.; Gordan, C.E., «Analysis of the Application Efficiency of TensorFlow and PyTorch in Convolutional Neural Network.», *Sensors*, vol. 22, 2022, doi: https://doi.org/10.3390/s22228872.
[52] Shobaki, W.A.; Milanova, M., «A Comparative Study of YOLO, SSD, Faster R-CNN, and More for Optimized Eye-Gaze Writing», *Sci*, vol. 7, fasc. 47, 2025, doi: https://doi.org/10.3390/sci7020047.
[53] Pieczyński, D.; Ptak, B.; Kraft, M.; Drapikowski, P., «LunarSim: Lunar Rover Simulator Focused on High Visual Fidelity and ROS 2 Integration for Advanced Computer Vision Algorithm

Development.», *Applied Sciences*, vol. 13, fasc. 12401, 2023, doi: https://doi.org/10.3390/app132212401.
[54] Xie, Y.; Tang, Z.; Song, A., «Motion Simulation and Human–Computer Interaction System for Lunar Exploration», *Applied Sciences*, vol. 12, fasc. 2312, 2022, doi: https://doi.org/10.3390/app12052312.
[55] Roman Wdowik e Artur Bełzo, «Artificial intelligence-based design of assemblies in the FreeCAD software», *TIAM*, vol. 127, 2025, doi: https://doi.org/10.7862/tiam.2025.1.6.
[56] Menapace, A.; Zanfei, A.; Herrera, M.; Brentan, B., «Graph Neural Networks for Sensor Placement: A Proof of Concept towards a Digital Twin of Water Distribution Systems», *Water*, vol. 16, fasc. 13, p. 18355, 2024, doi: https://doi.org/10.3390/w16131835.
[57] Waqas, M.; Jan, L.; Zafar, M.H.; Hassan, S.R.; Asif, R., «A Sensor Placement Approach Using Multi-Objective Hypergraph Particle Swarm Optimization to Improve Effectiveness of Structural Health Monitoring Systems.», *Sensors*, vol. 24, p. 1423, 2024.
[58] Aththanayake, Akeshi, Lininger, Andrew, Strangi, Cataldo, Griswold, Mark A. and Strangi, Giuseppe, «Tunable holographic metasurfaces for augmented and virtual reality», *Nanophotonics*, 2025, doi: https://doi.org/10.1515/nanoph-2024-0734.
[59] Makhataeva, Z.; Varol, H.A., «Augmented Reality for Robotics: A Review», *Robotics*, vol. 9, p. 21, 2020, doi: https://doi.org/10.3390/robotics9020021.
[60] Fogaça, D.R.; Grijalvo, M.; Sacomano Neto, M., «What Are Industry 4.0 and Industry 5.0 All About? An Integrative Institutional Model for the New Industrial Paradigms.», *Administrative Sciences*, vol. 15, doi: https://doi.org/10.3390/admsci15040118.
[61] Gourav Misra, Vivek Kumar, Arun Agarwal, Kabita Agarwal, «Internet of Things (IoT) – A Technological Analysis and Survey on Vision, Concepts, Challenges, Innovation Directions, Technologies, and Applications (An Upcoming or Future Generation Computer Communication System Technology)», *American Journal of Electrical and Electronic Engineering*, vol. 4, 2016, doi: 10.12691/ajeee-4-1-4.
[62] Nahavandi, S., «Industry 5.0—A Human-Centric Solution.», *Sustainability*, vol. 11, 2019, doi: https://doi.org/10.3390/su11164371.
[63] Almusaed, A.; Yitmen, I.; Almssad, A., «Reviewing and Integrating AEC Practices into Industry 6.0: Strategies for Smart and Sustainable Future-Built Environments», *Sustainability*, vol. 15, 2023, doi: https://doi.org/10.3390/su151813464.
[64] Baki, Y., «The Impact of Critical Listening and Critical Reading on Critical Thinking», vol. 15, fasc. 34, 2025, doi: https://doi.org/10.3390/bs15010034.
[65] Jiang, J.; Ma, Y., «Path Planning Strategies to Optimize Accuracy, Quality, Build Time and Material Use in Additive Manufacturing: A Review.», *Micromachines*, vol. 11, p. 633, 2020, doi: https://doi.org/10.3390/mi11070633.
[66] Miller, C.E., Tucker, A.W. and Zemlin, R.A., «Integer Programming Formulations and Traveling Salesman Problems», *Journal of the Association for Computing Machinery*, vol. 7, pp. 326–329, doi: https://doi.org/10.1145/321043.321046.
[67] Jiang, J.; Xu, X.; Stringer, J., «Support Structures for Additive Manufacturing: A Review.», *Manufacturing and Materials Processing*, vol. 2, 2018, doi: https://doi.org/10.3390/jmmp2040064.
[68] Martin P. Bendsøe , Ole Sigmund, *Topology Optimization:Theory, Methods, and Applications*, Second Edition. Springer, 2004.
[69] C. Dinesh, G. Cheung and I. V. Bajić, «Point Cloud Denoising via Feature Graph Laplacian Regularization», *IEEE Transactions on Image Processing*, vol. 29, pp. 4143–4158, 2020, doi: 10.1109/TIP.2020.2969052.
[70] Xian-Feng Han, Jesse S. Jin, Ming-Jie Wang, Wei Jiang, Lei Gao, Liping Xiao, «A review of algorithms for filtering the 3D point cloud», *Signal Processing: Image Communication*, vol. 57, 2017, doi: https://doi.org/10.1016/j.image.2017.05.009.

[71] Jegoon Ryu, Sei-ichiro Kamata, «An efficient computational algorithm for Hausdorff distance based on points-ruling-out and systematic random sampling», *Pattern Recognition*, vol. 114, 2021, doi: https://doi.org/10.1016/j.patcog.2021.107857.
[72] Degertekin, S.O.; Minooei, M.; Santoro, L.; Trentadue, B.; Lamberti, L., «Large-Scale Truss-Sizing Optimization with Enhanced Hybrid HS Algorithm», *Applied Sciences*, vol. 11, fasc. 3270, 2021, doi: https://doi.org/10.3390/app11073270.
[73] S.O. Degertekin, G. Yalcin Bayar, L. Lamberti, «Parameter free Jaya algorithm for truss sizing-layout optimization under natural frequency constraints», *Computers & Structures*, vol. 245, 2021, doi: https://doi.org/10.1016/j.compstruc.2020.106461.
[74] Ficarella,E.;Lamberti,L.;Degertekin,S.O., «Mechanical Identification of Materials and Structures with Optical Methods and Metaheuristic Optimization», *Materials*, vol. 12, 2019, doi: https://doi.org/10.3390/ma12132133.
[75] Li,H.;Zhang,Q.;Qin,X.;Yuantao,S., «Raw vibration signal pattern recognition with automatic hyper-parameter-optimized convolutional neural network for bearing fault diagnosis», *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science.*, vol. 234, pp. 343–360, 2019, doi: https://doi.org/10.1177/0954406219875756.
[76] Jihong Guo, Chao Liu, Jinfeng Cao, Dongxiang Jiang, «Damage identification of wind turbine blades with deep convolutional neural networks», *Renewable Energy*, vol. 174, pp. 122–133, 2021, doi: https://doi.org/10.1016/j.renene.2021.04.040.
[77] Paulo Vitor Silvestrin, Marcus Ritt, «An iterated tabu search for the multi-compartment vehicle routing problem», *Computers & Operations Research*, vol. 81, pp. 192–202, 2017, doi: https://doi.org/10.1016/j.cor.2016.12.023.
[78] Asif Afzal, Abdulrajak Buradi, Ravindra Jilte, Saboor Shaik, Abdul Razak Kaladgi, Muslum Arıcı, Chew Tin Lee, Sandro Nižetić, «Optimizing the thermal performance of solar energy devices using meta-heuristic algorithms: A critical review», *Renewable and Sustainable Energy Reviews*, vol. 173, 2023, doi: https://doi.org/10.1016/j.rser.2022.112903.
[79] Quang-Thanh Bui, «Metaheuristic algorithms in optimizing neural network: a comparative study for forest fire susceptibility mapping in Dak Nong, Vietnam», *Geomatics, Natural Hazards and Risk*, vol. 10, pp. 136–150, 2019, doi: https://doi.org/10.1080/19475705.2018.1509902.
[80] Zubaidi,S.L.;Abdulkareem,I.H.;Hashim,K.S.;Al-Bugharbee,H.;Ridha,H.M.;Gharghan,S.K.; Al-Qaim, F.F.; Muradov, M.; Kot, P.; Al-Khaddar, R., «Hybridised Artificial Neural Network Model with Slime Mould Algorithm: A Novel Methodology for Prediction of Urban Stochastic Water Demand», *Water*, vol. 12, 2020, doi: https://doi.org/10.3390/w12102692.
[81] Brion,D.A.J.;Pattinson,S.W., «Generalisable 3D printing error detection and correction via multi-head neural networks», *Nature Communications*, vol. 13, 2022, doi: https://doi.org/10.1038/s41467-022-31985-y.
[82] Ahmed M. Anter, Diego Oliva, Anuradha Thakare, Zhiguo Zhang, «AFCM-LSMA: New intelligent model based on Lévy slime mould algorithm and adaptive fuzzy C-means for identification of COVID-19 infection from chest X-ray images», *Advanced Engineering Informatics*, vol. 49, 2021, doi: https://doi.org/10.1016/j.aei.2021.101317.
[83] L. Meenachi, S. Ramakrishnan, «Metaheuristic Search Based Feature Selection Methods for Classification of Cancer», *Pattern Recognition*, vol. 119, 2021, doi: https://doi.org/10.1016/j.patcog.2021.108079.
[84] Diego Oliva, Noe Ortega-Sánchez, Salvador Hinojosa, Marco Pérez-Cisneros, *Modern Metaheuristics in Image Processing*. Boca Raton: CRC Press, 2022. [Online]. Disponibile su: https://doi.org/10.1201/9781003183501
[85] David Edward Goldberg, *Genetic algorithms in search, optimization, and machine learning*. USA: Reading, Mass. : Addison-Wesley Pub. Co., 1989. [Online]. Disponibile su: https://archive.org/details/geneticalgorithm0000gold

[86] John H. Holland, «Genetic Algorithms», *Scientific American*, vol. 267, pp. 66–73, 1992.
[87] Storn, R., Price, K., «Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces», *Journal of Global Optimization*, vol. 11, pp. 341–359, 1997, doi: https://doi.org/10.1023/A:1008202821328.
[88] Kenneth V. Price , Rainer M. Storn , Jouni A. Lampinen, *Differential Evolution A Practical Approach to Global Optimization*. Springer Berlin, Heidelberg, 2005. [Online]. Disponibile su: https://doi.org/10.1007/3-540-31306-0
[89] Xin Yao, Yong Liu, Guangming Lin, «Evolutionary programming made faster», *IEEE Transactions on Evolutionary Computation*, vol. 3, pp. 82–102, 1999, doi: 10.1109/4235.771163.
[90] Beyer, HG., Schwefel, HP., «Evolution strategies – A comprehensive introduction», *Natural Computing*, vol. 1, 2002, doi: https://doi.org/10.1023/A:1015059928466.
[91] Simon,D., «Biogeography-basedoptimization», *IEEE Transactions on Evolutionary Computation*, vol. 12, pp. 702–713, 2008, doi: 10.1109/TEVC.2008.919004.
[92] Vahideh Hayyolalam, Ali Asghar Pourhaji Kazem, «Black Widow Optimization Algorithm: A novel meta-heuristic approach for solving engineering optimization problems», *Engineering Applications of Artificial Intelligence*, vol. 87, 2020, doi: https://doi.org/10.1016/j.engappai.2019.103249.
[93] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, «Optimization by Simulated Annealing», *Science*, vol. 220, pp. 671–680, 1983, doi: 10.1126/science.220.4598.671.
[94] Peter J. M. Laarhoven , Emile H. L. Aarts, *Simulated Annealing: Theory and Applications*. in Mathematics and Its Applications. Springer Dordrecht, 1987. [Online]. Disponibile su: https://doi.org/10.1007/978-94-015-7744-1
[95] Kaveh, A., Talatahari, S., «A novel heuristic optimization method: charged system search», *Acta Mechanica*, vol. 213, pp. 267–289, 2010, doi: https://doi.org/10.1007/s00707-009-0270-4.
[96] Kaveh, A., Motie Share, M.A. & Moslehi, M., «Magnetic charged system search: a new meta-heuristic algorithm for optimization», *Acta Mechanica*, vol. 224, pp. 85–107, 2013, doi: https://doi.org/10.1007/s00707-012-0745-6.
[97] A. Kaveh, M. Khayatazad, «A new meta-heuristic method: Ray Optimization», *Computers & Structures*, vol. 112–113, pp. 283–294, 2012, doi: https://doi.org/10.1016/j.compstruc.2012.09.003.
[98] A. Kaveh, V.R. Mahdavi, «Colliding bodies optimization: A novel meta-heuristic method», *Computers & Structures*, vol. 139, pp. 18–27, 2014, doi: https://doi.org/10.1016/j.compstruc.2014.04.005.
[99] Kaveh, A., Bakhshpoori, T., «A new metaheuristic for continuous structural optimization: water evaporation optimization», *Structural and Multidisciplinary Optimization*, vol. 54, pp. 23–43, 2016, doi: https://doi.org/10.1007/s00158-015-1396-8.
[100] A. Kaveh, A. Dadras, «A novel meta-heuristic optimization algorithm: Thermal exchange optimization», *Advances in Engineering Software*, vol. 110, pp. 69–84, 2017, doi: https://doi.org/10.1016/j.advengsoft.2017.03.014.
[101] Afshin Faramarzi, Mohammad Heidarinejad, Brent Stephens, Seyedali Mirjalili, «Equilibrium optimizer: A novel optimization algorithm», *Knowledge-Based Systems*, vol. 191, 2020, doi: https://doi.org/10.1016/j.knosys.2019.105190.
[102] Abdel-Basset, M.; Mohamed, R.; Sallam, K.M.; Chakrabortty, R.K, «Light Spectrum Optimizer: A Novel Physics-Inspired Metaheuristic Optimization Algorithm», *Mathematics*, vol. 10, 2022, doi: https://doi.org/10.3390/math10193466.
[103] Bilal Alatas, «ACROA: Artificial Chemical Reaction Optimization Algorithm for global optimization», *Expert Systems with Applications*, vol. 38, 2011, doi: https://doi.org/10.1016/j.eswa.2011.04.126.

[104] Marjan Abdechiri, Mohammad Reza Meybodi, Helena Bahrami, «Gases Brownian Motion Optimization: an Algorithm for Optimization (GBMO)», *Applied Soft Computing*, vol. 13, pp. 2932–2946, 2013, doi: https://doi.org/10.1016/j.asoc.2012.03.068.
[105] Fatma A. Hashim, Essam H. Houssein, Mai S. Mabrouk, Walid Al-Atabany, Seyedali Mirjalili, «Henry gas solubility optimization: A novel physics-based algorithm», *Future Generation Computer Systems*, vol. 101, pp. 646–667, 2019, doi: https://doi.org/10.1016/j.future.2019.07.015.
[106] Osman K. Erol, Ibrahim Eksin, «A new optimization method: Big Bang–Big Crunch», *Advances in Engineering Software*, vol. 37, pp. 106–111, 2006, doi: https://doi.org/10.1016/j.advengsoft.2005.04.005.
[107] Esmat Rashedi, Hossein Nezamabadi-pour, Saeid Saryazdi, «GSA: A Gravitational Search Algorithm», *Information Sciences*, vol. 179, pp. 2232–2248, 2009, doi: https://doi.org/10.1016/j.ins.2009.03.004.
[108] «Principal components analysis by the galaxy-based search algorithm: A novel metaheuristic for continuous optimisation», *International Journal of Computational Science and Engineering*, vol. 6, pp. 132–140, 2011, doi: 10.1504/IJCSE.2011.041221.
[109] Abdolreza Hatamlou, «Black hole: A new heuristic optimization approach for data clustering», *Information Sciences*, vol. 222, pp. 175–184, 2013, doi: https://doi.org/10.1016/j.ins.2012.08.023.
[110] Vijay Kumar, Dinesh Kumar, «An astrophysics-inspired Grey wolf algorithm for numerical optimization and its application to engineering design problems», *Advances in Engineering Software*, vol. 112, pp. 231–254, 2017, doi: https://doi.org/10.1016/j.advengsoft.2017.05.008.
[111] Amjad A. Hudaib, Hussam N. Fakhouri, «Supernova Optimizer: A Novel Natural Inspired Meta-Heuristic», *Modern Applied Science*, vol. 12, pp. 32–50, 2018, doi: 10.5539/mas.v12n1p32.
[112] https://doi.org/10.1016/j.knosys.2015.12.022, «SCA: A Sine Cosine Algorithm for solving optimization problems», *Knowledge-Based Systems*, vol. 96, pp. 120–133, 2016, doi: https://doi.org/10.1016/j.knosys.2015.12.022.
[113] Iman Ahmadianfar, Ali Asghar Heidari, Amir H. Gandomi, Xuefeng Chu, Huiling Chen, «RUN beyond the metaphor: An efficient optimization algorithm based on Runge Kutta method», *Expert Systems with Applications*, vol. 181, 2021, doi: https://doi.org/10.1016/j.eswa.2021.115079.
[114] Laith Abualigah, Ali Diabat, Seyedali Mirjalili, Mohamed Abd Elaziz, Amir H. Gandomi, «The Arithmetic Optimization Algorithm», *Computer Methods in Applied Mechanics and Engineering*, vol. 376, 2021, doi: https://doi.org/10.1016/j.cma.2020.113609.
[115] Fred Glover, «Future paths for integer programming and links to artificial intelligence», *Computers & Operations Research*, vol. 13, pp. 533–549, 1986, doi: https://doi.org/10.1016/0305-0548(86)90048-1.
[116] Fred Glover , Manuel Laguna, *Tabu Search*. Springer New York, NY, 1997. [Online]. Disponibile su: https://doi.org/10.1007/978-1-4615-6089-0
[117] Zong Woo Geem, Joong Hoon Kim, Loganathan GV., «A New Heuristic Optimization Algorithm: Harmony Search», *Simulation*, vol. 76, pp. 60–68, 2001, doi: 10.1177/003754970107600201.
[118] Kang Seok Lee, Zong Woo Geem, «A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice», *Computer Methods in Applied Mechanics and Engineering*, vol. 194, pp. 3902–3933, 2005, doi: https://doi.org/10.1016/j.cma.2004.09.007.
[119] Yiying Zhang, Zhigang Jin, «Group teaching optimization algorithm: A novel metaheuristic method for solving global optimization problems», *Expert Systems with Applications*, vol. 148, 2020, doi: https://doi.org/10.1016/j.eswa.2020.113246.
[120] Matoušo-vá,I.;Trojovský,P.;Dehghani,M.;Trojovská,E.;Kostra,J.M, «Mother optimization algorithm: a new human-based metaheuristic approach for solving engineering

optimization», *Scientific Reports*, vol. 13, 2023, doi: https://doi.org/10.1038/s41598-023-37537-8.
[121] Gopi, S., Mohapatra, P., «Learning cooking algorithm for solving global optimization problems», *Scientific Reports*, vol. 14, 2024, doi: https://doi.org/10.1038/s41598-024-60821-0.
[122] Qingyang Zhang, Ronggui Wang, Juan Yang, Kai Ding, Yongfu Li, Jiangen Hu, «Collective decision optimization algorithm: A new heuristic optimization method», *Neurocomputing*, vol. 221, pp. 123–137, 2017, doi: https://doi.org/10.1016/j.neucom.2016.09.068.
[123] A. Kaveh, S. Talatahari, «Optimum design of skeletal structures using imperialist competitive algorithm», *Computers & Structures*, vol. 88, pp. 1220–1229, 2010, doi: https://doi.org/10.1016/j.compstruc.2010.06.011.
[124] Qamar Askari, Irfan Younas, Mehreen Saeed, «Political Optimizer: A novel socio-inspired meta-heuristic for global optimization», *Knowledge-Based Systems*, vol. 195, 2020, doi: https://doi.org/10.1016/j.knosys.2020.105709.
[125] Eberhart, R.C. and Kennedy, J., «A New Optimizer Using Particle Swarm Theory», in *Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya*, ott. 1995, pp. 39–43. doi: http://dx.doi.org/10.1109/MHS.1995.494215.
[126] Maurice Clerc, *Particle Swarm Optimization*. ISTE Ltd, 2006.
[127] M. Dorigo, V. Maniezzo and A. Colorni, «Ant system: optimization by a colony of cooperating agents», *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 26, pp. 29–41, feb. 1996, doi: 10.1109/3477.484436.
[128] Bonabeau, E., Dorigo, M. & Theraulaz, G., «Inspiration for optimization from social insect behaviour», *Nature*, vol. 406, pp. 39–42, 2000, doi: https://doi.org/10.1038/35017500.
[129] Marco Dorigo, Thomas Stützle, *Ant Colony Optimization*. The MIT Press, 2004. [Online]. Disponibile su: https://doi.org/10.7551/mitpress/1290.001.0001
[130] Karaboga, D., Basturk, B., «A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm», *Journal of Global Optimization*, vol. 39, pp. 459–471, 2007, doi: https://doi.org/10.1007/s10898-007-9149-x.
[131] Xin-She Yang, «Firefly Algorithm, Stochastic Test Functions and Design Optimisation», *International Journal of Bio-Inspired Computation*, vol. 2, pp. 78–84, 2010, doi: 10.1504/IJBIC.2010.032124.
[132] Amir Hossein Gandomi, Xin-She Yang, Amir Hossein Alavi, «Mixed variable structural optimization using Firefly Algorithm», *Computers & Structures*, vol. 89, pp. 2325–2336, 2011, doi: https://doi.org/10.1016/j.compstruc.2011.08.002.
[133] Seyedali Mirjalili, «The Ant Lion Optimizer», *Advances in Engineering Software*, vol. 83, pp. 80–98, 2015, doi: https://doi.org/10.1016/j.advengsoft.2015.01.010.
[134] J. Pierezan and L. Dos Santos Coelho, «Coyote Optimization Algorithm: A New Metaheuristic for Global Optimization Problems», presentato al 2018 IEEE Congress on Evolutionary Computation (CEC), Rio de Janeiro, Brazil, 2018, pp. 1–8. doi: 10.1109/CEC.2018.8477769.
[135] Fatma A. Hashim, Abdelazim G. Hussien, «Snake Optimizer: A novel meta-heuristic optimization algorithm», *Knowledge-Based Systems*, vol. 242, 2022, doi: https://doi.org/10.1016/j.knosys.2022.108320.
[136] Coufal, P.; Hubálovský, Š.; Hubálovská, M.; Balogh, Z., «Snow Leopard Optimization Algorithm: A New Nature-Based Optimization Algorithm for Solving Optimization Problems», *Mathematics*, vol. 9, 2021, doi: https://doi.org/10.3390/math9212832.
[137] X. -S. Yang and Suash Deb, «Cuckoo Search via Lévy flights», presentato al 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), Coimbatore, India, dic. 2009, pp. 210–214. doi: 10.1109/NABIC.2009.5393690.
[138] Xin-She Yang, Suash Deb, «Engineering Optimisation by Cuckoo Search», *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 1, pp. 330–343, 2010, doi: 10.1504/IJMMNO.2010.035430.

[139] Alireza Askarzadeh, «A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm», *Computers & Structures*, vol. 169, pp. 1–12, 2016, doi: https://doi.org/10.1016/j.compstruc.2016.03.001.
[140] Hoda Zamani, Mohammad H. Nadimi-Shahraki, Amir H. Gandomi, «Starling murmuration optimizer: A novel bio-inspired algorithm for global and engineering optimization», *Computer Methods in Applied Mechanics and Engineering*, vol. 392, 2022, doi: https://doi.org/10.1016/j.cma.2022.114616.
[141] Xin-She Yang, «A New Metaheuristic Bat-Inspired Algorithm», *Studies in Computational Intelligence*, vol. 284, pp. 65–74, 2010, doi: 10.1007/978-3-642-12538-6_6.
[142] Xin-She Yang, Amir Hossein Gandomi, «Bat algorithm: a novel approach for global engineering optimization», *Engineering Computations*, vol. 29, pp. 464–483, 2012, doi: https://doi.org/10.1108/02644401211235834.
[143] Seyedali Mirjalili, Andrew Lewis, «The Whale Optimization Algorithm», *Advances in Engineering Software*, vol. 95, pp. 51–67, 2016, doi: https://doi.org/10.1016/j.advengsoft.2016.01.008.
[144] Seyedali Mirjalili, Amir H. Gandomi, Seyedeh Zahra Mirjalili, Shahrzad Saremi, Hossam Faris, Seyed Mohammad Mirjalili, «Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems», *Advances in Engineering Software*, vol. 114, pp. 163–191, 2017, doi: https://doi.org/10.1016/j.advengsoft.2017.07.002.
[145] Afshin Faramarzi, Mohammad Heidarinejad, Seyedali Mirjalili, Amir H. Gandomi, «Marine Predators Algorithm: A nature-inspired metaheuristic», *Expert Systems with Applications*, vol. 152, 2020, doi: https://doi.org/10.1016/j.eswa.2020.113377.
[146] H. T. Sadeeq and A. M. Abdulazeez, «Giant Trevally Optimizer (GTO): A Novel Metaheuristic Algorithm for Global Optimization and Challenging Engineering Problems», *IEEE Access*, vol. 10, pp. 121615–121640, 2022, doi: 10.1109/ACCESS.2022.3223388.
[147] Shun-Fa Hwang, Rong-Song He, «Improving real-parameter genetic algorithm with simulated annealing for engineering problems», *Advances in Engineering Software*, vol. 37, pp. 406–418, 2006, doi: https://doi.org/10.1016/j.advengsoft.2005.08.002.
[148] M. Fesanghary, M. Mahdavi, M. Minary-Jolandan, Y. Alizadeh, «Hybridizing harmony search algorithm with sequential quadratic programming for engineering optimization problems», *Computer Methods in Applied Mechanics and Engineering*, vol. 197, 2008, doi: https://doi.org/10.1016/j.cma.2008.02.006.
[149] L. Lamberti, «An efficient simulated annealing algorithm for design optimization of truss structures», *Computers & Structures*, vol. 86, pp. 1936–1953, 2008, doi: https://doi.org/10.1016/j.compstruc.2008.02.004.
[150] A. Kaveh, S. Talatahari, «Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss», *Computers & Structures*, vol. 87, pp. 267–283, 2009, doi: https://doi.org/10.1016/j.compstruc.2009.01.003.
[151] J. -B. Park, Y. -W. Jeong, J. -R. Shin and K. Y. Lee, «An Improved Particle Swarm Optimization for Nonconvex Economic Dispatch Problems», *IEEE Transactions on Power Systems*, vol. 25, pp. 156–166, 2010, doi: 10.1109/TPWRS.2009.2030293.
[152] S.O. Degertekin, «Improved harmony search algorithms for sizing optimization of truss structures», *Computers & Structures*, vol. 92–93, pp. 229–241, 2012, doi: https://doi.org/10.1016/j.compstruc.2011.10.022.
[153] Ali R. Yildiz, «A new hybrid differential evolution algorithm for the selection of optimal machining parameters in milling operations», *Applied Soft Computing*, vol. 13, pp. 1561–1566, 2013, doi: https://doi.org/10.1016/j.asoc.2011.12.016.
[154] A. Kaveh, T. Bakhshpoori, E. Afshari, «An efficient hybrid Particle Swarm and Swallow Swarm Optimization algorithm», *Computers & Structures*, vol. 143, pp. 40–59, 2014, doi: https://doi.org/10.1016/j.compstruc.2014.07.012.

[155] Ting,T.O.;Yang,X.S.;Cheng,S.;Huang,K., «Hybrid Metaheuristic Algorithms: Past, Present, and Future», *Recent Advances in Swarm Intelligence and Evolutionary Computation; Springer: Cham, Switzerland*, vol. 585, pp. 73–81, 2015, doi: 10.1007/978-3-319-13826-8_4.
[156] Hathiram Nenavath, Ravi Kumar Jatoth, «Hybridizing sine cosine algorithm with differential evolution for global optimization and object tracking», *Applied Soft Computing*, vol. 62, pp. 1019–1043, 2018, doi: https://doi.org/10.1016/j.asoc.2017.09.039.
[157] Shubham Gupta, Kusum Deep, Seyedali Mirjalili, «An efficient equilibrium optimizer with mutation strategy for numerical optimization», *Applied Soft Computing*, vol. 96, 2020, doi: https://doi.org/10.1016/j.asoc.2020.106542.
[158] E. Ficarella, L. Lamberti, S.O. Degertekin, «Comparison of three novel hybrid metaheuristic algorithms for structural optimization problems», *Computers & Structures*, vol. 244, 2021, doi: https://doi.org/10.1016/j.compstruc.2020.106395.
[159] Rohit Salgotra, Urvinder Singh, Gurdeep Singh, Nitin Mittal, Amir H. Gandomi, «A self-adaptive hybridized differential evolution naked mole-rat algorithm for engineering optimization problems», *Computer Methods in Applied Mechanics and Engineering*, vol. 383, 2021, doi: https://doi.org/10.1016/j.cma.2021.113916.
[160] Welhazi, Y.; Guesmi, T.; Alshammari, B.M.; Alqunun, K.; Alateeq, A.; Almalaq, Y.; Alsabhan, R.; Abdallah, H.H, «A Novel Hybrid Chaotic Jaya and Sequential Quadratic Programming Method for Robust Design of Power System Stabilizers and Static VAR Compensator», *Energies*, vol. 15, 2022, doi: https://doi.org/10.3390/en15030860.
[161] Zheng R, Jia H, Abualigah L, Liu Q, Wang S., «An improved arithmetic optimization algorithm with forced switching mechanism for global optimization problems», *Mathematical Biosciences and Engineering*, vol. 19, pp. 473–512, 2022, doi: 10.3934/mbe.2022023.
[162] Bouaouda, A., Sayouti, Y., «Hybrid Meta-Heuristic Algorithms for Optimal Sizing of Hybrid Renewable Energy System: A Review of the State-of-the-Art», *Archives of Computational Methods in Engineering*, vol. 29, pp. 4049–4083, 2022, doi: https://doi.org/10.1007/s11831-022-09730-x.
[163] Yu-Jun Zhang, Yu-Fei Wang, Liu-Wei Tao, Yu-Xin Yan, Juan Zhao, Zheng-Ming Gao, «Self-adaptive classification learning hybrid JAYA and Rao-1 algorithm for large-scale numerical and engineering problems», *Engineering Applications of Artificial Intelligence*, vol. 114, 2022, doi: https://doi.org/10.1016/j.engappai.2022.105069.
[164] Shang, C., Zhou, Tt. & Liu, S., «Optimization of complex engineering problems using modified sine cosine algorithm», *Scientific Reports*, vol. 12, 2022, doi: https://doi.org/10.1038/s41598-022-24840-z.
[165] Ch, L.K., Kamboj, V.K. & Bath, S.K., «Hybridizing slime mould algorithm with simulated annealing algorithm: a hybridized statistical approach for numerical and engineering design problems», *Complex & Intelligent Systems*, vol. 9, pp. 1525–1582, 2023, doi: https://doi.org/10.1007/s40747-022-00852-0.
[166] Ye Chun, Xu Hua, Chen Qi & Ye Xin Yao, «Improved marine predators algorithm for engineering design optimization problems», *Scientific Reports*, vol. 14, 2024, doi: https://doi.org/10.1038/s41598-024-63826-x.
[167] Hassan, M.H., Kamel, S. & Mohamed, A.W., «Enhanced gorilla troops optimizer powered by marine predator algorithm: global optimization and engineering design», *Scientific Reports*, vol. 14, 2024, doi: https://doi.org/10.1038/s41598-024-57098-8.
[168] D. H. Wolpert and W. G. Macready, «No free lunch theorems for optimization», *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 67–82, 1997, doi: 10.1109/4235.585893.
[169] Ho, Y., Pepyne, D., «Simple Explanation of the No-Free-Lunch Theorem and Its Implications», *Journal of Optimization Theory and Applications*, vol. 115, pp. 549–570, 2002, doi: https://doi.org/10.1023/A:1021251113462.

[170] C. Furio, L. Lamberti, e C. I. Pruncu, «Mechanical and Civil Engineering Optimization with a Very Simple Hybrid Grey Wolf—JAYA Metaheuristic Optimizer», *Mathematics*, vol. 12, fasc. 22, p. 3464, nov. 2024, doi: 10.3390/math12223464.
[171] Mohammad H. Nadimi-Shahraki, Shokooh Taghian, Seyedali Mirjalili, Hoda Zamani, Ardeshir Bahreininejad, «GGWO: Gaze cues learning-based grey wolf optimizer and its applications for solving engineering problems», *Journal of Computational Science*, vol. 61, 2022, doi: https://doi.org/10.1016/j.jocs.2022.101636.
[172] Hsing-Chih Tsai, Jun-Yang Shi, «Potential corrections to grey wolf optimizer», *Applied Soft Computing*, vol. 161, 2024, doi: https://doi.org/10.1016/j.asoc.2024.111776.
[173] Zitar RA, Al-Betar MA, Awadallah MA, Doush IA, Assaleh K., «An Intensive and Comprehensive Overview of JAYA Algorithm, its Versions and Applications», *Archives of Computational Methods in Engineering*, vol. 29, pp. 763–792, 2022, doi: 10.1007/s11831-021-09585-8.
[174] Luiza Scapinello Aquino da Silva, Yan Lieven Souza Lúcio, Leandro dos Santos Coelho, Viviana Cocco Mariani & Ravipudi Venkata Rao, «A comprehensive review on Jaya optimization algorithm», *Artificial Intelligence Review*, vol. 56, pp. 4329–4361, 2023, doi: https://doi.org/10.1007/s10462-022-10234-0.
[175] S.O. Degertekin, L. Lamberti, I.B. Ugur, «Discrete sizing/layout/topology optimization of truss structures with an advanced Jaya algorithm», *Applied Soft Computing*, vol. 79, pp. 363–390, 2019, doi: https://doi.org/10.1016/j.asoc.2019.03.058.
[176] Chakraborty, U.K., «Semi-Steady-State Jaya Algorithm for Optimization», *Applied Sciences*, vol. 10, 2020, doi: https://doi.org/10.3390/app10155388.
[177] Yiying Zhang, Aining Chi, Seyedali Mirjalili, «Enhanced Jaya algorithm: A simple but efficient optimization method for constrained engineering design problems», *Knowledge-Based Systems*, vol. 233, 2021, doi: https://doi.org/10.1016/j.knosys.2021.107555.
[178] Guangcai Zhang, Chunfeng Wan, Songtao Xue, Liyu Xie, «A global-local hybrid strategy with adaptive space reduction search method for structural health monitoring», *Applied Mathematical Modelling*, vol. 121, pp. 231–251, 2023, doi: https://doi.org/10.1016/j.apm.2023.04.025.
[179] Trojovský,P., «A new human-based metaheuristic algorithm for solving optimization problems based on preschool education», *Scientific Reports*, vol. 13, 2023, doi: https://doi.org/10.1038/s41598-023-48462-1.
[180] Wei-Neng Chen; Jun Zhang; Ying Lin; Ni Chen; Zhi-Hui Zhan; Henry Shu-Hung Chung, «Particle Swarm Optimization With an Aging Leader and Challengers», *IEEE Transactions on Evolutionary Computation*, vol. 17, pp. 241–258, 2013, doi: 10.1109/TEVC.2011.2173577.
[181] Jinwei Qiao, Guangyuan Wang, Zhi Yang, Xiaochuan Luo, Jun Chen, Kan Li & Pengbo Liu, «A hybrid particle swarm optimization algorithm for solving engineering problem», *Scientific Reports*, vol. 14, 2024, doi: https://doi.org/10.1038/s41598-024-59034-2.
[182] Nikolaus Hansen, Sibylle D. Müller, Petros Koumoutsakos, «Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES)», *Evolutionary Computation*, vol. 11, pp. 1–18, 2003, doi: https://doi.org/10.1162/106365603321828970.
[183] R. Tanabe and A. S. Fukunaga, «Improving the search performance of SHADE using linear population size reduction», in *IEEE Congress on Evolutionary Computation (CEC)*, Beijing, China, pp. 1658–1665. doi: 10.1109/CEC.2014.6900380.
[184] N. H. Awad, M. Z. Ali and P. N. Suganthan, «Ensemble sinusoidal differential covariance matrix adaptation with Euclidean neighborhood for solving CEC2017 benchmark problems», presentato al 2017 IEEE Congress on Evolutionary Computation (CEC), Donostia, Spain, 2017, pp. 372–379. doi: 10.1109/CEC.2017.7969336.
[185] A. W. Mohamed, A. A. Hadi, A. M. Fattouh and K. M. Jambi, «LSHADE with semi-parameter adaptation hybrid with CMA-ES for solving CEC 2017 benchmark problems», presentato al

2017 IEEE Congress on Evolutionary Computation (CEC), Donostia, Spain, giu. 2017, pp. 145–152. doi: 10.1109/CEC.2017.7969307.

[186] Farid MiarNaeimi, Gholamreza Azizyan & Mohsen Rashki, «Multi-level cross entropy optimizer (MCEO): an evolutionary optimization algorithm for engineering problems», *Engineering with Computers*, vol. 34, pp. 719–739, 2018, doi: https://doi.org/10.1007/s00366-017-0569-z.

[187] Golamreza Azizyan,Farid Miarnaeimi, Mohsen Rashki and Naser Shabakhty, «Flying Squirrel Optimizer (FSO): A novel SI-based optimization algorithm for engineering problems», *Iranian Journal of Optimization*, vol. 11, pp. 177–205, 2019.

[188] Hasancebi,O.;Erdal,F.;Saka,M.P., «Adaptive Harmony Search Method for Structural Optimization», *Journal of Structural Engineering*, vol. 136, pp. 419–431, 2010, doi: https://doi.org/10.1061/(ASCE)ST.1943-541X.0000128.

[189] KazemzadehA-zad,S.;Hasancebi,O.;KazemzadehAzad,S.;Erol,O.K., «Upper Bound Strategy in Optimum Design of Truss Structures: A Big Bang-Big Crunch Algorithm Based Application», *Advances in Structural Engineering*, vol. 16, pp. 1035–1046, 2013, doi: https://doi.org/10.1260/1369-4332.16.6.1035.

[190] Amer Draa, Samira Bouzoubia, Imene Boukhalfa, «A sinusoidal differential evolution algorithm for numerical optimisation», *Applied Soft Computing*, vol. 27, pp. 99–126, 2015, doi: https://doi.org/10.1016/j.asoc.2014.11.003.

[191] Jasbir S. Arora, *Introduction to Optimum Design*, 3rd Edition. Boston, MA: Academic Press, 2012. [Online]. Disponibile su: https://archive.org/details/introductiontoop0000aror/page/884/mode/2up

[192] K. M. Sallam, S. M. Elsayed, R. K. Chakrabortty and M. J. Ryan, «Improved Multi-operator Differential Evolution Algorithm for Solving Unconstrained Problems», presentato al 2020 IEEE Congress on Evolutionary Computation (CEC), Glasgow, UK, lug. 2020, pp. 1–8. doi: 10.1109/CEC48606.2020.9185577.

[193] Huy Tang, Jaehong Lee, «Adaptive initialization LSHADE algorithm enhanced with gradient-based repair for real-world constrained optimization», *Knowledge-Based Systems*, 2022, doi: https://doi.org/10.1016/j.knosys.2022.108696.

[194] Shubham Gupta, Hammoudi Abderazek, Betül Sultan Yıldız, Ali Riza Yildiz, Seyedali Mirjalili, Sadiq M. Sait, «Comparison of metaheuristic optimization algorithms for solving constrained mechanical design optimization problems», *Expert Systems with Applications*, vol. 183, 2021, doi: https://doi.org/10.1016/j.eswa.2021.115351.

[195] Chakraborty,S.;Saha,A.K.;Sharma,S.;Sahoo,S.K.;Pal,G., «Comparative Performance Analysis of Differential Evolution Variants on Engineering Design Problems», *Journal of Bionic Engineering*, vol. 19, pp. 1140–1160, 2022, doi: https://doi.org/10.1007/s42235-022-00190-4.

[196] Abhishek Kumar, Swagatam Das, Ivan Zelinka, «A self-adaptive spherical search algorithm for real-world constrained optimization problems», in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, in GECCO '20. Cancún,Mexico: Association for Computing Machinery, lug. 2020, pp. 13–14. doi: https://doi.org/10.1145/3377929.3398186.

[197] Dhirendra Sharma, Syeda Darakhshan Jabeen, «Hybridizing interval method with a heuristic for solving real-world constrained engineering optimization problems», *Structures*, vol. 56, 2023, doi: https://doi.org/10.1016/j.istruc.2023.104993.

[198] Uray, E.; Carbas, S.; Geem, Z.W.; Kim, S., «Parameters Optimization of Taguchi Method Integrated Hybrid Harmony Search Algorithm for Engineering Design Problems», *Mathematics*, vol. 10, 2022, doi: https://doi.org/10.3390/math10030327.

[199] Paul,H.;Tay,A.O., «Optimal design of an industrial refrigeration system», in *Proceedings of the International Conference on Optimization Techniques and Applications*, Singapore, apr. 1987, pp. 427–435. [Online]. Disponibile su: https://www.gams.com/latest/noalib_ml/libhtml/noalib_refrigeration.html

[200] Alexandros Tsipianitis, Yiannis Tsompanakis, «Improved Cuckoo Search algorithmic variants for constrained nonlinear optimization», *Advances in Engineering Software*, vol. 149, 2020, doi: https://doi.org/10.1016/j.advengsoft.2020.102865.
[201] Abhishek Kumar, Guohua Wu, Mostafa Z. Ali, Rammohan Mallipeddi, Ponnuthurai Nagaratnam Suganthan, Swagatam Das, «A test-suite of non-convex constrained optimization problems from the real-world and some baseline results», *Swarm and Evolutionary Computation*, vol. 56, 2020, doi: https://doi.org/10.1016/j.swevo.2020.100693.
[202] A. Sciammarella, Federico M. Sciammarella, *Experimental Mechanics of Solids*. Chichester,UK: Wiley, 2012.
[203] C. Sciammarella, L. Lamberti, e A. Boccaccio, «General model for moiré contouring, part 1: theory», *Optical Engineering*, vol. 47, 2008, doi: https://doi.org/10.1117/1.2899039.
[204] Cesar A. Sciammarella, Luciano Lamberti, Antonio Boccaccio, Eustachio Cosola, and Domenico Posa, «General model for moiré contouring, part 2: applications», *Optical Engineering*, vol. 47, 2008, doi: https://doi.org/10.1117/1.2899040.
[205] General Stress Optics, Inc., *Holo-Moiré Strain Analyzer (HoloStrain)*. (2013). General Stress Optics, Inc., Chicago, IL, USA, 2013. Consultato: 20 settembre 2024. [Online]. Disponibile su: http://www.stressoptics.com
[206] Alexandros Tzanetos, Maude Blondin, «A qualitative systematic review of metaheuristics applied to tension/compression spring design problem: Current situation, recommendations, and research direction», *Engineering Applications of Artificial Intelligence*, vol. 118, 2023, doi: https://doi.org/10.1016/j.engappai.2022.105521.
[207] C. Furio, L. Lamberti, e C. I. Pruncu, «An Efficient and Fast Hybrid GWO-JAYA Algorithm for Design Optimization», *Applied Sciences*, vol. 14, fasc. 20, p. 9610, ott. 2024, doi: 10.3390/app14209610.
[208] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, e H. Chen, «Harris hawks optimization: Algorithm and applications», *Future Generation Computer Systems*, vol. 97, pp. 849–872, ago. 2019, doi: 10.1016/j.future.2019.02.028.
[209] Tu, B.; Wang, F.; Huo, Y.; Wang, X., «A hybrid algorithm of grey wolf optimizer and harris hawks optimization for solving global optimization problems with improved convergence performance», *Scientific Reports*, vol. 13, 2023, doi: https://doi.org/10.1038/s41598-023-49754-2.
[210] Ł. Knypiński, R. Devarapalli, e F. Gillon, «The hybrid algorithms in constrained optimization of the permanent magnet motors», *IET Science Measure & Tech*, vol. 18, fasc. 9, pp. 455–461, nov. 2024, doi: 10.1049/smt2.12185.
[211] Sharma, I.; Kumar, V.; Sharma, S., «A comprehensive survey on grey wolf optimization», *Recent Advances in Computer Science and Communications*, vol. 15, pp. 323–333, 2022, doi: 10.2174/2666255813999201007165454.
[212] Mohammad H. Nadimi-Shahraki, Shokooh Taghian, Seyedali Mirjalili, «An improved grey wolf optimizer for solving engineering problems», *Expert Systems with Applications*, vol. 166, 2021, doi: https://doi.org/10.1016/j.eswa.2020.113917.
[213] Li K, Li S, Huang Z, Zhang M, Xu Z., «Grey Wolf Optimization algorithm based on Cauchy-Gaussian mutation and improved search strategy», *Scientific Reports*, vol. 12, 2022, doi: 10.1038/s41598-022-23713-9.
[214] Shuidong Ma, Yiming Fang, Xiaodong Zhao, Zhendong Liu, «Multi-swarm improved Grey Wolf Optimizer with double adaptive weights and dimension learning for global optimization problems», *Mathematics and Computers in Simulation*, vol. 205, pp. 619–641, 2023, doi: https://doi.org/10.1016/j.matcom.2022.10.007.
[215] Qiu Y, Yang X, Chen S., «An improved gray wolf optimization algorithm solving to functional optimization and engineering design problems.», *Scientific Reports*, vol. 14, 2024, doi: 10.1038/s41598-024-64526-2.

[216] Zhang Y, Cai Y., «Adaptive dynamic self-learning grey wolf optimization algorithm for solving global optimization problems and engineering problems», *Mathematical Biosciences and Engineering*, vol. 21, 2024, doi: 10.3934/mbe.2024174.
[217] Narinder Singh, S. B. Singh, «Hybrid Algorithm of Particle Swarm Optimization and Grey Wolf Optimizer for Improving Convergence Performance», *Journal of Applied Mathematics*, vol. 2017, 2017, doi: https://doi.org/10.1155/2017/2030489.
[218] Xinming Zhang, Qiang Kang, Jinfeng Cheng, Xia Wang, «A novel hybrid algorithm based on Biogeography-Based Optimization and Grey Wolf Optimizer», *Applied Soft Computing*, vol. 67, pp. 197–214, 2018, doi: https://doi.org/10.1016/j.asoc.2018.02.049.
[219] Wang, JS., Li, SX, «An Improved Grey Wolf Optimizer Based on Differential Evolution and Elimination Mechanism», *Scientific Reports*, vol. 9, 2019, doi: https://doi.org/10.1038/s41598-019-43546-3.
[220] Qian Yang, Jinchuan Liu, Zezhong Wu, Shengyu He, «A fusion algorithm based on whale and grey wolf optimization algorithm for solving real-world optimization problems», *Applied Soft Computing*, vol. 146, 2023, doi: https://doi.org/10.1016/j.asoc.2023.110701.
[221] Optimization design of brushless DC motor based on improved JAYA algorithm, «84. Cheng, Y.; Lyu, X.; Mao, S.», *Scientific Reports*, vol. 14, 2024, doi: 10.1038/s41598-024-54582-z.
[222] Gholami, K.; Olfat, H.; Gholami, J., «An intelligent hybrid JAYA and crow search algorithms for optimizing constrained and unconstrained problems», *Soft Computing*, vol. 25, pp. 14393–14411, 2021, doi: https://doi.org/10.1007/s00500-021-06205-2.
[223] Gholami, J.; Nia, F.A.; Sanatifar, M.; Zawbaa, H.M, «Effective hybridization of JAYA and teaching–learning-based optimization algorithms for numerical function optimization.», *Soft Computing*, vol. 27, 2023, doi: https://doi.org/10.1007/s00500-023-08201-0.
[224] Ibrahim Mlaouhi,Najeh Ben Guedria & Chokri Bouraoui, «An efficient hybrid differential evolution–Jaya algorithm for enhancing vibration behaviour in automotive turbocharger systems», *Engineering Optimization*, vol. 56, pp. 1493–1515, 2023, doi: https://doi.org/10.1080/0305215X.2023.2260992.
[225] Wilson, E.M., *Engineering hydrology*. London: MacMillan, 1974. [Online]. Disponibile su: https://doi.org/10.1007/978-1-349-02417-9
[226] Yeou-Koung Tung, A. M. ASCE, «River Flood Routing by Nonlinear Muskingum Method», *Journal of Hydraulic Engineering*, 1985. [Online]. Disponibile su: https://doi.org/10.1061/(ASCE)0733-9429(1985)111:12(1447)
[227] W. Wang, W. Tian, D. Xu, K. Chau, Q. Ma, e C. Liu, «Muskingum Models' Development and their Parameter Estimation: A State-of-the-art Review», *Water Resources Management*, vol. 37, fasc. 8, pp. 3129–3150, giu. 2023, doi: 10.1007/s11269-023-03493-1.
[228] Luo, Jungang, Yang, Xue, e Xie, Jiancang, «Evaluation and Improvement of Routing Procedure for Nonlinear Muskingum Models», *International Journal of Civil Engineering*, pp. 47–59, 2016.
[229] X. Yuan, X. Wu, H. Tian, Y. Yuan, e R. M. Adnan, «Parameter Identification of Nonlinear Muskingum Model with Backtracking Search Algorithm», *Water Resources Management*, vol. 30, fasc. 8, pp. 2767–2783, giu. 2016, doi: 10.1007/s11269-016-1321-y.
[230] Geem, Z. W., «Issues in optimal parameter estimation for the nonlinear Muskingum flood routing model», *Engineering Optimization*, vol. 46, pp. 328–339, 2013, doi: https://doi.org/10.1080/0305215X.2013.768242.
[231] Reza Barati, «Parameter Estimation of Nonlinear Muskingum Models Using Nelder-Mead Simplex Algorithm», *Journal of Hydrologic Engineering*, p. 16 (11), 2011.
[232] L. Gu, R.J. Yang, C.H. Tho, M. Makowskit, O. Faruquet, Y.Li, «Optimisation and robustness for crashworthiness of side impact», *International Journal of Vehicle Design (IJVD), Vol. 26, No. 4*, 2001. [Online]. Disponibile su: https://doi.org/10.1504/IJVD.2001.005210

[233] Byeng D. Youn, Kyung K. Choi, «A new response surface methodology for reliability-based design optimization», *Computers & Structures*, vol. 82, pp. 241–256, 2004, doi: https://doi.org/10.1016/j.compstruc.2003.09.002.
[234] B. D. Youn, K. K. Choi, R.-J. Yang, e L. Gu, «Reliability-based design optimization for crashworthiness of vehicle side impact», *Structural and Multidisciplinary Optimization*, vol. 26, fasc. 3, pp. 272–283, feb. 2004, doi: 10.1007/s00158-003-0345-0.
[235] E. Pouriyanezhad, H. Rahami, e S. M. Mirhosseini, «Truss optimization using eigenvectors of the covariance matrix», *Engineering with Computers*, vol. 37, fasc. 3, pp. 2207–2224, lug. 2021, doi: 10.1007/s00366-020-00943-x.
[236] Hasan Tahsin Öztürk, Hamdi Tolga Kahraman, «Meta-heuristic search algorithms in truss optimization: Research on stability and complexity analyses», *Applied Soft Computing*, vol. 145, 2023, doi: https://doi.org/10.1016/j.asoc.2023.110573.
[237] Juliano Pierezan, Leandro dos Santos Coelho, Viviana Cocco Mariani, Emerson Hochsteiner de Vasconcelos Segundo, Doddy Prayogo, «Chaotic coyote algorithm applied to truss optimization problems», *Computers & Structures*, vol. 242, 2021, doi: https://doi.org/10.1016/j.compstruc.2020.106353.
[238] Kim, T.-H.; Byun, J.-I., «Truss Sizing Optimization with a Diversity-Enhanced Cyclic Neighborhood Network Topology Particle Swarm Optimizer», *Mathematics*, vol. 8, 2020, doi: https://doi.org/10.3390/math8071087.
[239] Pham, Vu Hong Son, Nguyen Dang, Nghiep Trinh, Nguyen, Van Nam, «Efficient Truss Design: A Hybrid Geometric Mean Optimizer for Better PerformanceView article page», *Applied Computational Intelligence and Soft Computing*, 2024, doi: https://doi.org/10.1155/2024/4216718.
[240] A. Khajeh, A. Kiani, M. Seraji, e H. Dashti, «Weight minimization of truss structures using an improved Harris hawks optimization algorithm», *Innovative Infrastructure Solutions*, vol. 8, fasc. 4, p. 112, mar. 2023, doi: 10.1007/s41062-023-01055-3.
[241] M. Yassami, P. Ashtari, «PSOHHO Hybrid Optimization Algorithm for Truss Optimization», *AUT Journal of Civil Engineering*, 2022. [Online]. Disponibile su: https://ajce.aut.ac.ir/article_5099_5e95610e68ea2b363b7b5b4c4f0757c2.pdf
[242] S. R. Massah e H. Ahmadi, «Weight Optimization of Truss Structures by the Biogeography-Based Optimization Algorithms», *Civil Engineering Infrastructures Journal*, vol. 54, fasc. 1, pp. 129–144, 2021, doi: 10.22059/ceij.2021.291862.1625.
[243] Kaveh, A. e Zakian, P., «Improved GWO algorithm for optimal design of truss structures», *Engineering with Computers*, vol. 34, pp. 685–707, 2018, doi: https://doi.org/10.1007/s00366-017-0567-1.
[244] Rafiq Awad, «Sizing optimization of truss structures using the political optimizer (PO) algorithm», *Structures*, vol. 33, pp. 4871–4894, 2021, doi: https://doi.org/10.1016/j.istruc.2021.07.027.
[245] Mendes Platt, G.; Yang, X.-S.; Silva Neto, A.J., *Optimization and Inverse Problems with Applications in Engineering*, vol. Computational Intelligence. Switzerland: Springer, 2019. [Online]. Disponibile su: https://doi.org/10.1007/978-3-319-96433-1
[246] Abd Elaziz, Mohamed, Elsheikh, Ammar H., Oliva, Diego, Abualigah, Laith, Lu, Songfeng, e Ewees, Ahmed A., «Advanced Metaheuristic Techniques for Mechanical Design Problems: Review», *Archives of Computational Methods in Engineering*, vol. 29, 2022, doi: https://doi.org/10.1007/s11831-021-09589-4.
[247] Turgut, O.E.; Turgut, M.S.; Kırtepe, E., «A systematic review of the emerging metaheuristic algorithms on solving complex op-timization problems», *Neural Computing and Applications*, vol. 35, 2023, doi: https://doi.org/10.1007/s00521-023-08481-5.
[248] L. M. Q. Abualigah, A c. di, *METAHEURISTIC OPTIMIZATION ALGORITHMS: optimizers,analysis, and applications*. Cambridge, MA: MORGAN KAUFMANN, 2024.

[249] E. Cuevas, A. Chavarin-Fajardo, C. Ascencio-Piña, e S. Garcia-De-Lira, *Optimization Strategies: A Decade of Metaheuristic Algorithm Development*, 1st ed. 2025. in Intelligent Systems Reference Library, no. 266. Cham: Springer Nature Switzerland, 2025. doi: 10.1007/978-3-031-81013-8.

[250] Karam M. Sallam, Saber M. Elsayed, Ripon K. Chakrabortty, Michael J. Ryan, «Multi-Operator Differential Evolution Algorithm for Solving Real-World Constrained Optimization Problems», presentato al 2020 IEEE Congress on Evolutionary Computation (CEC 2020), Virtual, Glasgow, United Kingdom: Yaochu Jin, lug. 2020, pp. 1–8.

[251] J. Gurrola-Ramos, A. Hernandez-Aguirre, e O. Dalmau-Cedeno, «COLSHADE for Real-World Single-Objective Constrained optimization Problems», in *2020 IEEE Congress on Evolutionary Computation (CEC)*, Glasgow, United Kingdom: IEEE, lug. 2020, pp. 1–8. doi: 10.1109/CEC48606.2020.9185583.

[252] Z. Yuan *et al.*, «An improved multi-operator differential evolution via a knowledge-guided information sharing strategy for global optimization», *Expert Systems with Applications*, vol. 269, p. 126403, apr. 2025, doi: 10.1016/j.eswa.2025.126403.

[253] J. Lee, R. Mendoza, V. M. P. Mendoza, e E. Jung, «Differential evolution with spherical search algorithm for nonlinear engineering and infectious disease optimization problems», *Applied Soft Computing*, vol. 168, p. 112446, gen. 2025, doi: 10.1016/j.asoc.2024.112446.

[254] G. Hu, Y. Guo, J. Zhong, e G. Wei, «IYDSE: Ameliorated Young's double-slit experiment optimizer for applied mechanics and engineering», *Computer Methods in Applied Mechanics and Engineering*, vol. 412, p. 116062, lug. 2023, doi: 10.1016/j.cma.2023.116062.

[255] G. Hu, C. Gong, X. Li, e Z. Xu, «CGKOA: An enhanced Kepler optimization algorithm for multi-domain optimization problems», *Computer Methods in Applied Mechanics and Engineering*, vol. 425, p. 116964, mag. 2024, doi: 10.1016/j.cma.2024.116964.

[256] B. Shu, G. Hu, M. Cheng, e C. Zhang, «MSFPSO: Multi-algorithm integrated particle swarm optimization with novel strategies for solving complex engineering design problems», *Computer Methods in Applied Mechanics and Engineering*, vol. 437, p. 117791, mar. 2025, doi: 10.1016/j.cma.2025.117791.

[257] G. You, Y. Hu, Z. Yang, e Y. Li, «Enhanced snow ablation optimizer using dynamic tangential flight and elite guidance strategy», *Sci Rep*, vol. 15, fasc. 1, p. 10036, mar. 2025, doi: 10.1038/s41598-025-93410-w.

[258] S. Fu, H. Huang, C. Ma, J. Wei, Y. Li, e Y. Fu, «Improved dwarf mongoose optimization algorithm using novel nonlinear control and exploration strategies», *Expert Systems with Applications*, vol. 233, p. 120904, dic. 2023, doi: 10.1016/j.eswa.2023.120904.

[259] Z. Wang, D. Dai, Z. Zeng, D. He, e S. Chan, «Multi-strategy enhanced Grey Wolf Optimizer for global optimization and real world problems», *Cluster Comput*, vol. 27, fasc. 8, pp. 10671–10715, nov. 2024, doi: 10.1007/s10586-024-04455-x.

[260] A. Naik, «Marine predators social group optimization: a hybrid approach», *Evolutionary Intelligence*, vol. 17, fasc. 4, pp. 2355–2386, ago. 2024, doi: 10.1007/s12065-023-00891-7.

[261] S. Kaur, L. Kaur, e M. Lal, «Artificial hummingbird algorithm with chaotic-opposition-based population initialization for solving real-world problems», *Neural Computing and Applications*, vol. 37, fasc. 27, pp. 22529–22572, set. 2025, doi: 10.1007/s00521-024-10621-4.

[262] C. Zhang *et al.*, «Piranha predation optimization algorithm (PPOA) for global optimization and engineering design problems», *Applied Soft Computing*, vol. 165, p. 112085, nov. 2024, doi: 10.1016/j.asoc.2024.112085.

[263] J. C. Ankrah, F. B. Effah, e E. Twumasi, «An Enhanced Semisteady-State Jaya Algorithm With a Control Coefficient and a Self-Adaptive Multipopulation Strategy», *Journal of Electrical and Computer Engineering*, vol. 2025, fasc. 1, p. 3036909, gen. 2025, doi: 10.1155/jece/3036909.

[264] M. Abdel-Basset, R. Mohamed, S. A. A. Azeem, M. Jameel, e M. Abouhawwash, «Kepler optimization algorithm: A new metaheuristic algorithm inspired by Kepler's laws of planetary

motion», *Knowledge-Based Systems*, vol. 268, p. 110454, mag. 2023, doi: 10.1016/j.knosys.2023.110454.
[265] M. Azizi, U. Aickelin, H. A. Khorshidi, e M. Baghalzadeh Shishehgarkhaneh, «Energy valley optimizer: a novel metaheuristic algorithm for global and engineering optimization», *Sci Rep*, vol. 13, fasc. 1, p. 226, gen. 2023, doi: 10.1038/s41598-022-27344-y.
[266] K. Li, H. Huang, S. Fu, C. Ma, Q. Fan, e Y. Zhu, «A multi-strategy enhanced northern goshawk optimization algorithm for global optimization and engineering design problems», *Computer Methods in Applied Mechanics and Engineering*, vol. 415, p. 116199, ott. 2023, doi: 10.1016/j.cma.2023.116199.
[267] Y. Zhao, C. Huang, M. Zhang, e C. Lv, «COLMA: a chaos-based mayfly algorithm with opposition-based learning and Levy flight for numerical optimization and engineering design», *The Journal of Supercomputing*, vol. 79, fasc. 17, pp. 19699–19745, nov. 2023, doi: 10.1007/s11227-023-05400-2.
[268] K. Wang, M. Guo, C. Dai, e Z. Li, «A novel heuristic algorithm for solving engineering optimization and real-world problems: People identity attributes-based information-learning search optimization», *Computer Methods in Applied Mechanics and Engineering*, vol. 416, p. 116307, nov. 2023, doi: 10.1016/j.cma.2023.116307.
[269] L. Wu, J. Wu, e T. Wang, «The improved grasshopper optimization algorithm with Cauchy mutation strategy and random weight operator for solving optimization problems», *Evolutionary Intelligence*, vol. 17, fasc. 3, pp. 1751–1781, giu. 2024, doi: 10.1007/s12065-023-00861-z.
[270] T. M. Luan, S. Khatir, M. T. Tran, B. De Baets, e T. Cuong-Le, «Exponential-trigonometric optimization algorithm for solving complicated engineering problems», *Computer Methods in Applied Mechanics and Engineering*, vol. 432, p. 117411, dic. 2024, doi: 10.1016/j.cma.2024.117411.
[271] Y. Fu, D. Liu, J. Chen, e L. He, «Secretary bird optimization algorithm: a new metaheuristic for solving global optimization problems», *Artificial Intelligence Review*, vol. 57, fasc. 5, p. 123, apr. 2024, doi: 10.1007/s10462-024-10729-y.
[272] Y. Lang e Y. Gao, «Dream Optimization Algorithm (DOA): A novel metaheuristic optimization algorithm inspired by human dreams and its applications to real-world engineering problems», *Computer Methods in Applied Mechanics and Engineering*, vol. 436, p. 117718, mar. 2025, doi: 10.1016/j.cma.2024.117718.
[273] Y. Xiao, H. Cui, R. A. Khurma, e P. A. Castillo, «Artificial lemming algorithm: a novel bionic meta-heuristic technique for solving real-world engineering optimization problems», *Artificial Intelligence Review*, vol. 58, fasc. 3, p. 84, gen. 2025, doi: 10.1007/s10462-024-11023-7.
[274] C. Zhong, G. Li, Z. Meng, H. Li, A. R. Yildiz, e S. Mirjalili, «Starfish optimization algorithm (SFOA): a bio-inspired metaheuristic algorithm for global optimization compared with 100 optimizers», *Neural Computing and Applications*, vol. 37, fasc. 5, pp. 3641–3683, feb. 2025, doi: 10.1007/s00521-024-10694-1.
[275] X. Cai, W. Wang, e Y. Wang, «Multi-strategy enterprise development optimizer for numerical optimization and constrained problems», *Sci Rep*, vol. 15, fasc. 1, p. 10538, mar. 2025, doi: 10.1038/s41598-025-93754-3.
[276] J. Wang, H. Ouyang, S. Li, W. Ding, e L. Gao, «Equilibrium optimizer-based harmony search algorithm with nonlinear dynamic domains and its application to real-world optimization problems», *Artif Intell Rev*, vol. 57, fasc. 7, p. 166, giu. 2024, doi: 10.1007/s10462-024-10793-4.
[277] K. Deb, «An efficient constraint handling method for genetic algorithms», *Computer Methods in Applied Mechanics and Engineering*, vol. 186, fasc. 2–4, pp. 311–338, giu. 2000, doi: 10.1016/S0045-7825(99)00389-8.
[278] C. A. Coello Coello, «Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art», *Computer Methods in Applied*

Mechanics and Engineering, vol. 191, fasc. 11–12, pp. 1245–1287, gen. 2002, doi: 10.1016/S0045-7825(01)00323-1.

[279] J. J. Liang, A. K. Qin, P. N. Suganthan, e S. Baskar, «Comprehensive learning particle swarm optimizer for global optimization of multimodal functions», *IEEE Trans. Evol. Computat.*, vol. 10, fasc. 3, pp. 281–295, giu. 2006, doi: 10.1109/TEVC.2005.857610.

[280] S. Kaur, L. K. Awasthi, A. L. Sangal, e G. Dhiman, «Tunicate Swarm Algorithm: A new bio-inspired based metaheuristic paradigm for global optimization», *Engineering Applications of Artificial Intelligence*, vol. 90, p. 103541, apr. 2020, doi: 10.1016/j.engappai.2020.103541.

[281] M. G. H. Omran e S. Al-Sharhan, «Improved continuous Ant Colony Optimization algorithms for real-world engineering optimization problems», *Engineering Applications of Artificial Intelligence*, vol. 85, pp. 818–829, ott. 2019, doi: 10.1016/j.engappai.2019.08.009.

[282] A. H. Gandomi e A. H. Alavi, «Krill herd: A new bio-inspired optimization algorithm», *Communications in Nonlinear Science and Numerical Simulation*, vol. 17, fasc. 12, pp. 4831–4845, dic. 2012, doi: 10.1016/j.cnsns.2012.05.010.

[283] A. Banharnsakun, T. Achalakul, e B. Sirinaovakul, «The best-so-far selection in Artificial Bee Colony algorithm», *Applied Soft Computing*, vol. 11, fasc. 2, pp. 2888–2901, mar. 2011, doi: 10.1016/j.asoc.2010.11.025.

[284] H. Peraza-Vázquez, A. Peña-Delgado, M. Merino-Treviño, A. B. Morales-Cepeda, e N. Sinha, «A novel metaheuristic inspired by horned lizard defense tactics», *Artificial Intelligence Review*, vol. 57, fasc. 3, p. 59, feb. 2024, doi: 10.1007/s10462-023-10653-7.

[285] J. O. Agushaka, A. E. Ezugwu, e L. Abualigah, «Dwarf Mongoose Optimization Algorithm», *Computer Methods in Applied Mechanics and Engineering*, vol. 391, p. 114570, mar. 2022, doi: 10.1016/j.cma.2022.114570.

[286] C. Furio, A. Ugenti, G. Reina, M. Foglia and L. Lamberti, «SmartExplorer1.0: a novel intelligent framework for path planning in Robotics», *IEEE Access*, vol. 13, 2025, doi: 10.1109/ACCESS.2025.3577464.

[287] Alenezi, M.R.; Almeshal, A.M., «Optimal Path Planning for a Remote Sensing Unmanned Ground Vehicle in a Hazardous Indoor Environment.», *Intelligent Control and Automation*, vol. 9, pp. 147–157, 2018, doi: 10.4236/ica.2018.94011.

[288] A. Ugenti, Vulpi F, Dominguez R, Cordes F, Milella A, Reina G, «On the role of feature and signal selection for terrain learning in planetary exploration robots», *Journal of Field Robotics*, vol. 39, pp. 355–370, 2022, doi: https://doi.org/10.1002/rob.22054.

[289] Ishigami, G.; Nagatani, K.; Yoshida, K., «Path planning and evaluation for planetary rovers based on dynamic mobility index.», presentato al In Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, CA, USA, set. 2011, pp. 601–606.

[290] Hart, P.; Nilsson, N.; Raphael, B., «A Formal Basis for the Heuristic Determination of Minimum Cost Paths.», *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, pp. 100–107, 1968, doi: 10.1109/TSSC.1968.300136.

[291] Zhang, J., Liu, Z., Wang, Y., Zhang, F., and Li, Y., «Research on effective path planning algorithm based on improved A* algorithm.», *Journal of Physics: Conference Series*, vol. 2188, 2022, doi: 10.1088/1742-6596/2188/1/012014.

[292] Wang H., Qi X., Lou S., Jing J., He H. and Liu W., «An Efficient and Robust Improved A* Algorithm for Path Planning. Symmetry», *Symmetry*, vol. 13, 2021, doi: 10.3390/sym13112213.

[293] Luoyang Luo, «An Improved A* Algorithm for AGV Path Planning.», presentato al 2023 IEEE International Conference on Image Processing and Computer Applications (ICIPCA), Changchun, China: IEEE, 2023, pp. 739–743. doi: 10.1109/ICIPCA59209.2023.10257733.

[294] Kabir, R.; Watanobe, Y.; Islam, M.R.; Naruse, K., «Enhanced Robot Motion Block of A-Star Algorithm for Robotic Path Planning», *Sensors*, vol. 24, 2024, doi: https://doi.org/10.3390/s24051422.

[295] H. Li, «Robotic Path Planning Strategy Based on Improved Artificial Potential Field», presentato al 2020 International Conference on Artificial Intelligence and Computer Engineering (ICAICE), Beijing, China, 2020, pp. 67–71. doi: 10.1109/ICAICE51518.2020.00019.
[296] Hamid Taghavifar, Subhash Rakheja, Giulio Reina, «A novel optimal path-planning and following algorithm for wheeled robots on deformable terrains», *Journal of Terramechanics*, vol. 96, 2021, doi: https://doi.org/10.1016/j.jterra.2020.12.001.
[297] Hao, G.; Lv, Q.; Huang, Z.; Zhao, H.; Chen, W., «UAV Path Planning Based on Improved Artificial Potential Field Method», *Aerospace*, vol. 10, 2023, doi: https://doi.org/10.3390/aerospace10060562.
[298] Xing, T.; Wang, X.; Ding, K.; Ni, K.; Zhou, Q., «Improved Artificial Potential Field Algorithm Assisted by Multisource Data for AUV Path Planning», *Sensors*, vol. 23, 2023, doi: https://doi.org/10.3390/s23156680.
[299] T. Wang, X. Chen, C. Lei and L. Yang, «Artificial Potential Field-Enhanced Deep Reinforcement Learning for Motion Planning», presentato al 8th International Conference on Robotics, Control and Automation (ICRCA), Shanghai, China: IEEE, 2024, pp. 459–465. doi: 10.1109/ICRCA60878.2024.10649312.
[300] J. J. Kuffner and S. M. LaValle, «RRT-connect: An efficient approach to single-query path planning», presentato al IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065), San Francisco, CA, USA: IEEE, 2000, pp. 995–1001. doi: 10.1109/ROBOT.2000.844730.
[301] LaValle SM, Kuffner JJ., «Randomized Kinodynamic Planning», *The International Journal of Robotics Research*, vol. 20, 2001, doi: https://doi.org/10.1177/02783640122067453.
[302] Karaman S, Frazzoli E., «Sampling-based algorithms for optimal motion planning», *The International Journal of Robotics Research*, vol. 30, pp. 846–894, 2011, doi: 10.1177/0278364911406761.
[303] Honghui Fan, Jiahe Huang, Xianzhen Huang, Hongjin Zhu, Huachang Su, «BI-RRT*: An improved path planning algorithm for secure and trustworthy mobile robots systems», *Heliyon*, vol. 10, 2024, doi: 10.1016/j.heliyon.2024.e26403.
[304] A. Leanza, A. Ugenti, R. Galati, and G. Reina, «Predicting the impact of wheel passage on terrain unevenness», *Journal of Terramechanics*, vol. 117, 2025, doi: https://doi.org/10.1016/j.jterra.2024.101015.
[305] E. W. Dijkstra, «A note on two problems in connexion with graphs», *Numerische Mathematik*, vol. 1, pp. 269–271, 1959, doi: https://doi.org/10.1007/BF01386390.
[306] T. H. Cormen, C. E. Leiserson, R. Rivest, and C. Stein, *Introduction to algorithms*. in Introduction to Computational Thinking. Cambridge, MA, USA: MIT Press, 2009. [Online]. Disponibile su: https://books.google.it/books?hl=en&lr=&id=RSMuEAAAQBAJ&oi=fnd&pg=PR13&dq=+Introduction+to+algorithms&ots=a3l80Z4DTO&sig=ih9r_bebALkmdY2ZaGyQ_IFojNE&redir_esc=y#v=onepage&q=Introduction%20to%20algorithms&f=false
[307] Khatib O., «Real-Time Obstacle Avoidance for Manipulators and Mobile Robots», *The International Journal of Robotics Research*, vol. 5, 1986, doi: https://doi.org/10.1177/027836498600500106.
[308] L. E. Kavraki, P. Svestka, J. . -C. Latombe and M. H. Overmars, «Probabilistic roadmaps for path planning in high-dimensional configuration spaces», *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 566–580, 1996, doi: 10.1109/70.508439.
[309] K. Tamura and K. Yasuda, «Spiral Dynamics Inspired Optimization», *Journal of Advanced Computational Intelligence and Intelligent Informatics.Tokyo, Japan*, vol. 15, pp. 1116–1122, 2011, doi: 10.20965/jaciii.2011.p1116.
[310] J. R. Sánchez-Ibáñez, C. J. Pérez-Del-Pulgar, and A. García-Cerezo, «Path Planning for Autonomous Mobile Robots: A Review», *Sensors*, vol. 21, 2021, doi: https://doi.org/10.3390/s21237898.

[311] Ye Zhang, Wang Zhao, Jingyu Wang, Yuan Yuan, «Recent progress, challenges and future prospects of applied deep reinforcement learning : A practical perspective in path planning», *Neurocomputing*, vol. 608, 2024, doi: https://doi.org/10.1016/j.neucom.2024.128423.
[312] R. Jaramillo-Martínez, E. Chavero-Navarrete, and T. Ibarra-Pérez, «Reinforcement-Learning-Based path planning: A reward function strategy», *Applied Sciences*, vol. 14, 2024, doi: https://doi.org/10.3390/app14177654.
[313] Liu, L.; Wang, X.; Yang, X.; Liu, H.; Li, J.; Wang, P., «Path planning techniques for mobile robots: Review and prospect», *Expert Systems with Applications*, vol. 227, 2023, doi: https://doi.org/10.1016/j.eswa.2023.120254.
[314] Li, Z., Zhu, C., & Gold, C., *Digital Terrain Modeling: Principles and Methodology*. Boca Raton: CRC Press, 2004. [Online]. Disponibile su: https://doi.org/10.1201/9780203357132
[315] Mitasova, Helena e Mitas, Lubos, «Multiscale Soil Erosion Simulations For Land Use Management», in *Landscape Erosion and Evolution Modeling*, Boston, MA: Springer, 2001, pp. 321–347. [Online]. Disponibile su: https://doi.org/10.1007/978-1-4615-0575-4_11
[316] T. Gillespie, *Fundamentals of Vehicle Dynamics*. Warrendale, PA, USA: Society of Automotive Engineers, 2021. [Online]. Disponibile su: https://books.google.it/books?hl=en&lr=&id=LeybEAAAQBAJ&oi=fnd&pg=PP1&dq=+Fundamentals+of+Vehicle+Dynamics&ots=WVG-m8UtTA&sig=5zgoczvvVSvo8PGLV3nOIwuBosE&redir_esc=y#v=onepage&q=Fundamentals%20of%20Vehicle%20Dynamics&f=false
[317] Jazar, Reza N., *Vehicle dynamics: theory and application.* Springer Nature, 2025. [Online]. Disponibile su: https://doi.org/10.1007/978-3-031-74458-7
[318] S. Wang and Y. Li, «Dynamic rocker-bogie: Kinematical analysis in a high-speed traversal stability enhancement», *International Journal of Aerospace Engineering*, 2016. [Online]. Disponibile su: https://doi.org/10.1155/2016/5181097
[319] Dean Karnopp, *Vehicle Dynamics, Stability, and Control*. Boca Raton: CRC Press, 2013. [Online]. Disponibile su: https://doi.org/10.1201/b13767
[320] K. Rajwar, K. Deep, and S. Das, «An exhaustive review of the Metaheuristic algorithms for search and optimization: Taxonomy, applications, and open challenges», *Artificial Intelligence Review*, vol. 56, pp. 13187–13257, 2023, doi: 10.1007/s10462-023-10470-y.
[321] S. R. Lawande, G. Jasmine, J. Anbarasi, and L. I. Izhar, «A systematic review and analysis of intelligence-based pathfinding algorithms in the field of video games», *Applied Sciences*, vol. 12, 2022, doi: https://doi.org/10.3390/app12115499.
[322] Ge, S., Cui, Y., «Dynamic Motion Planning for Mobile Robots Using Potential Field Method», *Autonomous Robots*, vol. 13, pp. 207–222, doi: https://doi.org/10.1023/A:1020564024509.
[323] E. Rimon and D. E. Koditschek, «The construction of analytic diffeomorphisms for exact robot navigation on star worlds», presentato al Proceedings, 1989 International Conference on Robotics and Automation, Scottsdale, AZ, USA: IEEE, mag. 1989. doi: 10.1109/ROBOT.1989.99962.
[324] J.-O. Kim and P. K. Khosla, «Real-time obstacle avoidance using harmonic potential functions," in IEEE Transactions on Robotics and Automation», *IEEE Transactions on Robotics and Automation*, vol. 8, pp. 338–349, 1992, doi: 10.1109/70.143352.
[325] D. Zhu and J. C. Latombe, «New heuristic algorithms for efficient hierarchical path planning», *IEEE Transactions on Robotics and Automation*, vol. 7, pp. 9–20, 1991, doi: 10.1109/70.68066.
[326] C. Warren, «Global path planning using artificial potential fields», presentato al Proceedings, 1989 International Conference on Robotics and Automation, Scottsdale, AZ, USA: IEEE, 1989, pp. 316–321. doi: 10.1109/ROBOT.1989.100007.
[327] Gargiulo, A.M.; di Stefano, I.; Genova, A., «Model-Based Slippage Estimation to Enhance Planetary Rover Localization with Wheel Odometry», *Applied Sciences*, vol. 11, 2011, doi: https://doi.org/10.3390/app11125490.

[328] Chao Chen; Shibin Su; Minglei Shu; Chong Di; Weihua Li; Pengyao Xu, «Slippage Estimation via Few-Shot Learning Based on Wheel-Ruts Images for Wheeled Robots on Loose Soil», *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, pp. 6555–6566, 2024, doi: 10.1109/TITS.2023.3344799.
[329] Christopher Brunskill, Nildeep Patel, Thibault P. Gouache, Gregory P. Scott, Chakravarthini M. Saaj, Marcus Matthews, Liang Cui, «Characterisation of martian soil simulants for the ExoMars rover testbed», *Journal of Terramechanics*, vol. 48, 2011, doi: https://doi.org/10.1016/j.jterra.2011.10.001.
[330] Trojnacki, M., «Tracking Control of a Four-Wheeled Skid-Steered Robot with Slip Compensation and Application of the Drive Unit Model», *Electronics*, vol. 14, 2025, doi: https://doi.org/10.3390/electronics14030444.
[331] Shen, Y.; Zou, M.; Cao, H.; Pan, D.; Yuan, B.; He, L., «Current-Based Analysis and Validation of a Wheel–Soil Interaction Model for the Trafficability of a Planetary Rover», *Aerospace*, vol. 11, 2024, doi: https://doi.org/10.3390/aerospace11110892.
[332] Ronnier Frates Rohrich; Luís Felipe Messias; Jose Lima; André Schneider de Oliveira, «Collective Mapping of Gas Leakages to Determine Safe Routes Using Multi-Robot System», *IEEE Access*, vol. 12, pp. 179764–179784, 2024, doi: 10.1109/ACCESS.2024.3503654.
[333] A. Haraldsen, M. Wiig, and K. Y. Pettersen, «A theoretical analysis of the velocity obstacle method for nonholonomic vehicles and underactuated surface vessels», *IEEE Transactions on Control Systems Technology*, vol. 32, pp. 1801–1816, 2024, doi: 10.1109/TCST.2024.3375023.
[334] Simon Fong, Suash Deb, Ankit Chaudhary, «A review of metaheuristics in robotics, Computers & Electrical Engineering», *Computers & Electrical Engineering*, vol. 43, pp. 278–291, 2015, doi: https://doi.org/10.1016/j.compeleceng.2015.01.009.
[335] G. Dantzig, R. Fulkerson, S. Johnson, «Solution of a Large-Scale Traveling-Salesman Problem», *Journal of the Operations Research Society of America*, vol. 2, fasc. 4, pp. 393–410, 1954, doi: http://dx.doi.org/10.1287/opre.2.4.393.
[336] Masiala Mavungu, «Control and Trajectory Planning of an Autonomous Bicycle Robot», *Computation-MDPI*, vol. 10, fasc. 11, p. 194, 2022, doi: https://doi.org/10.3390/computation10110194.
[337] Hugo Barbosa, Marc Barthelemy, Gourab Ghoshal, Charlotte R. James, Maxime Lenormand, Thomas Louail, Ronaldo Menezes, José J. Ramasco, Filippo Simini, Marcello Tomasini, «Human mobility: Models and applications», vol. 734, pp. 1–74, 2018.
[338] B. Lan e Y. Suzuki, «Traveling salesman problem with time windows and a drone-utilizing intermediate points (TSPTWD-IP)», *Computers & Industrial Engineering*, vol. 198, p. 110641, dic. 2024, doi: 10.1016/j.cie.2024.110641.
[339] M. Held e R. Karp, «A Dynamic Programming Approach to Sequencing Problems».
[340] A. R. Saiyed, «The Traveling Salesman problem».
[341] T. Tawanda, P. Nyamugure, S. Kumar, e E. Munapo, «A Labelling Method for the Travelling Salesman Problem», *Applied Sciences*, vol. 13, fasc. 11, p. 6417, mag. 2023, doi: 10.3390/app13116417.
[342] S. P. Tiwari, S. Kumar, e D. K. K. Bansal, «A Survey of Metaheuristic Algorithms for Travelling Salesman Problem», *International Journal Of Engineering Research*, vol. 1, fasc. 5, 2014.
[343] Łukasz Str, Rafał Skinderowicz and Urszula Boryczka and Arkadiusz Nowakowski, «A Self-Adaptive Discrete PSO Algorithm with Heterogeneous Parameter Values for Dynamic TSP», *Entropy*, vol. 21, p. 738, 2019, doi: https://doi.org/10.3390/e21080738.
[344] Y. Zhou, H. Ríos, M. Mera, A. Polyakov, G. Zheng, A. Dzul, «Trajectory Tracking in Unicycle Mobile Robots: A Homogeneity–based Control Approach», *IFAC-PapersOnLine*, vol. 56, fasc. 2, pp. 54–59, 2023, doi: https://doi.org/10.1016/j.ifacol.2023.10.1404.

[345] K. Wang, P. Li, F. Wu and X. -M. Sun, «Switching Anti-Windup Synthesis for Linear Systems With Asymmetric Actuator Saturation», *IEEE Transactions on Cybernetics*, vol. 54, fasc. 6, pp. 3796–3809, 2024, doi: 10.1109/TCYB.2023.3264913.
[346] S. Li, M. Cui, C. K. Ahn, Z. Xiang and I. Ghous, «Positivity and Saturated Stabilization of Singular Switched Positive Systems Under Mode-Dependent Minimum Dwell Time», *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 54, fasc. 3, pp. 1401–1413, 2024.
[347] C. Chekuri and K. Quanrud, «Approximating the Held-Karp Bound for Metric TSP in Nearly-Linear Time», presentato al 2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS), Berkeley, CA, USA, 2017, pp. 789–800. doi: 10.1109/FOCS.2017.78.
[348] Yong Wang, Zunpu Han, «Ant colony optimization for traveling salesman problem based on parameters optimization», *Applied Soft Computing*, vol. 107, 2021.
[349] Xu, L.; You, J.; Yuan, H., «Real-Time Parametric Path Planning Algorithm for Agricultural Machinery Kinematics Model Based on Particle Swarm Optimization», *Agriculture*, vol. 13, 2023, doi: https://doi.org/10.3390/agriculture13101960.
[350] J. V. C. F. de Lima, E. M. Belo, e V. A. da S. Marques, «Multi-agent path planning with nonlinear restrictions», *Evolutionary Intelligence*, vol. 14, fasc. 1, pp. 191–201, mar. 2021, doi: 10.1007/s12065-020-00534-1.
[351] A. Habib e M. Akram, «Optimizing traveling salesman problem using tabu search metaheuristic algorithm with Pythagorean fuzzy uncertainty», *Granular Computing*, vol. 9, fasc. 1, p. 16, gen. 2024, doi: 10.1007/s41066-023-00424-8.
[352] A. Nourmohammadzadeh e S. Voß, «A Hybrid Metaheuristic for the Clustered Travelling Salesman Problem», in *Operations Research Proceedings 2022*, O. Grothe, S. Nickel, S. Rebennack, e O. Stein, A c. di, Cham: Springer International Publishing, 2023, pp. 385–392.
[353] Omar Cheikhrouhou, Ines Khoufi, «A comprehensive survey on the Multiple Traveling Salesman Problem: Applications, approaches and taxonomy, Computer Science Review», *Computer Science Review*, vol. 40, 2021.
[354] L. C. W. Dixon; G. P. Szegö, «The Global Optimization Problem: An Introduction», in *Towards Global Optimization*, vol. 2, Amsterdam: North-Holland, 1978, pp. 1–15.
[355] Victor Picheny, Tobias Wagner, David Ginsbourger, «A benchmark of kriging-based infill criteria for noisy optimization», *Structural and Multidisciplinary Optimization*, vol. 48, 2013, doi: 10.1007/s00158-013-0919-4.
[356] Rastrigin, L. A., *Systems of extremal control*. Moscow: Mir, 1974.
[357] Holland, J. H., *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. MIT Press, 1992.
[358] Momin Jamil, Xin-She Yang, Hans-Jürgen Zepernick, «Test Functions for Global Optimization: A Comprehensive Survey», in *Swarm Intelligence and Bio-Inspired Computation*, Elsevier, Xin-She Yang, Zhihua Cui, Renbin Xiao, Amir Hossein Gandomi, Mehmet Karamanoglu, 2013, pp. 193–222.
[359] Himmelblau, D. M., *Applied nonlinear programming*. McGraw-Hill, 1972.
[360] Ragsdell, K. e Phillips, D., «Optimal design of a class of welded structures using geometric programming», vol. 98, pp. 1021–1025, 1976.
[361] Timoshenko, S. e Gere, J., *Theory of Elastic Stability*. McGraw-Hill: McGraw-Hill, 1961.
[362] Baishya, N.J., Sharma, D. & Dixit, U.S., «Optimization of Pressure Vessel Under Thermo-Elastic Condition», *Journal of The Institution of Engineers (India): Series C*, pp. 389–400, 2014.
[363] Bai Li, Zhijiang Shao, «Simultaneous dynamic optimization: A trajectory planning method for nonholonomic car-like robots», *Advances in Engineering Software*, vol. 87, pp. 30–42, 2015, doi: https://doi.org/10.1016/j.advengsoft.2015.04.011.
[364] A. Gasparetto, V. Zanotto, «Optimal trajectory planning for industrial robots, Advances in Engineering Software», *Advances in Engineering Software*, vol. 41, pp. 548–556, 2010, doi: https://doi.org/10.1016/j.advengsoft.2009.11.001.

APPENDIX 1-MATLAB JAYA CODE APPLIED TO A DAM'S DESIGN

THE MAIN JAYA FILE

The MATLAB code has been divided into three files: one containing the JAYA algorithm code, and the other two files containing, respectively, the objective function and the function incorporating the constraints.

The algorithm can be identified in 7 steps.

1. Initialization:
    - Initialization of design variable bounds, low and up (XL and XU).
    - Initialization of constants and parameters (NPOP, ITLIM, penfact).

2. Jaya Algorithm:
    - Generation of an initial population (XPOP) within the specified bounds.
    - Evaluation of objective function (FUNZOB) and constraints (CONSTRAINTS) for each member of the population.
    - Identification of the best and worst designs in the current population.

3. Iteration Loop:
    - Iterative optimization process (DO iter=1, ITLIM) where new designs are generated, and the population is updated.
    - New designs are generated based on the best and worst designs from the previous iteration using random variations.
    - The fitness of new designs is evaluated, and the population is updated to see if the new designs are better.

4. Output:
    - Print information about each iteration, including the best and worst designs.
    - Final optimal design and associated information.

5. Subroutines:
    - FUNZOB: Evaluates the objective function based on design variables.
    - CONSTRAINTS: Evaluates constraints and updates the total violation measure.

6. File Output:
    - The program opens a file ('ITERATIONS.txt') for writing information about each iteration.

7. Pauses and Stops:
    - pause statements are used for manual inspection during execution.
    - The program ends with a stop.

The optimization goal is to find design variables that minimize the objective function under the specified constraints.

The main file is named MAIN_JAYA_DAM

CODE	DESCRIPTION

```
close all
clear all
NDV = 9;%number design variable
XVAR = zeros (1, NDV); %design variable
XU = zeros (1, NDV);%upper value design variable
XL = zeros (1, NDV); %lower value design variable
G = zeros (1, 8); % constraints design
%NSA:number of analisys for computational cost
%evaluation
NSA = 0;
XPOP = zeros(1000, 9); %
Xnew = zeros(1000, 9);
Xbest = zeros(1, 9); %best solution
Xworst = zeros(1, 9);%worst solution
WT = zeros(1, 1000); %total volume
WPEN = zeros(1, 1000);%volume penalty
XDES = zeros(1, 9);%solution
XCM = zeros (1, 9); %center of mass
% dam geometry- range of possible values
XL(1) = 5;
XU(1) = 20;
XL(2) = 20;
XU(2) = 50;
XL(3) = 1;
XU(3) = 20;
XL(4) = 40;
XU(4) = 80;
XL(5) = 10;
XU(5) = 30;
XL(6) = 30;
XU(6) = 80;
XL(7) = 20;
XU(7) = 40;
XL(8) = 50;
XU(8) = 150;
XL(9) = 80;
XU(9) = 150;
fileID = fopen('ITERATIONS.txt', 'w');
% ------------------------------------------
% INITIALIZE JAYA ALGORITHM INTERNAL PARAMETERS
%------------------------------------------%
%ITLIM: maximum number of iterations for a loop.
% To be incremented if convergence is slow.
ITLIM = 650;
% NPOP: number of individuals or population size
% in an optimization algorithm.
NPOP = 5;
% penfact: factor that influences the severity of
% the penalty for constraint violations. It must
% be implemented if constraints must be stricter.
penfact = 1e3;
for kkk = 1:NPOP
for jjj = 1:NDV
RAND = rand();
```

Initialization of design variable bounds, low and up (XL and XU).

XL(1), XL(2), XL(3), XL(4), XL(5), XL(6), XL(7), XL(8), XL(9): Sets the first element of the lower bounds of the design variables

XU(1), XU(2) , XU(3) , XU(4) , XU(5) , XU(6) , XU(7) , XU(8) , XU(9) : Sets the first element of the upper bounds of the design variables

Opens a file named 'ITERATIONS.txt' for writing ('w').
The file handle is stored in the variable fileID.
This file is likely to be used for writing some output during the execution of the program.

for kkk = 1:NPOP: Loop over each individual in the population (NPOP individuals).

for jjj = 1:NDV: generation of a random candidate solution within the specified bounds for each decision variable.

```
XPOP(kkk, jjj) = XL(jjj) + RAND * (XU(jjj) -
XL(jjj));
XVAR(jjj) = XPOP(kkk, jjj);
End

% Call objective function FUNZOB, output FO.
FO=FUNZOB(XVAR);
% Assign the objective function value to the WT
%vector at the kkk-th position.
WT(kkk) = FO;
% Call function CONSTRAINTS, input ICOMP,TOTVNS,NSA
%outputs ICOMP and TOTVNS to evaluate constraints
%associated with the candidate solution.
[ICOMP,TOTVNS,NSA]= CONSTRAINTS(NSA,XVAR,G);
% compute the penalty-weighted objective function
% value and store it in the WPEN array.
WPEN(kkk) = WT(kkk) + penfact * TOTVNS;
% print the objective function value and penalty-
%weighted objective function value to the console.
fprintf(fileID, '%f %f\n', WT(kkk), WPEN(kkk));
end
%-----------------------------------------------
% FIND BEST AND WORST DESIGNS
%-----------------------------------------------
% initialize variables that will be used to
%track the maximum and minimum penalty-weighted
%objective function values and their
%corresponding indices in the population.
WMAX = 0;
WMIN = 1e50;
IMAX = 0;
IMIN = 0;
for kkk = 1:NPOP
if WPEN(kkk) < WMIN
WMIN = WPEN(kkk);
IMIN = kkk;
end
if WPEN(kkk) > WMAX
WMAX = WPEN(kkk);
IMAX = kkk;
end
end
for j = 1:NDV
Xbest(j) = XPOP(IMIN, j);
Xworst(j) = XPOP(IMAX, j);
End
% Print the index (IMIN) and the penalty-weighted
% objective function value (WMIN) of the best
% individual to the console or a file (assuming 6
% is the file identifier).
fprintf(fileID, '%d %f\n', IMIN, WMIN);
% GENERATE NEW DESIGN AND UPDATE POPULATION
```

For each individual, it compares the penalty-weighted objective function value (WPEN(kkk)) with the current minimum (WMIN) and maximum (WMAX) values.

If the current individual has a smaller penalty-weighted objective function value than the current minimum, update WMIN and store the index IMIN of this individual.

If the current individual has a larger penalty-weighted objective function value than the current maximum, update WMAX and store the index IMAX of this individual.

Extract Best and Worst Individuals' Decision Variables:
for j = 1:NDV: After finding the indices (IMIN and IMAX) of the best and worst individuals, extract their decision variables from the population matrix (XPOP).

Xbest(j) stores the decision variable values of the best individual.

Xworst(j) stores the decision variable values of the worst individual.

```
for iter = 1:ITLIM
for i = 1:NPOP
for j = 1:NDV
Xnew(i, j) = 0;
end
end
for kkk = 1:NPOP
for jjj = 1:NDV
RAND1 = rand();
RAND2 = rand();
dx1 = RAND1 * (Xbest(jjj) - abs(XPOP(kkk, jjj)));
dx2 = RAND2 * (Xworst(jjj) - abs(XPOP(kkk, jjj)));
Xnew(kkk, jjj) = abs(XPOP(kkk, jjj) + dx1 - dx2);
if Xnew(kkk, jjj) < XL(jjj)
Xnew(kkk, jjj) = XL(jjj);
end
if Xnew(kkk, jjj) > XU(jjj)
Xnew(kkk, jjj) = Xbest(jjj);
end
end
for jjj = 1:NDV
XVAR(jjj) = Xnew(kkk, jjj);
end
FO=FUNZOB(XVAR);
WTn = FO;
if WTn > WPEN(kkk)
continue;
end
[ICOMP,TOTVNS,NSA]= CONSTRAINTS(NSA,XVAR,G);
WPENn = WTn + penfact * TOTVNS;
if WPENn <= WPEN(kkk)
for jjj = 1:NDV
XPOP(kkk, jjj) = Xnew(kkk, jjj);
end
WT(kkk) = WTn;
WPEN(kkk) = WPENn;
end
End
% inizialize volumes and iterations
WMAX = 0;
WMIN = 1e50;
IMAX = 0;
IMIN = 0;
%Find the worst individual in the Population.
%This loop identifies the index (IMAX)
%and value (WMAX) of the individual with the
%highest penalty-weighted objective in
% the population.
for kkk = 1:NPOP
if WPEN(kkk) < WMIN
WMIN = WPEN(kkk);
IMIN = kkk;
end
if WPEN(kkk) > WMAX
WMAX = WPEN(kkk);
IMAX = kkk;
end
```

for iter = 1:ITLIM: This loop initializes a new population matrix (Xnew) for the next iteration, beginning by the value zero.
for kkk = **1:NPOP:** For each individual in the population (NPOP individuals), generate random perturbations (RAND1 and RAND2) for each decision variable (NDV variables)
dx1, dx2: perturbations based on the best and worst individuals' decision variable values.
if Xnew(kkk, jjj) < XL(jjj): limits to the Perturbations, if the perturbed decision variable (Xnew(kkk, jjj)) is below the lower limit (XL(jjj)), the lower limit is setted. And the same for the upper limit.
for jjj = 1:NDV: Update the decision variable values for the current individual to the perturbed values stored in Xnew.
if WTn > WPEN(kkk): Check if the objective function value of the perturbed solution (WTn) is greater than the penalty-weighted objective function value of the current individual (WPEN(kkk)).
If it is, skip to the next iteration (continue), implying that the perturbed solution is not an improvement over the current individual.
if WPENn <= WPEN(kkk): Checking if the penalty-weighted objective for the perturbed solution (WPENn) is less than or equal to the penalty-weighted objective for the current individual (WPEN(kkk)).
If the perturbed solution is better, update the population XPOP(kkk, jjj) , WT(kkk) objective function value for the current individual, WPEN(kkk penalty-weighted

```
End
% Find the best individual in the population
% After finding the worst individual, copy the
% decision variable values of the best individual
% (Xbest) from the population.
% The best individual is identified by the index
% stored in IMIN.
for j = 1:NDV
Xbest(j) = XPOP(IMIN, j);
Xworst(j) = XPOP(IMAX, j);
End
% Calculate the penalty weighted objective of
% the best individual (WOPT) using the index
% stored in IMIN

WOPT = WPEN(IMIN);
% Print information about the iteration, the
% minimum penalty-weighted objective (WMIN), the
% maximum penalty-weighted objective (WMAX) to
% the console
fprintf(fileID, '%d %f %f\n', iter, WMIN, WMAX);
fprintf(fileID, '%d %f %f\n', iter, WMIN, WMAX);
End
 %---------------------------------------
% OPTIMUM DESIGN
%---------------------------------------
for j = 1:NDV
XDES(j) = Xbest(j);
fprintf(fileID, 'XDES(%d) = %f\n', j, XDES(j));
fprintf(fileID, 'XDES(%d) = %f\n', j, XDES(j));
end
fprintf(fileID, 'DESIGN FINALE %f kg\n', WOPT);
fprintf(fileID, 'DESIGN FINALE %f kg\n', WOPT);
fprintf(fileID, 'NUMERO DI ANALISI %d\n', NSA);
for j = 1:NDV
XVAR(j) = Xbest(j);
end
[ICOMP,TOTVNS,NSA]= CONSTRAINTS(NSA,XVAR,G);
FO=FUNZOB(XVAR);
FOpen = FO + penfact * TOTVNS;
fprintf(fileID, '%f %d %f\n', FO, ICOMP, TOTVNS);
fprintf(fileID, '%f %f\n', FO, FOpen);
fclose(fileID);
pause;
pause;
return;
```

objective for the current individual.
for kkk = 1:NPOP: iterate through the population (NPOP individuals) and update WMIN, WMAX, IMIN, and IMAX based on the penalty-weighted objective values (WPEN(kkk)).
for j = 1:NDV : copy the design variables of the best individual (Xbest) to the variable XDES.
Print the design variables of the best individual to the console and the file.
Print information about the final design to both the console and the file specified by fileID.
WOPT is the penalty-weighted objective of the best individual.
NSA is the number of analyses performed.
for j = 1:NDV: copy the design variables of the best individual (Xbest) to the variable XVAR, evaluate the constraints (CONSTRAINTS(ICOMP, TOTVNS)) based on the design variables, evaluate the objective function (FUNZOB(FO)) based on the design variables.
FOpen = FO + penfact * TOTVNS: calculate the total objective with penalty (FOpen) by adding the penalty term (penfact * TOTVNS) to the original objective value (FO).
Print information about the final objective to both the console and the file.
Include the original objective value (FO), the constraint index (ICOMP), and the total violation of constraints (TOTVNS).
Close the file specified by fileID.
Pauses and stop execution are used for manual inspection or

interaction during the execution of the program.

THE JAYA OBJECTIVE FUNCTION

The file containing the Objective function is called FUNZOB.m

This subroutine calculates the objective function FO (output) based on the values of the elements in the XVAR array and some constants.

CODE

```
function FO = FUNZOB(XVAR)
Hdam = 150.0;
W1 = Hdam* XVAR(1,1);
W2 = XVAR(1,3) * (XVAR(1,4) + XVAR(1,6));
W3 = XVAR(1,5) * XVAR(1,6);
W4a = 0.5 * (XVAR(1,2) * XVAR(1,3));
W4b = 0.5 * (XVAR(1,4) * XVAR(1,5));
W4c = 0.5 * (XVAR(1,6) * XVAR(1,7));
W4d = 0.5 * (XVAR(1,8) * XVAR(1,9));
W4 = W4a + W4b + W4c + W4d;
FO = W1 + W2 + W3 + W4;
end
```

DESCRIPTION

Hdam: total height of the dam in meters

W1, W2, W3, W4a, W4b, W4c, W4d, W4: partial dam's volumes
FO:objective function FO, made by the dam's total volume.

return: Ends the subroutine, returning the calculated objective function FO as the output.

THE JAYA CONSTRAINTS' FUNCTION

The file containing the constraints is called CONSTRAINTS.m

CODE

```
function [ICOMP,TOTVNS,NSA]= CONSTRAINTS(NSA,XVAR,G)
NSA = NSA + 1;
ICOMP = 1;
% dam parameters%
Hdam = 150.0;
h_water = 145.0;
h_sediments = 7.0;
hwsq = h_water^2.0;
hSsq = h_sediments^2.0;
%specific weights
GammaC = 23.0; %concrete
GammaW = 9.81; % water
GammaS = 15.0;% sediments
PIGREC = 3.14159265;
Phi = (50.0/180.0)*PIGREC;%conversion to radiant
theta = atan(XVAR(6)/XVAR(7));
% b total base length of the dam
b = XVAR(1) + XVAR(3) + XVAR(5) + XVAR(7) +
XVAR(9);
bsq = b^2.0;
% h3 effective height of the dam above a certain
%reference level, which is the water level reduced
% by the heights of sediment and another section.
h3 = h_water - XVAR(4) - XVAR(6);
% horizontal distance or length associated with
% the geometry of the dam, calculated based on
% the difference in heights (h3) and the ratio
% of two elements of the XVAR vector.
dx3 = h3*(XVAR(3)/XVAR(2));
% alfa safety factor for weights (5%)
Alpha = 0.05;
% f static friction coefficient
f = 0.7;
%stress in N/mq
Sigma = 3500.0;
SigmaMax = 25000.0;
%volume
W1 = Hdam*XVAR(1,1);
W2 = XVAR(1,3)*(XVAR(1,4)+XVAR(1,6));
W3 = XVAR(1,5)*XVAR(1,6);
W4a = 0.5*(XVAR(1,2)*XVAR(1,3));
W4b = 0.5*(XVAR(1,4)*XVAR(1,5));
W4c = 0.5*(XVAR(1,6)*XVAR(1,7));
W4d = 0.5*(XVAR(1,8)*XVAR(1,9));
W4 = W4a + W4b + W4c + W4d;
Vdam = W1 + W2 + W3 + W4;
```

DESCRIPTION

ICOMP is an indicator of success or failure based on the conditions specified in the **CONSTRAINTS** subroutine.
TOTVNS: total violation of constraints
NDV: Number of Design Variables.
XVAR, XU, XL: design variables, upper bounds, and lower bounds.
G: constraint values.
NSA: Number of analyses
Hdam: total height of the dam
h_water: height of the water portion in the dam
h_sediments: height of the sediment portion in the dam
GammaC: specific weight of concrete in N/m^3
GammaW: specific weight of water in N/m^3
GammaS: specific weight of sediments in N/m^3
SigmaMax= concrete maximum allowable stress or pressure in the dam structure.

```
Weight_dam = GammaC*Vdam;
A1 = 0.5*(XVAR(1,6)*XVAR(1,7));
A2 = 0.5*(XVAR(1,4)*XVAR(1,5));
A3 = 0.5*h3*dx3;
A4 = XVAR(1,4)*XVAR(1,7);
A5 = h3*(XVAR(1,5)+XVAR(1,7));
Vwater = A1 + A2 + A3 + A4 + A5;
Weight_water = GammaW*Vwater;
%dam's center of mass
DamX1 = W4c*(2*XVAR(1,7)/3);
DamX2 = W4b*(XVAR(1,7)+2*XVAR(1,5)/3);
DamX3 = W4a*(XVAR(1,7)+XVAR(1,5)+2*XVAR(1,3)/3);
DamX4 = W3*(XVAR(1,7)+XVAR(1,5)/2);
DamX5 = W2*(XVAR(1,7)+XVAR(1,5)+XVAR(1,3)/2);
DamX6 =
W1*(XVAR(1,7)+XVAR(1,5)+XVAR(1,3)+XVAR(1,1)/2);
DamX7 = W4d*(b-2*XVAR(1,9)/3);
Xbar_dam =
(DamX1+DamX2+DamX3+DamX4+DamX5+DamX6+DamX7)/Vdam;
%water's center of mass
WatX1 = A1*(XVAR(1,7)/3);
WatX2 = A4*(XVAR(1,7)/2);
WatX3 = A2*(XVAR(1,7)+XVAR(1,5)/3);
WatX4 = A5*(XVAR(1,5)+XVAR(1,7)/2);
WatX5 = A3*(XVAR(1,5)+XVAR(1,7)+dx3/3);
Xbar_water =
(WatX1+WatX2+WatX3+WatX4+WatX5)/Vwater;
%hydrostatic pressure
PsH = 0.5*GammaS*hSsq*(1-sin(Phi))/(1+sin(Phi));
PsV = 0.5*GammaS*hSsq*tan(theta);
%forces
FE = Alpha*Weight_dam;%uplift force
Fuplift = 0.5*GammaW*h_water;%buoyancy force
FhH = 0.5*GammaW*hwsq;
FhV = Weight_water;
% sum of horizontal forces acting on the dam.
SumFH = FhH + PsH;
% sum of vertical forces acting on the dam.
SumFV = PsV + Weight_dam + FhV - Fuplift - FE;
%resistant torque
SumMR1 = (Weight_dam - FE)*(b - Xbar_dam);
SumMR2 = Weight_water*(b - Xbar_water);
SumMR3 = PsV*(b - h_sediments*(XVAR(7)/XVAR(6))/3);
% total sum of resisting moments
SumMR = SumMR1 + SumMR2 + SumMR3;
%overturning torque
SumMO1 = FhH*h_water/3;
SumMO2 = Fuplift*(2*b/3);
SumMO3 = PsH*h_sediments/3;
SumMO = SumMO1 + SumMO2 + SumMO3;
```

PsH: horizontal hydrostatic pressure due to sediment. It's calculated based on the submerged weight of the sediment and the geometry of the dam.
PsV: vertical hydrostatic pressure due to sediment.
FE: uplift force acting on the dam. It represents the buoyancy force trying to lift the dam due to water pressure beneath it. The Alpha factor represents a safety factor.
Fuplift: buoyancy force due to the water. It is a constant force trying to lift the dam.
FhH: horizontal force due to water pressure. It considers the hydrostatic pressure acting horizontally on the dam.
FhV: vertical force due to water pressure. It represents the weight of the water above the dam.
SumMR1: torque due to the weight of the dam and the arch force multiplied by the distance between the point of force application and the centroid of the dam.
SumMR2: torque due to the weight of the water relative to the centroid of the water multiplied by the distance between the point of force application and the centroid of the water.
SumMR3: torque due to the vertical lateral pressure relative to a specific point, calculated based on the dam's geometry and sediment distribution multiplied by the distance between the point of force application and the centroid of the vertical lateral pressure.
SumMO1: moment due to the buoyancy force of the water.
SumMO2: moment due to the uplift force.
SumMO3: torque due to the horizontal component of the sediment force.
SumMO: total horizontal moment in the dam.

```
%safety factor calculation
SFS = (f*SumFV + Sigma*b)/SumFH;
SFO = SumMR/SumMO;
%resultant stress
%sigmau=stress at the top of the dam(tension side)
SigmaU = SumFV/b - 6*SumMO/bsq;
%sigmaD=stress at the bottom of the dam
%(compression side)
SigmaD = SumFV/b + 6*SumMO/bsq;
%constraint functions
G(1) = (4 - SFS);
G(2) = (1.5 - SFO);
G(3) = (-SigmaU);
G(4) = (-SigmaD);
G(5) = (SigmaU - SigmaMax);
G(6) = (SigmaD - SigmaMax);
G(7) = (XVAR(2) + XVAR(4) + XVAR(6) - Hdam)^2;
G(8) = XVAR(4) + XVAR(6) - XVAR(8);
TOTVNS = 0;
for jvin = 1:8
if G(jvin) > 0
ICOMP = 0;
TOTVNS = TOTVNS + (G(jvin)^2);
end
end
if TOTVNS > 0
ICOMP = 0;
end
end
```

SFS: safety factor for stress. It is calculated based on the ratio of the allowable stress (Sigma) to the calculated stress (SumFH).
SFO: safety factor for overturning. It is calculated based on the ratio of the resisting moment (SumMR) to the overturning moment (SumMO).
G: constraints used to impose limits and conditions on the design variables and performance measures.

The optimization algorithm seeks to find a solution where all these constraints are satisfied.

If any of these constraints is not met (i.e., G(i)>0), it indicates a violation of the corresponding constraint.

The total violation is then used to evaluate the feasibility of the design.

Check if the G(jvin) is greater than zero, indicating a violation.
If any constraint is violated, set ICOMP to zero, indicating a failure.
End the if statement.
End the loop over constraints.

If the total violation (TOTVNS) is greater than zero, set ICOMP to zero, indicating a failure.

In summary: If there are no constraint violations, ICOMP remains equal to one, indicating a successful design. If there are violations, ICOMP is set to zero.

End the subroutine.

APPENDIX 2- MATLAB GWO CODE APPLIED TO A GENERIC FUNCTION

THE "MAIN" GWO FILE

The MATLAB code has been divided into four files: one containing the main commands, a second including the objective function, a third incorporating GWO algorithm code and the internal variables' initialization, the last containing the final plot code. The main algorithm can be identified in 7 steps.

1. Initialization of Parameters:
    - SearchAgents_no=30: Sets the number of search agents to 30.
    - Function_name='F10': Specifies the name of the objective function.
    - Max_iteration=500: Defines the maximum number of iterations for the optimization process.

2. Loading Function Details:
    - Get_Functions_details: Calls a function to retrieve details (lower bound, upper bound, dimension, and function handle) for the specified test function.

3. Executing GWO Algorithm:
    - GWO(SearchAgents_no,Max_iteration,lb,ub,dim,fobj): Invokes the Grey Wolf Optimizer (GWO) algorithm, passing the parameters like the number of search agents, maximum iterations, and function details. Receives outputs such as the best score, best position, and GWO convergence curve.

4. Plotting Parameter Space:
    - Creates a figure window with a specified position.
    - Sets up a subplot for the parameter space plot.
    - Calls func_plot to visualize the search space of the specified test function.
    - Sets titles and labels for the parameter space plot.

5. Plotting Objective Space:
    - Sets up a subplot for the objective space plot.
    - Uses semilogy to plot the GWO convergence curve on a semilogarithmic scale.
    - Sets titles and labels for the objective space plot.

6. Adjusting Plot Settings:
    - axis tight: Adjusts the axis limits to fit the data.
    - Enables grid lines and a box around the plot.

7. Displaying Results:

    - Displays in the command window the best solution obtained by GWO (Best_pos) and the best optimal value of the objective function found by GWO (Best_score).

CODE	DESCRIPTION

clear all
clc
SearchAgents_no=30; % Number of search agents
Function_name='F10'; % Name of the test function
Max_iteration=500; % Maximum numbef of iterations
% Load details of the selected OBJECTIVE function
[lb,ub,dim,fobj]=Get_Functions_details(Function_name);
[Best_score,Best_pos,GWO_cg_curve]=GWO(SearchAgents_no,Max_iteration,lb,ub,dim,fobj);
% Figure window with a specified position.
figure('Position',[500 500 660 290])
%Draw search space
% Sets up a subplot for the parameter space plot in a 1x2 grid.
subplot(1,2,1);
func_plot(Function_name);
% Sets the title for the first subplot.
title('Parameter space')
% Labels the x-axis and y-axis for the first subplot
xlabel('x_1');
ylabel('x_2');
% Labels the z-axis for the first subplot with the name of the test function
zlabel([Function_name,'(x_1 , x_2)'])
%Draw objective space
% Sets up a subplot for the objective space plot in a 1x2 grid
subplot(1,2,2);
% Plots the GWO convergence curve using a semilogarithmic scale
semilogy(GWO_cg_curve,'Color','r')
% Sets the title for the second subplot
title ('Objective space')
% Labels the x-axis and y-axis for the second subplot
xlabel('Iteration');
ylabel('Best score obtained so far');
% Adjusts the axis limits to fit the data
axis tight
% Enables grid lines and a box around the plot
grid on
box on
% Adds a legend to the plot indicating that the curve corresponds to the GWO algorithm
legend('GWO')
display(['The best solution obtained by GWO is : ', num2str(Best_pos)]);
display(['The best optimal value of the objective function found by GWO is : ', num2str(Best_score)]);
	[lb,ub,dim,fobj]=Get_Functions_details(Function_name) : Calls a function named Get_Functions_details to retrieve details (lb lower bound, ub upper bound, dim number of variables, and fobj function handle)
	[Best_score,Best_pos,GWO_cg_curve]=GWO(SearchAgents_no,Max_iteration,lb,ub,dim,fobj): Call the GWO algorithm function with parameters such as the number of search agents, maximum iterations, function details, and receives outputs like the best score, best position, and the GWO convergence curve.
	func_plot(Function_name): Calls a function func_plot to plot the search space of the specified test function.
	grid on and box on: Enables grid lines and a box around the plot.
	
	Displays the best solution obtained by GWO in the command window.
	
	Displays the best optimal value of the objective function found by GWO in the command window.

THE "GWO" FILE

The file develops the Grey Wolf Optimizer (GWO) algorithm, as follows:

1. Function Declaration GWO:

   Outputs: fitness scores of the alfa wolf position (Alpha_score), alfa wolf position (Alpha_pos), Convergence_curve

   Inputs: number of search agents (SearchAgents_no), number of maximum iterations (Max_iter), lower and upper bounds of the search space (lb, ub), number of variables (dim), objective function (fobj).

2. Initialization: the initial positions of the alpha, beta and delta wolves are setted to zero, while the fitness scores related to the alpha, beta, delta positions are initialized with infinity.
3. Search Agents Initialization: the positions of the search agents are calculated with a function named initialization. Inputs: number of search agents, number of variables, upper and lower bounds of the search space.
4. Convergence Curve Initialization: the convergence curve values during iterations are stored in a vector, initially defined by zeros.

5. Main Loop: l is incremented until the value of maximum iterations to obtain convergence.

6. Search Agent Position Update: iterations through each search agent, ensuring they remain within the search space bounds, are developed to calculate the new positions of the alpha, beta, and delta wolves considering the fitness scores for each search agent.

7. Adaptive Parameter Update: the coefficient a=2-l 2/(Max_iter) is linearly decreased from 2 to 0 over the course of iterations.

8. Position Update Equations: the position of each search agent is updated involving random numbers and the positions of the alpha, beta, and delta wolves, following the equations proposed for the GWO algorithm.

9. Convergence Curve Update: the fitness score of the alpha wolf is stored in the convergence curve vector.

10. Output: visualization of the final alpha score, alpha position, and convergence curve values

CODE	DESCRIPTION
% Grey Wolf Optimizer function [Alpha_score,Alpha_pos,Convergence_curve]=GWO(SearchAgents_no,Max_iter,lb,ub,dim,fobj) % initialize alpha, beta, and delta_pos Alpha_pos=zeros(1,dim); Alpha_score=inf; %change this to -inf for maximization problems Beta_pos=zeros(1,dim); Beta_score=inf; %change this to -inf for maximization problems Delta_pos=zeros(1,dim); Delta_score=inf; %change this to -inf for maximization problems %Initialize the positions of search agents Positions=initialization(SearchAgents_no,dim,ub,lb); Convergence_curve=zeros(1,Max_iter); l=0;% Loop counter % Main loop while l<Max_iter for i=1:size(Positions,1) % Return back the search agents that go beyond the boundaries of the search space Flag4ub=Positions(i,:)>ub; Flag4lb=Positions(i,:)<lb; Positions(i,:)=(Positions(i,:).*(~(Flag4ub+Flag4lb)))+ub.*Flag4ub+lb.*Flag4lb; % Calculate objective function for each search agent fitness=fobj(Positions(i,:)); % Update Alpha, Beta, and Delta if fitness<Alpha_score Alpha_score=fitness; % Update alpha Alpha_pos=Positions(i,:); end if fitness>Alpha_score && fitness<Beta_score Beta_score=fitness; % Update beta Beta_pos=Positions(i,:); end if fitness>Alpha_score && fitness>Beta_score && fitness<Delta_score Delta_score=fitness; % Update delta Delta_pos=Positions(i,:); end end a=2-l*((2)/Max_iter); % a decreases linearly from 2 to 0	**Function "GWO":** INPUT: number of search agents, maximum iterations, lower and upper bounds of the search space, dimensionality (number of variables), and the objective function. **Initialization** of alpha, beta, delta wolves and their fitness scores **Function "initialization":** The vector "Positions" is assigned values computed by the "initialization" function. INPUT: number of search agents, number of variables, lower and upper bounds. **Convergence loop.** **Search Agent Position Update** **fitness=fobj(Positions(i,:)): Function "fobj":** The vector "fitness" is assigned values computed by the "fobj" function. INPUT: vector "Positions" **a=2-l*((2)/Max_iter): Adaptive coefficient update**

```
% Update the Position of search agents including
omegas
for i=1:size(Positions,1)
for j=1:size(Positions,2)
r1=rand(); % r1 is a random number in [0,1]
r2=rand(); % r2 is a random number in [0,1]
A1=2*a*r1-a; % Coefficient vector for alpha
C1=2*r2; % Coefficient vector for alpha
D_alpha=abs(C1*Alpha_pos(j)-Positions(i,j)); %
Distance vector for alpha
X1=Alpha_pos(j)-A1*D_alpha;
r1=rand();
r2=rand();
A2=2*a*r1-a; % Coefficient vector for beta
C2=2*r2; % Coefficient vector for beta
D_beta=abs(C2*Beta_pos(j)-Positions(i,j)); %
Distance vector for beta
X2=Beta_pos(j)-A2*D_beta;
r1=rand();
r2=rand();
A3=2*a*r1-a; % Coefficient vector for delta
C3=2*r2; % Coefficient vector for delta
D_delta=abs(C3*Delta_pos(j)-Positions(i,j)); %
Distance vector for delta
X3=Delta_pos(j)-A3*D_delta;

Positions(i,j)=(X1+X2+X3)/3;
end
end
l=l+1;
Convergence_curve(l)=Alpha_score;
end
```

Position Update Equations
Definition of the coefficient a, r_1, r_2, and the GWO vectors A, C, D

X1:alpha wolf's new position

X2:beta wolf's new position

X3:delta wolf's new position

Positions: search agents' new position
Increment of iterations
Convergence Curve Update

THE "INITIALIZATION" FILE

This file initializes the starting positions of search agents, configuring them according to specified boundaries for each dimension within the search space.

CODE	DESCRIPTION

`% This function initialize the first population of search agents`
`function`
`Positions=initialization(SearchAgents_no,dim,ub,lb)`
	It returns a matrix Positions representing the initial positions of the search agents. INPUT: number of search agents , number of variables, upper and lower bounds for each variable.

`% Number of Boundaries Determination` `Boundary_no= size(ub,2); % number of boundaries`	This line calculates the number of boundaries by determining the size of the ub matrix along its second dimension.

`% Initialization for Equal Boundaries: if the boundaries of all variables are equal and user enter a single number for both ub and lb`
`if Boundary_no==1`
`Positions=rand(SearchAgents_no,dim).*(ub-lb)+lb;`
`End`
	if Boundary_no == 1
	If all variables share the same upper and lower boundaries, the function initializes positions using random values in the specified range for each dimension.

`%Initialization for Different Boundaries:if each variable has a different lb and ub` `if Boundary_no>1`	**if Boundary_no > 1** If each variable has different upper and lower boundaries, the function initializes positions for each dimension separately using random values in the specified range. for i = 1:dim
`% Loop over each dimension`	get the upper and lower boundaries for the current dimension.

`for i=1:dim`
`ub_i=ub(i);`
`lb_i=lb(i);`
	The function initializes positions for the current dimension using random values.

`Positions(:,i)=rand(SearchAgents_no,1).*(ub_i-lb_i)+lb_i;`
`end`
`end`	

THE "GET_FUNCTION_DETAILS" FILE

This file defines a function that returns details (lower bound, upper bound, the number of variables, and objective function) for benchmark functions commonly used in optimization problems and here specified by their index (F1, F2, ..., F23).

The "switch F" command is a control flow statement that allows to select one of many cases to be executed based on the value of a variable or expression. For instance, if F is equal to 'F1', the code inside the case 'F1' will be executed, setting the lower bound (lb), upper bound (ub), dimension (dim), and objective function (fobj) for the benchmark function 'F1'.

This structure makes the code modular and allows for easy extension or modification when adding new benchmark functions.

CODE

```
% lb is the lower bound: lb=[lb_1,lb_2,...,lb_d]
% up is the upper bound: ub=[ub_1,ub_2,...,ub_d]
% dim is the number of variables (dimension of the
problem)
% fobj is the objective function
function [lb,ub,dim,fobj] =
Get_Functions_details(F)
switch F
case 'F1'
fobj = @F1;
lb=-100;
ub=100;
dim=30;
case 'F2'
fobj = @F2;
lb=-10;
ub=10;
dim=30;
case 'F3'
fobj = @F3;
lb=-100;
ub=100;
dim=30;
case 'F4'
fobj = @F4;
lb=-100;
ub=100;
dim=30;
case 'F5'
fobj = @F5;
lb=-30;
ub=30;
dim=30;
```

DESCRIPTION

This function takes a single input F, which represents the index of the benchmark function and gives in output lb,ub,dim,fobj

switch F checks the value of F and executes the corresponding case.

case 'F1'

fobj = @F1: This line sets the variable fobj to be a function handle @F1. The function handle allows the objective function 'F1' to be used later in the optimization process.

Functions: For each benchmark function (F1 to F23), there is a specific case that sets the lower bound (lb), upper bound (ub), dimension of the space search (dim), and the objective function (fobj) using function handles.

```
case 'F6'
fobj = @F6;
lb=-100;
ub=100;
dim=30;
case 'F7'
fobj = @F7;
lb=-1.28;
ub=1.28;
dim=30;
case 'F8'
fobj = @F8;
lb=-500;
ub=500;
dim=30;
case 'F9'
fobj = @F9;
lb=-5.12;
ub=5.12;
dim=30;
case 'F10'
fobj = @F10;
lb=-32;
ub=32;
dim=30;
case 'F11'
fobj = @F11;
lb=-600;
ub=600;
dim=30;
case 'F12'
fobj = @F12;
lb=-50;
ub=50;
dim=30;
case 'F13'
fobj = @F13;
lb=-50;
ub=50;
dim=30;
case 'F14'
fobj = @F14;
lb=-65.536;
ub=65.536;
dim=2;
case 'F15'
fobj = @F15;
lb=-5;
ub=5;
dim=4;
```

```
case 'F16'
fobj = @F16;
lb=-5;
ub=5;
dim=2;
case 'F17'
fobj = @F17;
lb=[-5,0];
ub=[10,15];
dim=2;
case 'F18'
fobj = @F18;
lb=-2;
ub=2;
dim=2;
case 'F19'
fobj = @F19;
lb=0;
ub=1;
dim=3;
case 'F20'
fobj = @F20;
lb=0;
ub=1;
dim=6;
case 'F21'
fobj = @F21;
lb=0;
ub=10;
dim=4;
case 'F22'
fobj = @F22;
lb=0;
ub=10;
dim=4;
case 'F23'
fobj = @F23;
lb=0;
ub=10;
dim=4;
end
End

% F1
function o = F1(x)
o=sum(x.^2);
end
% F2
function o = F2(x)
o=sum(abs(x))+prod(abs(x));
end
```

This second **end** marks the end of the switch case statement.

Objective Function Definitions. These functions are used to calculate the objective value based on the input x

```
% F3
function o = F3(x)
dim=size(x,2);
o=0;
for i=1:dim
o=o+sum(x(1:i))^2;
end
end
% F4
function o = F4(x)
o=max(abs(x));
end
% F5
function o = F5(x)
dim=size(x,2);
o=sum(100*(x(2:dim)-(x(1:dim-1).^2)).^2+
+(x(1:dim-1)-1).^2);
end
% F6
function o = F6(x)
o=sum(abs((x+.5)).^2);
end
% F7
function o = F7(x)
dim=size(x,2);
o=sum([1:dim].*(x.^4))+rand;
end
% F8
function o = F8(x)
o=sum(-x.*sin(sqrt(abs(x))));
end
% F9
function o = F9(x)
dim=size(x,2);
o=sum(x.^2-10*cos(2*pi.*x))+10*dim;
end
% F10
function o = F10(x)
dim=size(x,2);
o=-20*exp(-.2*sqrt(sum(x.^2)/dim))-
exp(sum(cos(2*pi.*x))/dim)+20+exp(1);
end
% F11
function o = F11(x)
dim=size(x,2);
o=sum(x.^2)/4000-prod(cos(x./sqrt([1:dim])))+1;
end
% F12
function o = F12(x)
dim=size(x,2);
o=(pi/dim)*(10*((sin(pi*(1+(x(1)+1)/4)))^2)+
+sum((((x(1:dim-1)+1)./4).^2).*
*(1+10.*((sin(pi.*(1+(x(2:dim)+
+1)./4)))).^2))+((x(dim)+1)/4)^2)+
+sum(Ufun(x,10,100,4));
end
```

o=0: sets the initial value of the output to zero.

%F12- Utility Function (Ufun): The Ufun function is a utility function.
This function is used to calculate a specific component of the overall expression.
The purpose of Ufun in this context is to calculate a part of the benchmark function that involves conditions based on some values. The utility function Ufun is commonly used in mathematical formulations to introduce non-linearity or specific characteristics that make the optimization problem more challenging.

```
% F13
function o = F13(x)
dim=size(x,2);
o=.1*((sin(3*pi*x(1)))^2+sum((x(1:dim-1)-
1).^2.*(1+(sin(3.*pi.*x(2:dim))).^2))+...
((x(dim)-
1)^2)*(1+(sin(2*pi*x(dim)))^2))+sum(Ufun(x,5,100,4));
end
% F14
function o = F14(x)
aS=[-32 -16 0 16 32 -32 -16 0 16 32 -32 -16 0 16 32 -32
-16 0 16 32 -32 -16 0 16 32;,...
-32 -32 -32 -32 -32 -16 -16 -16 -16 -16 0 0 0 0 0 16 16
16 16 16 32 32 32 32 32];
for j=1:25
bS(j)=sum((x'-aS(:,j)).^6);
end
o=(1/500+sum(1./([1:25]+bS))).^(-1);
end
% F15
function o = F15(x)
aK=[.1957 .1947 .1735 .16 .0844 .0627 .0456 .0342 .0323
.0235 .0246];
bK=[.25 .5 1 2 4 6 8 10 12 14 16];bK=1./bK;
o=sum((aK-
((x(1).*(bK.^2+x(2).*bK))./(bK.^2+x(3).*bK+x(4)))).^2);
end
% F16
function o = F16(x)
o=4*(x(1)^2)-2.1*(x(1)^4)+(x(1)^6)/3+x(1)*x(2)-
4*(x(2)^2)+4*(x(2)^4);
end
% F17
function o = F17(x)
o=(x(2)-(x(1)^2)*5.1/(4*(pi^2))+5/pi*x(1)-6)^2+10*(1-
1/(8*pi))*cos(x(1))+10;
end
% F18
function o = F18(x)
o=(1+(x(1)+x(2)+1)^2*(19-14*x(1)+3*(x(1)^2)-
14*x(2)+6*x(1)*x(2)+3*x(2)^2))*...
(30+(2*x(1)-3*x(2))^2*(18-32*x(1)+12*(x(1)^2)+48*x(2)-
36*x(1)*x(2)+27*(x(2)^2)));
end
% F19
function o = F19(x)
aH=[3 10 30;.1 10 35;3 10 30;.1 10 35];cH=[1 1.2 3
3.2];
pH=[.3689 .117 .2673;.4699 .4387 .747;.1091 .8732
.5547;.03815 .5743 .8828];
o=0;
for i=1:4
o=o-cH(i)*exp(-(sum(aH(i,:).*((x-pH(i,:)).^2))));
end
end
```

```
% F20
function o = F20(x)
aH=[10 3 17 3.5 1.7 8;.05 10 17 .1 8 14;3 3.5 1.7
10 17 8;17 8 .05 10 .1 14];
cH=[1 1.2 3 3.2];
pH=[.1312 .1696 .5569 .0124 .8283 .5886;.2329 .4135
.8307 .3736 .1004 .9991;...
.2348 .1415 .3522 .2883 .3047 .6650;.4047 .8828
.8732 .5743 .1091 .0381];
o=0;
for i=1:4
o=o-cH(i)*exp(-(sum(aH(i,:).*((x-pH(i,:)).^2))));
end
end
% F21
function o = F21(x)
aSH=[4 4 4 4;1 1 1 1;8 8 8 8;6 6 6 6;3 7 3 7;2 9 2
9;5 5 3 3;8 1 8 1;6 2 6 2;7 3.6 7 3.6];
cSH=[.1 .2 .2 .4 .4 .6 .3 .7 .5 .5];
o=0;
for i=1:5
o=o-((x-aSH(i,:))*(x-aSH(i,:))'+cSH(i))^(-1);
end
end
% F22
function o = F22(x)
aSH=[4 4 4 4;1 1 1 1;8 8 8 8;6 6 6 6;3 7 3 7;2 9 2
9;5 5 3 3;8 1 8 1;6 2 6 2;7 3.6 7 3.6];
cSH=[.1 .2 .2 .4 .4 .6 .3 .7 .5 .5];
o=0;
for i=1:7
o=o-((x-aSH(i,:))*(x-aSH(i,:))'+cSH(i))^(-1);
end
end
% F23
function o = F23(x)
aSH=[4 4 4 4;1 1 1 1;8 8 8 8;6 6 6 6;3 7 3 7;2 9 2
9;5 5 3 3;8 1 8 1;6 2 6 2;7 3.6 7 3.6];
cSH=[.1 .2 .2 .4 .4 .6 .3 .7 .5 .5];
o=0;
for i=1:10
o=o-((x-aSH(i,:))*(x-aSH(i,:))'+cSH(i))^(-1);
end
End

function o=Ufun(x,a,k,m)
o=k.*((x-a).^m).*(x>a)+k.*((-x-a).^m).*(x<(-a));
end
```

function o=Ufun(x,a,k,m)
x: Input vector or array.
A,k,m: Parameters.
The function calculates a value o based on the following conditions:
If x > a, it calculates k * ((x - a).^m).
If x < -a, it calculates k * ((-x - a).^m).
If neither condition is satisfied, it assigns a value of 0.
Essentially, this function models a piecewise mathematical expression with different forms based on the ranges of x.

THE FILE "FUNC_PLOT"

This file generates a surface plot of benchmark functions. The function takes a string parameter func_name, representing the name of the benchmark function to be plotted. The code uses the Get_Functions_details function to get information about the selected benchmark function.

The switch-case block determines the grid ranges for x and y based on the selected benchmark function. The nested loops then evaluate the benchmark function for each combination of x and y within the specified ranges, and the resulting values are used to create a surface plot. This allows visualizing the behavior of the benchmark function in a 3D space.

CODE

```
% This function draws the benchmark functions
function func_plot(func_name)


[lb,ub,dim,fobj]=Get_Functions_details(func_name);


switch func_name
case 'F1'
x=-100:2:100; y=x; %[-100,100]
case 'F2'
x=-100:2:100; y=x; %[-10,10]
case 'F3'
x=-100:2:100; y=x; %[-100,100]
case 'F4'
x=-100:2:100; y=x; %[-100,100]
case 'F5'
x=-200:2:200; y=x; %[-5,5]
case 'F6'
x=-100:2:100; y=x; %[-100,100]
case 'F7'
x=-1:0.03:1; y=x %[-1,1]
case 'F8'
x=-500:10:500;y=x; %[-500,500]
case 'F9'
x=-5:0.1:5; y=x; %[-5,5]
case 'F10'
x=-20:0.5:20; y=x;%[-500,500]
case 'F11'
x=-500:10:500; y=x;%[-0.5,0.5]
case 'F12'
x=-10:0.1:10; y=x;%[-pi,pi]
case 'F13'
x=-5:0.08:5; y=x;%[-3,1]
case 'F14'
x=-100:2:100; y=x;%[-100,100]
case 'F15'
x=-5:0.1:5; y=x;%[-5,5]
```

DESCRIPTION

This line defines the function func_plot that takes one input argument, func_name.

This line retrieves details about the benchmark function specified by func_name. It gets the lower bound (lb), upper bound (ub), dimension (dim), and function handle (fobj).

switch func_name
This block defines grid ranges for x and y based on the selected benchmark function using a switch-case structure.

```
case 'F16'
x=-1:0.01:1; y=x;%[-5,5]
case 'F17'
x=-5:0.1:5; y=x;%[-5,5]
case 'F18'
x=-5:0.06:5; y=x;%[-5,5]
case 'F19'
x=-5:0.1:5; y=x;%[-5,5]
case 'F20'
x=-5:0.1:5; y=x;%[-5,5]
case 'F21'
x=-5:0.1:5; y=x;%[-5,5]
case 'F22'
x=-5:0.1:5; y=x;%[-5,5]
case 'F23'
x=-5:0.1:5; y=x;%[-5,5]
end
L=length(x);
f=[];
for i=1:L
for j=1:L
if strcmp(func_name,'F15')==0 &&
strcmp(func_name,'F19')==0 &&
strcmp(func_name,'F20')==0 &&
strcmp(func_name,'F21')==0 &&
strcmp(func_name,'F22')==0 &&
strcmp(func_name,'F23')==0
f(i,j)=fobj([x(i),y(j)]);
end
if strcmp(func_name,'F15')==1
f(i,j)=fobj([x(i),y(j),0,0]);
end
if strcmp(func_name,'F19')==1
f(i,j)=fobj([x(i),y(j),0]);
end
if strcmp(func_name,'F20')==1
f(i,j)=fobj([x(i),y(j),0,0,0,0]);
end
if strcmp(func_name,'F21')==1 ||
strcmp(func_name,'F22')==1
||strcmp(func_name,'F23')==1
f(i,j)=fobj([x(i),y(j),0,0]);
end
end
end
surfc(x,y,f,'LineStyle','none');
end
```

Calculates the **L=length** of the grid (x or y), assuming they have the same length.
f = [ ]: initializes an empty matrix

strcmp stands for "string compare," and it returns 1 (true) if the strings are identical and 0 (false) otherwise.

This loops evaluates the function for each combination of x and y values and stores the results in the matrix f.

Generates a surface plot of the benchmark function using the surfc function. The 'LineStyle', 'none' option is used to hide the grid lines.

APPENDIX 3- PID+ADMM ON SINGLE CART-POLE.

3.1 MATLAB CODE FOR LQR+ADMM ON DUAL CART-POLE

```
% Clear workspace and close all figures
clear all;
close all;

% Constants and System Properties
M = 1;     % Mass of the cart (kg)
m = 0.1;   % Mass of the pole (kg)
g = 9.81;  % Acceleration due to gravity (m/s^2)
cart_length = 0.2;  % Length of the cart (meters)
cart_width = 0.1;   % Width of the cart (meters)
cart_height = 0.05;  % Height of the cart (meters)
pole_length = 1.0;  % Length of the pole (meters)
pole_radius = 0.02;  % Radius of the pole (meters)

% Simulation parameters
simulation_time = 10; % Total simulation time (seconds)
dt = 0.01;          % Time step (seconds)

% Desired pole extremity position
x_desired = 0.0;     % Desired cart position (meters)
theta_desired = 45;  % Desired pole angle (degrees)

% PID Controller Gains for cart position and pole angle
Kp_xc = 100;  % Proportional gain for cart position
Ki_xc = 10;   % Integral gain for cart position
Kd_xc = 20;   % Derivative gain for cart position

Kp_theta = 500; % Proportional gain for pole angle
Ki_theta = 100; % Integral gain for pole angle
Kd_theta = 50;  % Derivative gain for pole angle

% Define the state-space representation of the system (continuous-time)
A = [0, 1, 0, 0;
    0, 0, (-12*m*g)/(13*M+m), 0;
    0, 0, 0, 1;
    0, 0, 12*(M+m)*g/(pole_length*(13*M+m)), 0];
B = [0,0; 13/(13*M+m),0; 0,0; 0,-12/(pole_length*(13*M+m))];

% ADMM Parameters for Sensor Coordination
rho = 0.1;              % Penalty parameter
max_iterations = 100;    % Maximum ADMM iterations

% Initial conditions
x1 = [0; 0; deg2rad(0); 0]; % Initial state [cart_pos, cart_vel, pole_angle, pole_ang_vel]

% Preallocate arrays to store states and control inputs
num_steps = simulation_time / dt;
state_history = zeros(4, num_steps);
control_history = zeros(2, num_steps); % Two control inputs: cart position and pole angle

% ADMM Variables (Assuming two sensors for cart position and pole angle)
lambda_xc = 0;    % Lagrange multiplier for cart position sensor
lambda_theta = 0; % Lagrange multiplier for pole angle sensor

% Weight for the coordination objective
```

```
alpha = 0.01;

% Weight for the cart position constraint
beta = 0.005;

% Path generation (linear path on the y-axis)
path_y = linspace(0, 100, num_steps);

% Visualization Setup
figure(1); % Figure 1 for cart-pole animation
hold on;
axis([-1, 11, 0, 100]); % Adjusted x-axis limits for 10 meters of motion
xlabel('X-axis (meters)');
ylabel('Y-axis (meters)');
zlabel('Z-axis (meters)');
title('Cart-Pole Simulation');

% Initialize the cart and pole graphics objects with empty handles
cart = [];
pole = [];

% Simulation Loop
error_xc_prev = 0;
error_theta_prev = 0;

% Create a folder on the desktop to save figures
desktop_path = fullfile(pwd, 'DIOESISTE');
if ~exist(desktop_path, 'dir')
   mkdir(desktop_path);
end

for t = 1:num_steps
   % Calculate PID control inputs for cart position and pole angle
   error_xc = x_desired - x1(1);
   error_theta = deg2rad(theta_desired) - x1(3);

   u_xc = Kp_xc * error_xc + Ki_xc * sum(error_xc) * dt + Kd_xc * (error_xc - error_xc_prev) / dt;
   u_theta = Kp_theta * error_theta + Ki_theta * sum(error_theta) * dt + Kd_theta * (error_theta - error_theta_prev) / dt;

   % Update state using the discrete-time dynamics
   x1 = x1 + (A * x1 + B * [u_xc; u_theta]) * dt;

   % Store state and control input history
   state_history(:, t) = x1;
   control_history(:, t) = [u_xc; u_theta];

   % ADMM Updates (Coordination between sensors)
   % Calculate the coordination objective based on squared difference
   coordination_objective = alpha * (x1(1) - x1(3))^2;

   % Calculate the cart position constraint
   cart_position_constraint = beta * max(0, abs(x1(1)) - (cart_length / 2));

   % Combine the coordination objective and constraint
   objective_and_constraint = coordination_objective + cart_position_constraint;

   % Update lambda_xc and lambda_theta based on the combined objective and constraint
```

```
    lambda_xc = lambda_xc + rho * (x1(1) - x1(3)) + alpha * rho * (x1(1) - x1(3)) - beta * rho * sign(x1(1));
    lambda_theta = lambda_theta + rho * (x1(3) - x1(1)) - alpha * rho * (x1(1) - x1(3));

    % Remember previous errors for derivative terms
    error_xc_prev = error_xc;
    error_theta_prev = error_theta;

    % Update cart position for visualization
    cart_x = x1(1) - cart_length / 2;  % Updated cart position translation
    cart_y = path_y(t) - cart_width / 2;
    cart_z = cart_height / 2; % Adjusted for cart height

    pole_x = x1(1);
    pole_y = path_y(t);
    pole_z = pole_length / 2; % Adjusted for pole length

   % Update cart and pole graphics objects or create them if not exist
    if isempty(cart)
        cart = rectangle('Position', [cart_x, cart_y, cart_length, cart_width], 'FaceColor', 'b');
    else
        set(cart, 'Position', [cart_x, cart_y, cart_length, cart_width]);
    end

    if isempty(pole)
        pole = plot3([pole_x, pole_x], [pole_y, pole_y - pole_length], [pole_z, pole_z], 'r', 'LineWidth', 2);
    else
        set(pole, 'XData', [pole_x, pole_x], 'YData', [pole_y, pole_y - pole_length], 'ZData', [pole_z, pole_z]);
    end

    % Draw the linear path for the cart along the y-axis
    plot3([0, 10], [0, 100], [0, 0], 'k--', 'LineWidth', 2); % Adjusted the path for 10 meters

% Set axis limits
    axis([-1, 11, 0, 100]); % Adjusted the x-axis limits for 10 meters

    % Save the current figure as an image
    saveas(gcf, fullfile(desktop_path, sprintf('figure_%04d.png', t)));

    % Pause to control the animation speed
    pause(dt);
end

% Plot results (you can adapt this part as needed)
time = 0:dt:(simulation_time - dt);
figure;
subplot(2, 1, 1);
plot(time, state_history(1, :), 'b', 'LineWidth', 2);
xlabel('Time (s)');
ylabel('Cart Position (m)');
title('Cart Position Control');

subplot(2, 1, 2);
plot(time, rad2deg(state_history(3, :)), 'r', 'LineWidth', 2);
xlabel('Time (s)');
ylabel('Pole Angle (degrees)');
title('Pole Angle Control');
```

```
figure;
subplot(2, 1, 1);
plot(time, control_history(1, :), 'g', 'LineWidth', 2);
xlabel('Time (s)');
ylabel('Cart Control Input');
title('Cart Control Input (u_xc)');

subplot(2, 1, 2);
plot(time, control_history(2, :), 'm', 'LineWidth', 2);
xlabel('Time (s)');
ylabel('Pole Control Input');
title('Pole Control Input (u_theta)');
```

3.2 MATLAB CODE WITH LQR AND ADMM

```
close all
clear all

% Cart-Pole Control with LQR and ADMM

% Define system parameters
M = 1.0;          % Mass of the cart (kg)
m = 0.3;          % Mass of the pole (kg)
L = 0.6;          % Length of the pole (m)
g = 9.81;         % Acceleration due to gravity (m/s^2)

% Define cart and pole dimensions
cart_length = 0.4;  % Cart length (m)
cart_width = 0.2;   % Cart width (m)
pole_length = 0.6;  % Pole length (m)
pole_width = 0.1;   % Pole width (m)

% Define initial conditions
xc_0 = 0.0;        % Initial cart position (m)
xcdot_0 = 0.05;    % Initial cart velocity (m/s)
theta_0 = pi/4;    % Initial pole angle (45 degrees)
thetadot_0 = 0.0;   % Initial pole angular velocity (rad/s)
% Desired final position for the cart
xc_final = 0.7;    % Final cart position (m)

% Define system matrices (A, B, C, D) for state-space representation
A = [0, 1, 0, 0;
    0, 0, (m*g)/M, 0;
    0, 0, 0, 1;
    0, 0, ((M+m)*g)/(L*M), 0];
B = [0; -1/M; 0; 1/(L*M)];
C = eye(4);
D = zeros(4, 1);

% LQR controller design
Q = diag([1, 1, 1, 1]);  % State cost matrix
R = 0.01;             % Control cost
K = lqr(A, B, Q, R);    % Compute the LQR gain matrix

% ADMM parameters
rho = 1.0;  % ADMM penalty parameter (tune as needed)
max_iterations = 100;  % Maximum number of ADMM iterations
```

```
% Simulation parameters
t_max = 10.0;      % Maximum simulation time (s)
dt = 0.001;        % Time step
t = 0:dt:t_max;
n = length(t);

% Initialize arrays to store state history
xc_history = zeros(1, n);
theta_history = zeros(1, n);
u_history = zeros(1, n);

% Create a figure for visualization
figure(1);
hold on;
axis([-1 1 -0.2 1.2]); % Adjust axis limits based on your needs
grid on;
xlabel('Cart Position (m)');
ylabel('Pole Angle (rad)');
title('Cart-Pole Control with LQR and ADMM');

% ADMM initialization
lambda_xc = zeros(1, n);  % Lagrange multiplier for cart position constraint
lambda_theta = zeros(1, n);  % Lagrange multiplier for pole angle constraint

% Simulate the system and control with ADMM
for i = 1:n
    % State vector
    x = [xc_0; xcdot_0; theta_0; thetadot_0];

    % LQR control law
    u = -K * x;

    % ADMM update for cart position constraint
    lambda_xc(i) = lambda_xc(i) + rho * (xc_0 - x(1) - lambda_xc(i) / rho);

    % ADMM update for pole angle constraint
    lambda_theta(i) = lambda_theta(i) + rho * (theta_0 - x(3) - lambda_theta(i) / rho);

    % Update system dynamics
    x_dot = A * x + B * u;

    % Update state variables
    xc_0 = x(1) + x_dot(1) * dt;
    xcdot_0 = x(2) + x_dot(2) * dt;
    theta_0 = x(3) + x_dot(3) * dt;
    thetadot_0 = x(4) + x_dot(4) * dt;

    % Store state history and control input
    xc_history(i) = xc_0;
    theta_history(i) = theta_0;
    u_history(i) = u;

    % Visualize the cart-pole system
    cla; % Clear previous frame
    % Draw cart
    cart_x = [xc_0 - cart_length/2, xc_0 + cart_length/2, xc_0 + cart_length/2, xc_0 - cart_length/2];
```

```
    cart_y = [0, 0, cart_width, cart_width];
    fill(cart_x, cart_y, 'b');
    % Draw pole
    pole_x = [xc_0, xc_0 + pole_length*sin(theta_0)];
    pole_y = [cart_width, cart_width + pole_length*cos(theta_0)];
    plot(pole_x, pole_y, 'r', 'LineWidth', 2);

    % Pause for visualization (adjust as needed)
    pause(0.01);
end

% Plot results (if needed)
figure;
subplot(2, 1, 1);
plot(t, xc_history, 'b', 'LineWidth', 2);
hold on;
plot([0, t_max], [xc_final, xc_final], 'r--', 'LineWidth', 2);
xlabel('Time (s)');
ylabel('Cart Position (m)');
title('Cart Position Control with LQR and ADMM');
legend('Actual Position', 'Desired Position');
grid on;
subplot(2, 1, 2);
plot(t, theta_history, 'b', 'LineWidth', 2);
hold on;
plot([0, t_max], [pi/4, pi/4], 'r--', 'LineWidth', 2);
xlabel('Time (s)');
ylabel('Pole Angle (rad)');
title('Pole Angle Control with LQR and ADMM');
legend('Actual Angle', 'Desired Angle');
grid on;

figure;
plot(t, u_history, 'b', 'LineWidth', 2);
xlabel('Time (s)');
ylabel('Control Input (u)');
title('Control Input with LQR and ADMM');
grid on;
```

APPENDIX 4- LQR+ADMM CONTROL OF TWO CART-POLE SYSTEMS: MATLAB CODE

```
close all
clear all

% Cart-Pole Control with LQR and ADMM for Two Systems

% Define system parameters
M = 1.0;          % Mass of the cart (kg)
m = 0.3;          % Mass of the pole (kg)
L = 0.6;          % Length of the pole (m)
g = 9.81;         % Acceleration due to gravity (m/s^2)

% Define cart and pole dimensions
cart_length = 0.4;  % Cart length (m)
cart_width = 0.2;   % Cart width (m)
pole_length = 0.6;  % Pole length (m)
pole_width = 0.1;   % Pole width (m)

% Define initial conditions for the first cart-pole system
xc_0_1 = 0.0;        % Initial cart position (m)
xcdot_0_1 = 0.05;    % Initial cart velocity (m/s)
theta_0_1 = pi/4;    % Initial pole angle (45 degrees)
thetadot_0_1 = 0.0;  % Initial pole angular velocity (rad/s)

% Define initial conditions for the second cart-pole system
xc_0_2 = 1.0;        % Initial cart position (m)
xcdot_0_2 = 0.05;    % Initial cart velocity (m/s)
theta_0_2 = pi/4;    % Initial pole angle (45 degrees)
thetadot_0_2 = 0.0;  % Initial pole angular velocity (rad/s)

% Desired final positions for both cart-pole systems
xc_final_1 = 0.7;    % Final cart position for the first system (m)
xc_final_2 = 1.7;    % Final cart position for the second system (m)

% Define system matrices (A, B, C, D) for state-space representation
A = [0, 1, 0, 0;
    0, 0, (m*g)/M, 0;
    0, 0, 0, 1;
    0, 0, ((M+m)*g)/(L*M), 0];
B = [0; -1/M; 0; 1/(L*M)];
C = eye(4);
D = zeros(4, 1);

% LQR controller design
Q = diag([1, 1, 1, 1]);  % State cost matrix
R = 0.01;            % Control cost
K = lqr(A, B, Q, R);    % Compute the LQR gain matrix

% ADMM parameters
rho = 1.0;  % ADMM penalty parameter (tune as needed)
max_iterations = 100;  % Maximum number of ADMM iterations

% Simulation parameters
t_max = 10.0;       % Maximum simulation time (s)
dt = 0.001;         % Time step
```

```
t = 0:dt:t_max;
n = length(t);

% Initialize arrays to store state history for both systems
xc_history_1 = zeros(1, n);
theta_history_1 = zeros(1, n);
u_history_1 = zeros(1, n);
xc_history_2 = zeros(1, n);
theta_history_2 = zeros(1, n);
u_history_2 = zeros(1, n);

% Create figures for visualization
figure(1);
subplot(1, 2, 1);
hold on;
axis([-1 2 -0.2 1.2]); % Adjust axis limits based on your needs
grid on;
xlabel('Cart Position (m)');
ylabel('Pole Angle (rad)');
title('Cart-Pole System 1');

subplot(1, 2, 2);
hold on;
axis([-1 2 -0.2 1.2]); % Adjust axis limits based on your needs
grid on;
xlabel('Cart Position (m)');
ylabel('Pole Angle (rad)');
title('Cart-Pole System 2');

% ADMM initialization
lambda_xc_1 = zeros(1, n);    % Lagrange multiplier for cart position constraint (System 1)
lambda_theta_1 = zeros(1, n); % Lagrange multiplier for pole angle constraint (System 1)
lambda_xc_2 = zeros(1, n);    % Lagrange multiplier for cart position constraint (System 2)
lambda_theta_2 = zeros(1, n); % Lagrange multiplier for pole angle constraint (System 2)

% Simulate the systems and control with ADMM
for i = 1:n
    % State vector for the first system
    x_1 = [xc_0_1; xcdot_0_1; theta_0_1; thetadot_0_1];

    % State vector for the second system
    x_2 = [xc_0_2; xcdot_0_2; theta_0_2; thetadot_0_2];

    % LQR control law for both systems
    u_1 = -K * x_1;
    u_2 = -K * x_2;

    % ADMM update for cart position constraint (System 1)
    lambda_xc_1(i) = lambda_xc_1(i) + rho * (xc_0_1 - x_1(1) - lambda_xc_1(i) / rho);

    % ADMM update for pole angle constraint (System 1)
    lambda_theta_1(i) = lambda_theta_1(i) + rho * (theta_0_1 - x_1(3) - lambda_theta_1(i) / rho);

    % ADMM update for cart position constraint (System 2)
    lambda_xc_2(i) = lambda_xc_2(i) + rho * (xc_0_2 - x_2(1) - lambda_xc_2(i) / rho);

    % ADMM update for pole angle constraint (System 2)
```

```
    lambda_theta_2(i) = lambda_theta_2(i) + rho * (theta_0_2 - x_2(3) - lambda_theta_2(i) / rho);

    % Update system dynamics for both systems
    x_dot_1 = A * x_1 + B * u_1;
    x_dot_2 = A * x_2 + B * u_2;

    % Update state variables for both systems
    xc_0_1 = x_1(1) + x_dot_1(1) * dt;
    xcdot_0_1 = x_1(2) + x_dot_1(2) * dt;
    theta_0_1 = x_1(3) + x_dot_1(3) * dt;
    thetadot_0_1 = x_1(4) + x_dot_1(4) * dt;

    xc_0_2 = x_2(1) + x_dot_2(1) * dt;
    xcdot_0_2 = x_2(2) + x_dot_2(2) * dt;
    theta_0_2 = x_2(3) + x_dot_2(3) * dt;
    thetadot_0_2 = x_2(4) + x_dot_2(4) * dt;

    % Store state history and control input for both systems
    xc_history_1(i) = xc_0_1;
    theta_history_1(i) = theta_0_1;
    u_history_1(i) = u_1;

    xc_history_2(i) = xc_0_2;
    theta_history_2(i) = theta_0_2;
    u_history_2(i) = u_2;

    % Visualize the cart-pole systems
    subplot(1, 2, 1);
    cla; % Clear previous frame
    % Draw cart (System 1)
    cart_x_1 = [xc_0_1 - cart_length/2, xc_0_1 + cart_length/2, xc_0_1 + cart_length/2, xc_0_1 - cart_length/2];
    cart_y_1 = [0, 0, cart_width, cart_width];
    fill(cart_x_1, cart_y_1, 'b');
    % Draw pole (System 1)
    pole_x_1 = [xc_0_1, xc_0_1 + pole_length*sin(theta_0_1)];
    pole_y_1 = [cart_width, cart_width + pole_length*cos(theta_0_1)];
    plot(pole_x_1, pole_y_1, 'r', 'LineWidth', 2);
    title('Cart-Pole System 1');

    subplot(1, 2, 2);
    cla; % Clear previous frame
    % Draw cart (System 2)
    cart_x_2 = [xc_0_2 - cart_length/2, xc_0_2 + cart_length/2, xc_0_2 + cart_length/2, xc_0_2 - cart_length/2];
    cart_y_2 = [0, 0, cart_width, cart_width];
    fill(cart_x_2, cart_y_2, 'b');
    % Draw pole (System 2)
    pole_x_2 = [xc_0_2, xc_0_2 + pole_length*sin(theta_0_2)];
    pole_y_2 = [cart_width, cart_width + pole_length*cos(theta_0_2)];
    plot(pole_x_2, pole_y_2, 'r', 'LineWidth', 2);
    title('Cart-Pole System 2');

    % Pause for visualization (adjust as needed)
    pause(0.01);
end

% Plot results for both systems
figure;
```

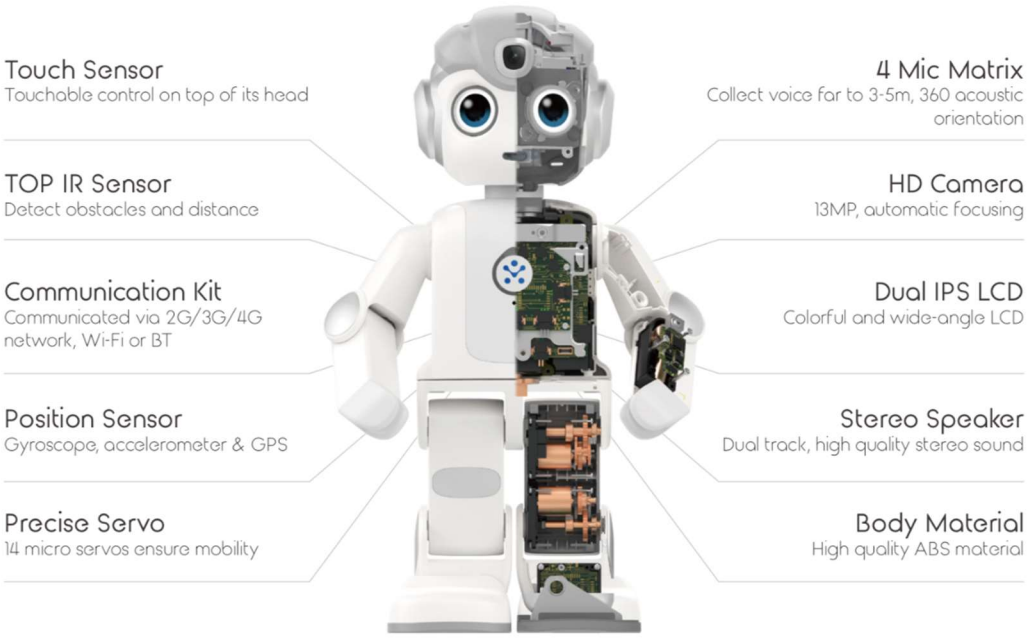
```
subplot(2, 1, 1);
plot(t, xc_history_1, 'b', 'LineWidth', 2);
hold on;
plot(t, xc_history_2, 'g', 'LineWidth', 2);
plot([0, t_max], [xc_final_1, xc_final_1], 'r--', 'LineWidth', 2);
plot([0, t_max], [xc_final_2, xc_final_2], 'm--', 'LineWidth', 2);
xlabel('Time (s)');
ylabel('Cart Position (m)');
title('Cart Position Control with LQR and ADMM (System 1 and 2)');
legend('System 1', 'System 2', 'Desired Position 1', 'Desired Position 2');
grid on;

subplot(2, 1, 2);
plot(t, theta_history_1, 'b', 'LineWidth', 2);
hold on;
plot(t, theta_history_2, 'g', 'LineWidth', 2);
plot([0, t_max], [pi/4, pi/4], 'r--', 'LineWidth', 2);
xlabel('Time (s)');
ylabel('Pole Angle (rad)');
title('Pole Angle Control with LQR and ADMM (System 1 and 2)');
legend('System 1', 'System 2', 'Desired Angle');
grid on;

figure;
subplot(2, 1, 1);
plot(t, u_history_1, 'b', 'LineWidth', 2);
xlabel('Time (s)');
ylabel('Control Input (u)');
title('Control Input with LQR and ADMM (System 1)');
grid on;

subplot(2, 1, 2);
plot(t, u_history_2, 'g', 'LineWidth', 2);
xlabel('Time (s)');
ylabel('Control Input (u)');
title('Control Input with LQR and ADMM (System 2)');
grid on;
```

APPENDIX 5- ALPHA MINI ROBOT INFORMATION



MAIN COMPONENTS	DESCRIPTION
Power Generator	Generates the energy required for the device's operation
Type C cable	Commonly used USB connector for charging and data transfer
Button turn on/off	Turn the device on or off
Camera (intelligent composition, autofocus, one-tap sharing functions, facial memorization and recognition)	Camera with advanced features such as intelligent composition, autofocus, easy sharing, and facial memorization and recognition
Microphone	Captures sound for the device's audio functions
LCD Eyes	Liquid crystal displays designed to resemble eyes and display information or expression
LED mouth	Formed by LED lights, used to express states or interact visually
Infrared Sensor	Detects infrared radiation for object detection or remote control
SIM Slot	Used for cellular connectivity
Stereo Speakers	Emit sound in stereo format
Sling hole	A hole designed for attaching a strap or lanyard
Touch sensor	A sensor that detects physical contact or pressure, often used for user interaction.

Mechanic Info

Measurements	Height 245 m, width 149mm, thickness 112 mm
Weight	700g
Joints	14 (3 head joints, 1 waist joint,6 leg jints)
Network Standard	4G
Wifi	2.4G (Wlan 802.11 b/g/n, BT 4.2, GPS
User Storage	RAM 2GB/ROM16GB
SIM Card	Nano SIM (4FF, single card slot)
Speakers	2-channel stereo speakers, 0.8bW*2
Microphones	4 microphone array, 1 communication microphone
Display screen	1.22 inch, 262K color, TFT display screen, 240*240 pixel
Camera	13 million pixels
Sensors	GPS, accelerometer, gyroscope, touch sensor, infrared sensor
Battery	Li-Polymer lithium polymer battery, 3.85 V, 4060mAh
Power adapter	Input AC 100V 240 V 50/60 Hz 0.5A Output DC 5V 3A/9V 2A/12V 1.5A
Operating temperatures	0°C-40°C
Control Methods	Power button, volume button, App, voice control

USEFUL LINKS:

https://docs.ubtrobot.com/alphamini/#/en-us/DEVELOPMENT-GUIDANCE

https://docs.ubtrobot.com/alphamini/#/en-us/PRODUCT-INSTRUCTION

https://github.com/GiovTemp/AlphaMiniServer

https://github.com/marklogg/ubt_alphamini_app_demo

https://ide.ubtrobot.com/#/

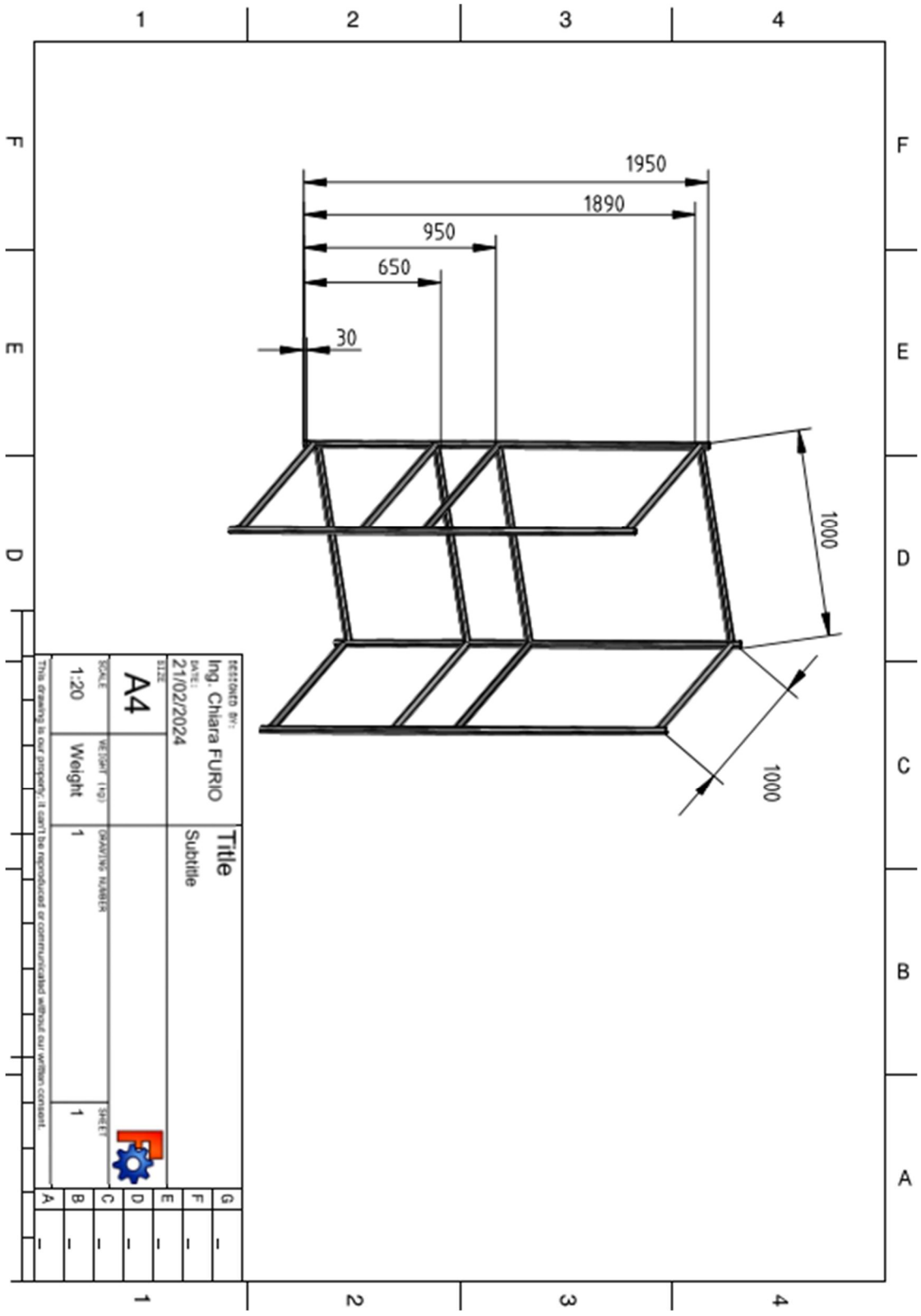
https://developer.android.com/studio?hl=it

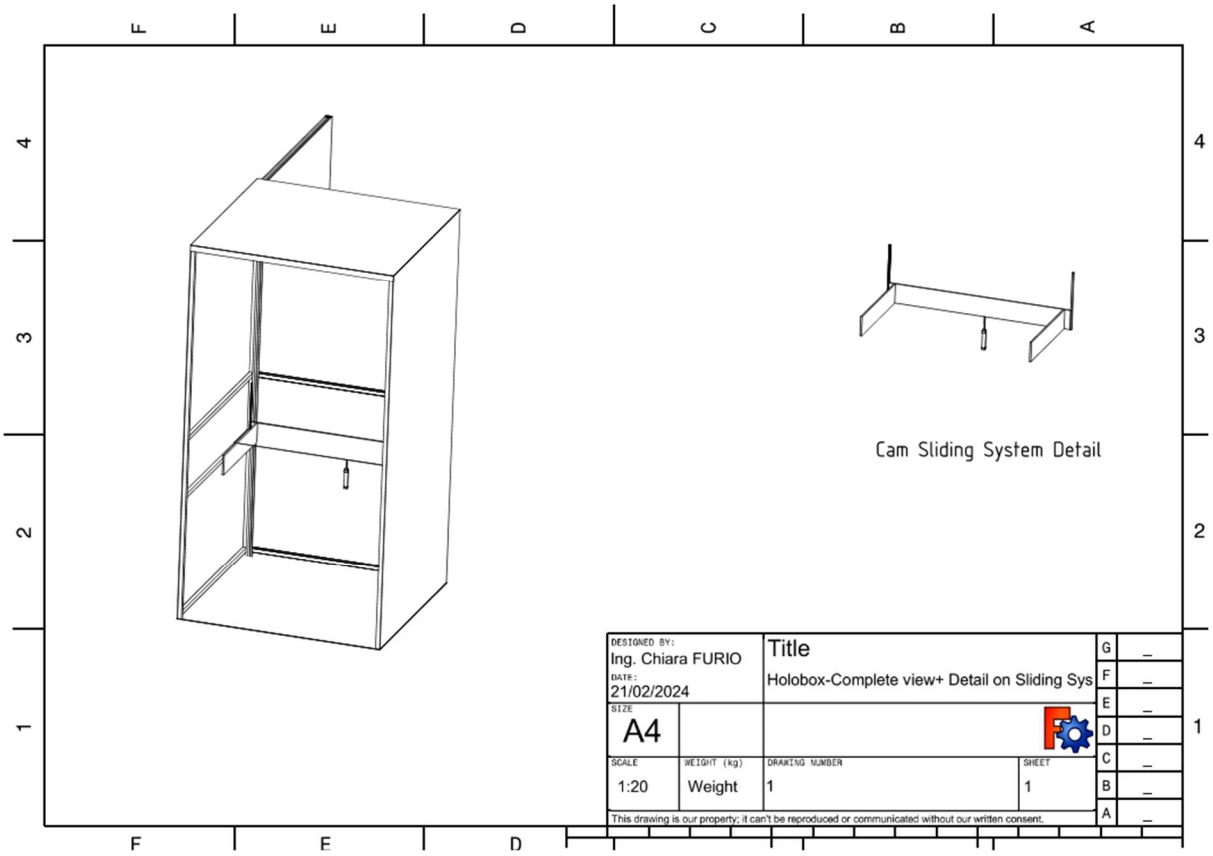
https://alphacephei.com/vosk/

https://pypi.org/project/gTTS/

https://www.docker.com/

APPENDIX 6- HOLOBOX FREECAD OPTIMIZATION

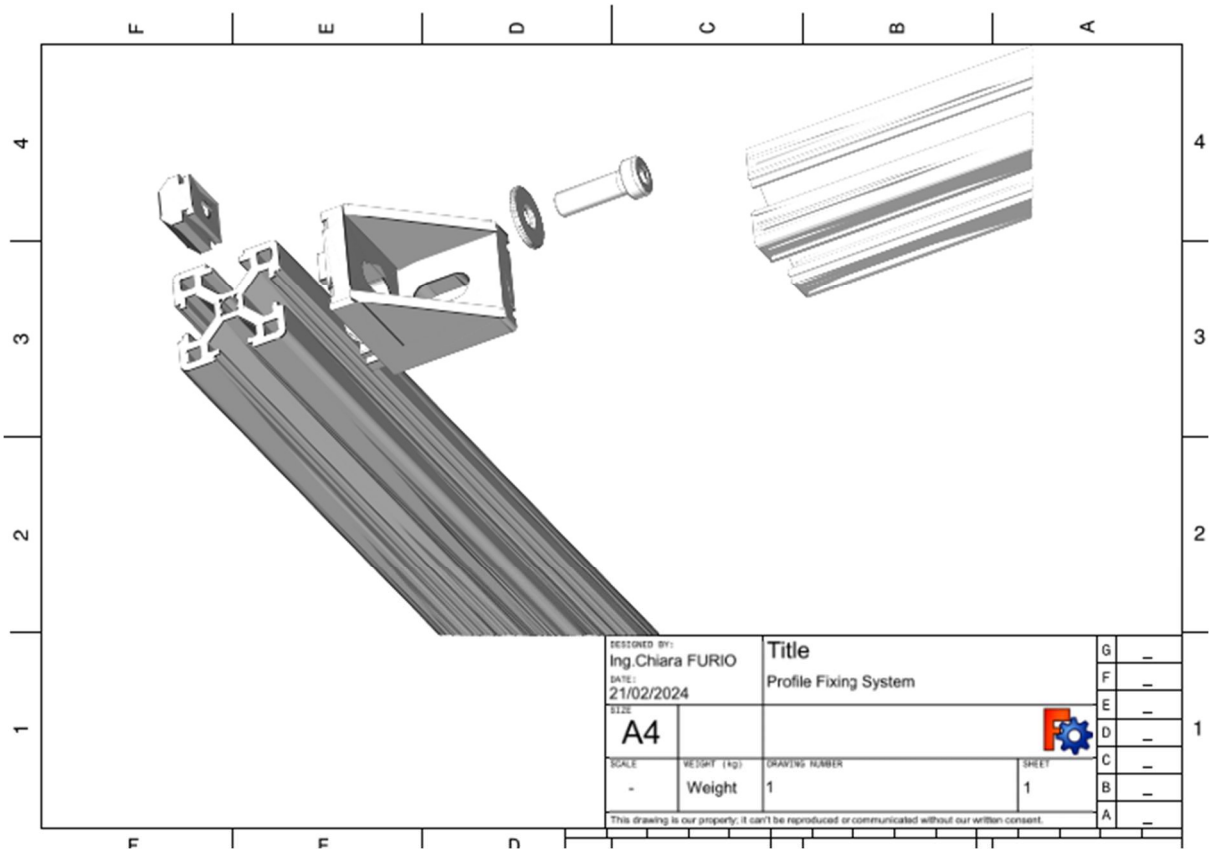




Cam Sliding System Detail

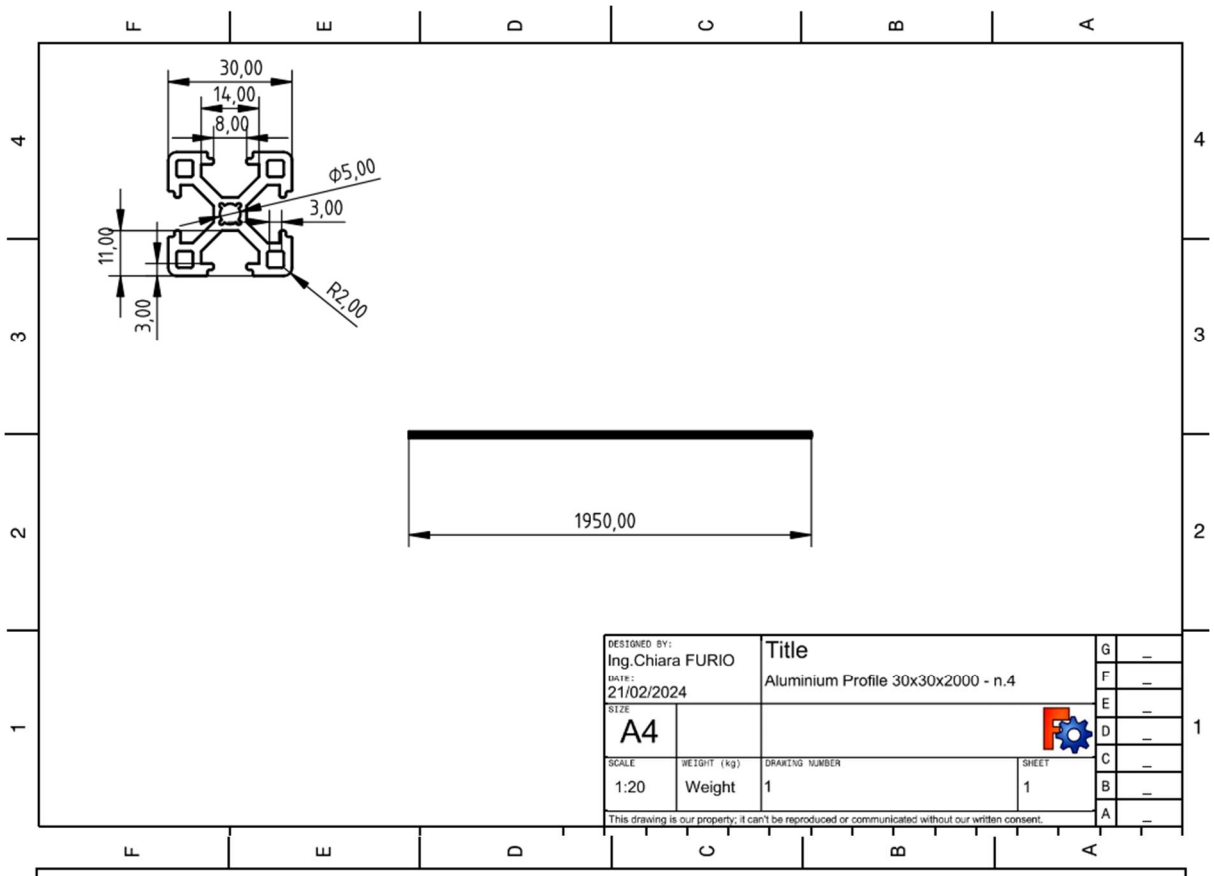
DESIGNED BY: Ing. Chiara FURIO		Title: Holobox-Complete view+ Detail on Sliding Sys	
DATE: 21/02/2024			
SIZE: A4			
SCALE: 1:20	WEIGHT (kg): Weight	DRAWING NUMBER: 1	SHEET: 1

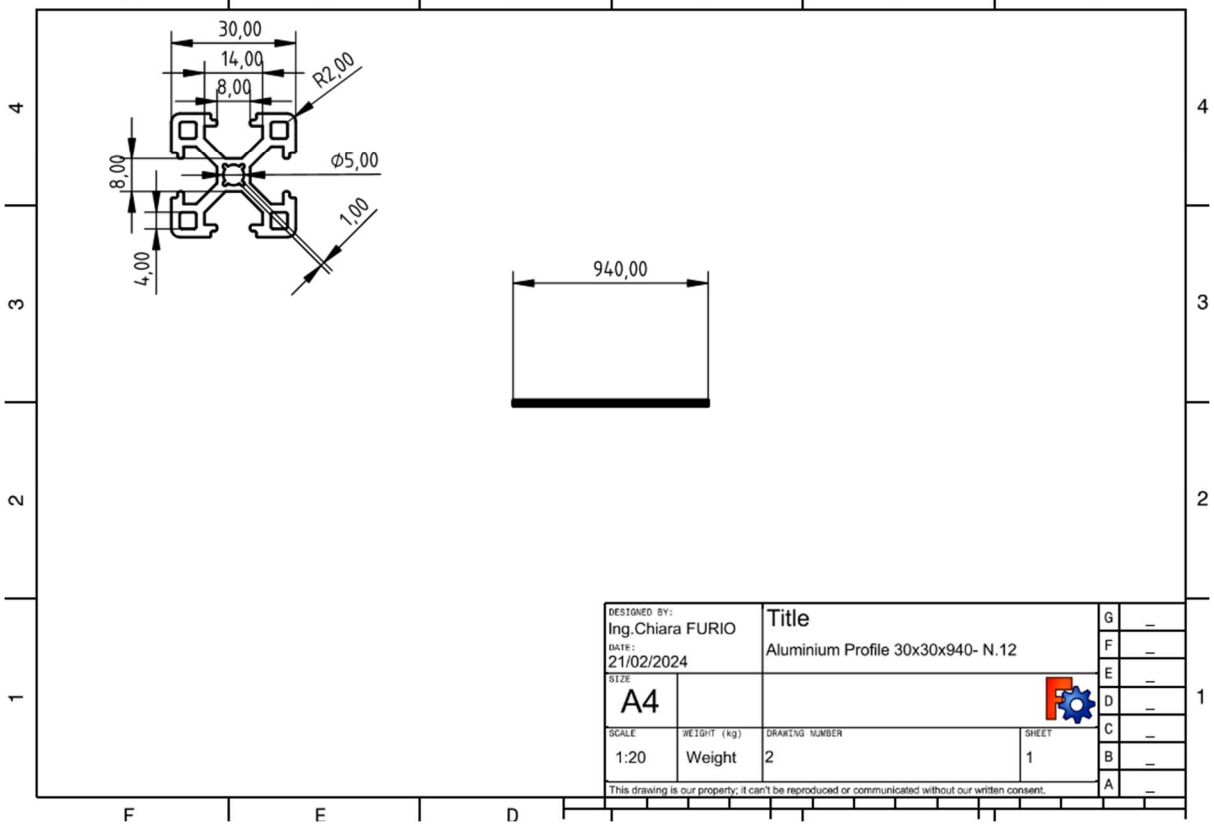
This drawing is our property; it can't be reproduced or communicated without our written consent.

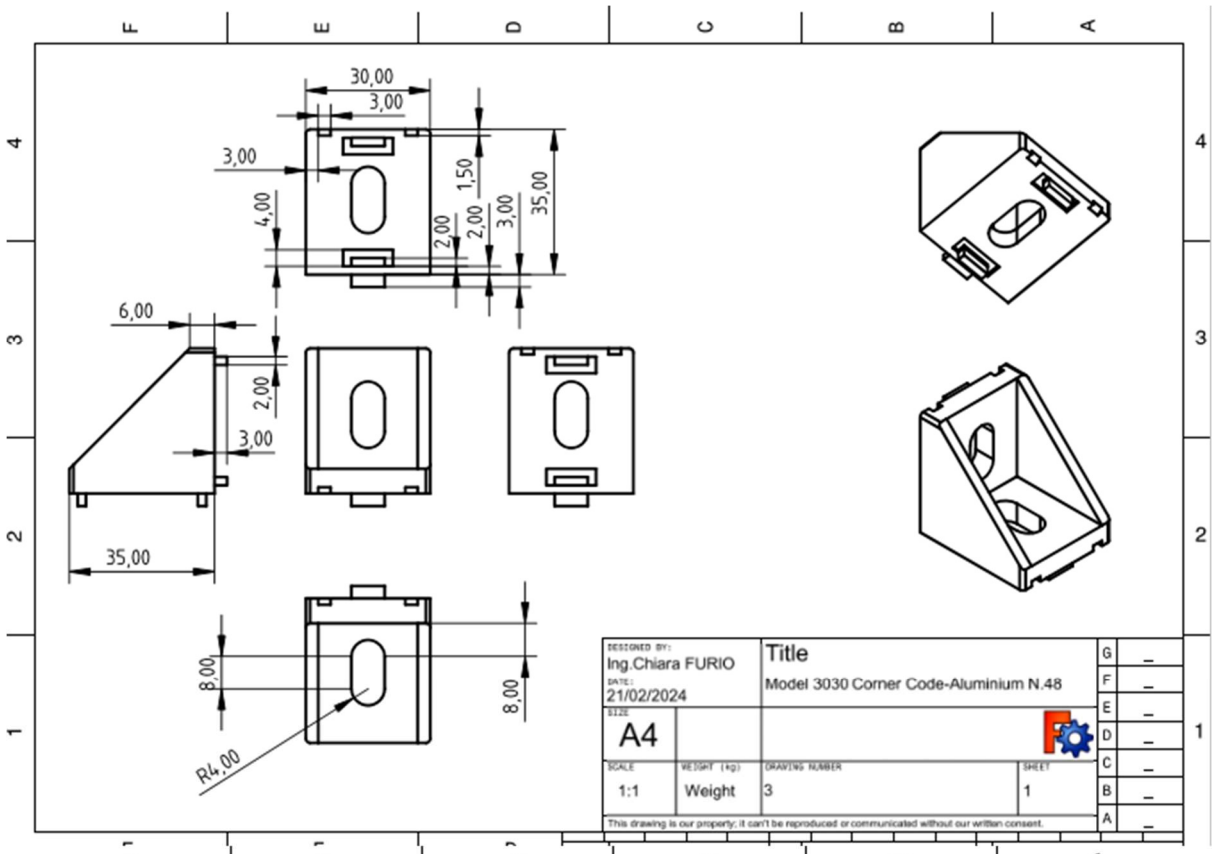


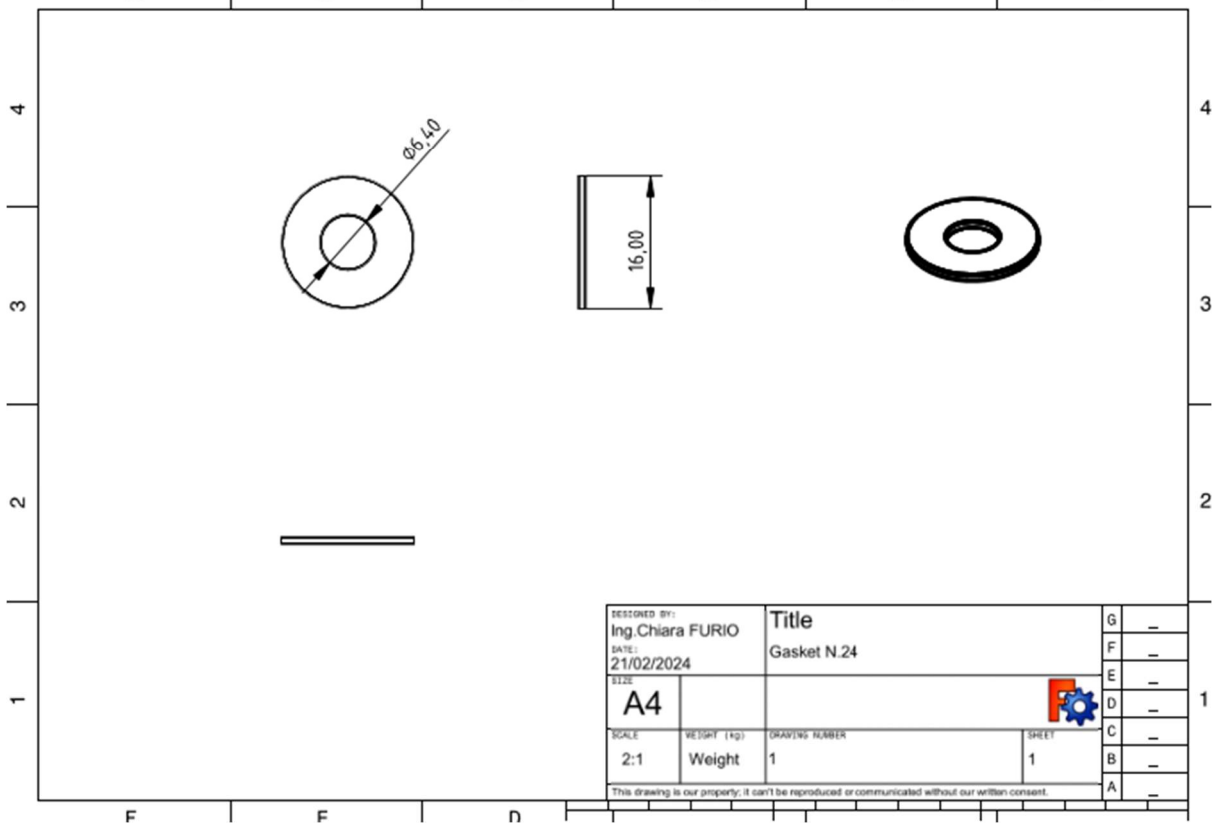
DESIGNED BY: Ing.Chiara FURIO		Title: Profile Fixing System	
DATE: 21/02/2024			
SIZE: A4			
SCALE: -	WEIGHT (kg): Weight	DRAWING NUMBER: 1	SHEET: 1

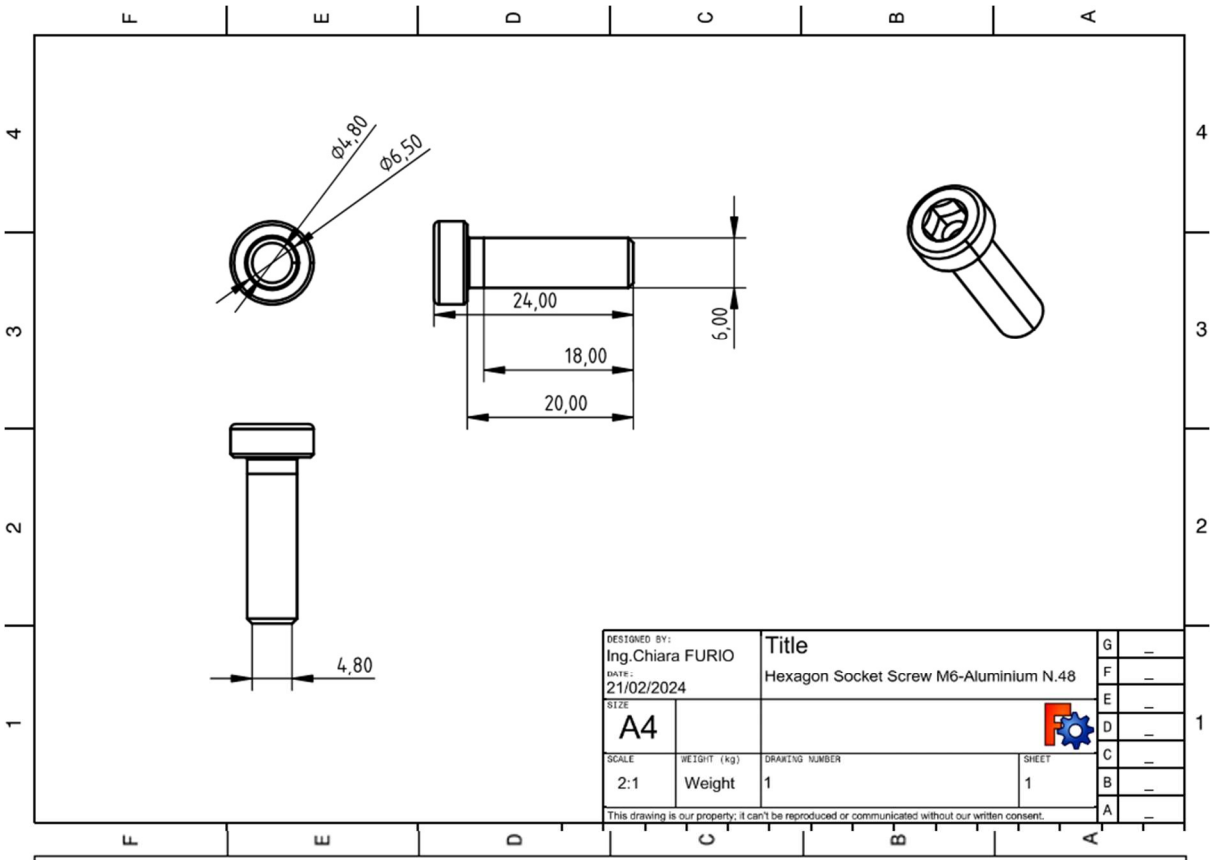
This drawing is our property; it can't be reproduced or communicated without our written consent.

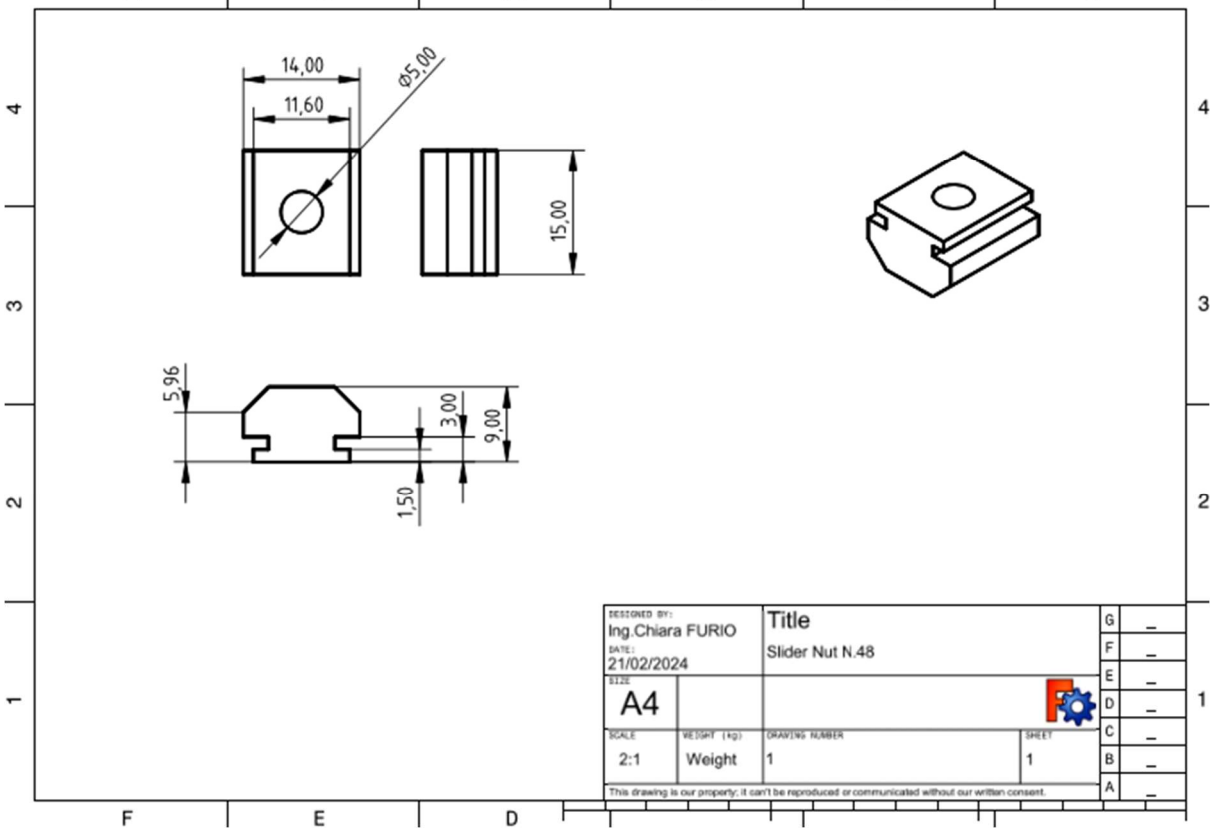

30,00
14,00
8,00
ϕ5,00
3,00
11,00
3,00
R2,00
1950,00
DESIGNED BY:
Ing.Chiara FURIO
DATE:
21/02/2024
Title
Aluminium Profile 30x30x2000 - n.4
SIZE
A4
SCALE
1:20
WEIGHT (kg)
Weight
DRAWING NUMBER
1
SHEET
1
This drawing is our property; it can't be reproduced or communicated without our written consent.

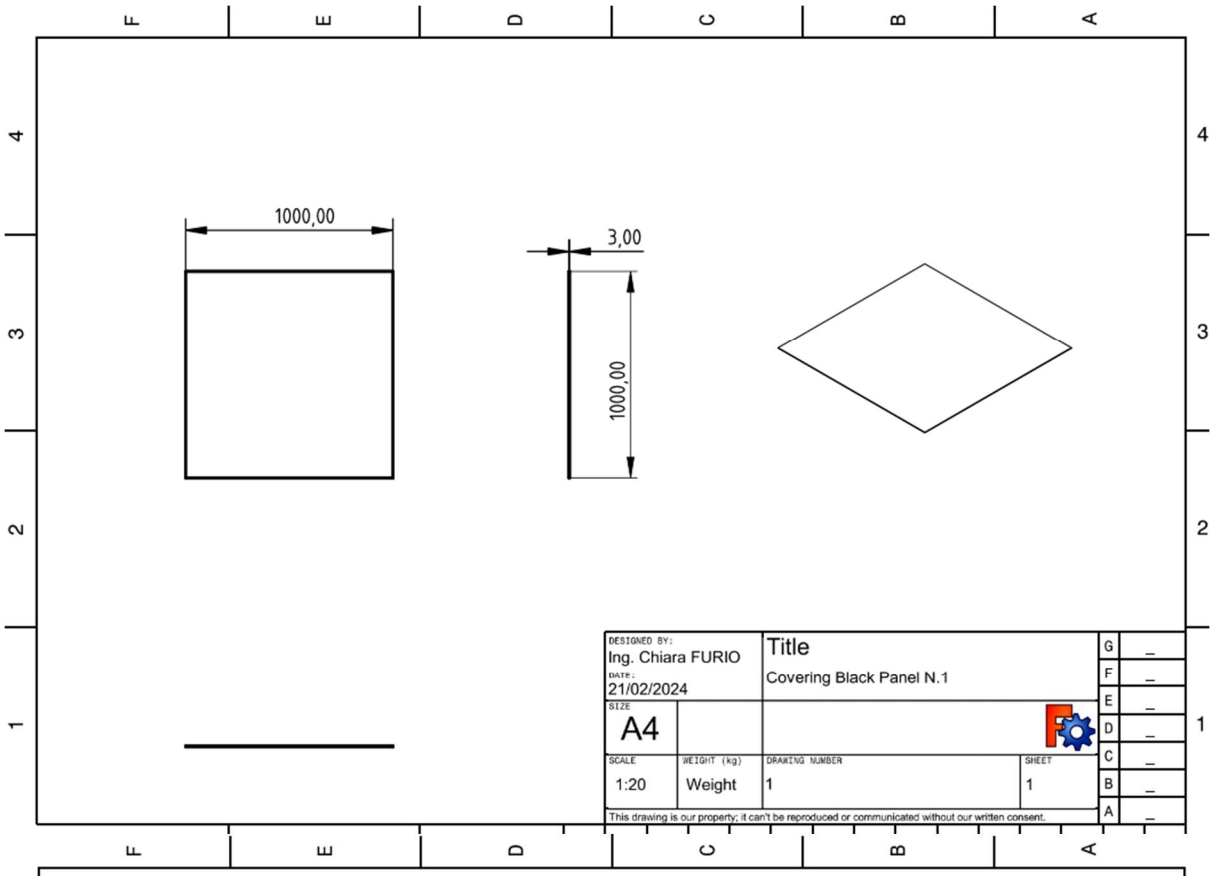

30,00
14,00
8,00
R2,00
8,00
ϕ5,00
1,00
4,00
940,00
DESIGNED BY:
Ing.Chiara FURIO
DATE:
21/02/2024
Title
Aluminium Profile 30x30x940- N.12
SIZE
A4
SCALE
1:20
WEIGHT (kg)
Weight
DRAWING NUMBER
2
SHEET
1
This drawing is our property; it can't be reproduced or communicated without our written consent.

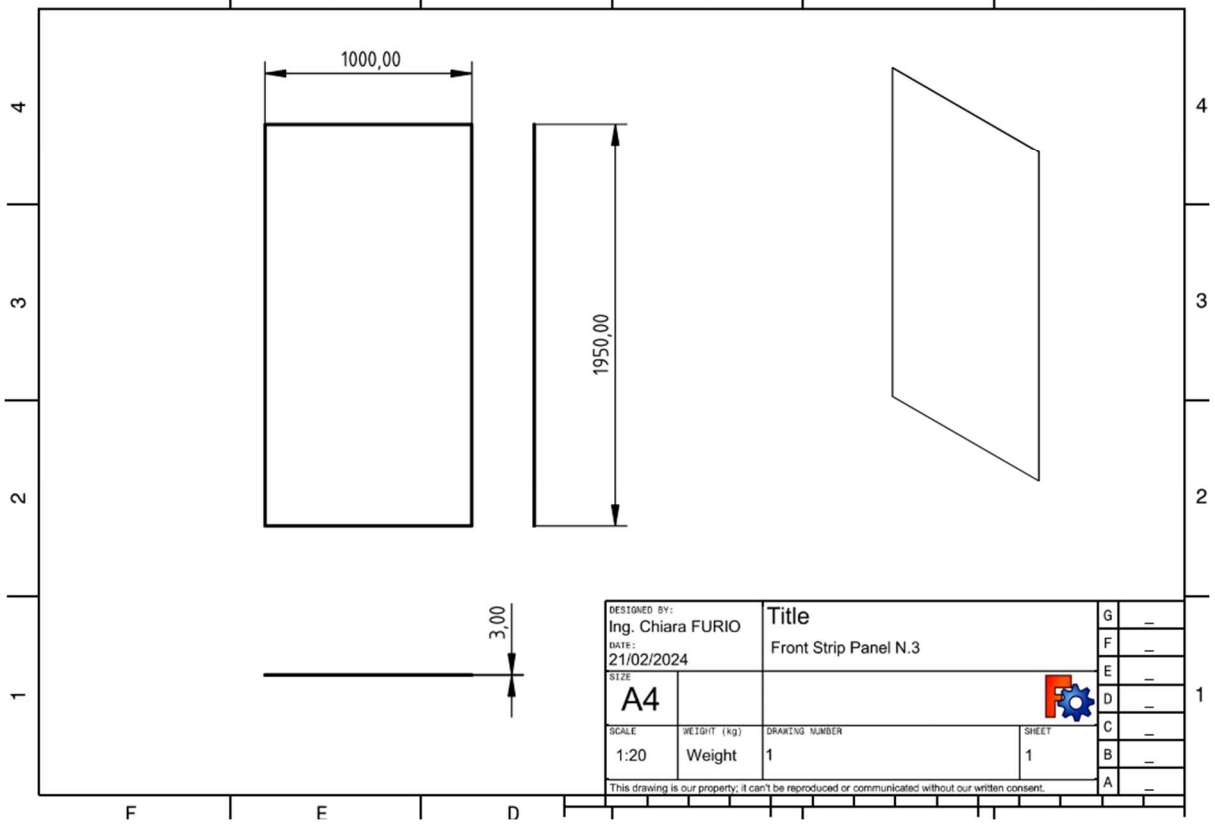

30,00
3,00
3,00
4,00
1,50
2,00
2,00
3,00
35,00
6,00
2,00
3,00
35,00
8,00
8,00
R4,00
DESIGNED BY:
Ing.Chiara FURIO
DATE:
21/02/2024
Title
Model 3030 Corner Code-Aluminium N.48
SIZE
A4
SCALE
1:1
WEIGHT (kg)
Weight
DRAWING NUMBER
3
SHEET
1
This drawing is our property; it can't be reproduced or communicated without our written consent.

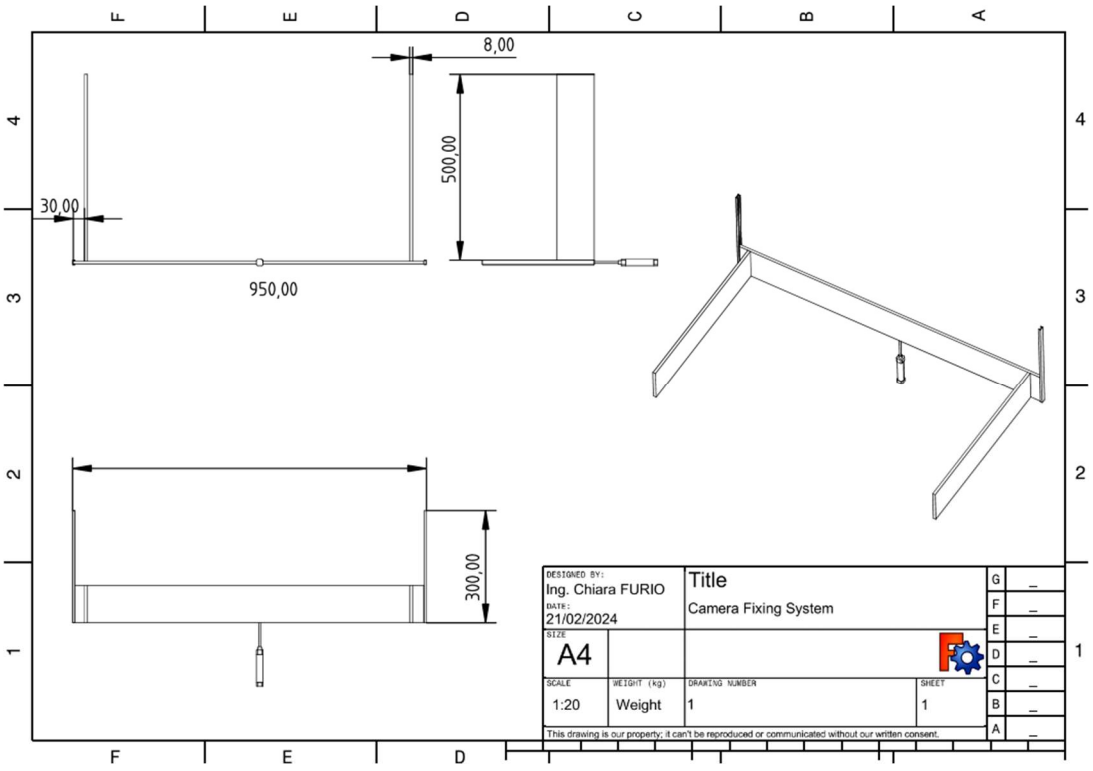

Ø6,40
16,00
DESIGNED BY:
Ing.Chiara FURIO
DATE:
21/02/2024
Title
Gasket N.24
SIZE
A4
SCALE
2:1
WEIGHT (kg)
Weight
DRAWING NUMBER
1
SHEET
1
This drawing is our property; it can't be reproduced or communicated without our written consent.


Φ4,80
Φ6,50
24,00
18,00
20,00
6,00
4,80
DESIGNED BY:
Ing.Chiara FURIO
DATE:
21/02/2024
Title
Hexagon Socket Screw M6-Aluminium N.48
SIZE
A4
SCALE
2:1
WEIGHT (kg)
Weight
DRAWING NUMBER
1
SHEET
1
This drawing is our property; it can't be reproduced or communicated without our written consent.


14,00
11,60
Φ5,00
15,00
5,96
3,00
9,00
1,50
DESIGNED BY:
Ing.Chiara FURIO
DATE:
21/02/2024
Title
Slider Nut N.48
SIZE
A4
SCALE
2:1
WEIGHT (kg)
Weight
DRAWING NUMBER
1
SHEET
1
This drawing is our property; it can't be reproduced or communicated without our written consent.


1000,00
3,00
1000,00
DESIGNED BY:
Ing. Chiara FURIO
DATE:
21/02/2024
Title
Covering Black Panel N.1
SIZE
A4
SCALE
1:20
WEIGHT (kg)
Weight
DRAWING NUMBER
1
SHEET
1
This drawing is our property; it can't be reproduced or communicated without our written consent.


1000,00
1950,00
3,00
DESIGNED BY:
Ing. Chiara FURIO
DATE:
21/02/2024
Title
Front Strip Panel N.3
SIZE
A4
SCALE
1:20
WEIGHT (kg)
Weight
DRAWING NUMBER
1
SHEET
1
This drawing is our property; it can't be reproduced or communicated without our written consent.



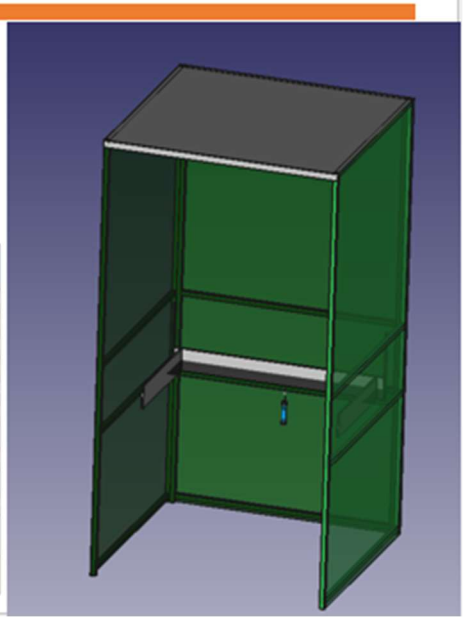
HOW TO CREATE AN IMMERSIVE SPACE

HOLOBOX:

TOTAL BULK=1000x1000x1950

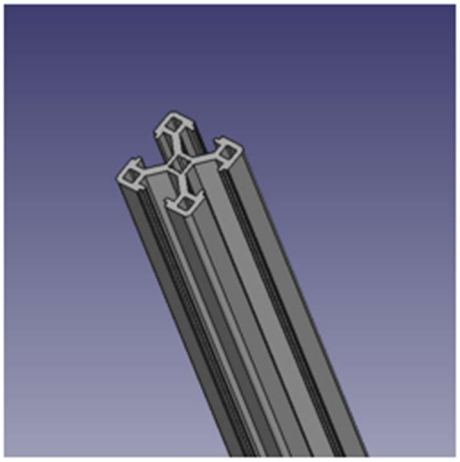
MATERIAL: ALLUMINIUM

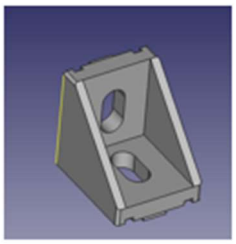
eq. ium.	ITEM DESCRIPTION	DIMENSIONS	QTY
1	Aluminium Profile 30x30x1950 AW6060- groove 8mm	30 x 30 x 1950	4
2	Aluminium Profile 30x30x940 AW6060-grove 8mm	30 x 30 x 940	12
3	Model 3030 Corner Code-Aluminium	30x35x35	48
4	Flat Gasket M6x1-Aluminium (lot of 50 pieces)	M6x1	24
5	Hexagon Socket Screw M6-Aluminium	M6	48
6	Slider Nut for groove 8mm-Aluminium (lot of 50 pieces)	M6	48
7	Black or Green polypropylene panel for covering	1000x1000x3	1
8	Black of Green polypropylene panel for front strip + support base	1000x2000x3	3
9	Black Curtain fireproof 1000x2100	1000x2000	1
10	TP-Link TC70 Spherical Surveillance Camera IP Security Camera Indoor Ceiling/Wall		3
11	Ball Bearing Drawer Slides Guide, 10 kg Load Capacity, Full Extension, Bearings Total Extraction	50 cm	1
12	Linear Actuator 6000N 300mm / 1320lb 12 Inches, Stroke DC 12V Electric Motor	Return Length: 18,7 inch=48cm	1

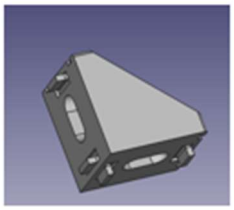


REALITY VS VIRTUAL REALITY

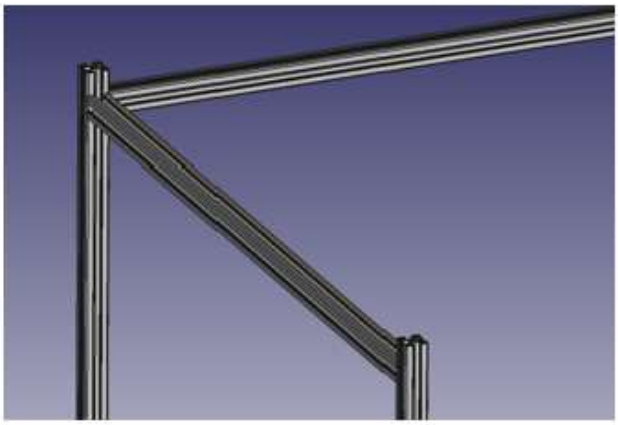


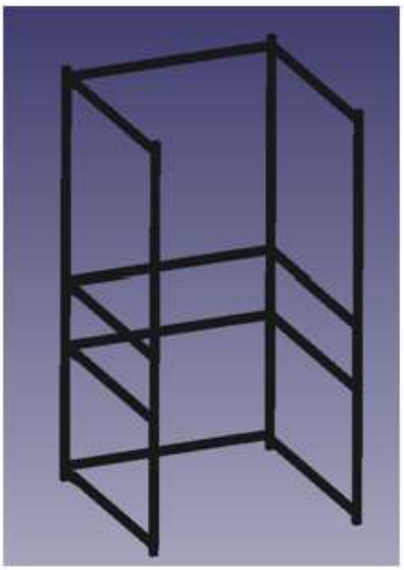






BASE STRUCTURE

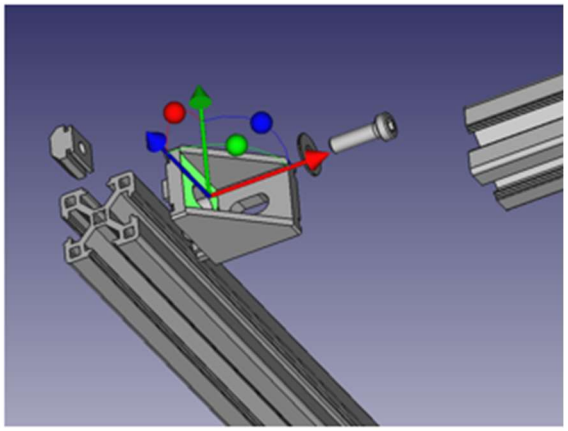




VERTICAL AND HORIZONTAL PROFILES FIXING SYSTEMS

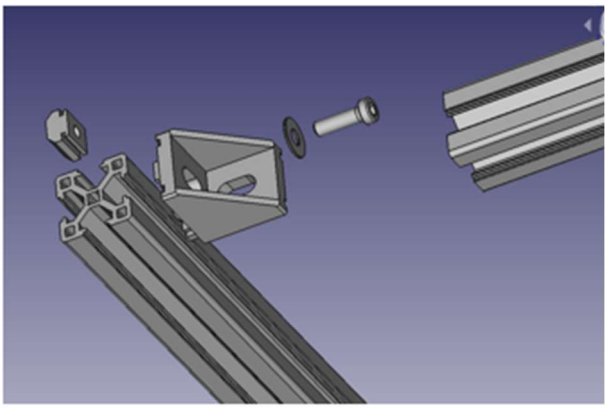
MATERIAL: ALLUMINIUM

- VERTICAL PROFILE 30x30x1950
- HORIZONTALPROFILE 30x30x940

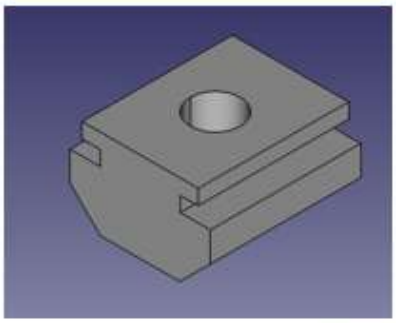


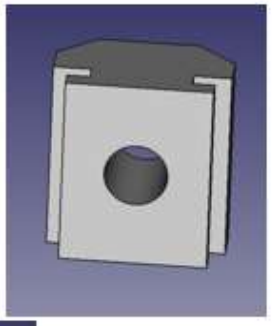
PROFILES FIXING SYSTEMS

- ✓ Corner Model 3030-Aluminium
- ✓SLIDER NUT
- ✓FLAT GASKET
- ✓HEXAGON SOCKET SCREW





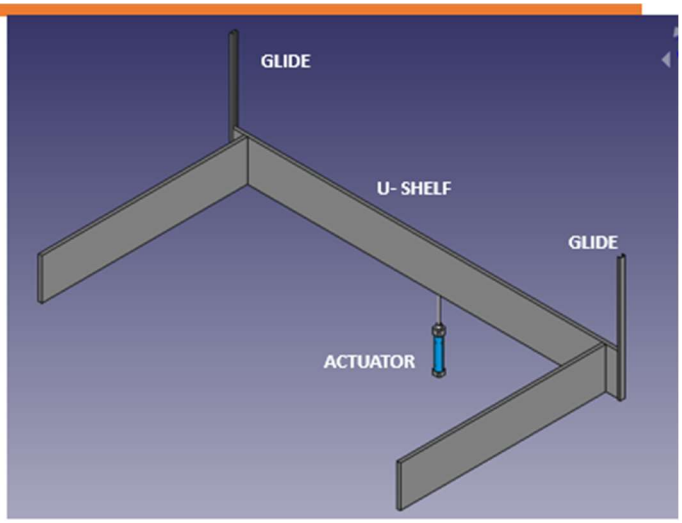


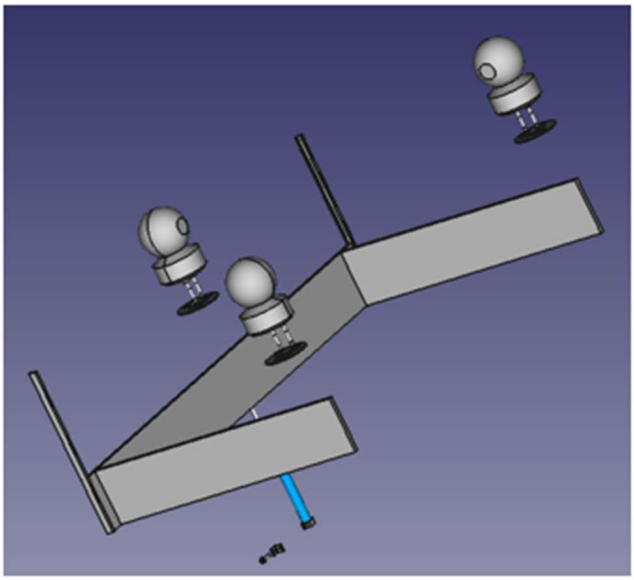


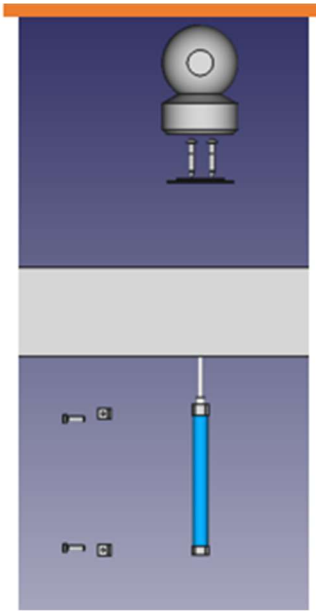


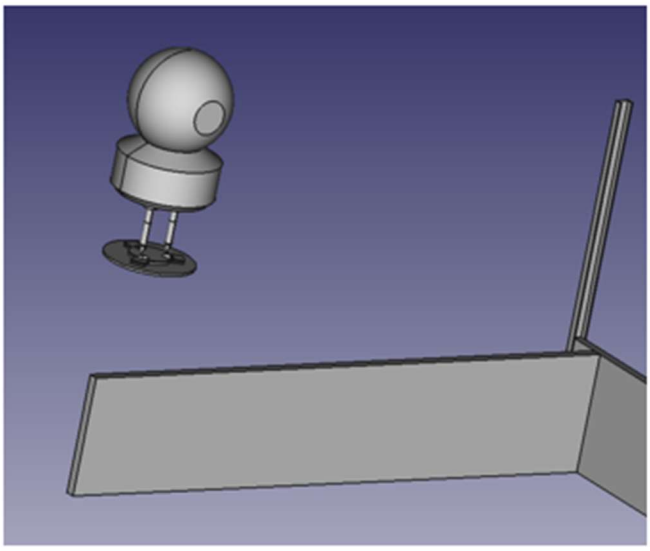
SLINDING SYSTEM FOR CAMERAS:

- GLIDES
- U-SHAPED SHELVES
- ELECTRICAL ACTUATOR



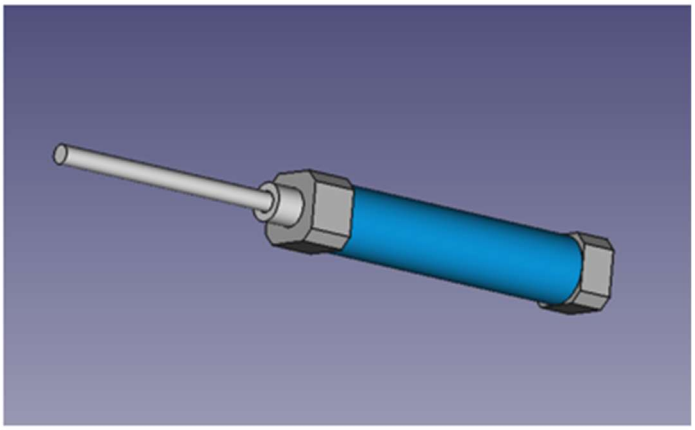






AUTOMATIC SLINDING SYSTEM FOR CAMERAS:

ELECTRICAL ACTUATOR



APPENDIX 7- REVERSE ENGINEERING AND 3D PRINTING TOOLS

Reverse engineering is a structured process that enables the reconstruction of a digital model from an existing physical object, typically in the absence of original design data. It is employed across various domains to facilitate analysis, replication, optimization, or integration into modern digital workflows. The process generally involves the following key phases:

- Data Acquisition – The physical object is scanned using technologies such as laser scanners, structured light, or photogrammetry to generate a high-resolution point cloud representing its surface geometry.

- Point Cloud Processing – The acquired data is cleaned and filtered to remove noise and outliers, preparing it for accurate surface reconstruction.

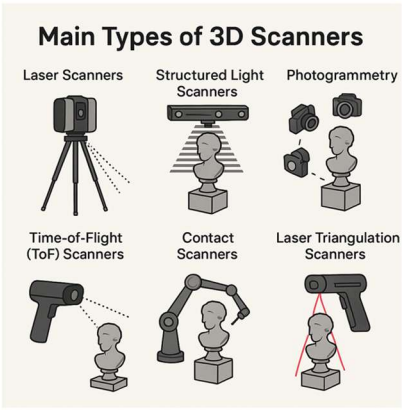
- Mesh Reconstruction – A polygonal mesh is generated from the point cloud, capturing the object's geometry in a form suitable for editing and analysis.

- CAD Modelling – The mesh is converted into a parametric or NURBS-based CAD model to enable precise design modifications, simulations, and integration into engineering environments.

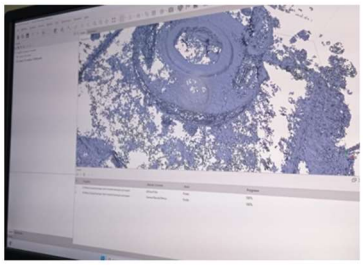
- Validation and Optimization – The digital model is assessed for dimensional accuracy and optimized according to the intended application, such as manufacturing or additive fabrication.

MAIN 3D SCANNERS

- **LASER SCANNERS: Use laser beams to measure distances with high accuracy. Ideal for large objects and architectural surveys.** *E.g.: Terrestrial laser scanners, handheld laser scanners.*
- **STRUCTURED LIGHT SCANNERS: Project a pattern of light onto the object and analyze the deformation to calculate depth. Fast and precise.** *Common in industrial and medical applications.*
- **PHOTOGRAMMETRY: Reconstructs 3D models from 2D images taken at multiple angles.** *Cost-effective, good for detailed textures.*
- **TIME-OF-FLIGHT (TOF) SCANNERS: Measure the time it takes for light to bounce back from a surface.** *Used in mobile devices and robotics.*
- **CONTACT SCANNERS: Physically touch the object with a probe to record 3D data.** *Highly accurate, but slow and requires object stability.*
- **LASER TRIANGULATION SCANNERS: Combine laser projection and a camera to triangulate object surface points.** *Common in handheld and desktop scanners.*



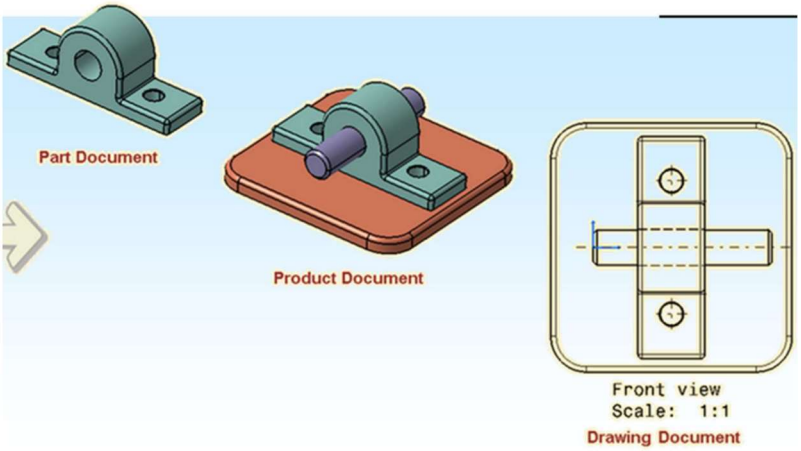
Photogrammetry


CATIA V5

CATIA V5 IS A CAD (COMPUTER-AIDED DESIGN) SOFTWARE DEVELOPED BY DASSAULT SYSTÈMES.

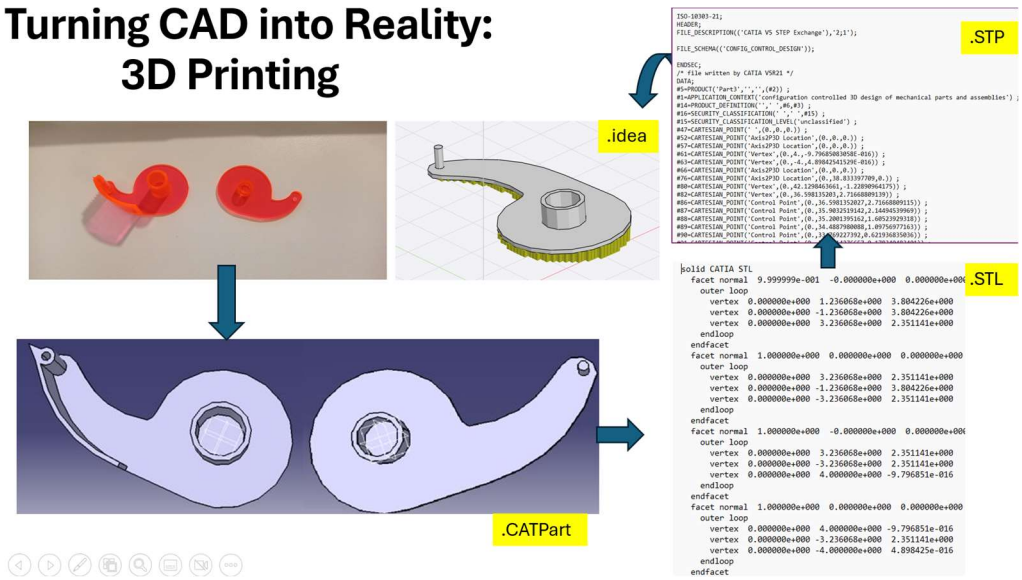
It is used for 3D modeling, simulation, and product lifecycle management, especially in industries like aerospace, automotive, and industrial design.



KEY TOOLS FOR REVERSE ENGINEERING

Software	Main Purpose
Rhino	NURBS-based 3D modeling for precise freeform surfaces. Ideal for design, prototyping, and industrial modeling.
CATIA	Advanced CAD/CAM/CAE software used in aerospace, automotive, and mechanical engineering for 3D design and product simulation.
Geomagic	Converts 3D scan data into usable CAD models. Specializes in mesh editing, surface reconstruction, and inspection.
ideaMaker	Slicing software for 3D printing. Prepares models for print, optimizes paths, and supports multiple printer settings.





Case studyIMPLEMENTATION OF A RIM

GOAL: PERFORM A REAL-WORLD TRIAL OF A RIM USING CATIA V5:

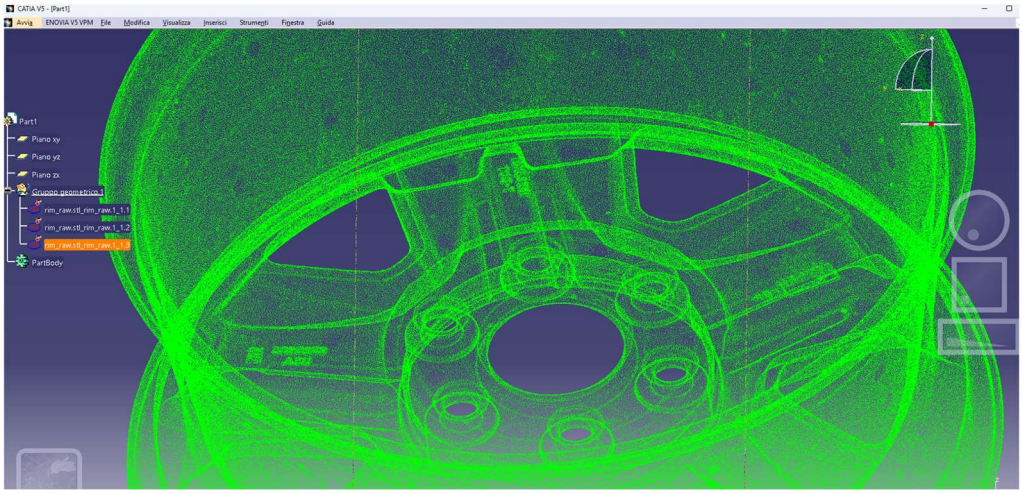
- **ANALYZE THE CLOUD POINTS AND DEFINE A MESH**
- **DETERMINE GEOMETRICAL FUTURES**

POSSIBLE CONNECTION WITH OPTIMIZATION TECHNICS:

- **OPTIMIZE AN OPTIMAL NUMBER OF POINTS CLOUDS OR OPTIMIZE THE NOZZLE'S PATH OR OPTIMIZE SUPPORTS FOR THE PART CONSIDERING EXECUTION TIME, ACCURACY OR ENERGY CONSUMPTION**



CLOUD POINTS: Higher scanning accuracy allows for a denser and more detailed point cloud, enabling the creation of a more precise and realistic CAD model.

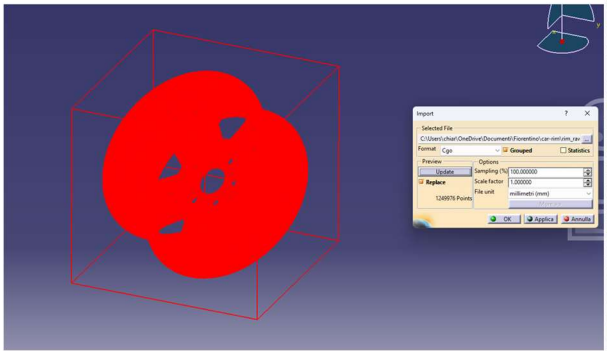


Rim.asci_cgo file

Structure of an ASCII CGO File

A CGO file consists of a sequence of floating-point numbers, each representing commands and parameters for drawing graphical objects.

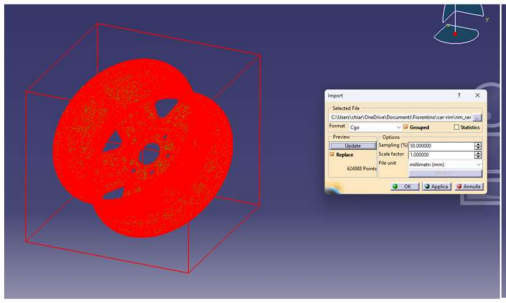
POINT NUMBERS: 1249976




```
-203.478622 -318.520660 477.730682
-204.745514 -318.545563 478.012207
-203.878418 -319.027588 478.929565
-202.078415 -319.020447 478.629547
-203.278412 -319.414001 479.762024
-201.778412 -319.161377 478.929565
-202.078415 -319.512238 479.829559
-201.511703 -318.688568 477.811646
-202.678421 -318.180939 476.829559
-204.776886 -317.913940 476.685699
-205.065369 -317.412109 475.730743
-205.378265 -317.804230 476.529419
-205.372589 -317.576904 476.083588
-205.361176 -317.126495 475.200104
-203.581772 -317.143890 474.928925
-201.716751 -317.903625 476.155457
-200.858246 -318.406647 477.101013
-200.278412 -318.724304 477.729553
-199.378418 -318.243896 476.529541
-199.378418 -318.655334 477.429565
-199.378418 -319.047729 478.329559
-200.278412 -319.248962 478.929565
-201.178421 -319.603668 479.829559
-199.978424 -319.545715 479.529541
-197.878418 -319.113831 478.329559
-198.178421 -318.814026 477.664154
-198.178421 -318.560272 477.129547
-198.478424 -318.153778 476.229553
-199.377991 -317.921326 475.822571
-197.366089 -317.914001 475.629547
-196.678421 -318.366974 476.529541
-196.678421 -318.814026 477.552094
-196.978424 -319.260345 478.629547
-197.578415 -319.388153 478.929565
-198.178421 -319.627625 479.529541
-196.678421 -319.702454 479.529541
-196.078415 -319.462677 478.929565
-195.778412 -319.714020 479.392944
-196.078415 -319.085297 478.029541
-195.778412 -318.663055 477.129547
-195.085327 -318.814026 477.429565
-194.975098 -319.114014 478.029541
-194.278412 -319.414001 478.602600
-194.283035 -319.714020 479.229553
-194.878418 -320.053528 480.129547
-195.836716 -320.014008 480.129547
-196.678421 -320.061371 480.429565
-197.578415 -320.046783 480.429565
-198.740189 -320.014008 480.429565
```

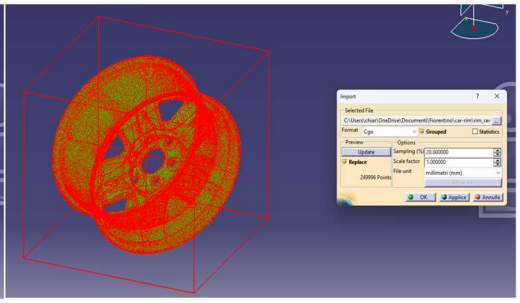
Rim.asci_cgo file

Sampling 50%



POINT
NUMBERS:
624988

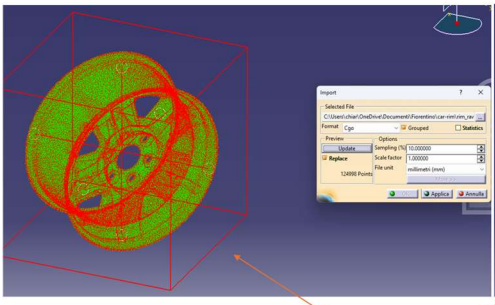
Sampling 20%



POINT
NUMBERS:
249996

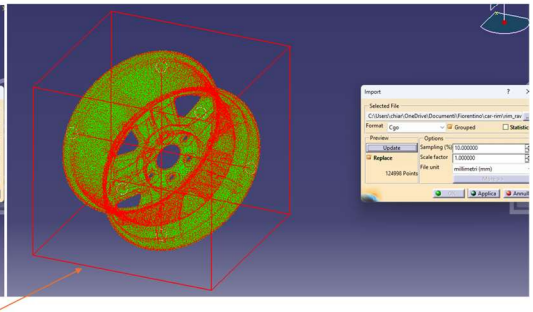
Rim.asci_cgo file

Sampling 10%



POINT
NUMBERS:
124998

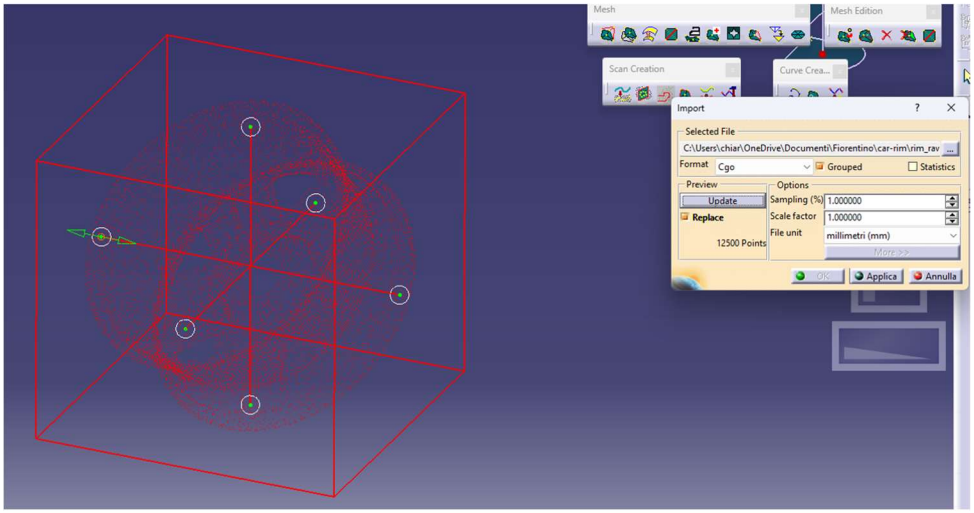
Sampling 5%



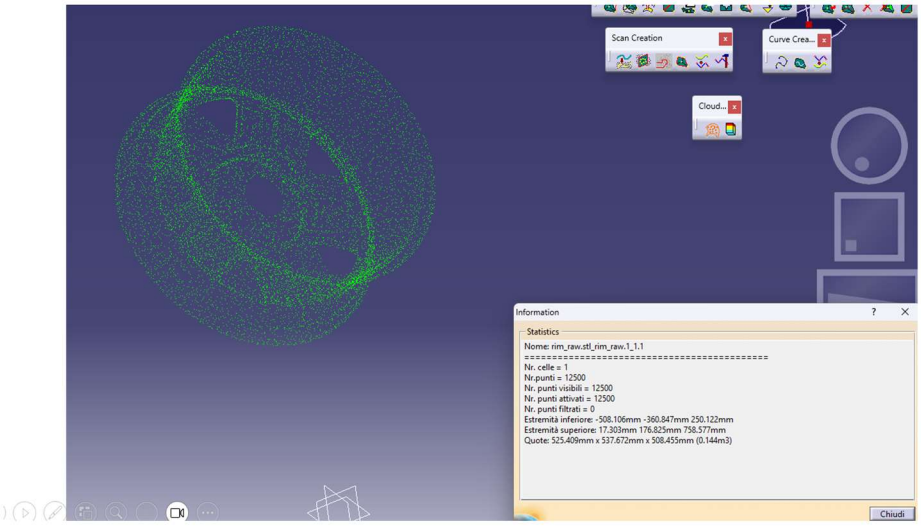
POINT
NUMBERS:
62499

CAPTURING VOLUME

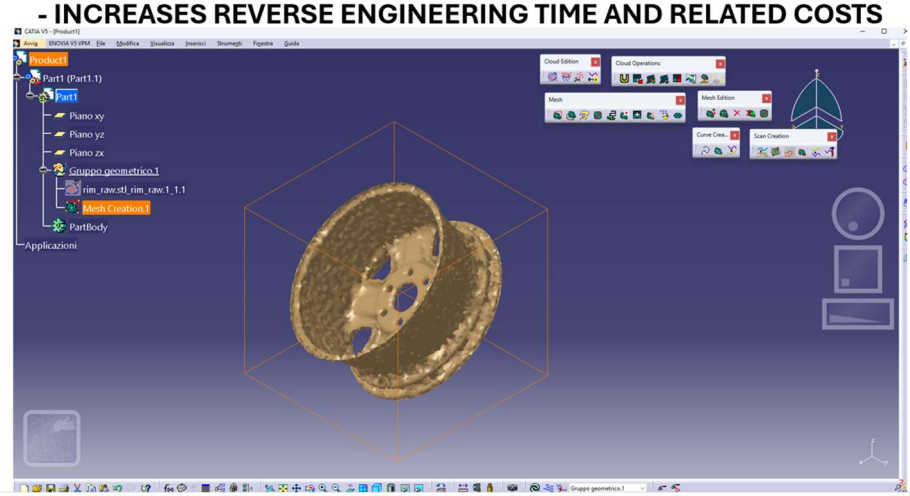
CLOUD POINTS SAMPLING 1%- POINTS NUMBER: 12500



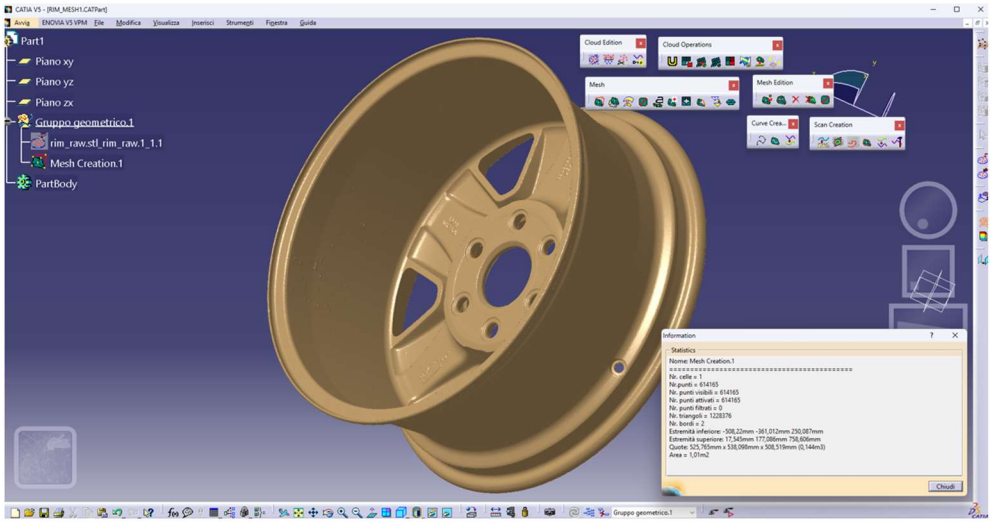
CLOUD POINTS SAMPLING 1%- POINTS NUMBER: 12500



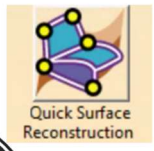
MESHING WITH A LOW NUMBER OF POINTS :
- REDUCES SCANNING TIME AND RELATED COSTS
- INCREASES REVERSE ENGINEERING TIME AND RELATED COSTS



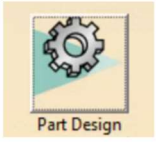
GOOD MESHING FOR HIGH PRECISION APPLICATIONS



THE MAIN REVERGE ENGINEERING TOOLS:



- **Data Acquisition:** Gather point cloud data
- **Preprocessing:** Clean the point cloud .
- **Surface Reconstruction** with **Delaunay Triangulation**(Meshing by connecting points), **Poisson Surface Reconstruction** (Converting noisy point clouds into smooth surfaces), **Alpha Shapes** (Forming surfaces from point neighborhoods), **Ball Pivoting** (Meshing by rolling a virtual ball over the points).
- **Marching Cubes/Octree:** For volumetric data.
- **Post-Processing:** Smooth meshes, fill holes.

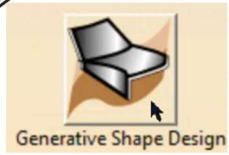


- **Creation of Solid Geometries**
- **Solid Modeling Operations** (extrusion, drilling, milling, chamfering, and filleting)
- **Parametric Design**
- **Assembly and Positioning Functions:** Parts created in Part Design can be assembled using constraints between parts.
- **Analysis Tools:** ensure parts are ready for production.

THE MAIN REVERGE ENGINEERING TOOLS:

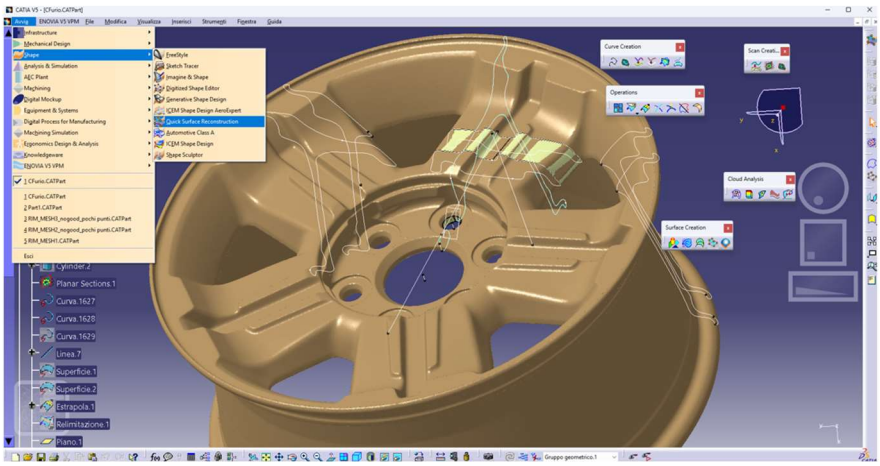


- **Importing & Cleaning Point Clouds :** Removes noise and unwanted points.
- **Mesh Creation & Editing:** Converts point clouds into editable polygonal meshes.
- **Surface Optimization & Repair :** Fixes holes and distortions.
- **CAD Surface Conversion**

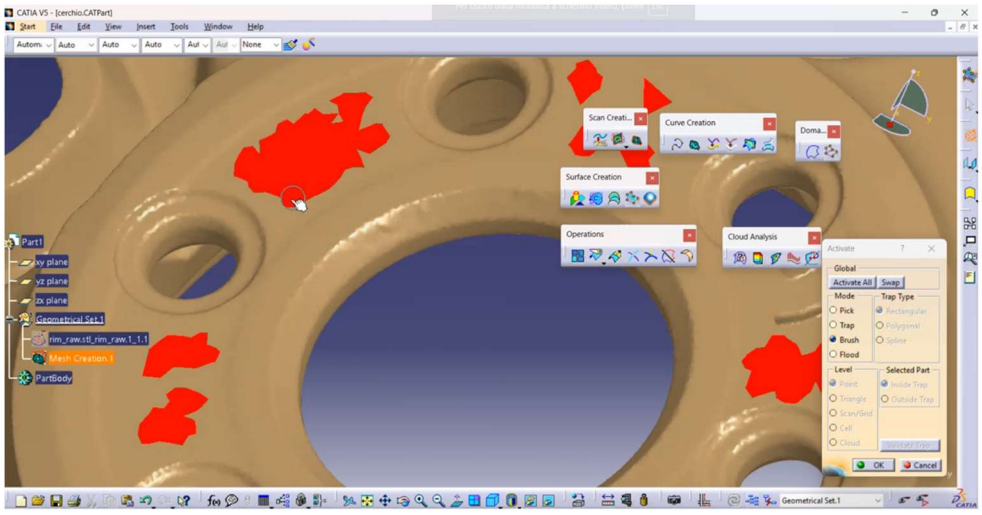


- **Creation of Complex Surfaces:** modeling advanced 3D surfaces
- **Parametric Design:** Surfaces and shapes are controlled by parameters that can be easily modified
- **Surface Continuity Control:** Ensures smooth, well-connected surfaces
- **Integration with Solid Modeling:** Supports the transition between complex surfaces and solid geometries
- **Surface Validation:** Allows for the quality checking of surfaces to ensure they are defect-free and ready for production.

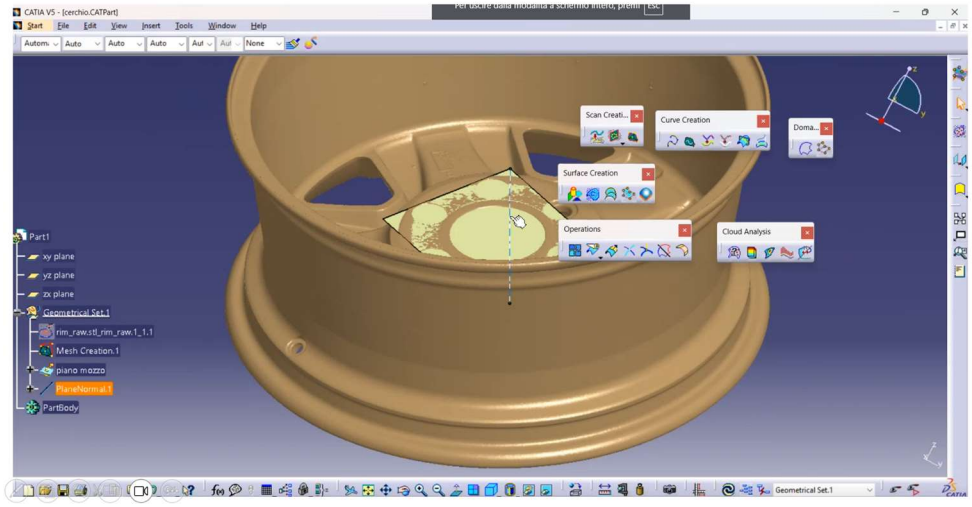
THE MAIN REVERGE ENGINEERING TOOLS:
QUICK SURFACES RECOSTRUCTION



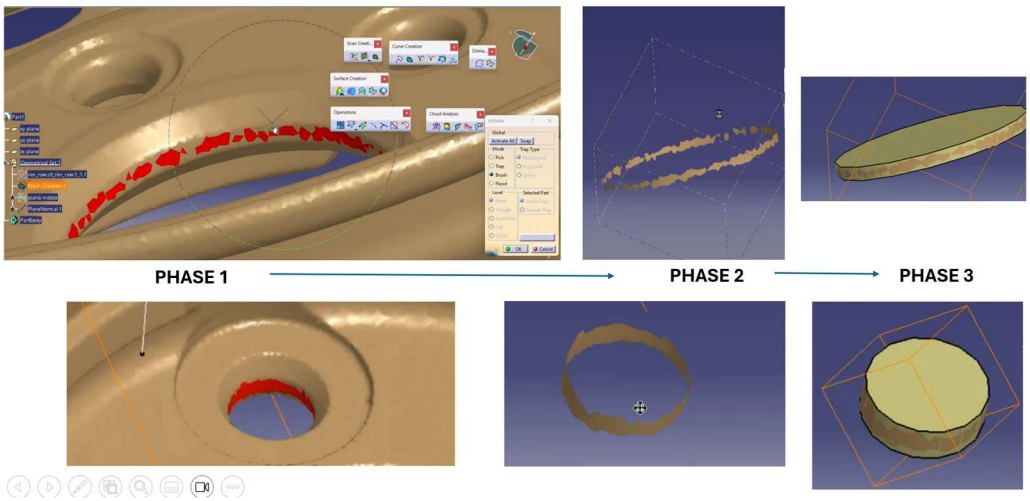
STEP 1: "BRUSH" COMMAND



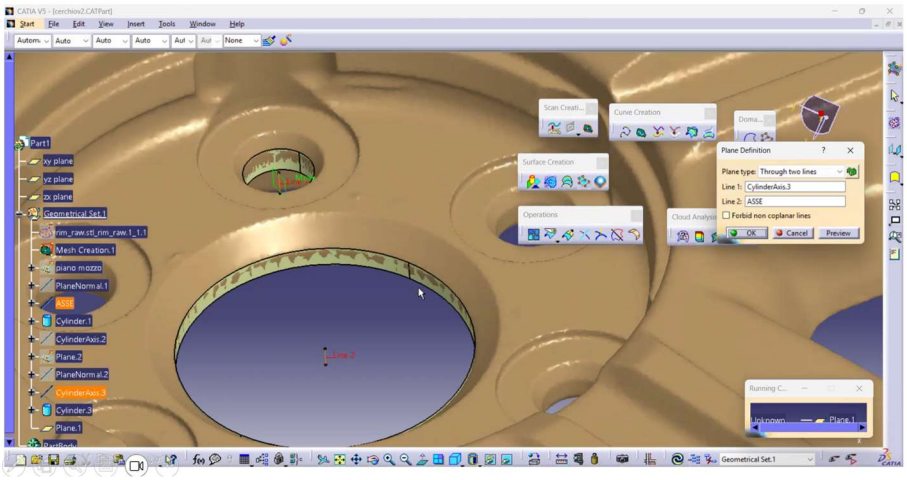
It permits creation of reference surfaces and related axis



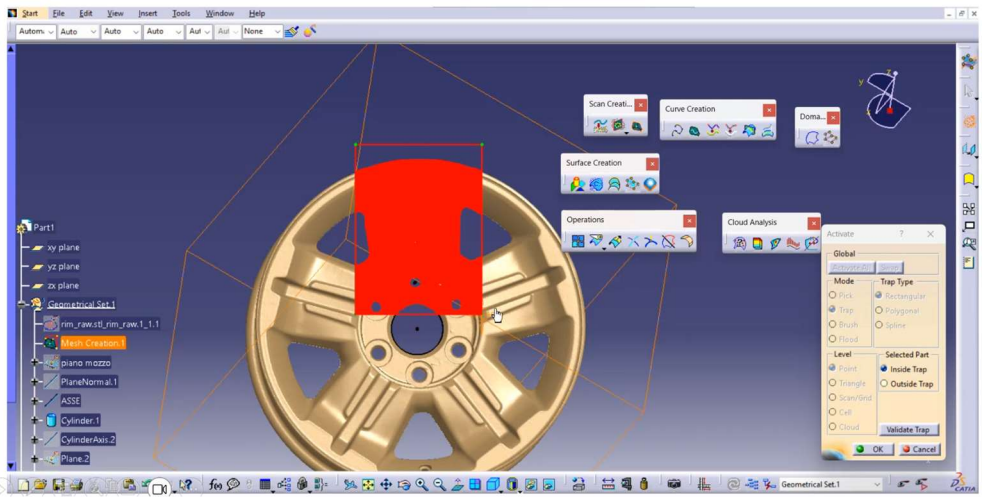
BRUSHING ON THE MESH, SURFACES ARE RECOGNITED

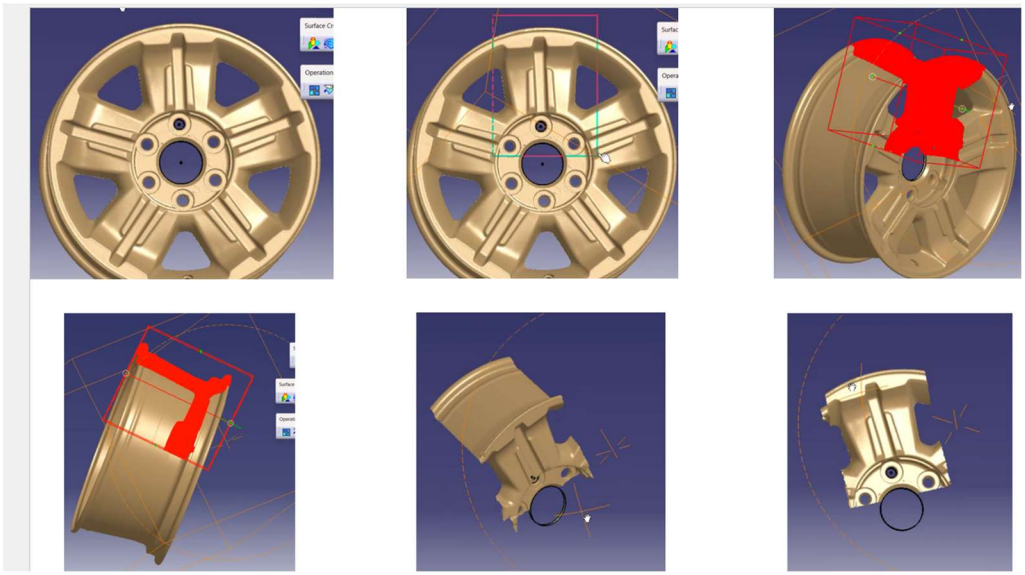


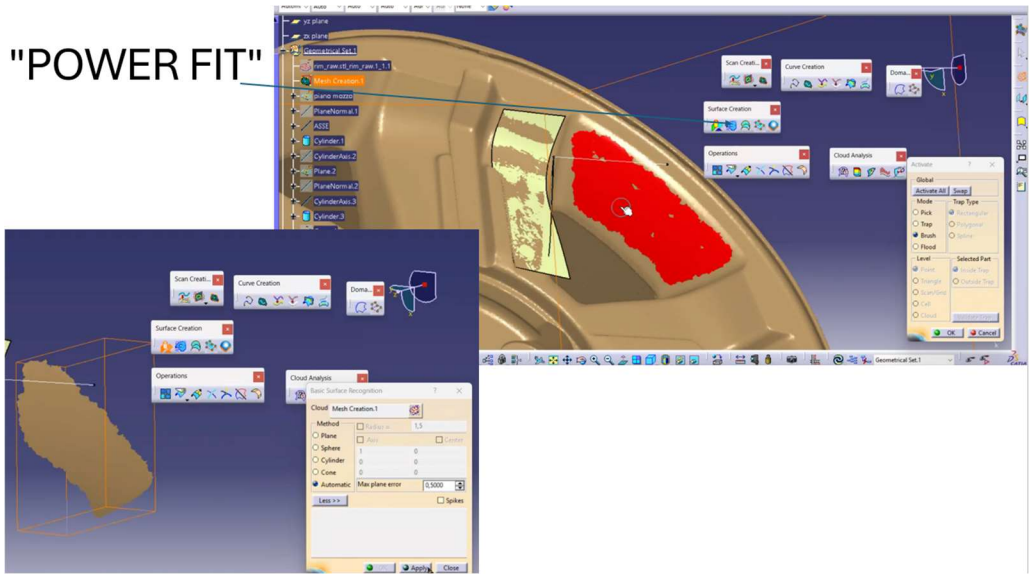
STEP 3: Construction of a meaningful section plane that allows obtaining a relevant section of the object



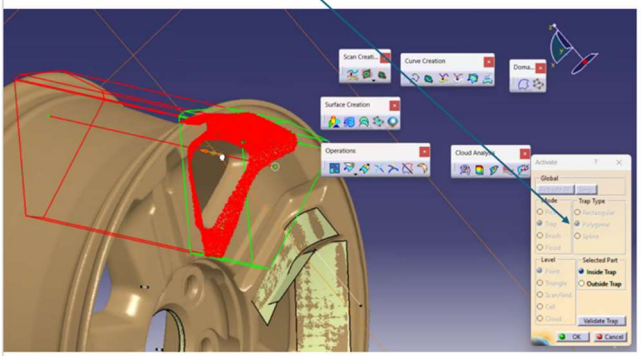
STEP 4: Capturing a part of the object with command "TRAP"

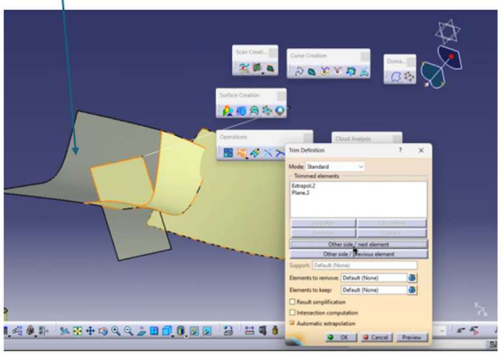




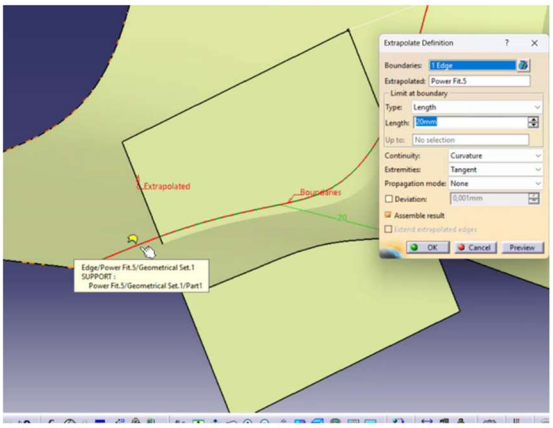

"POWER FIT"

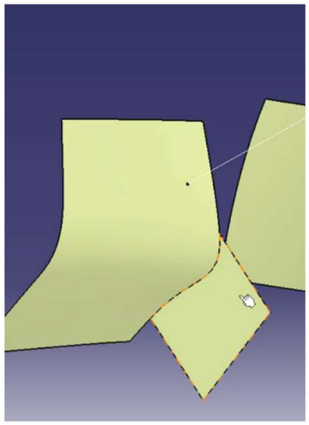
Polygonal trap to generate curved surfaces



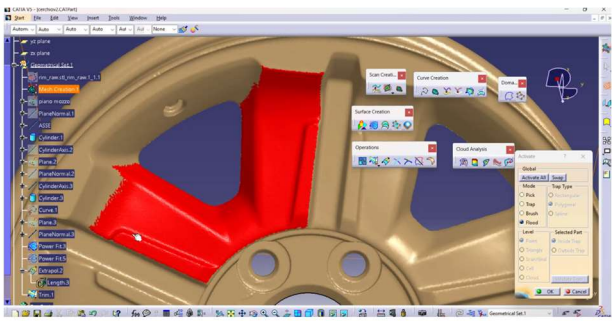


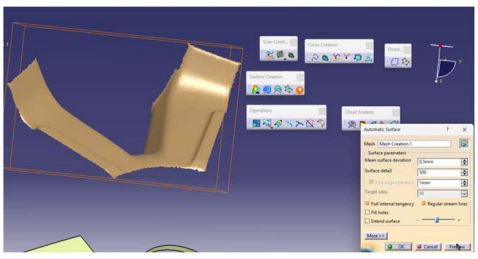
Trim function

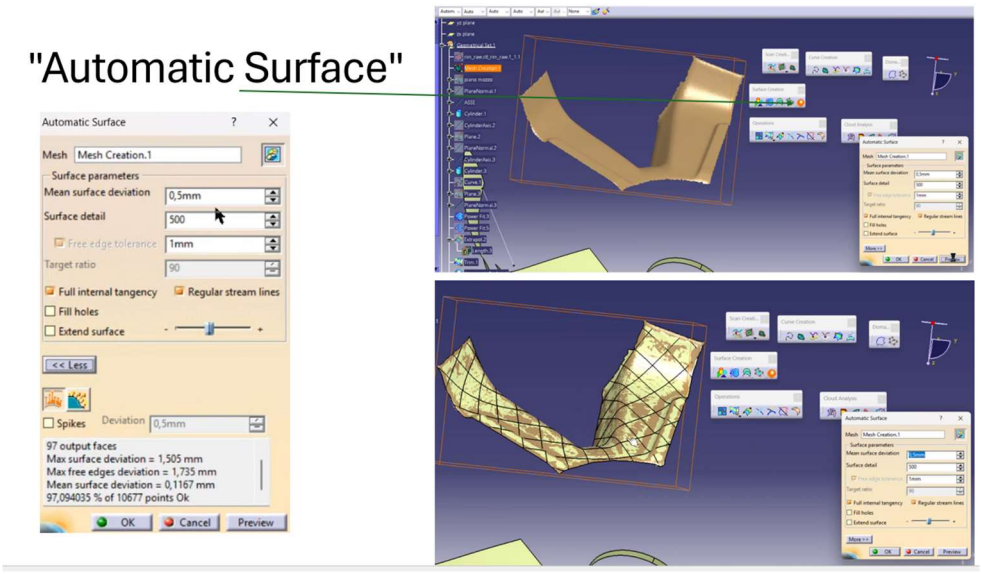


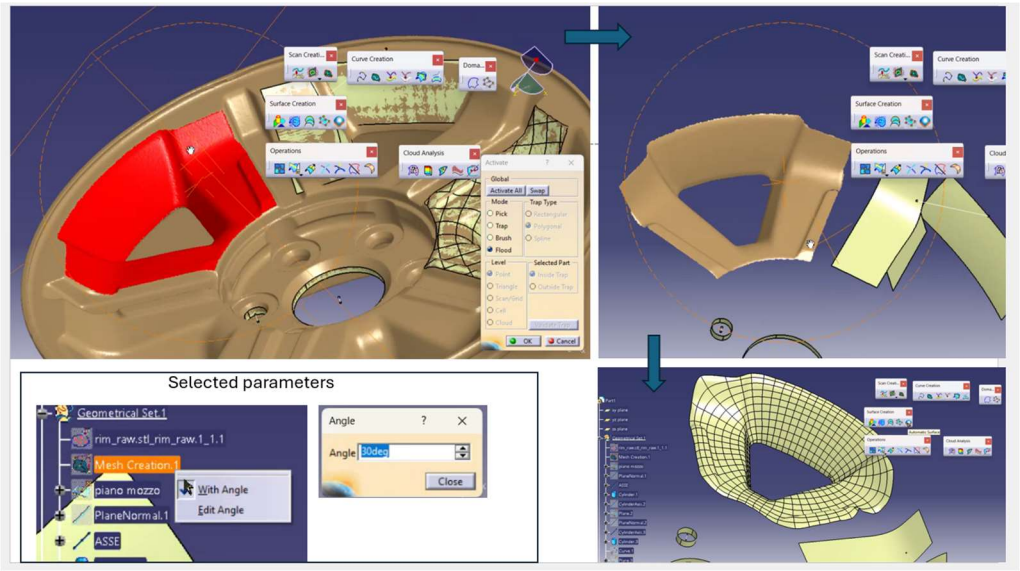


Flood Command

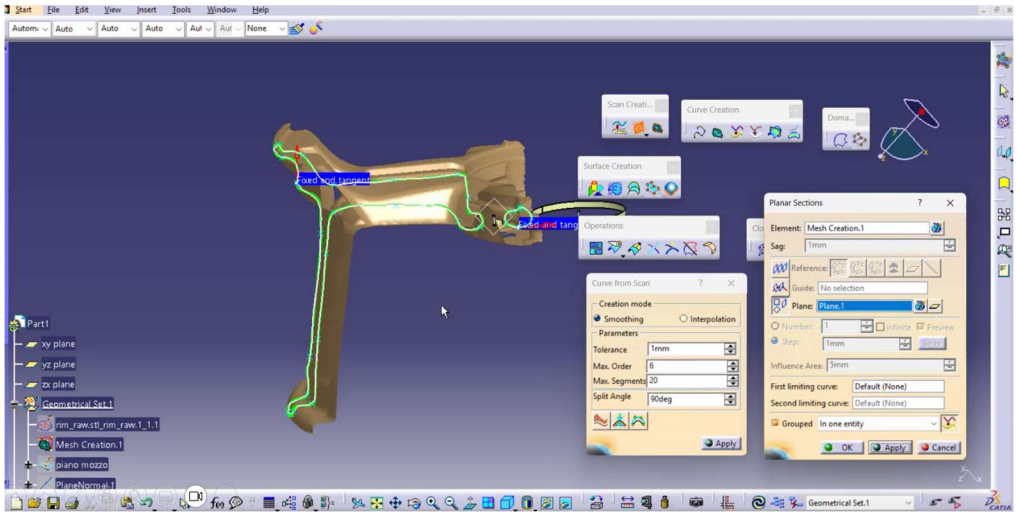




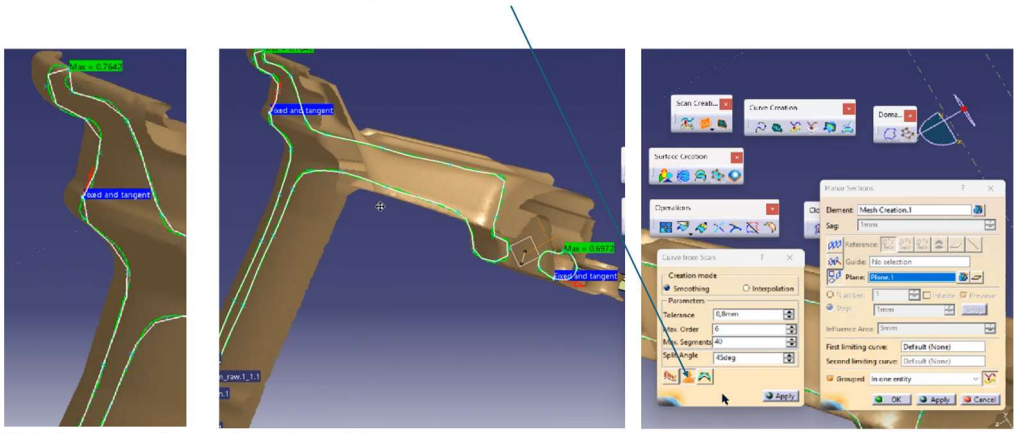




STEP 5: Curve Creation

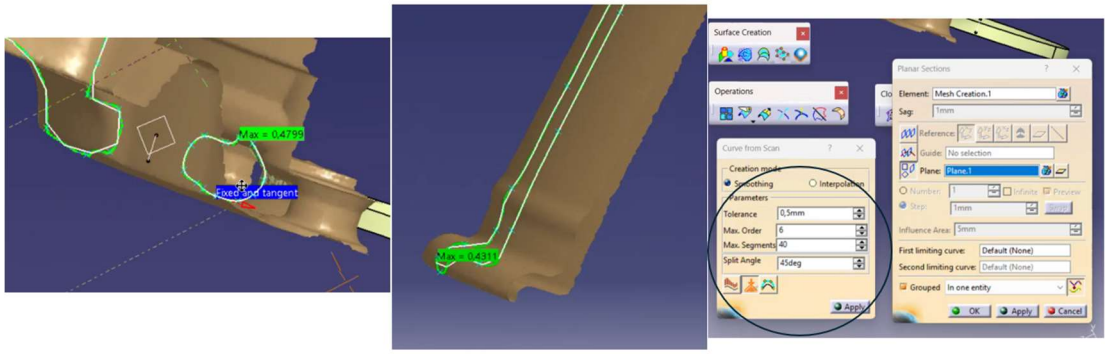


Using the "Curve Deviation" command, you can see how much the created curve deviates from the original one.

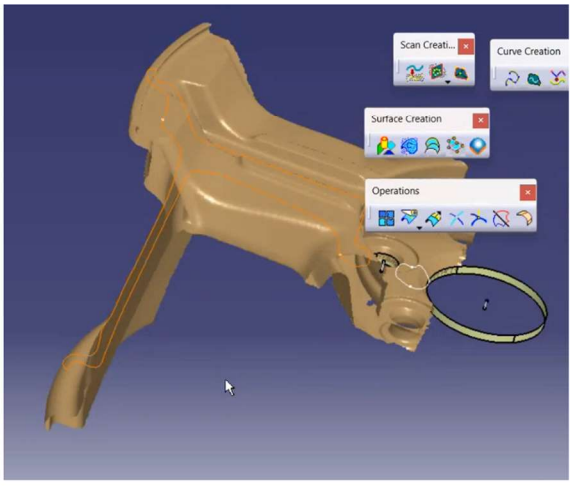


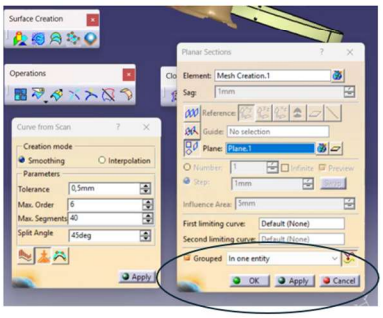
White line= created curve; Green Line= original curve

By increasing the number of segments and reducing the tolerance value, it's possible to reduce the maximum deviation.

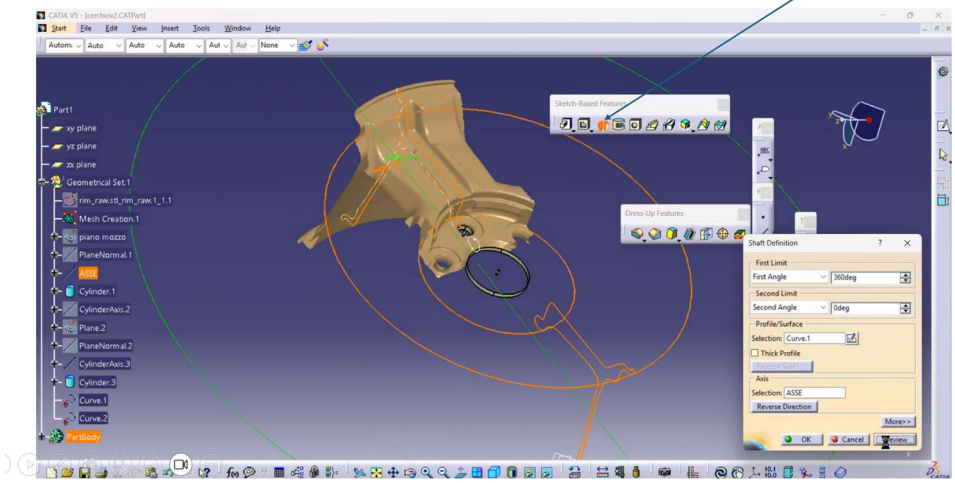


FINAL CURVE

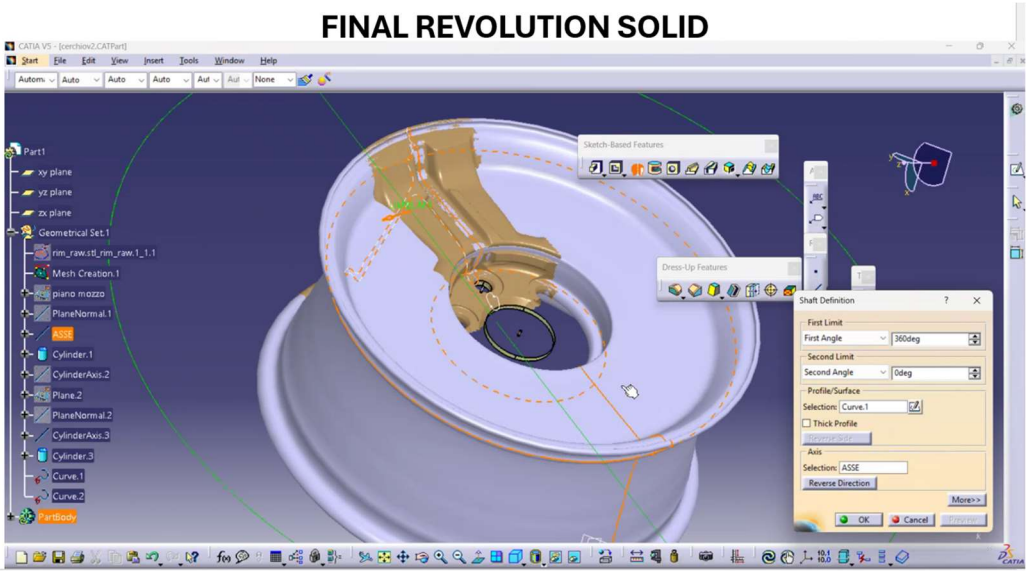




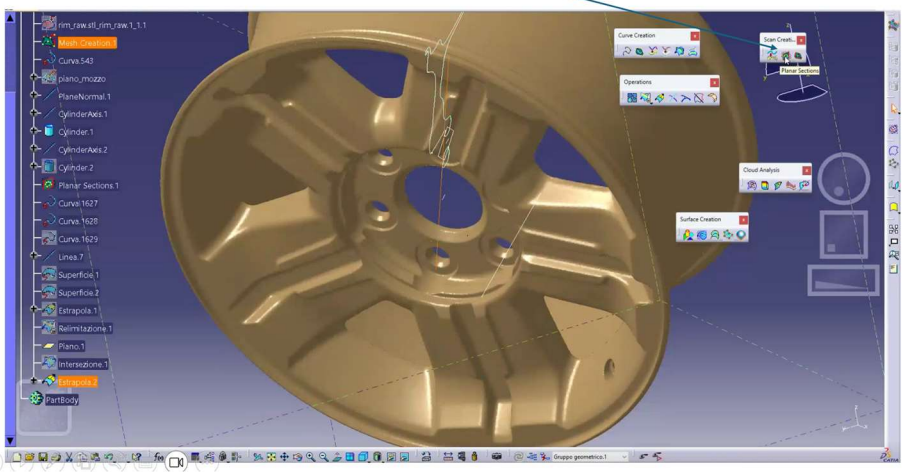
STEP 6: Starting from the curve, create a solid using the Revolve function in the Mechanical Design Part workbench.

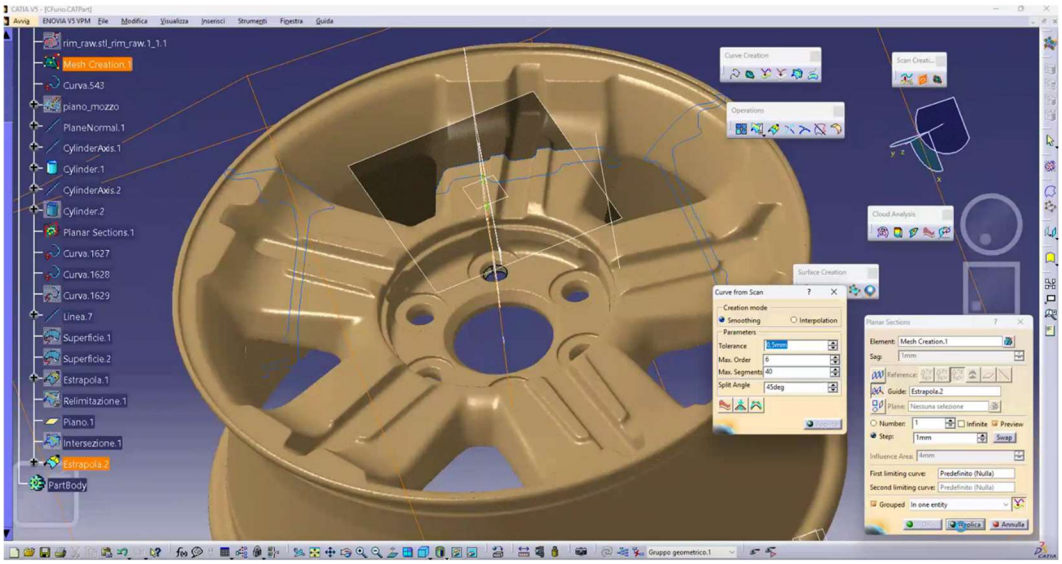


FINAL REVOLUTION SOLID

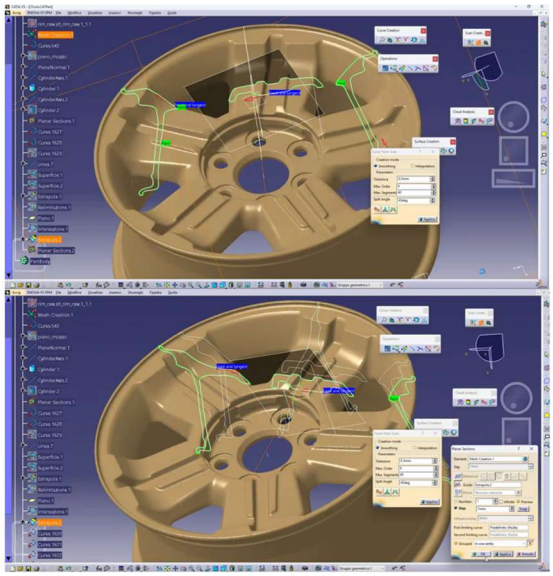


STEP 7: Planar Section

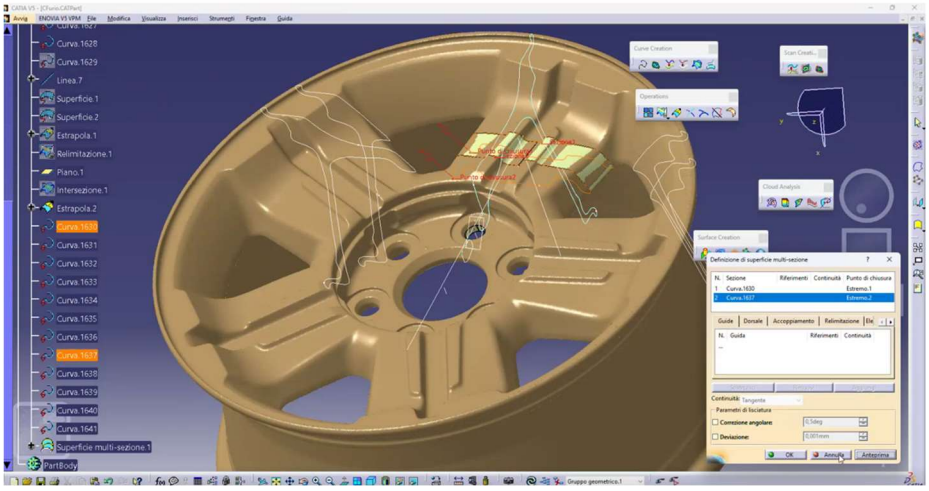




FINAL SECTIONS



MULTI-SECTION



FINAL FILE "rim.Stl"

Geometry Information:

- The object's surface is divided into triangles (triangular mesh).
- Each triangle is described by three vertices and a surface normal.

Binary or ASCII Format:

- **ASCII STL:** A human-readable file with a text-based structure that explicitly lists all triangles and their coordinates.
- **Binary STL:** More compact and faster to process, containing the same information in a non-human-readable format.



FILES ".Stl"

STL FILES DEFINE AN OBJECT'S SURFACE GEOMETRY BUT LACK INFORMATION ON:

- COLOR
- MATERIAL
- PRINT SETTINGS
- INTERNAL STRUCTURE.

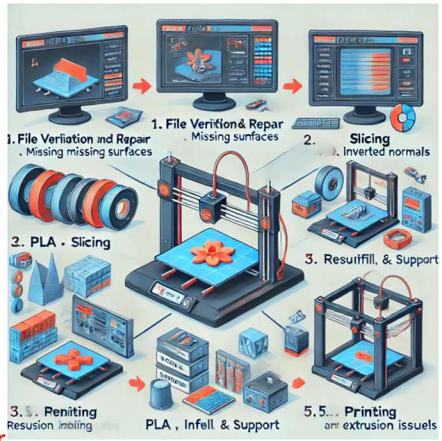


ADDITIONAL CONFIGURATIONS ARE NEEDED FOR OPTIMAL RESULTS.

FINAL FILE "rim.Stl"

TO ACHIEVE A HIGH-QUALITY 3D PRINT FROM AN STL FILE:

- **File verification and repair** : Use software like Rhino, Geomagic Wrap to fix errors such as missing surfaces or inverted normals.
- **Slicing** – Convert the STL into G-code using slicing software (e.g. PrusaSlicer, or Simplify3D) while setting resolution, infill, supports, and print speed.
- **Material selection** – Choose the appropriate filament (PLA, ABS, PETG, resin, etc.) based on the desired properties.
- **Printer setup** – Ensure proper calibration, including bed leveling and filament loading.
- **Printing and monitoring** – Start the print and supervise the initial stages to prevent adhesion or extrusion issues.



APPENDIX 8- CATALOGUE OF STUDIED OPTIMIZATION CASES

8.1 Applied Mathematics:

8.1.1 Rosenbrock

The Rosenbrock Function [354], [355] is a benchmark for evaluating global optimization algorithms. It is commonly referred to as the "banana function" or "the valley" and is a non-convex, non-linear unimodal function. While its valley is straightforward to identify, achieving convergence to the global minimum is notably challenging.

The original formula is:

$$f(\overline{x}) = \frac{1}{3.755 \cdot 10^5}\left(\sum_{i=1}^{3}\left(100 \cdot \left(\overline{x}_{i+1} - \overline{x}_i^2\right)^2 + ((1 - \overline{x}_i)^2) - 3.827 \cdot 10^5\right)\right)$$

$$with\ \overline{x_i} = 15 \cdot x_i - 5,\ i = 1,2,3,4$$

The global optimum is $f(\overline{x}) = \overline{0}$, obtained for $\overline{x} = \overline{1}$.

Figure A8.1 represent the classical surface plot of this function.

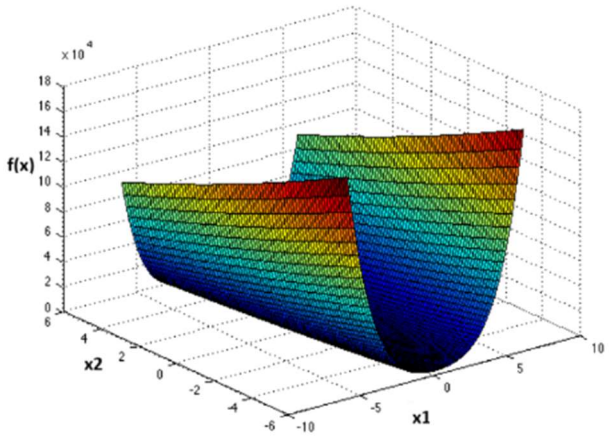


Figure A8.1- Rosenbrock function plot in two-dimensional form

8.1.2 Rastrigin

The Rastrigin Function [356], [357], [358] is a non-convex, non-linear multimodal function [356],[357], [358]. It is characterized by a large search space and a high number of regularly

distributed local minimaConsequently, it is frequently employed as a benchmark to evaluate the performance of global optimization algorithms.
The original formula is:

$$f(\overline{x}) = 10 \cdot d + \sum_{i=1}^{d} (x_i^2 - 10 \cdot \cos(2 \cdot \pi \cdot x_i))$$

where d is the dimension.

The boundaries conditions are:

$$x_i \in [-5.12,\ 5.12]\ with\ i = 1,2, \dots, d$$

The global optimum is $f(\overline{x}) = \overline{0}$, obtained for $\overline{x} = \overline{0}$. Figure A8.2 represent the classical surface plot of this function. Figure 2 represent the classical surface plot of this function.

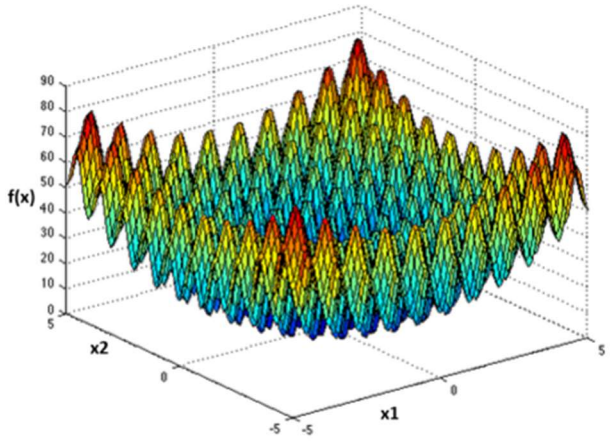


Figure A8.2- Rastrigin function plot in two-dimensional form

8.1.3 Himmelblau's Function

The Himmelblau Function [359] is a non-convex, nonlinear, multimodal function characterized by multiple local minima and several global minima.

The original formula is the following:

$$f(\overline{x}) = (x_1 + x_2^2 - 7)^2 + (x_1^2 + x_2^2 - 11)^2$$

subject to the following bounds:
$-6 \leq x_i \leq 6,\ i = 1,2$

The function value at each minimum is:

$$f_{min}(\overline{x}) = 0$$

The coordinates of these global minima are:

$\overline{x}_1 = [3\ 2]$
$\overline{x}_2 = [3.584428340330\ -1.848126526964]$
$\overline{x}_3 = [-3.779310253378\ -3.283185991286]$
$\overline{x}_4 = [-2.805118086953\ 3.131312518250]$

Figure A8.3 represent the classical surface plot of this function.

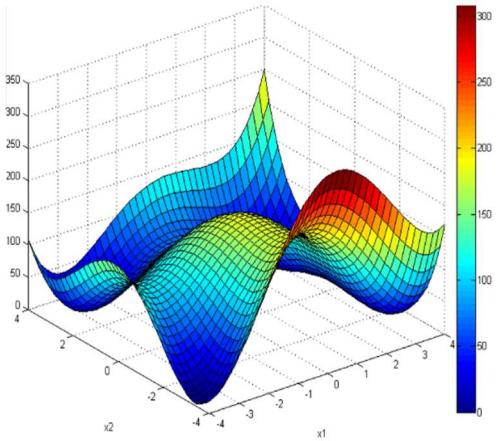


Figure A8.3- Himmelblau function plot in different two-dimensional forms

8.2 Mechanics:

8.2.1 Tension/compression spring design (case 1: 3 variables and 3 constraints)

The optimization of a helical spring [191] (see Figure A8.4) pertains to the minimization of its total weight through the manipulation of three design variables (X_1= d= wire's diameter, X_2= D= coil's diameter, X_3= N= number of coils) while adhering to predefined constraints.

While the number of constraints may be limited, the intricacy of this case study lies in the nuanced challenge of error propagation during approximation. The inclusion of exponents in the formulas renders the truncation of significant digits throughout the computational process, thereby introducing potential inaccuracies.

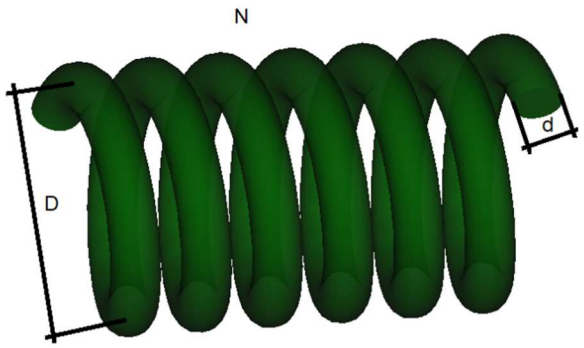


Figure A8.4- Representation of a tension/compression spring for optimization design

The optimization process is guided by constraints on geometrical parameters, deflection, surge frequency, and shear stress, as detailed in ensuing section outlining the mathematical framework for the problem.

The boundary conditions are given by

$0.05 < X_1 < 2$

$0.25 < X_2 < 1.3$

$2 < X_3 < 15$

The following parameters are defined as follows:

$g = 386.0\ ips^2$	Acceleration of Gravity
$\gamma = 0{,}285\ \frac{lb}{in^3}$	Specific weight
$G = 11{,}5 \cdot 10^6\ psi$	Shearing Modulus
$\rho = \frac{\gamma}{g}$	Mass Density
$\tau_{max} = 80000\ psi$	Allowable Shear Stress
$Q = 2$	Quantity of Inactive Coils
$P = 10.0\ lb$	Concentrated Load
$\Delta = 0.5\ \ in$	Minimum deflection
$\varpi_0 = 100.0\ Hz$	Lower limit for surge wave frequency
$D_0 = 1.5\ in$	Limit on outer diameter of Coil

$$K = \frac{d^4 \cdot G}{8 \cdot D^3 \cdot N} = \frac{x_1^4 \cdot G}{8 \cdot x_2^3 \cdot x_3}$$ Spring Stiffness

$$\tau = \frac{8 \cdot k \cdot P \cdot D}{\pi \cdot d^3} = \frac{8 \cdot k \cdot P \cdot x_2}{\pi \cdot x_1^3}$$ Shear Stress

$$k = \frac{4 \cdot D - d}{4 \cdot (D - d)} + \frac{0.615 \cdot d}{D} = \frac{4 \cdot x_2 - x_1}{4 \cdot (x_2 - x_1)} + \frac{0.615 \cdot x_1}{x_2}$$ Wahl's factor

$$\varpi = \frac{d}{2 \cdot P \cdot N \cdot D^2} \cdot \sqrt{\frac{G}{2 \cdot \rho}} = \frac{x_1}{2 \cdot P \cdot x_3 \cdot x_2^2} \cdot \sqrt{\frac{G}{2 \cdot \rho}}$$ Surge Wave Frequency

Note that the Wahl correction factor, being dimensionless, streamlines the computation of spring compression by considering the amplification of external stress at the curvature of the spring coil. Notably, the inner surface experiences greater stress compared to the outer surface. This factor estimates the reduction in fatigue strength while recognizing the impact of a notch in the material.

The objective function is defined as follows:

$$F_{obj}(\bar{X}) = \frac{1}{4} \cdot (N + Q) \cdot \pi^2 \cdot D \cdot d^2 \cdot \rho = \frac{1}{4} \cdot (X_3 + Q) \cdot \pi^2 \cdot X_2 \cdot X_1^2 \cdot \rho$$

The optimization problem therefore consists in minimizing the spring's mass.

To ensure the feasibility and reliability of the design, the following engineering constraints are imposed: The spring's deflection should exceed the Minimum deflection:

$G(1) = \Delta - \Delta_{cal} \leq 0$

where $\Delta_{cal} = \frac{P}{K}$

The sum of the wire and coil diameters cannot exceed the Imposed Limit on outer diameter of Coil:

$$G(2) = d + D = x_1 + x_2 \leq D_0$$

The frequency should be greater than the Lower limit for surge wave frequency, to prevent resonance phenomena:

$$G(3) = \varpi_0 - \varpi \leq 0$$

The shear stress cannot exceed the Allowable Shear Stress:

$$G(4) = \tau - \tau_{max} \leq 0$$

8.2.2 Car's Crashworthiness

In the automotive field, the design and evaluation of fuselage crashworthiness for occupant survivability constitute a major optimization challenge. Vehicle pillars are alphabetically categorized according to their position and structural function. The A-pillar, located at the front, supports the roof near the windshield. Moving rearward, the B-pillar connects the front and rear side windows, maintaining overall body rigidity. In sedans and hatchbacks, the C-pillar is positioned at the rear, while station wagons and SUVs also include a D-pillar, which provides structural support at the vehicle's far rear.

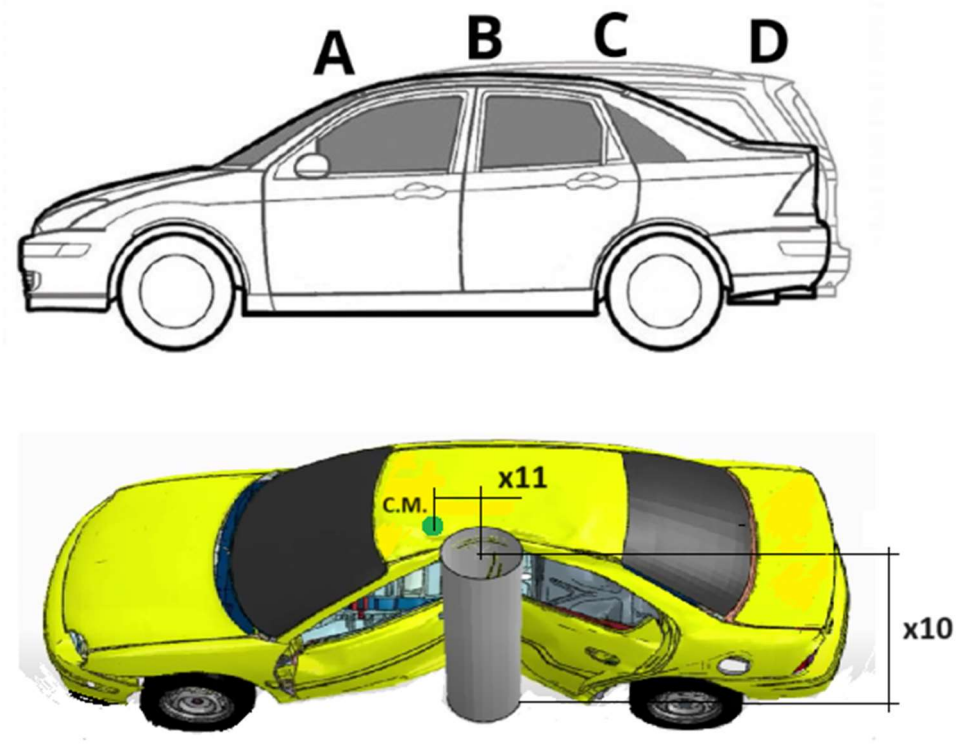


Figure A8.2.3.1a- Designation of vehicle pillars and definition of parameters X_{10} and X_{11} correlated with the barrier in the lateral crashworthiness test.

During side-impact testing, the B-pillar region is typically the most affected; hence, several design variables are associated with this area.

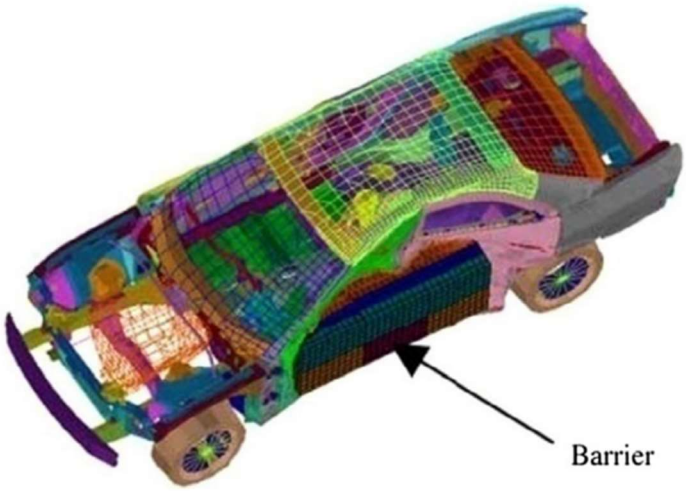


Figure A8.2.3.1.b- Side-impact crashworthiness design.

Crashworthiness analysis in side impacts is directly linked to the risk of injury to occupants located near the collision point. The primary objective of crashworthiness optimization is to ensure that the vehicle absorbs and dissipates impact energy effectively, thereby minimizing the forces transmitted to the occupants. The goal is to develop a structure that provides maximum protection without significantly increasing weight or compromising other performance characteristics. Key components involved in this optimization include side beams, door reinforcements, and B-pillars, all designed to effectively resist and manage crash loads.

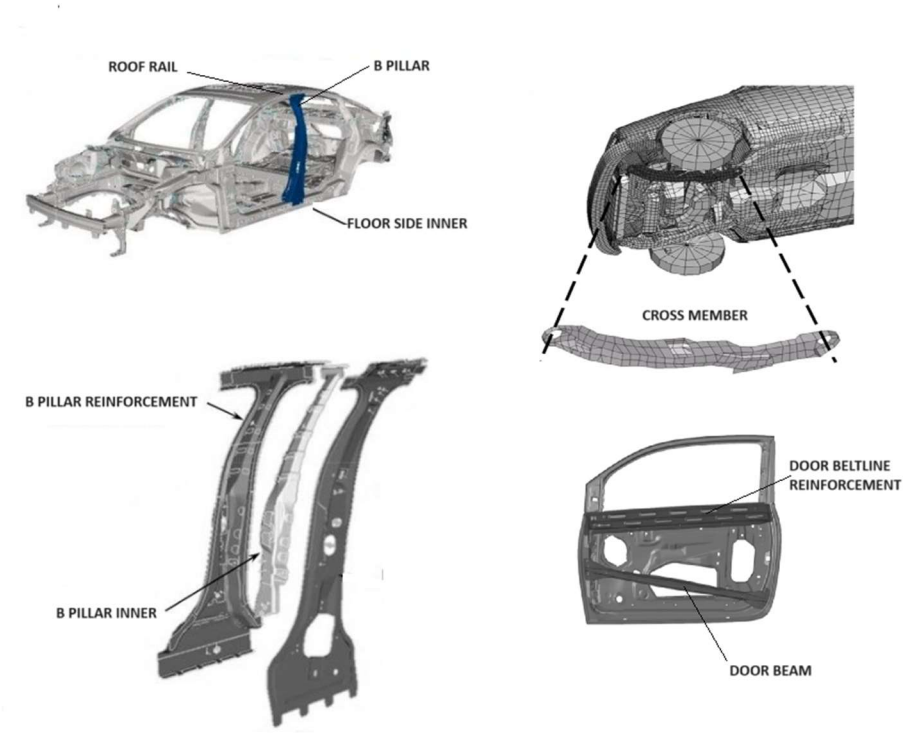


Figure 5. Schematic representation of vehicle components used in crashworthiness testing.

The optimization problem involves 11 design parameters and 10 constraints. Seven parameters describe geometric features: X_1 = B-pillar inner thickness, X_2 = B-pillar reinforcement thickness, X_3 = floor side inner thickness, X_4 = cross-member thickness, X_5 = door beam thickness, X_6 = door beltline reinforcement, X_7 = roof rail thickness. Two parameters refer to material selection: X_8 = shape factor of the B-pillar inner, X_9 = shape factor of the floor side inner. Finally, two parameters define barrier characteristics used in the simulation: X_{10} = barrier height, X_{11} = impact position relative to the vehicle centre.

The shape factor is a dimensionless parameter that quantifies the efficiency of a material or structural element in absorbing energy during an impact. It depends on the materials used for the B-pillar inner and floor side inner. Optimization of this factor enables a balance between energy absorption capacity and weight reduction.

The energy exchanged during an impact, determined by amplitude, duration, pulse shape, and acceleration direction, can be analysed to estimate human tolerance levels and predict the probability of compression fractures in the lower spine due to vertical accelerations (from pelvis to head). Occupant survivability requires that kinetic energy from the crash be absorbed through controlled structural deformation, ensuring that loads and accelerations remain within human tolerance limits.

The boundary conditions imposed on the design variables are:

$0.5 < X_i < 1.5 \quad \text{for } i = 1,2,\dots,7$

$0.192 < X_9 < 0.345$

$-30 < X_{10} < 30$

$$-30 < X_{11} < 30$$

The objective function to be minimized is:

$$F_{\text{obj}}(\bar{X}) = 1.98 + 4.90 \cdot X_1 + 6.67 \cdot X_2 + 6.98 \cdot X_3 + 4.01 \cdot X_4 + 1.78 \cdot X_5 + 2.73 \cdot X_7$$

The optimization is constrained by the impact tolerance of various human body regions (head, neck, thorax, spine, abdomen, extremities) and corresponding injury criteria, expressed as:

1) Load of abdomen $F_A\ [kN]$:

$$G_1(\bar{X}) = F_A = 1.16 - 0.3717 \cdot X_2 \cdot X_4 - 0.00931 \cdot X_2 \cdot X_{10} - 0.484 \cdot X_3.X_9 + 0.01343 \cdot \cdot X_6 \cdot X_{10} \leq 1\ kN$$

2) Dummy upper chest $VC_U\ \left[\frac{m}{s}\right]$:

$$G_2(\bar{X}) = VC_U = 0.261 - 0.0159 \cdot X_1 \cdot X_2 - 0.188 \cdot X_1 \cdot X_8 - 0.019 \cdot X_2 \cdot X_7 + 0.0144 \cdot X_3 \cdot X_5 + 0.0008757 \cdot X_5 \cdot X_{10} + 0.080405 \cdot X_6 \cdot X_9 + 0.00139 \cdot X_8 \cdot X_{11} + +0.00001575 \cdot X_{10} \cdot X_{11} \leq 0.32\ \frac{m}{s}$$

3) Dummy middle chest $VC_M\ \left[\frac{m}{s}\right]$:

$$G_3(\bar{X}) = VC_M = 0.214 + 0.00817 \cdot X_5 - 0.131 \cdot X_1 \cdot X_8 - 0.0704 \cdot X_1 \cdot X_9 + +0.03099 \cdot X_2 \cdot X_6 - 0.018 \cdot X_2 \cdot X_7 + 0.0208 \cdot X_3 \cdot X_8 + 0.121 \cdot X_3 \cdot X_9 - -0.00364 \cdot X_5 \cdot X_6 + 0.0007715 \cdot X_5 \cdot X_{10} - 0.0005354 \cdot X_6 \cdot X_{10} + +0.00121 \cdot X_8 \cdot X_{11} \leq 0.32\ \frac{m}{s}$$

4) Dummy lower chest $VC_L\ \left[\frac{m}{s}\right]$:

$$G_4(\bar{X}) = VC_L = 0.074 - 0.061 \cdot X_2 - 0.163 \cdot X_3 \cdot X_8 + 0.001232 \cdot X_3 \cdot X_{10} - -0.166 \cdot X_7 \cdot X_9 + 0.227 \cdot X_2^2 \leq 0.32\ \frac{m}{s}$$

5) Dummy rib deflection $\Delta_{UR}\ [mm]$

$$G_5(\bar{X}) = \Delta_{UR} = 28.98 + 3.818 \cdot X_3 - 4.2 \cdot X_1 \cdot X_2 + 0.0207 \cdot X_5 \cdot X_{10} +$$

$$+6.63 \cdot X_6 \cdot X_9 - 7.7 \cdot X_7 \cdot X_8 + 0.32 \cdot X_9 \cdot X_{10} \leq 32\ mm$$

6) Middle rib deflection Δ_{MR} $[mm]$:

$$G_6(\bar{X}) = \Delta_{MR} = 33.86 + 2.95 \cdot X_3 + 0.1792 \cdot X_{10} - 5.057 \cdot X_1 \cdot X_2 - 11.0 \cdot X_2 \cdot X_8 -$$
$$-0.0215 \cdot X_5 \cdot X_{10} - 9.98 \cdot X_7 \cdot X_8 + 22.0 \cdot X_8 \cdot X_9 \leq 32\ mm$$

7) Lower rib deflection $\Delta_{LR}[mm]$:

$$G_7(\bar{X}) = \Delta_{LR} = 46.36 - 9.9 \cdot X_2 - 12.9 \cdot X_1 \cdot X_8 + 0.1107 \cdot X_3 \cdot X_{10} \leq 32\ mm$$

8) Public force F_P $[kN]$:

$$G_8(\bar{X}) = F_P = 4.72 - 0.5 \cdot X_4 - 0.19 \cdot X_2 \cdot X_3 - 0.0122 \cdot X_4 \cdot X_{10} +$$
$$+0.009325 \cdot X_6 \cdot X_{10} + 0.000191 \cdot X_{11}^2 \leq 4\ kN$$

9) Velocity of V-Pillar at middle point V_{MBP} $\left[\frac{mm}{ms}\right]$:

$$G_9(\bar{X}) = V_{MBP} = 10.58 - 0.674 \cdot X_1 \cdot X_2 - 1.95 \cdot X_2 \cdot X_8 + 0.02054 \cdot X_3 \cdot X_{10} -$$
$$-0.0198 \cdot X_4 \cdot X_{10} + 0.028 \cdot X_6 \cdot X_{10} \leq 9.9\ \frac{mm}{ms}$$

10) Velocity of front door at V-Pillar V_{FD} $\left[\frac{mm}{ms}\right]$:

$$G_{10}(\bar{X}) = V_{FD} = 16.45 - 0.489 \cdot X_3 \cdot X_7 - 0.843 \cdot X_5 \cdot X_6 + 0.0432 \cdot X_9 \cdot X_{10} -$$
$$-0.0556 \cdot X_9 \cdot X_{11} - 0.000786 \cdot X_{11}^2 \leq 15.7\ \frac{mm}{ms}$$

8.2.3 Welded Beam

The welded beam design [360], [361] consists of a rigid member attached to a beam, with a load applied to the member's end (see Figure A8.5).

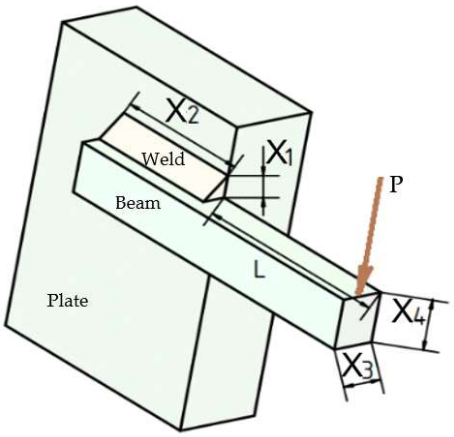


Figure A8.5- Schematic representation of the welded beam

The scope of the optimization problem is to achieve minimum production cost while maintaining adherence to specified design constraints, ensuring that the welded beam meets industry standards for shear stress, bending stress, buckling load, and end deflection.

The total production cost encompasses labour costs, which are a function of the weld dimensions, and the costs associated with weld and beam materials.

The primary objective is to optimize the beam's dimensions (X_1= thickness of the weld bead, X_2= length of the weld bead, X_3= height of the member's cross-sectional area, X_4= thickness of the member's cross-sectional area) to minimize the overall production cost.

This involves a careful consideration of the weld and member dimensions considering the continuous nature of $X_1$and X_2. They are strategically manipulated to find integer multiples that strike a balance between structural performance and cost efficiency.

The boundary conditions are given by:

$0.1 \leq X_1 \leq 2.0$ in

$0.1 \leq X_2 \leq 10.0$ in

$10.0 \leq X_3 \leq 0.1$ in

$0.1 \leq X_4 \leq 2.0$ in

The following parameters are defined as follows:

$E = 30 \cdot 10^6\ psi$	Young Modulus
$G = 12 \cdot 10^6\ psi$	Shearing Modulus
$L = 14\ inch$	Member's Overhang
$\tau_{max} = 13600\ psi$	Allowable Shear Stress

$\sigma_{max} = 30000\ psi$	Allowable Normal Stress
$\delta_{max} = 0{,}25\ inch$	Allowable Deflection
$P = 6000\ lb$	Concentrated Load
$C_1 = 0.10471\ \$/\ inch^3$	Welded Material's Cost per unit volume
$C_2 = 0.04811\ \$/inch^3$	Bar's Cost per unit volume
$C_3 = 1.00000\ \$/inch^3$	Labor's Cost per unit volume
$V_{WM} = X_1^2 \cdot X_2$	Welded Material's Volume
$V_B = X_3 \cdot X_4 \cdot (L + X_2)$	Bar's Volume
$\tau' = \frac{P}{\sqrt{2} \cdot (x_1 \cdot x_2)}$	Primary Stress
$\tau'' = \frac{M \cdot R}{J}$	Secondary Torsional Stress
$M = P \cdot \left(L + \frac{x_2}{2}\right)$	Bending Moment
$R = \sqrt{\left(\frac{x_2}{2}\right)^2 + \left(\frac{x_1 + x_3}{2}\right)^2}$	Radial Distance
$J = 2 \cdot \sqrt{2} \cdot x_1 \cdot x_2 \cdot \left(\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right)$	Polar Moment of Inertia
$\tau = \sqrt{\tau'^2 + 2 \cdot \tau' \cdot \tau'' \cdot \frac{x_2}{2 \cdot R} + \tau''^2}$	Total Shear Weld Stress
$\sigma = \frac{6 \cdot P \cdot L}{x_4 \cdot x_3^2}$	Bar's Bending Stress
$\delta = \frac{4 \cdot P \cdot L^3}{E \cdot x_4 \cdot x_3^2}$	Bar's deflection, with the assumption that the bar is a cantilever of length L

$$P_c = 4.013 \cdot \frac{E}{L^2} \cdot \sqrt{\frac{x_3^2 \cdot x_4^6}{36}} \cdot \left(1 - \frac{x_3}{2 \cdot L} \cdot \sqrt{\frac{E}{4 \cdot G}}\right)$$

Bar's Buckling load for a narrow rectangular section

The objective function, representing the total material cost, is defined as follows:

$$F_{obj}(\bar{X}) = (C_1 + C_3) \cdot V_{WM} + C_2 \cdot V_B = 1.1047 \cdot x_1^2 \cdot x_2 + 0.04811 \cdot x_3 \cdot x_4 \cdot (L + x_2)$$

The optimization process is guided by seven constraints on shear stress, bending stress, buckling load P_c , and end deflection δ as described below:

The Shear Stress cannot exceed the Allowable Shear Stress:

$$G(1) = \tau - \tau_{max} \leq 0$$

The Normal Stress cannot exceed the Allowable Normal Stress:

$$G(2) = \sigma - \sigma_{max} \leq 0$$

The Deflection cannot exceed the Allowable Deflection:

$$G(3) = \delta - \delta_{max} \leq 0$$

The Thickness of the Weld Bead cannot exceed the Thickness of the Member:

$$G(4) = x_1 - x_4 \leq 0$$

The Applied Load cannot exceed the Buckling Load:

$$G(5) = P - P_c \leq 0$$

The Thickness of the Weld Bead must be greater than a minimum value:

$$G(6) = 0.125 - x_1 \leq 0$$

The Cost of the material cannot exceed a minimum value:

$$G(7) = C_1 \cdot x_1^2 + C_2 \cdot x_3 \cdot x_4 \cdot (L + x_2) - 5 \leq 0$$

that is

$$= 0.10471 \cdot x_1^2 + 0.4811 \cdot x_3 \cdot x_4 \cdot (L + x_2) - 5 \leq 0$$

8.2.4 Pressure Vessel

The classical study [362] aims to minimize the total fabrication cost by constraining the volume and length of a cylindrical pressure vessel closed with two welded hemi-spherical head (see Figure A8.6).

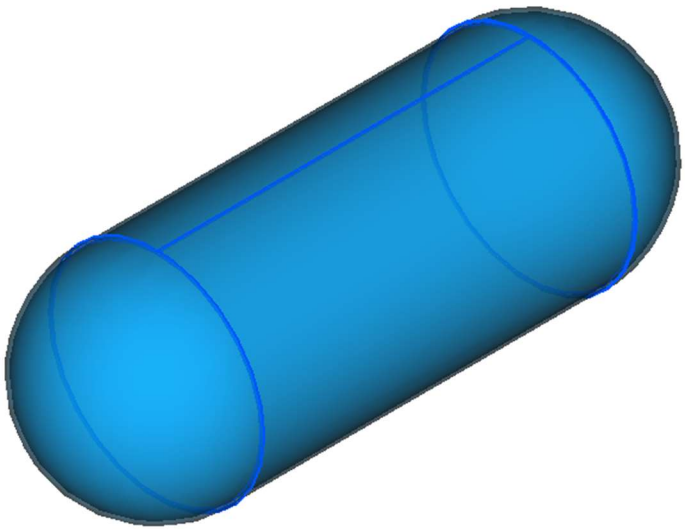


Figure A8.6- 3D representation of a Pressure Vessel

The evaluation focuses on the maximum shear stress, varying thickness under thermo-elastic conditions and steady-state temperature. The welding cost associated with the pressure vessel is determined based on a 60° single V-groove butt joint. The optimization process introduces constraints derived from on safe design limits, including considerations for stress, volume and length. The primary objective is to reduce the maximum shear stress to a specified threshold while ensuring the safety and operability of the pressure vessel at specified pressure and temperature levels.

The following assumptions are adopted for the analysis of this case study: the cylindrical shell operates under plane stress conditions, the external pressure is significantly smaller than internal or gauge pressure, the temperature remains constant over time, the material properties exhibit no temperature-dependent variations, the weld joint connecting the spherical head and cylindrical shell is robust, ensuring joint integrity, uniform density characterizes both the welding material and the material of the pressure vessel.

The four design variables are described in Figure A8.7: $T_s = X_1$= thickness of sheet, $T_h = X_2$= thickness of hemispherical heads, $R = X_3$ R= radius of hemispherical heads, $L = X_4$= length of cylindrical shell.

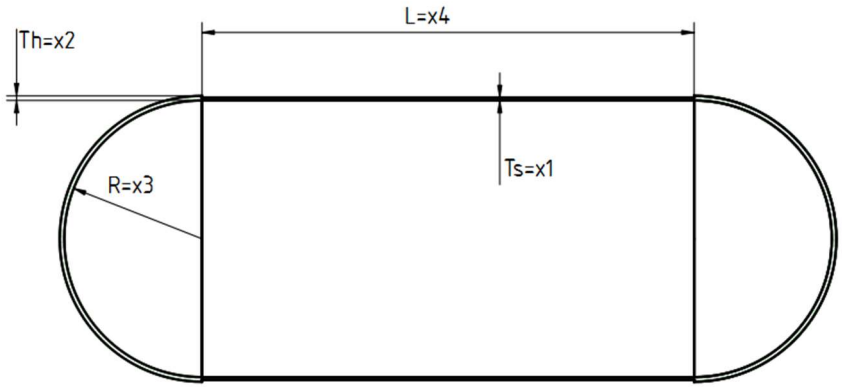


Figure A8.7- Cross-sectional view illustrating the design variables

The boundary conditions are given by:

$0 < X_1 < 100$

$0 < X_2 < 100$

$10 < X_3 < 200$

$10 < X_4 < 200$

The following parameters are defined as follows:

$T_a = 100°C$	Internal temperature
$T_b = 25°C$	External temperature
$E = 210\ GPa$	Young modulus
$p = 0{,}010\ GPa$	Internal pressure
$C_s = 0{,}0124 \cdot 10^{-3}$ \$/kg	Cost of weld material related to the shell per kg
$C_h = 0{,}1979 \cdot 10^{-3}$ \$/kg	Cost of welding material associated with the hemispherical shells, per kg
$C_w = 0{,}3283 \cdot 10^{-3}$ \$/kg	Cost of weld material per kg
$\rho \approx 7850\ kg/m^3$	Steel density
$A=\left(\frac{T_s}{\cos 30°}\right)^2 \cdot \left(\frac{60°}{360°}\right)$	Area of one-sixth sector of a circle
$C_1 = 2 \cdot \pi \cdot \rho \cdot L \cdot C_w \cdot = 3.1661 \cdot A \cdot X_1^2 \cdot X_4$	Cost of longitudinal weld for a cylindrical shell. The value 2 denotes

the welding of two rolled sheets longitudinally at two locations to create a cylindrical shell

$$C_2 = 4 \cdot \pi^2 \cdot \rho \cdot R \cdot C_w \cdot \left(\frac{T_h}{\cos 30°}\right)^2 \cdot \left(\frac{60°}{360°}\right) = 1.7781 \cdot X_2 \cdot X_3^2$$

Cost of welding hemispherical heads to the cylindrical shell

$$C_3 = 2 \cdot \pi \cdot \rho \cdot R \cdot L \cdot T_s \cdot C_s + 4 \cdot \pi \cdot \rho \cdot R^2 \cdot T_h \cdot C_h = 0.6224 \cdot X_1 \cdot X_3 \cdot X_4 + 19.84 \cdot (X_1^2 \cdot X_3)$$

Cost related to the material

$$\sigma_{rr_s} = \frac{p \cdot R}{((R+T_s)^2 - R^2)} \cdot \left(1 - \frac{(R+T_s)^2}{r^2}\right) + \\ + \frac{\alpha \cdot E}{2} \cdot (T_b - T_a) \cdot \\ \cdot \left[-\frac{\ln \frac{r}{R}}{\ln \frac{R+T_s}{R}} + \left(1 - \frac{R^2}{r^2}\right) \cdot\cdot \frac{(R+T_s)^2}{(R+T_s)^2 - R^2}\right]$$

Radial shear stress for the cylindrical shell

$$\sigma_{\theta\theta_s} = \frac{p \cdot R^2}{((R+T_s)^2 - R^2)} \cdot \left(1 + \frac{(R+T_s)^2}{r^2}\right) + \\ + \frac{\alpha \cdot E}{2} \cdot\cdot (T_b - T_a) \cdot \frac{1 + \ln \frac{r}{R}}{\ln \frac{R+T_s}{R}} + \\ + \left(1 + \frac{R^2}{r^2}\right) \cdot \frac{(R+T_s)^2}{(R+T_s)^2 - R^2}$$

Tangential shear stress for the cylindrical shell

$$\sigma_{rr_h} = \frac{p \cdot R^3}{((R+T_h)^3 - R^3)} \cdot \left(1 - \frac{(R+T_h)^3}{r^3}\right) - \\ - \frac{\alpha \cdot E}{1-\nu} \cdot (T_b - T_a) \cdot \frac{R \cdot (R+T_h)}{(R+T_h)^3 - R^3} \cdot \left(1 - \frac{R}{r}\right) \cdot \\ \cdot \left(1 - \frac{R+T_h}{r}\right) \cdot \left(2 \cdot R + T_h + \frac{R \cdot (R+T_h)}{r}\right)$$

Radial shear stress for the hemispherical shell

$$\sigma_{\theta\theta_h} = \sigma_{\phi\phi} = \frac{p \cdot R^3}{(R+T_h)^3 - R^3} \cdot \left(1 + \frac{(R+T_h)^3}{2 \cdot r^3}\right) - \\ - \frac{\alpha \cdot E}{1-\nu} \cdot (T_b - T_a) \cdot \frac{R \cdot (R+T_h)}{(R+T_h)^3 - R^3} \cdot \\ \cdot \left[2 \cdot R + T_h - ((R+T_h)^2 + R \cdot (R+T_h) + R^2) \cdot \right. \\ \left. \cdot \frac{1}{2 \cdot r} - \frac{R^2 \cdot (R+T_h)^2}{2 \cdot r^3}\right]$$

Longitudinal shear stress for the hemispherical shell

$$\tau = \frac{\max(\sigma_{rr}, \sigma_{\theta\theta}, \sigma_{\phi\phi}) - \min(\sigma_{rr}, \sigma_{\theta\theta}, \sigma_{\phi\phi})}{2}$$

Tresca shear stress

It should be noted that the subscript "s" denotes the shell portion, while "h" denotes the hemispherical zone.

The objective function, representing the total material cost, is defined as follows:

$F_{obj}(\bar{X}) = C_1 + C_2 + C_3$

Therefore, for the present problem, four engineering constraints are defined.

The shear stress should not exceed the allowable shear stress for both the cylindrical shell and the spherical heads:

$$\tau < \tau_{max}$$

To simplify, the previous formulas are replaced by the following two empirical geometric constraints:

$G(1) = 0.0193 \cdot R - \tau_s \leq 0$

$G(2) = 0.00954 \cdot R - \tau_h \leq 0$

$G(3) = -\pi \cdot R^2 \cdot L - \frac{4}{3} \cdot \pi \cdot R^3 + 1296000 \leq 0$

$G(4) = L - 240 \leq 0$

8.2.5 Topology: Characterization of a flat composite panel subjected to axial compression

This represents a nonlinear and multi-physical optimization problem aimed at the inverse identification of material properties and structural characteristics of composite structures under mechanical loading. Specifically, the objective is to determine unknown material constants and layup configurations that best replicate experimentally observed structural responses, typically under conditions that may include instabilities such as buckling [74].

The case developed as example investigation is a flat IM7/977-2 graphite-epoxy composite panel with dimensions 430 × 165 × 3 mm. According to the manufacturer, the layup consists of one-third of the plies oriented at 0°, one-third at 90°, one-third at ±45°, and the target values of elastic constants are: E_L=148300 MPa, E_T=7450 MPa, G_{LT}=4140 MPa and ν_{LT}=0.01510.

The optimization problem involves seven design variables: the four elastic constants $X_1 = E_L$, $X_2 = E_T$, $X_3 = G_{LT}$ and $X_4 = \nu_{LT}$, and the three-ply orientations $X_5 = \theta_0$, $X_6 = \theta_{90}$ and $X_7 = \theta_{45}$ of the laminate (respectively corresponding to nominal ply angles 0°, 90° and ±45°).

Assuming $\bar{X} = (E_L, E_T, G_{LT}, \nu_{LT}, \theta_0, \theta_{90}, \theta_{45})$, the objective function to be minimized is given by:

$$\Omega(\bar{X}) = \sqrt{\sum_{p=1}^{N_{CNT}} \left(\frac{w_{norm,FEM}^{p} - w_{norm,EXP}^{p}}{w_{norm,FEM}^{p}} \right)^2}$$

where Ω denotes the functional error used to quantify the normalized discrepancy between the experimentally measured fundamental buckling mode shape ($w_{norm,EXP}$) and the numerically predicted mode shape from a finite element model ($w_{norm,FEM}$) evaluated at each measurement point p, for a total of N_{CNT} control points on the panel.

The boundary conditions are given by:

$10000 \leq X_1, X_2 \leq 1000000$

$500 \leq X_3 \leq 20000$

$0.001 \leq X_4 \leq 0.1$

$-50 \leq X_5 \leq 50$

$0 \leq X_6 \leq 91$

$10 \leq X_7 \leq 70$

Therefore, for the present problem, two engineering constraints are defined:

$G(1) = E_T - E_L \leq 0$

$G(2) = \nu_{LT}(\frac{E_T}{E_L}) - 1 \leq 0$

The buckling mode shape of the composite panel was experimentally measured using the white light double-illumination projection moiré setup depicted in Figure A8.8.

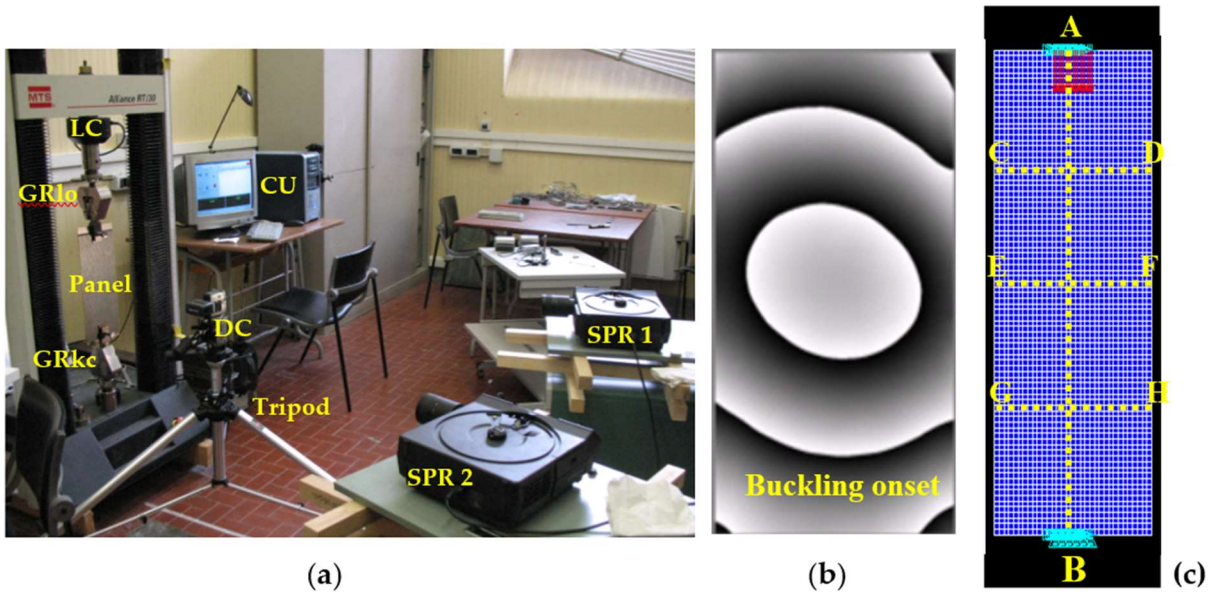


Figure A8.8– (a) Experimental setup employed for the measurement of the buckling mode shape of a composite panel using white-light double-illumination projection moiré. LC: load cell; GRlo and GRkc: upper and lower grips of the testing machine; Panel: test specimen; DC and Tripod: digital camera with tripod; SPR1 and SPR2: slide projectors, CU: control unit. (b) Moiré fringe pattern recorded at the onset of buckling, illustrating the out-of-plane displacement field. (c) Finite element model with control paths (AB, CD, EF, GH) employed to evaluate the error functional Ω by comparing experimental and numerical displacement distributions.

The specimen is subjected to axial compression by gradually displacing the upper grip (GRlo) of the testing machine while keeping the lower grip (GRkc) fixed, thereby inducing incremental end-shortening. The applied axial load is continuously monitored using a load cell (LC).

Two slide projectors (SPR1, SPR2), symmetrically positioned with respect to the optical axis, project line gratings onto the specimen surface. Their superposition generates an interference pattern that deforms as the specimen buckles. A digital camera (DC), mounted on a tripod and aligned orthogonally to the specimen, captures the moiré fringes at successive load steps.

By comparing each moiré image with a reference of the undeformed configuration, the out-of-plane displacement field is reconstructed and the buckling mode shape identified (see Figure A8.8.b).

The experimental displacement data are then compared with the finite element model (FEM) under equivalent boundary and loading conditions. The FEM model has been implemented in ANSYS® using 2465 8-node shell elements. As shown in Figure A8.8.c, control paths (AB, CD, EF, GH) defined on the panel serve as locations where displacements are extracted both experimentally and numerically. The vertical path AB has been chosen due to the panel's typical Euler buckling mode, with maximum displacement at the center, while the additional horizontal paths increase the number of control nodes for the Ω functional. These datasets are used to compute the error functional (Ω), quantifying the difference between measured and simulated buckling mode shapes. This procedure enabled the calibration of uncertain material parameters, thereby improving the fidelity of the FEM and ensuring consistency with the experimental observations.

Additional information on the projection moiré technique and experimental set up is given in [74], [202], [203], [204], and the recorded images are processed with HoloStrain [205].

8.3 Robotics: Path Planning

This optimization problem [363], [364] represents a classical robotics' problem: computing an optimal 2D trajectory from a start point (A) to an end point (B) by means of a cubic spline, subject to obstacle avoidance constraints. The trajectory is defined by a set of control points (NCP) and is constrained to pass through a series of base points (NBP), while avoiding a specified number of obstacles (NOB) (see Figure A8.9). Additionally, the path must begin at point A, terminate at point B, and ensure collision-free motion throughout the environment.

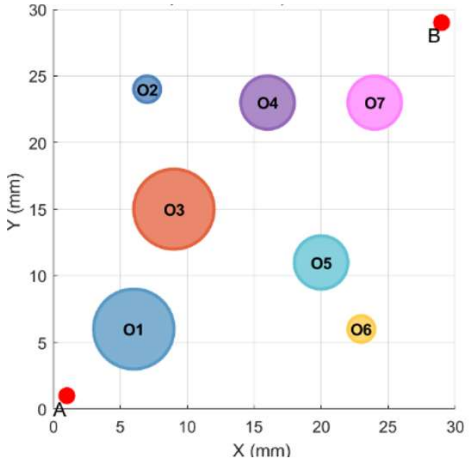


Figure A8.9- Planar representation of a typical path planning scenario, including the start point (A), the end point (B), and the obstacles distributed within the environment.

The objective is to minimize the total length of the spline trajectory while ensuring no collisions and maintaining a safe distance from the circles representing the obstacles. In an obstacle-free environment, the shortest path between A and B would be a straight line. Nevertheless, obstacles can be effectively handled by employing cubic splines, which generate smooth and continuous trajectories that are optimal for vehicle manoeuvrability.

The NBP determines the flexibility of the trajectory in adapting to different environments. In addition, the NCP, which represents the set of evaluation nodes along the spline where the path is sampled and optimized, directly impacts the resolution and accuracy of the spline trajectory representation. A higher number of NCP provides a finer discretization of the path, allowing for more precise control and smoother curves, which can improve obstacle avoidance and path feasibility.

However, increasing NCP also raises the computational complexity of the optimization problem, as more evaluation points require additional collision checks, constraint evaluations, and optimization variables to process. Conversely, too few nodes may lead to an insufficiently detailed path, potentially missing collisions or producing fewer smooth trajectories, which can negatively affect vehicle manoeuvrability.

Boundary conditions are defined by restricting the design variables to remain within the spatial bounds established by the start point (A) and the end point (B).

The trajectory $\mathscr{C}$ is a complex curve composed of NCP−1 cubic spline segments; each defined between successive control points among the total of NCP points. While continuous, the overall path may exhibit non-smooth behaviour depending on the configuration of the control points.

The spline is evaluated at multiple points to determine the (X, Y) coordinates of the interpolated curve represented by a cubic polynomial $Y_i = S_i(\bar{X})$ between each pair adjacent control points$(X_i,\ Y_i)$ and (X_{i+1}, Y_{i+1}) :

$$S_i(\bar{X}) = a_i \cdot \bar{X}^3 + b_i \cdot \bar{X}^2 + c_i \cdot \bar{X} + d_i$$

The length of the $\mathscr{C}$ trajectory is calculated as

$$W = \sum_{i=1}^{NCP} \left(\sqrt{(X_{i+1} - X_i)^2 + (Y_{i+1} - Y_i)^2}\right) + \ PENALTY$$

where "PENALTY" equals zero if the trajectory does not collide with any of the NOB obstacles.

The term "Penalty" aggregates the squared violations of constraint inequalities, thereby amplifying the impact of any collision events.

The optimization problem is to minimize the total trajectory length: Minimize W.

The design variables are subject to a specific number of constraints, defined as:

NCON= (NCP+ NBP) × NOB.

These constraints are formulated to ensure that each NCP and each NBP maintains a safe distance from all obstacles (NOB), thereby guaranteeing collision-free trajectory generation.

Each obstacle is represented by a circular region of center (x_p, y_p) and radius ρ_p. Non-collision conditions represent the nonlinear constraints of the path planning problem. Specifically, the Euclidean distance between any control (x_c, y_c) or base point (x_b, y_b) of the trajectory and the center (x_p, y_p) of an obstacle must be greater than the obstacle radius ρ_p:

$$G(i) = \frac{\rho_p(j)}{\sqrt{\left(x_c(r) - x_p(j)\right)^2 + \left(y_c(r) - y_p(j)\right)^2}} - 1 < 0$$

$$G(ii) = \frac{\rho_p(j)}{\sqrt{\left(x_b(k) - x_p(j)\right)^2 + \left(y_b(k) - y_p(j)\right)^2}} - 1 < 0$$

where

i=1, …, NCP × NOB

ii=1, …, NBP × NOB

j= 1, 2, ..., NOB

r= 1, 2, ..., NCP

k= 1, 2, ..., NBP

8.4 Fluid Machines: Industrial Refrigeration System

The original optimization problem [199] aims to minimize the manufacturing and operational costs of an industrial refrigeration system (see Figure A8.10).

The design is characterized by 14 variables and is subject to 15 non-linear constraints, which are determined by the system's geometry and the heat transfer characteristics of the refrigeration cycle. Analysis of the design variables suggests that a two-step configuration provides the optimal solution, while variations in other factors have negligible impact on the

cost.

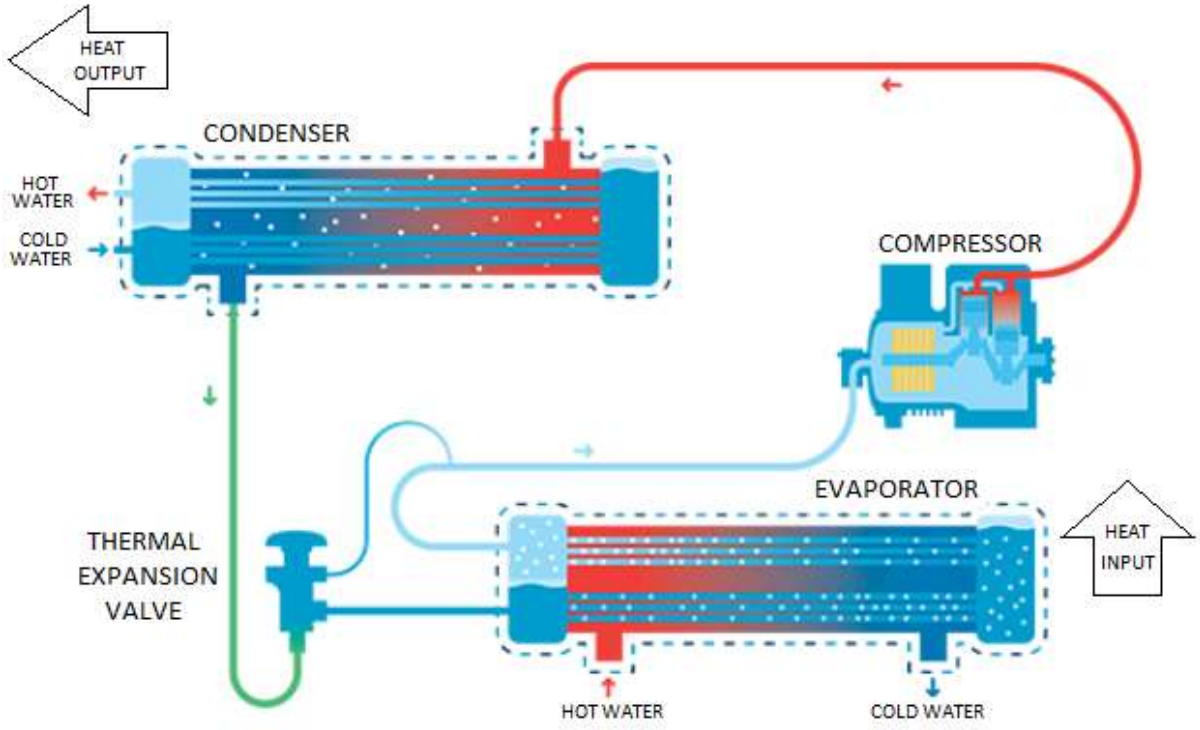


Figure A8.10- Representation of an Industrial refrigeration System

The system is required to meet the following performance specifications and design features, following the standard compliances ARI, ASTM, ASME, ASHRAE:

- Cycle type: Standard vapor-compression cycle
- Condenser and evaporator type: horizontal, multi-pass, shell-and-tube
- Expansion device: Thermostatic expansion valve (off-the-shelf)
- Compressor: Commercially available (off-the-shelf)
- Tube external diameter for both condenser and evaporator: 0.0318 m
- Tube material: Copper Type L, wall thickness 2.5 mm, absolute roughness of $0.127 \cdot 10^{-6}\ m$
- Tube arrangement: Equilateral triangular pitch, pitch-to-diameter ratio 1.25, selected for maximum header strength relative to shell-side flow area
- Tubesheet, flange, end-cover material: Carbon steel ASTM-A285 Grade A, maximum allowable stress 165 MPa
- Shell material: Carbon steel SA53 Grade B, maximum allowable stress 103 MPa
- Refrigerating capacity: 615.3 kW (175 tons)
- Leaving chilled water temperature: 6.7°C

- Entering condenser water temperature: 28°C
- Condensing temperature: 40°C
- Evaporating temperature: 5°C
- Refrigerant: R-22 (see Figure A8.11)

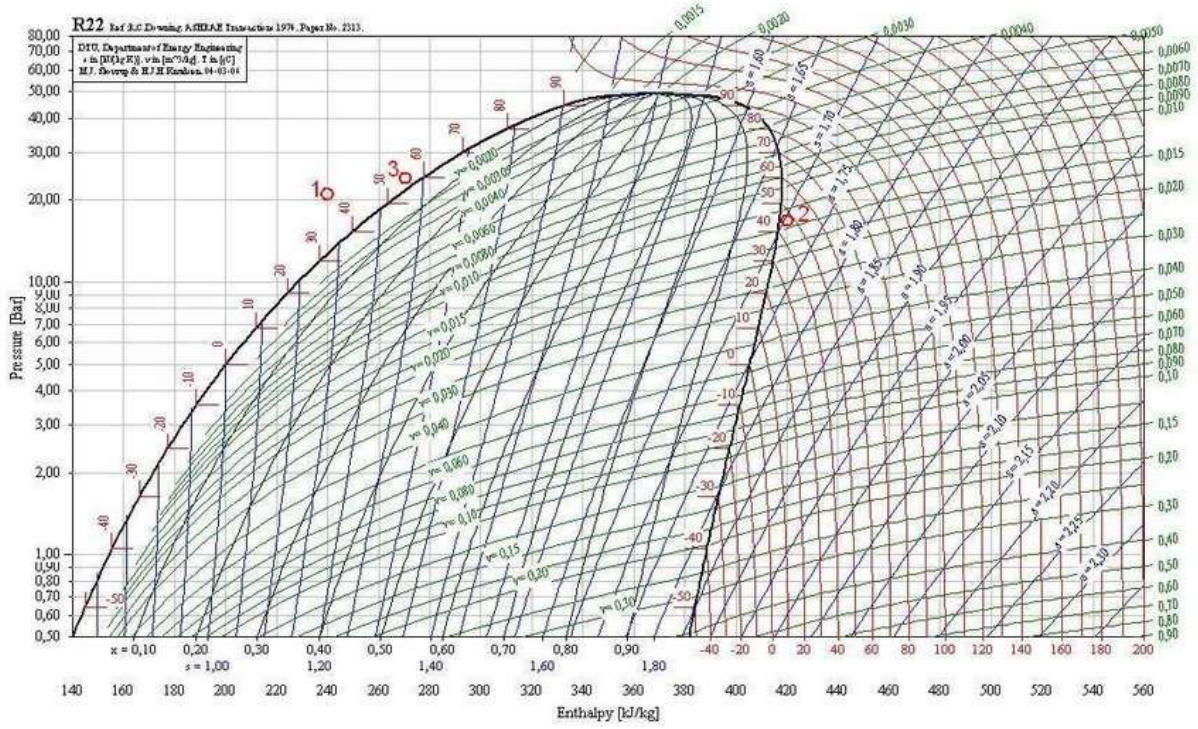


Figure A8.11- R22 Mollier Diagram

The 14 design variables are:

- shell diameter of the evaporator ($X_1 = D_{se}$)
- shell diameter of the condenser ($X_2 = D_{sc}$)
- shell thickness of the evaporator ($X_3 = T_{se}$)
- shell thickness of the condenser ($X_4 = T_{sc}$)
- tubesheet thickness for the evaporator ($X_5 = t_{se}$)
- tubesheet thickness for the condenser ($X_6 = t_{sc}$)
- fluid velocity in the evaporator ($X_7 = V_e$)
- fluid velocity in the condenser ($X_8 = V_c$)
- number of tube passes for the evaporator ($X_9 = P_e$)
- number of tube passes for the condenser ($X_{10} = P_c$)
- tube length for the evaporator ($X_{11} = L_e$)
- tube length for the condenser ($X_{12} = L_c$)
- condenser effectiveness ($X_{13} = \varepsilon$)
- inlet/outlet fluid temperature difference ($X_{14} = \Delta T_c$).

with the following boundary conditions on design variables

$$0.001 < x_i < 5 \qquad i = 1,2, \ldots .14$$

The objective function to be minimized is:

$$\begin{aligned} W(\overline{X}) = {} & 63098.88 \cdot X_2 \cdot X_4 \cdot X_{12} + 5441.5 \cdot X_2^2 \cdot X_{12} + 115055.5 \cdot X_2^{1.664} \cdot X_6 \\ & + 6172.27 \cdot X_2^2 \cdot X_6 + 63098.88 \cdot X_1 \cdot X_3 \cdot X_{11} + 5441.5 \cdot X_1^2 \cdot X_{11} \\ & + 115055.5 \cdot X_1^{1.664} \cdot x_5 + 6172.27 \cdot X_1^2 \cdot X_5 + 140.53 \cdot X_1 \cdot X_{11} \\ & + 281.29 \cdot X_3 \cdot X_{11} + 70.26 \cdot X_1^2 + 281.29 \cdot X_1 \cdot X_3 + 281.29 \cdot X_3^2 \\ & + 14437 \cdot X_8^{1.8812} \cdot X_{12}^{0.3424} \cdot \frac{X_{10}}{X_{14}} \cdot X_1^2 \cdot \frac{X_7}{X_9} + 20470.2 \cdot X_7^{2.893} \cdot X_{11}^{0.316} \cdot X_1^2 \end{aligned}$$

Subject to:

$G(1) = \frac{1.524}{X_7} - 1 \leq 0$ (EVAPORATOR VELOCITY LOWER LIMIT)

$G(2) = \frac{1.524}{X_8} - 1 \leq 0$ (CONDENSER VELOCITY LOWER LIMIT)

$G(3) = 0.07789 \cdot X_1 - 2 \cdot \frac{X_9}{X_7} - 1 \leq 0$ (EVAPORATOR FLOW BALANCE)

$G(4) = \frac{7.05305 \cdot X_1^2 \cdot X_{10}}{X_2 \cdot X_8 \cdot X_9 \cdot X_{14}} - 1 \leq 0$ (CONDENSER FLOW BALANCE)

$G(5) = \frac{0.0833 \cdot X_{14}}{X_{13}} - 1 \leq 0$ (CONDENSER EFFECTIVENESS)

$G(6) = 47.136 \cdot X_2^{0.333} \cdot \frac{X_{12}}{X_{10}} - 1.333 \cdot X_{13}^{2.1195} \cdot X_8 + \frac{62.08 \cdot X_{13}^{2.1195} \cdot X_8^{0.2}}{X_{10} \cdot X_{12}} - 1 \leq 0$
(CONDENSER ENERGY BALACE)

$G(7) = 0.04771 \cdot X_{10} \cdot X_8^{1.8812} \cdot X_{12}^{0.3424} - 1 \leq 0$ (CONDENSER TUBE-SIDE FLOW CONSTRAINT)

$G(8) = 0.0488 \cdot X_9 \cdot X_7^{1.893} \cdot X_{11}^{0.316} - 1 \leq 0$ (EVAPORATOR TUBE-SIDE FLOW CONSTRAINT)

$G(9) = 0.0099 \cdot \frac{X_1}{X_3} - 1 \leq 0$ (EVAPORATOR SHELL THICKNESS CONSTRAINT)

$G(10) = 0.0193 \cdot \frac{X_2}{X_4} - 1 \leq 0$ (CONDENSER SHELL THICKNESS CONSTRAINT)

$G(11) = 0.0298 \cdot \frac{X_1}{X_5} - 1 \leq 0$ (EVAPORATOR TUBESHEET THICKNESS CONSTRAINT)

$G(12) = 0.056 \cdot \frac{X_2}{X_6} - 1 \leq 0$ (CONDENSER TUBESHEET THICKNESS CONSTRAINT)

$G(13) = \frac{2}{X_9} - 1 \leq 0$ (MINIMUM TUBE PASSES IN EVAPORATOR)

$G(14) = \frac{2}{X_{10}} - 1 \leq 0$ (MINIMUM TUBE PASSES IN CONDENSER)

$G(15) = \frac{X_{12}}{X_{11}} - 1 \leq 0$ (CONDENSER/EVAPORATOR TUBE LENGTH RATIO CONSTRAINT)

8.5 Hydraulics:

8.5.1 Muskingum 3 variables

The Muskingum case represents a typical nonlinear unconstrained mathematical inverse problem applied in hydrological modelling [225], [226], [227].

Generally, the Muskingum model can be applied to achieve various objectives such as minimizing flood risks, maximizing water supply reliability, and optimizing the operation of hydraulic structures. In this study, it is used to estimate the outflow hydrograph from a river reach, given the inflow hydrograph.

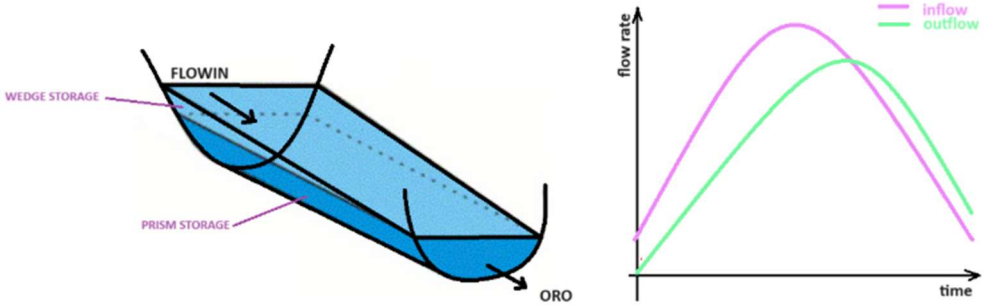


Figure A8.12- Scheme of a of a river reach traversed by a water flow and typical time series graph of inflow and outflow behaviour

The objective is to minimize the difference between the observed and the modelled river outflows over a span of 126 hours, divided into 21 intervals of 6 hours each. These outflows are compared at 22 specific control times.

The objective function is defined as follows:

$$Min(SSQ) = \sum_{t=1}^{NCT} \left(O_{oro,t} - O_{cro,t}\right)^2$$

where $O_{oro,t}$ and $O_{cro,t}$ represent the observed and computes river outflows at time t, respectively, and NCT=22 is the number of control points.

The computed river outflow $O_{cro,t}$ is determined using the nonlinear Muskingum model:

$$O_{cro,t} = \frac{1}{1-h} \cdot \left(\frac{S_t}{k}\right)^{\frac{1}{r}} \cdot (1-h) \cdot I_t$$

Where:

- k=X_1, $h = X_2$, $r = X_3$, are the Muskingum model parameters to be calibrated (optimization variables)

- S_t is the channel storage at time t
- I_t is the measured river inflow at time t.

The channel storage S_t is updated according to:

$$S_{t+1} = S_t + \dot{S}_t \cdot \Delta t$$

with the storage rate given by:

$$\dot{S}_t = \frac{1}{1-h} \cdot \left(\frac{S_t}{k}\right)^{\frac{1}{r}} + \frac{1}{1-h} \cdot I_t$$

The boundary conditions on the design variables are:

$$0.01 < k < 0.2,\ 0.2 < h < 0.3,\ 1.5 < r < 2.5$$

8.5.2 Muskingum 25 variables

This case adopts the same objective function as the 3-variable problem; however, the design variables include not only the three parameters k, h, and r previously defined, but also the storage values S_t at each time step, yielding a total of 25 optimization variables.

The boundary conditions are defined as follows:

$$0.01 < k < 0.2,\ 0.2 < h < 0.3,\ 1.5 < r < 2.5,\ 1 < S_t < 1000 \cdot m^3$$

In addition, the storage values are constrained within a narrow range ($\pm 0.01\%$) around the specified target values X_{TRG}, defined as:

$$X_{TRG} = [27.7515;\ 26.7557;\ 35.2807;\ 80.5214;\ 147.4765;\ 199.3944;\ 244.3567;$$

$$279.5318;\ 296.3119;\ 301.2047;\ 291.969;\ 263.4177;\ 229.9541;\ 187.8578;\ 146.8396;$$

$$107.3812;\ 75.8607;\ 54.8967;\ 41.0549;\ 30.8404;\ 25.7578;\ 20.5118]$$

The corresponding boundary conditions are:

$$0.999 \cdot X_{TRG} < X_i < 1.001 \cdot X_{TRG} \qquad for\ i = 4, \ldots, 25$$

8.6 Hydraulic Engineering: Dam case 1- only vertical component for earthquake

This optimization problem is based on a real case study originating from an academic investigation of a dam located in the Shafaroud region of northern Iran (see Figure A8.12).

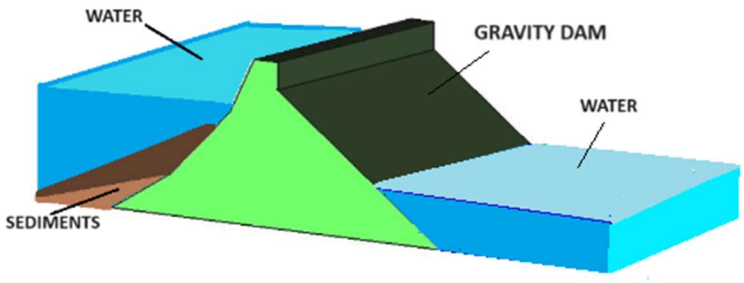


Figure A8.12- Gravity Dam in 3D.

Figure A8.13 shows the dam cross-section subject to optimization and provides a clear representation of the variables $\bar{X} = (X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9)$.

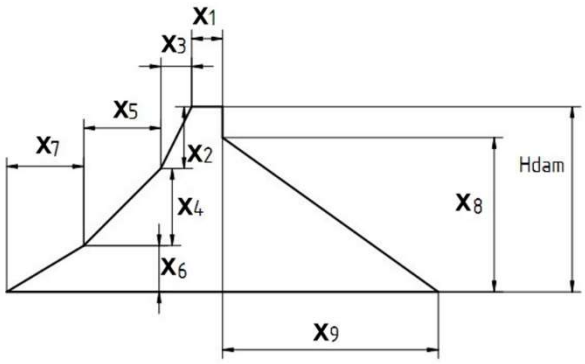


Figure A8.13- Identification of variables.

The dam's base is specified as follows:

$$b = X_1 + X_3 + X_5 + X_7 + X_9$$

To assess the structural stability of the dam, it is essential to evaluate the total horizontal and vertical forces acting on its cross-section (see Figure A8.14).

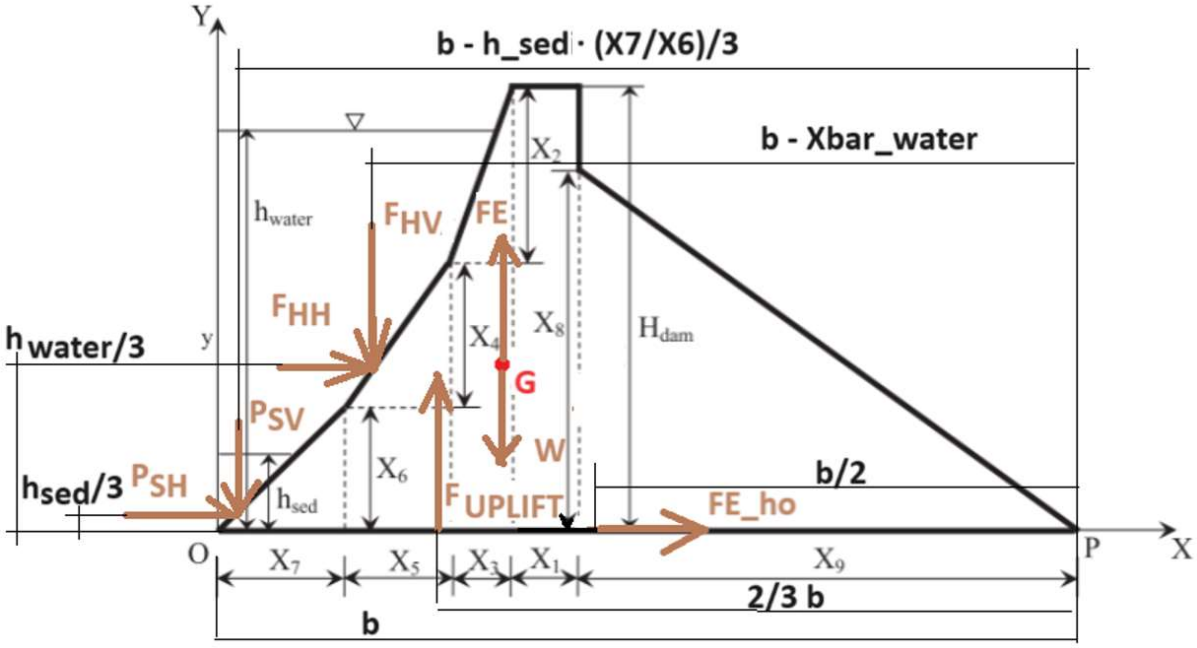


Figure A8.14- Forces acting on the cross section of a gravity dam subjected to a horizontal earthquake force (FE_ho).

The setting of the design parameters is defined as follows:

$H_{dam} = 150\ m$ Dam height

$h_{water} = 145\ m$ Water level

$h_{sed} = 7\ m$	Sediment thickness
$\gamma_{water} = 9{,}81\,\frac{kN}{m^3}$	Water's specific weight
$\gamma_{concrete} = 23\ \frac{kN}{m^3}$	Concrete's specific weight
$\gamma_{sed} = 15\ \frac{kN}{m^3}$	Sediment's specific weight
$\sigma_{max} = 25000kPa$	Concrete's allowable stress
$\sigma = 3500\ kPa$	Load on the base of the dam
$f = 0.7$	Static friction coefficient
$\alpha = 0.05$	Safety factor for weights
$\varphi = 50°$	Sediment's internal friction angle

The x-coordinate of centre of mass of the dam is defined as:

$$X_{bar_dam} = \frac{(DamX1 + DamX2 + DamX3 + DamX4 + DamX5 + DamX6 + DamX7)}{V_{dam}}$$

where

$$DamX1 = \frac{1}{2} \cdot (X_6 \cdot X_7) \cdot \left(\frac{2}{3} \cdot X_7\right)$$

$$DamX2 = \frac{1}{2} \cdot (X_4 \cdot X_5) \cdot \left(X_7 + \frac{2}{3} \cdot X_5\right)$$

$$DamX3 = \frac{1}{2} \cdot (X_2 \cdot X_3) \cdot \left(X_7 + X_5 + \frac{2}{3} \cdot X_3\right)$$

$$DamX4 = X_5 \cdot X_6 \cdot \left(X_7 + \frac{X_5}{2}\right)$$

$$DamX5 = X_3 \cdot (X_4 + X_6) \cdot \left(X_7 + X_5 + \frac{X_3}{2}\right)$$

$$DamX6 = H_{dam} \cdot X_1 \cdot \left(X_7 + X_5 + X_3 + \frac{X_1}{2}\right)$$

$$DamX7 = \frac{1}{2} \cdot (X_8 \cdot X_9) \cdot \left(b - \frac{2}{3} \cdot X_9\right)$$

The x-coordinate of the centre of mass of the water is given by:

$$X_{bar_water} = \frac{WatX1 + WatX2 + WatX3 + WatX4 + WatX5}{V_{water}}$$

where

$$WatX1 = \frac{1}{2} \cdot (X_6 \cdot X_7) \cdot \frac{X_7}{3}$$

$$WatX2 = X_4 \cdot X_7 \cdot \frac{X_7}{2}$$

$$WatX3 = \frac{1}{2} \cdot X_4 \cdot X_5 \cdot \left(X_7 + \frac{X_5}{3}\right)$$

$$WatX4 = (h_{water} - X_4 - X_6) \cdot (X_5 + X_7) \cdot \left(X_5 + \frac{X_7}{2}\right)$$

$$WatX5 = \frac{1}{2} \cdot (h_{water} - X_4 - X_6)^2 . \frac{X_3}{X_2} \cdot \left(X_5 + X_7 + (h_{water} - X_4 - X_6) . \frac{X_3}{3 \cdot X_2}\right)$$

The objective function minimizes the total volume of the dam, calculating the cross-sectional area per unit length, as follows:

$$F_{obj}(\bar{X}) = H_{dam} \cdot X_1 + X_3 \cdot (X_4 + X_6) + X_5 \cdot X_6 + \frac{X_2 \cdot X_3}{2} + \frac{X_4 \cdot X_5}{2} + \frac{X_6 \cdot X_7}{2} + \frac{X_8 \cdot X_9}{2}$$

The total horizontal and vertical forces acting on the dam are given by:

$$\Sigma F_H = P_{SH} + F_{HH} + FE_{ho}$$

$$\Sigma F_V = P_{SV} + F_{HV} - F_{Uplift} + W - \mathrm{FE}$$

where

$P_{SH} = \frac{1}{2} \gamma_{sed} \cdot h_{sed}^2 \cdot \frac{1 - sin\varphi}{1 + sin\varphi}$ Horizontal sediment pressure on the dam

$P_{SV} = \frac{1}{2} \gamma_{sed} \cdot h_{Ssed}^2 \cdot tan\theta$ Vertical sediment pressure on the dam

$F_{HH} = \frac{1}{2} \gamma_{water} \cdot h^2$ Horizontal component of water pressure

$F_{HV} = \frac{1}{2} \gamma_{water} \cdot V_{water}$ Vertical component of water pressure

$F_{UPLIFT} = \frac{1}{2} \gamma_{water} \cdot h$ Uplift force

FE_{ho}	Earthquake force
$\mathrm{FE} = \alpha \cdot W$	Earthquake uplift force
$\theta = \tan^{-1}\left(\frac{x_6}{x_7}\right)$	Angle of repose of the sediments
$W = \gamma_{concrete} \cdot V_{dam}$	Weight of the dam
$V_{dam} = H_{dam} \cdot X_1 + X_3 \cdot (X_4 + X_6) + X_5 \cdot X_6 + \frac{1}{2} \cdot X_2 \cdot X_3 + \frac{1}{2} \cdot X_4 \cdot X_5 + \frac{1}{2} \cdot X_6 \cdot X_7 + \frac{1}{2} \cdot X_8 \cdot X_9$	Volume of the dam
$V_{water} = \frac{1}{2} \cdot X_6 \cdot X_7 + \frac{1}{2} \cdot X_4 \cdot X_5 + \frac{1}{2} \cdot (h_w - X_4 - X_6)^2 \cdot \frac{X_3}{X_2} + X_4 \cdot X_7 + (h_{water} - X_4 - X_6) \cdot (X_5 + X_7)$	Volume of the water
$SFS = \frac{f \cdot \Sigma F_V + \sigma \cdot b}{\Sigma F_H}$	Safety factor against dam sliding
$SFO = \frac{\Sigma M_R}{\Sigma M_O}$	Safety factor against dam overturning
$\sigma_U = \frac{\Sigma F_V}{b} - 6 \cdot \frac{\Sigma M_O}{b^2}$	Vertical fatigue specific loads in upstream
$\sigma_D = \frac{\Sigma F_V}{b} + 6 \cdot \frac{\Sigma M_O}{b^2}$	Vertical fatigue specific loads in downstream
$M_O = \frac{F_{HH} \cdot h_{water}}{3} + F_{UPLIFT} \cdot \frac{2}{3} \cdot b + \frac{P_{SH} \cdot h_{sed}}{3}$	Total overturning moment
$M_R = (W - \mathrm{FE}) \cdot \left(b - X_{bar_dam}\right) + W_{water} \cdot \left(b - X_{bar_water}\right) + \frac{1}{3} \cdot P_{SV} \cdot \left(b - h_{sed} \cdot \frac{x_7}{x_6}\right)$	Total resisting moment

The eight design constraints governing the optimization of the dam cross-section are defined as follows:

The safety factor against dam sliding is imposed to be more than a fixed value:

$SFS \geq 4$

The safety factor against dam overturning exceeds a predetermined threshold:
$SFO \geq 1{,}5$

The vertical fatigue specific loads in upstream is constrained to fall within the range of zero to the respective fatigue limit, inclusively.

$0 \leq \sigma_u \leq \sigma_{max}$

The vertical fatigue specific loads in downstream is limited to a range extending from zero up to the corresponding fatigue limit, including both endpoints

$0 \leq \sigma_D \leq \sigma_{max}$

A geometric constraint defines the proportion of the heights:

$H_{dam} \geq X_2 + X_4 + X_6$

A geometric constraint is defined on the vertical coordinates:

$X_4 + X_6 \leq X_8$

The boundary conditions on design variables, expressed in meters, are imposed as follows:

$5 < X_1 < 20, 20 < X_2 < 50{,}1 < X_3 < 20, 40 < X_4 < 80, 10 < X_5 < 30, 30 < X_6 < 80,$

$20 < X_7 < 40, 50 < X_8 < 150, 80 < X_9 < 150.$

8.7 Civil Engineering: Two hundred bar truss design

This design example represents a classic medium-scale truss weight minimization problem, which has been addressed by numerous researchers utilizing both gradient-based and metaheuristic algorithms. It pertains to a planar truss structure consisting of 200 bars and 77 nodes, as illustrated in the following drawing:

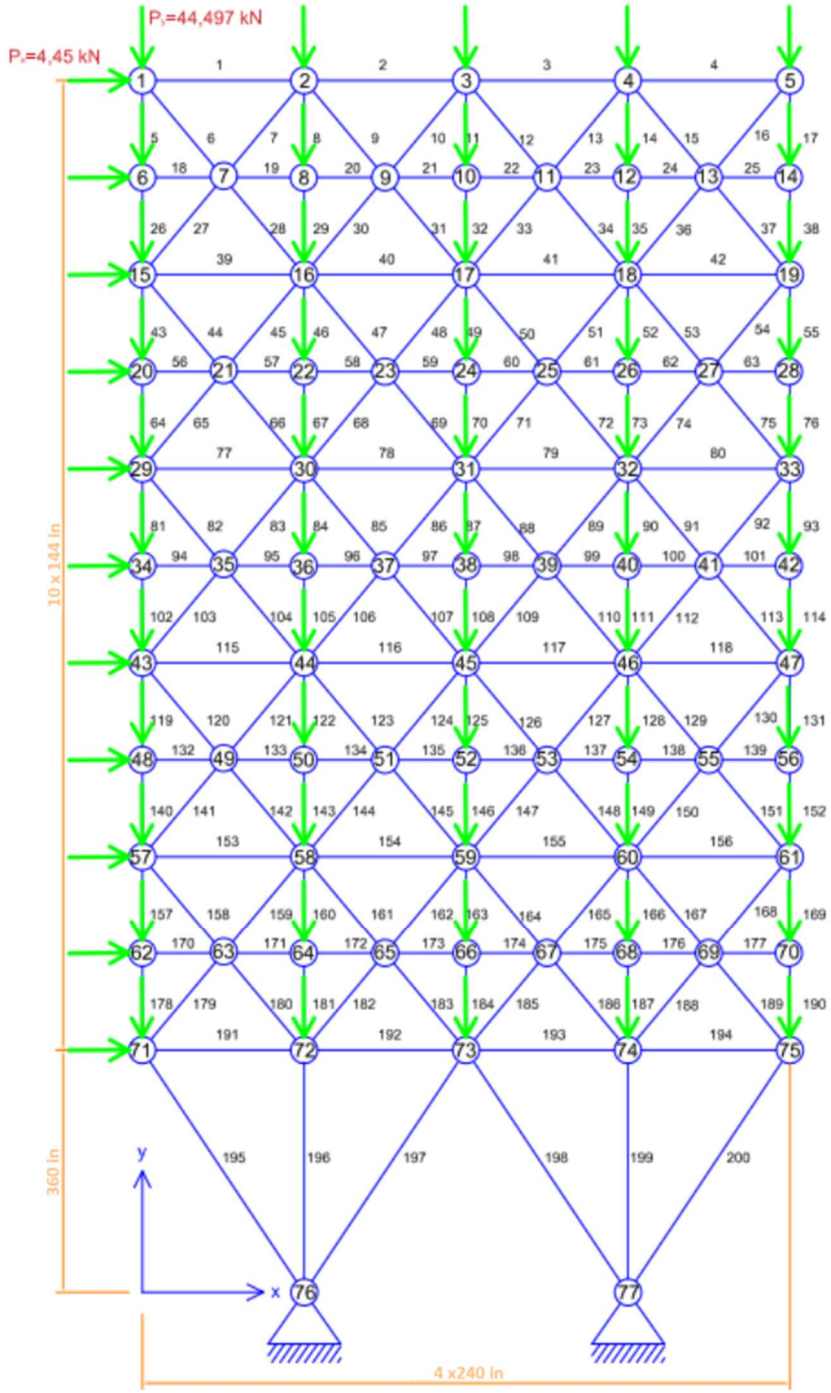


Figure 2. Scheme of the planar 200-bar truss structure and applied loads

This problem involves finding the optimal combination of 29 variables under 1200 nonlinear constraints to minimize the total weight of the structure while ensuring that all elements operate within safe stress limits.

The 29 design variables correspond to the cross-sectional area of 29 groups of bars with identical geometric characteristics that make up the structure.

The identified 29 groups of elements are specified in the following table:

Table 1 Groups of Elements in the Planar 200-Bar Truss Structure

GROUP	ELEMENTS	GROUP	ELEMENTS
1	1, 2, 3, 4	16	82, 83, 85, 86, 88, 89, 91, 92, 103, 104, 106, 107, 109, 110, 112, 113
2	5, 8, 11, 14, 17	17	115,116,117,118
3	19, 20, 21, 22, 23, 24	16	119, 122, 125, 128, 131
4	18, 25, 56, 63, 94, 101, 132, 139, 170, 177	19	133, 134, 135, 136, 137, 138
5	26, 29, 32, 35, 38	20	140, 143, 146, 149, 152
6	6, 7, 9, 10, 12, 13, 15, 16, 27, 28, 30, 31, 33, 34, 36, 37	21	120, 121, 123, 124, 126, 127, 129, 130, 141, 142, 144, 145, 147, 148, 150, 151
7	39, 40, 41, 42	22	153, 154,155, 156
8	43,46,49,52,55	23	157, 160, 163, 166, 169
9	57,58,59,60,61,62	24	171, 172, 173, 174, 175, 176
10	64, 67, 70, 73, 76	25	178, 181, 184, 187, 190
11	44, 45, 47, 48, 50, 51, 53, 54, 65, 66, 68, 69, 71, 72, 74, 75	26	158, 159, 161, 162, 164, 165, 167, 168, 179, 180, 182, 183, 185, 186, 188, 189
12	77, 78, 79, 80	27	191, 192, 193, 194
13	81, 84, 87, 90, 93	28	195, 197, 198, 200
14	95, 96, 97, 98, 99, 100	29	196,199
15	102, 105, 108, 111, 114		

The structure is subjected to three types of loads:
(a) 4.45 kN (i.e. 1000 lbf) acting in the positive X-direction at node points 1, 6, 15, 20, 29, 34, 43, 48, 57, 62 and 71.
(b) 44.497 kN (i.e. 10,000 lbf) acting in the negative Y-direction at node points 1, 2, 3, 4, 5, 6, 8, 10, 12, 14, 15, 16, 17, 18, 19, 20, 22, 24, 26, 28, 29, 30, 31, 32, 33, 34, 36, 38, 40, 42, 43, 44, 45, 46, 47, 48, 50, 52, 54, 56, 57, 58, 59, 60,61,62,64, 66, 68, 70, 71, 72, 73, 74, 75;
(c) Loading conditions (a) and (b) acting together.
The design parameters are determined using the following values: truss width dimx=4·240 in (~24,384 m), bar truss height dimy=10·144 in+360 in=1800 in (~ 45,720 m), Young Modulus $E = 2.06913 \cdot 10^{11}\, Pa$, density $\rho = 7833.412832\ \frac{Kg}{m^3}$, ultimate tensile strength (UTS) $\sigma_{lim} = 10000\, psi$, allowable tensile stress $\sigma_t = 6897.11257626 \cdot 10^5\ \frac{N}{m^2}$, allowable compression stress $\sigma_c = -6897.11257626 \cdot 10^5\ \frac{N}{m^2}$.
The minimization of the total weight of the truss structure is defined with some boundary conditions, as follows:

$$Min\left(F_{obj}(x_i)\right) = Min\left(\rho \cdot \sum_{j=1}^{N_{el}} A_j \cdot L_j\right)\ \ with\ i = 1,2, \dots 29\ and\ N_{el} = 200 \qquad (4)$$
$$0{,}1\ in^2 \leq A_i \leq 100\ in^2\ with\ i = 1,2, \dots 29$$

where A is the cross-sectional area, L is the length of the single element, and N_{el} the total number of elements.
The truss is composed of 200 elements, with each element having 2 types of constraints (compression and tension). Additionally, each node in the plane has 3 load conditions, totalling 1200 constraints.

The constraints are defined to ensure that the stresses in each truss element do not exceed the permissible values for both compression σ_c and tension σ_t, as follows:

$$G(k) = \frac{\sigma_j}{\sigma_c} - 1 \leq 0 \qquad with\, j = 1,2,\ldots,200\ and\ k = 1,2,\ldots 1200 \tag{5}$$
$$G(k) = \frac{\sigma_j}{\sigma_t} - 1 \leq 0 \qquad with\, j = 1,2,\ldots,200\ and\ k = 1,2,\ldots 1200$$

where σ_j represents the stress in element j under a specific loading condition:

$$\sigma_j = \frac{P_j}{A_j} \qquad with\, j = 1,2\ldots,200$$

Displacements for each node are calculated using the formula that relates the thermal stiffness matrix and the global stiffness matrix to them, as follows:

$$\overline{P} = (k + k_{thermal}) \cdot \overline{u}$$

where P is the vector of applied forces, k is the global stiffness matrix

$$k = IG_{glob_j} = \begin{bmatrix} \cos^2\theta_x & \cos\theta_x \sin\theta_y & -\cos^2\theta_x & -\cos\theta_x \sin\theta_y \\ \cos\theta_x \sin\theta_y & \sin^2\theta_y & -\cos\theta_x \sin\theta_y & -\sin^2\theta_y \\ -\cos^2\theta_x & -\cos\theta_x \sin\theta_y & \cos^2\theta_x & -\cos\theta_x \sin\theta_y \\ -\cos\theta_x \sin\theta_y & -\sin^2\theta_y & \cos\theta_x \sin\theta_y & \sin^2\theta_y \end{bmatrix} \cdot \frac{E \cdot A_j}{\dim e\, l_j}$$

and k_{therm} is the global thermal stiffness matrix

$$k_{thermal} = IGTCDEL_j = \begin{bmatrix} \cos(\theta_x) & \sin(\theta_y) & -\cos(\theta_x) & -\sin(\theta_y) \\ -\sin(\theta_x) & -\cos(\theta_y) & \sin(\theta_x) & \cos(\theta_y) \end{bmatrix} \cdot \frac{E \cdot A_j}{\dim e\, l_j}$$

where θ_x and θ_y are the angle between the truss element and the x-axis or the y-axis, and $\dim e\, l_j$ is the length of the truss element j.