Design optimization of large-scale structures with advanced meta-heuristic methods

Politecnico di Bari

Department of Mechanics, Mathematics and Management
MECHANICAL AND MANAGEMENT ENGINEERING

Ph.D. Program

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

Final Dissertation

Design optimization of large-scale structures with advanced meta-heuristic methods

by

Seyed Mohammad MINOOEI MAZLOOM

Supervisor:

Prof. Luciano Lamberti

Coordinator of Ph.D. Program:

Prof. Giuseppe Pompeo Demelio

Course n°34, 01/11/2018-31/01/2022

Politecnico di Bari

Department of Mechanics, Mathematics and Management
MECHANICAL AND MANAGEMENT ENGINEERING

Ph.D. Program

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

Final Dissertation

Design optimization of large-scale structures with advanced meta-heuristic methods

by

Seyed Mohammad MINOOEI MAZLOOM

Referees:

Prof. Francesco Panella

Prof. Sadik Ozgur Degertekin

Supervisor:

Prof. Luciano Lamberti

Coordinator of Ph.D. Program:

Prof. Giuseppe Pompeo Demelio

Course n°34, 01/11/2018-31/01/2022

Table of Contents

Acknowledgments

At the end of my PhD course in mechanical engineering I feel the need to thank many people that contributed at different extent to bring me to this point.

First of all, I would like to thank my beloved family for their loving support and unsteady encouragement during the three years of my doctoral studies. They always remained close to me even if the physical distance between us was some thousands of kilometers. I am very happy and incredibly proud to have been born in my family.

I also would like to thank Politecnico di Bari (PoliBa) for having given me the opportunity of attending my PhD studies in Mechanical Engineering in an internationally reputed department such as the Department of Mechanics, Mathematics and Management (DMMM) of PoliBa. The support and guidance given to me over these three years by the PhD program coordinator, Prof. Giuseppe Pompeo Demelio, is gratefully acknowledged. The support of the admistrative staffs of the Doctorate School of PoliBa and DMMM and their help in successfully dealing with the many bureaucratic intricacies of university life also is acknowledged.

Very special thanks to my supervisor, Prof. Luciano Lamberti. He has introduced me into the challenging and fashinating world of Structural Optimization and involved me also in the wonderful field of cell mechanics. His impressive expertise in research, deep knowledge in many fields, and high dedication to students will be a constant inspiration for me should I decide to have a career either in the industry or in the academy.

I would like to thank Prof. Benedikt Kriegesmann of the Technical University of Hamburg (TUHH), Germany. I worked from December 2019 to May 2021 in the Structural Optimization for Lightweight Design group coordinated by Prof. Kriegesmann. Although the COVID-19 pandemic greatly limited my operation at TUHH and most of tasks originally planned had to be suppressed, Prof. Kriegesmann was a true gentleman and helped me a great deal to expand my knowledge on structural behavior of shell structures.

Last, but not least, I would like to thank my colleagues (especially Mr. Lorenzo Santoro) who shared with me these years spent attending the PhD at PoliBa.

Abstract:

Metaheuristic algorithms nowadays represent the standard approach to engineering optimization. A very challenging field is large scale structural optimization entailing hundreds of design variables and thousands of nonlinear constraints on element stresses and nodal displacements. However, a very few studies documented the use of metaheuristic algorithms in large scale structural optimization. In order to fill this gap, an enhanced hybrid harmony search (HS) algorithm for weight minimization of large scale truss structures is presented in this PhD dissertation. The new algorithm, Large Scale Structural Optimization – Hybrid Harmony Search JAYA (LSSO–HHSJA), developed here combines a well established method like HS with a very recent method like JAYA, which has however the simplest and inherently most powerful search engine amongst metaheuristic optimizers. All stages of LSSO-HHSJA are aimed at reducing the number of structural analyses required in large scale structural optimization. The basic idea is to move along descent directions to generate new trial designs: directly through the use of gradient information in the HS phase, indirectly by correcting trial designs with JA-based operators that push search towards the best design currently stored in the population or the best design included in some local neighborhood of the currently analyzed trial design. The proposed algorithm is tested in three large scale weight minimization problems of truss structures. Optimization results obtained for the three benchmark examples with up to 280 sizing variables and 37374 nonlinear constraints prove the efficiency of the proposed LSSO–HHSJA algorithm, which is very competive with other HS and JAYA variants as well as with commercial gradient-based optimizers. The possibility of using the same hybridization strategy for another metaheuristic algorithm such as Big Bang–Big Crunch is also investigated by solving two highly nonlinear design problems including up to 84 variables, (i) shape optimization of a concrete dam and (ii) discrete layout optimization of a planar steel frame, to prove the feasibility of the proposed approach.

Keywords: *Large scale structural optimization*; *Metaheuristic algorithms*; *Harmony search*; *Big Bang-Big Crunch; JAYA*; *Truss structures*; *Sizing and shape optimization*.

Outline of the dissertation

The dissertation focuses on large-scale weight minimization of 2D and 3D truss structures. This is a relevant topic in structural design in view of the complexity introduced in the optimization process by the presence of hundreds of variables and thousands of constraints. Gradient-based optimizers may get stuck at local optima thus missing the global optimum; this is very likely to happen when the design space is non-convex, especially if the optimization problem includes many design variables. Furthermore, computational cost entailed by gradient evaluation is often unaffordable in the case of large-scale problems. Metaheuristic algorithms are becoming the standard approach to structural optimization and may outperform gradient-based algorithms in terms of their inherent ability of finding the global optimum if the search engine is properly selected. Basically, the number of structural analyses entailed by the metaheuristic search must not exceed the number of structural analyses entailed by gradient evaluation.

This dissertation follows an "intermediate" strategy attempting to enhance the global search ability of metaheuristic algorithms with the local search ability of gradient-based algorithms. Feasible or almost feasible descent directions are always generated taking care to limit the number of structural analyses yet moving rapidly towards the best regions of design space. The harmony search optimization algorithm and the JAYA optimization algorithm are combined into a new metaheuristic optimizer, the LSSO–HHSJA algorithm. LSSO–HHSJA is successfully tested on three large-scale sizing optimization problems of truss structures including up to 280 design and 37374 nonlinear constraints.

The possibility of using the same hybridization strategy for another metaheuristic algorithm such as Big Bang–Big Crunch is also investigated by solving two highly nonlinear design problems including up to 84 variables, (i) shape optimization of a concrete dam and (ii) discrete layout optimization of a planar steel frame, to prove the feasibility of the proposed approach.

The dissertation is structured as follows. The Introduction gives a general overview of metaheuristic optimization and illustrates the motivations, rationale and employed methodologies of this research work. Section 2 reviews the most common metaheuristic algorithms for structural optimization. Section 3 recalls the basic formulations of HS and JAYA used as start point for the present study. Section 4 describes the new hybrid algorithm LSSO–HHSJA developed here. Section 5 recalls the formulation of the truss sizing design

problem, outlines the implementation of the optimization process, describes the three benchmark problems and discusses optimization results. Finally, Section 6 summarizes the main findings of this study and outlines directions of future research. Details of structural modeling for the spatial towers optimized in the dissertation are presented in Appendix A. Appendix B describes the other hybrid formulation combining Big Bang–Big Crunch and JAYA and reports the optimization results obtained for the shape optimization of a concrete dam and the discrete layout optimization of a planar steel frame.

CHAPTER 1

INTRODUCTION

Metaheuristic optimization methods inspired by evolution theory, life sciences and zoology, physics and astronomy, human sciences etc are successfully utilized in science and engineering. For example, genetic algorithms (GA) [1], differential evolution [2], simulated annealing (SA) [3], particle swarm optimization (PSO) [4], ant colony optimization (ACO) [5], firefly algorithm (FFA) [6], cuckoo search (CS) [7], ant lion optimizer [8], tabu search (TS) [9], harmony search (HS) [10], teaching-learning based optimization (TLBO) [11], JAYA [12], big bang–big crunch (BBBC) [13], charged system search (CSS) [14], ray optimization (RO) [15], colliding bodies optimization [16], water evaporation optimization (WEO) [17], cyclical parthenogenesis algorithm (CPA) [18], and coyote optimization algorithm (COA) [19] are representative metaheuristic methods with several variants documented in optimization literature.

Generally speaking, optimization algorithms may be trajectory-based or population-based. In the former case, a single design is elaborated in each iteration thus building a "trajectory" connecting the initial solution with the optimal solution and passing through all intermediate solutions found in each iteration. Trajectory-based algorithms include gradient-based methods using linear or quadratic approximations of the optimization problem as well as some metaheuristic algorithms such as, for example, simulated annealing, randomized local search and simplified variants of differential evolution. Population-based algorithms update a population of candidate solutions until the search process converges to the optimum design after a predefined number of iterations or function evaluations. This category practically includes all metaheuristic algorithms except those classified above as trajectory-based algorithms.

While metaheuristic algorithms practically became the standard approach to engineering optimization (see, for example, the review articles and books [20-24] focusing on structural optimization), the "no free lunch" theorem [25,26] had an important consequence in the fact that no metaheuristic algorithm can prove itselfsuperior over all other algorithms in all problems. A general-purpose universal optimization algorithm does not exist by the theoretical point of view. However, an algorithm can outperform its competitors if is specialized to the specific problem at hand. For this reason, most of the metaheuristic algorithms lost their appeal just after a very few years. This is not the case of Harmony Search (HS) [10], which was developed almost 20 years ago butremains a very popular optimization algorithm.

The HS method is a population-based metaheuristic algorithm that simulates the process of searching for a perfect state of harmony performed by jazz players. The

relationship of HS to music is well summarized in Yang [27]. Three possible options are available to a skilled musician who is improvising: (1) play any famous piece of music (a series of pitches in harmony) exactly from his or her memory; (2) play something similar to a known piece (thus adjusting the pitch slightly); (3) compose new or random notes. These three options were formalized by Geem *et al*. [10] into the harmony search optimization algorithm. A set of N_{POP} randomly generated solutions are stored in a matrix called harmony memory [HM]. New trial designs may be extracted from [HM] or randomly selected accordingto the value taken by the harmony memory considering rate (*HMCR*) parameter. Design variables may be modified according to the value taken by the pitch adjustment rate (*PAR*) parameter; the bandwith parameter (*bw*) quantifies the amount of variation given to a variable in the pitch adjustment operation.

Structural optimization is an important field of engineering concerned with the optimum design of structures [28–30]. The most common goal in structural optimization is to minimize the weight of the structure in presence of limitations on deformations, stresses, critical loads, natural frequencies etc. However, other objectives can be considered such as, for example, to minimize stresses or maximize stiffness. Structural optimization problems may be of three types: (i) sizing optimization where design variables correspond to geometric dimensions such as, for example, the cross-sectional areas of the elements forming the structure; (ii) shape optimization where design variables define the profile of the structure (e.g. coordinates of nodes); (iii) topology optimization where design variables define thedistribution of material in the structure.

The easiness of implementation of HS soon attracted many structural optimization experts that developed several variants of the algorithm after the pioneering studies by Lee and Geem [31,32]. In particular, researchers attempted to (i) minimize sensitivity of convergence behavior to the setting of *HMCR*, *PAR* and *bw* parameters; (ii) reduce the number of structural analyses and speed up the search processby removing trial solutionsyieldingnoimprovements in design. These abilities turn very useful especially in optimization of large-scale structures. For example, Saka [33] used an adaptive error strategy to handle slightly infeasible designs. Maheri and Narimani [34] considered also designs that are worse than the worst design stored in [HM] yet distant from local optima. Murren and Khandelwal [35] generated random trial solutions within intelligently specified search neighborhoods. Mahdavi *et al*. [36] dynamically updated the *PAR* and *bw* parameters in the search process, while Carbas and Saka [37] used another dynamic scheme for updating the *HMCR* and *PAR* parameters. Hasancebi *et al*. [38] probabilistically selected HS parameters

to adapt search to varying features of design space. A self-adaptive HS algorithm was implemented also by Degertekin [39]. Kaveh and Naiemi [40] developed a multi-adaptive HS variant updating internal parameters linearly or exponentially. Geem and Sim [41] developed a parameter-setting-free technique selectingspecific operators for each variable stored in [HM]. Turky and Abdullah [42] developed a multi-population approach where each sub-population operates on a different region of search space. Finally, HS was hybridized with other metaheuristic methods [43-48], gradient-based optimizers [49] and approximate line search strategiesexploiting explicitly available gradient information [50-52]. In a very recent study, Ficarella *et al*. [53] synthetized all approaches mentioned above by combining the HS metaheuristic engine with search direction mechanisms, gradient information based search and 1-D simulated annealing search.While the different studies published in the literature considered structures of increasing complexity over the years, it has to be noted that only Ref. [51] directly deals with large-scale structural optimization problems including more than 250 sizing variables.

The JAYA algorithm was developed by Rao [12] in 2016. This population-based method relies on the simplest search strategy ever implemented in metaheuristic optimization: trial solutions are generated always moving towards the best design and away from the worst design of the population. Remarkably, JAYA needs only two standard control parameters such as population size and limit number of iterations. The very simple formulation and inherently efficient search strategy explain why JAYA is probably the most exploited metaheuristic algorithm in the last few years. However, while JAYA often achieves a high success rate in finding the global optimum regardless of population size and initial solutions, the number of required analyses is not always significantly lower than those reported for the other state-of-the-art metaheuristic methods (see, for example, Ref. [53]). In this regard, Degertekin *et al*. [54–56] developed an efficient JAYA variant for weight minimization of truss structures trying to avoid unnecessary structural analyses that would not yield design improvements. While this enhancement made JAYA suitable also for structural optimization problems, since only one test case in [54–56] included more than 200 sizing variables,similar to HS, also JAYA's performance in large-scale structural optimization should further be investigated.

The above arguments indicate that a very few studies focused on the use of metaheuristic algorithms in large scale structural optimization problems. The same conclusion can be drawn for all methods besides HS and JAYA. In order to overcome this limitation, a novel hybrid harmony search – JAYA algorithm for large scale sizing

optimization of truss structures is described in this PhD dissertation. As mentioned above, both HS and JAYA are highly attractive algorithms for the engineering community: the former has a well established practice over 20 years while the latter has a inherently powerful search engine. Here, these algorithms are combined in order to solve large scale structural optimization problems where it is essential to limit the total number of structural analyses required by the search process as each analysis is computationally expensive. The HS engine is the backbone of the proposed algorithm but trial designs and search directions generated in the HS phase are enhanced by the JAYA strategy. Explicit gradient information available from the formulation of the truss optimization problem are utilized to perturb design.

The new algorithm, denoted as LSSO–HHSJA (i.e. *Large Scale Structural Optimization - Hybrid Harmony Search JAYA*), is in essence an enhanced hybrid HS algorithm with multiple line searches, which uses the JAYA search strategy to minimize the number of structural analyses required in the optimization process. This makes LSSO–HHSJA suitable for computationally expensive large-scale structural optimization problems including multiple loading conditions, hundreds of variables and thousands of nonlinear constraints on nodal displacements and element stresses. In large-scale structural optimization, metaheuristic algorithms are competitive with gradient-based algorithms as long as the number of structural analyses per design cycle $N_{AN\text{-}cycle}$ is significantly smaller than the number of optimization variables NDV. In metaheuristic optimization, $N_{AN\text{-}cycle}$ is usually equal to or at most twice the population size N_{POP}. In gradient-based optimization, $N_{AN\text{-}cycle}$ is at least equal to $(NDV+1)$ if one assumes that forward finite differences are used for computing gradients of nonlinear constraints with respect to design variables. A vast number of studies demonstrated that population-based metaheuristic optimizers can find global optima or nearly global optimum designs even if $N_{POP} << NDV$. Basically, by improving the quality of the trial designs generated in each iteration, it is possible to efficiently search the design space even with up to one order of magnitude smaller populations than the number of design variables. Such a condition is satisfied also by the formulation of the proposed LSSO–HHSJA algorithm.

In summary, the proposed LSSO–HHSJA formulation attempts to reduce the number of structural analyses entailed by optimum design of large scale structures. For this purpose, trial designs are generated by perturbing optimization variables along descent directions where it is very likely to reduce cost function in a fast way. Such a goal is accomplished both directly, using explicitly available gradient information in the HS phase, and indirectly,

correcting trial designs with JA-based operators that push search towards the best designs included in the current population or in a neighborhood of the currently analyzed trial design.

The LSSO–HHSJA algorithm is tested in three large scale weight minimization problems of truss structures: (i) a planar 200-bar truss subject to five independent loading conditions, optimized with 200 sizing variables and 3500 nonlinear constraints; (ii) a spatial 1938-bar tower subject to three independent loading conditions, optimized with 204 sizing variables and 20700 nonlinear constraints; (iii) a spatial 3586-bar tower subject to three independent loading conditions, optimized with 280 sizing variablesand 37374 nonlinear constraints. Truss structures are pin-connected skeletal structures very often selected by designers for testing novel structural optimization algorithms.

Optimization results demonstrate the validity of the proposed approach: LSSO–HHSJA is very competitive with other HS and JAYA variants as well as commercial gradient-based optimizers.

The possibility of using the same hybridization strategy for another metaheuristic algorithm such as Big Bang–Big Crunch (BBBC) is also investigated by solving two highly nonlinear design problems including up to 84 variables, (i) shape optimization of a concrete dam and (ii) discrete layout optimization of a planar steel frame, to prove the feasibility of the proposed approach. BBBC has been selected for hybridization with line search and JAYA-based strategies because it is a well established metaheuristic algorithm widely used in structural optimization and has a simpler formulation than HS. However, computational efficiency of BBBC is affected by the considerably high computational cost of the explosion phase.

CHAPTER 2

METAHEURISTIC ALGORITHMS FOR STRUCTURAL OPTIMIZATION

2.1. Overview

Structural optimization problems may entail highly non-convex and nonlinear design spaces with hundreds of design variables and constraints. Random generation of trial designs allows in principle to explore larger fractions of the search space than the local approximations built in the case of gradient-based optimization. However, purely random search often results in a large number of analyses yielding just marginal improvements in design or may even generate unfeasible trial designs. In order to rationalize the search process, metaheuristic optimization methods have been developed inspired by some principle of theory of evolution, medicine, biology and zoology, physics and astronomy, human sciences etc. These methods have been successfully utilized practically in every field of science and engineering.

The basic difference between metaheuristic optimization algorithms and gradient-based optimization algorithms is that the latter methods have only a local search capability. In fact, they analyze candidate solutions of the optimization problem in a neighborhood of the selected point at which gradients are evaluated. By increasing step size it is possible to consider a larger neighborhood but the quality of the approximation may decay as we move far from the base-point of the approximation. Conversely, metaheuristic algorithms do not use gradient information as they explore randomly the design space. The step size given to each design variable is a randomly defined fraction of the whole interval of variation of the currently perturbed variable (x^U–x^L). Hence, the optimizer is free to "explore" the whole design space without being limited to search in the neighborhood of a given solution. This process is named as "exploration" in the metaheuristic optimization terminology. The global search carried out in the exploration phase must be combined by a local search in the neighborhood of the best solutions identified by exploration. This process is the "exploitation" stage of metaheuristic optimization. The efficiency of a metaheuristic algorithm directly relates to how good is the balance of exploration and exploitation. Exploration dominates in the early stage of the search process and is progressively replaced by exploitation as the global optimum is approached towards the end of the optimization process. However, exploitation may be restored every time search stagnates about local optima or design is not improved over many iterations.

Another difference between metaheuristic algorithms and gradient-based algorithms is the fact that the former methods usually operate on a population of candidate designs while the latter methods develop just one design in each iteration. The efficiency of metaheuristic search strongly depends on how population is updated in each optimization iteration and convergence speed is affected by the ability of the optimizer to quickly update population

with high quality designs. In this context, the optimizer should possess the capability of bypassing local minima.

Genetic algorithms (GA) and evolution strategies (ES) [2,57–59] and simulated annealing (SA) [3,60] were among the first metaheuristic optimization methods to be developed and are still widely utilized nowadays. The basic difference between GA and SA is that the former algorithm operates with a population of candidate designs while the latter algorithm, at least in its classical implementation, considers one trial design at a time and then further develops it.

Swarm intelligence and colony based algorithms mostly inspired by animals' behavior are other population-based algorithms developed since early '1990s. They still gather the attention of optimization experts that keep proposing new algorithms: the most popular methods are particle swarm optimization (PSO) [4,61], ant colony optimization (ACO) [5,62,63], artificial bee colony (ABC) [64], firefly algorithm (FFA) [6,65], hunting search (HuS) [66], eagle strategy (ES) [67], bat algorithm (BA) [68,69], dolphin echolocation [70], cuckoo search (CS) [7], ant lion optimizer [8], bacterial/social foraging (BFA) [71,72], crow search [73], fruit fly optimization [74].

Progress in medicine and cell biology also led to develop new metaheuristic algorithms in the last fifteen years [75,76]. Social sciences and human activities are since almost 20 years another important source of inspiration for metaheuristic algorithms yet not so popular as swarm intelligence methods: among others, we can mention tabu search (TS) [9], harmony search (HS) [10], imperialist competitive algorithm (ICA) [77], teaching-learning based optimization (TLBO) [11,78,79], mine-blast algorithm (MBA) [80] and search group algorithm (SGA) [81].

Astronomy, physics (electromagnetism, optics, classical mechanics etc.) and natural phenomena provided another prolific field of inspiration especially in the last two decades: for example, big bang–big crunch (BBBC) [13], gravitational search algorithm (GSA) [82], charged system search (CSS) and magnetic charged system search (MCSS) [14,83], ray optimization (RO) [15], colliding bodies optimization (CBO) [16]; chaos theory was embedded at various extents in the original PSO, HS, BBBC and FFA formulations [84–87]; water cycle algorithm (WCA) [88], flower pollination algorithm (FPA) [89], water evaporation optimization (WEO) [17], thermal exchange optimization (TEO) [90], and cyclical parthenogenesis algorithm (CPA) [18].

Comprehensive reviews of the applications of metaheuristic optimization algorithms to structural design problems with critical assessment of the relative merits of different methods

can be found in [20–24,91]. A number of studies also compared the performance of metaheuristic algorithms in specific optimization problems such as large-scale and real size truss structures and steel frames [92–94,98], reinforced concrete frames [95], cellular beams [96], layout optimization [97] and optimization with frequency constraints [99].

The literature survey made in this introduction indicates that metaheuristic algorithms became the standard approach to solving engineering problems, for example structural optimization problems. The easiness of implementation and the rapidly increasing available computational power allows to run population-based algorithms also on standard computers. Since practically all metaheuristic algorithms utilize a population of candidate designs, the main drawback of these methods is the large number of function evaluations/structural analyses required in the search process. In order to overcome this limitation, researchers have developed hybrid formulations either combining two or more metaheuristic algorithms [100–108], or metaheuristic search with gradient based optimization [49] and approximate line search [109].

However, according to the "no free lunch" theorem [25,26], there is no metaheuristic algorithm that always outperforms all other algorithms in all optimization problems. Unlike gradient-based optimizers that are still used as optimization tools in commercial software although their formulations have remained substantially unvaried for almost 30 years, most of the newly developed metaheuristic algorithms added very little to the optimization practice and their appeal quickly vanished after a very few years. This suggests that rather than proposing a new metaheuristic algorithm that improves the existing methods just marginally it is better to significantly improve the most common algorithms trying to make them competitive with gradient-based optimizers in terms of computational cost yet preserving the inherent ability of metaheuristic optimizers to globally explore design space.

From the large number of papers published in the last decade (just to limit the analysis to recent years) and analyzing the number of citations gathered by them in the technical literature, it appears that PSO, GA, SA, HS and BBBC are amongst the most popular metaheuristic algorithms in structural optimization. Critical comparisons of these optimization engines reveal that SA is the only metaheuristic method inherently capable to bypass local optima. However, HS and BBBC include very important features that should be possessed by any population-based algorithm. HS stores all candidate designs (i.e. those forming the population and additional designs kept in memory from previous iterations) in a memory. Values assigned to optimization variables can be extracted from this memory to form new trial designs. This allows to carry out an adaptive search yet avoiding stagnation.

BBBC utilizes the concept of center of mass, which makes it possible to follow the evolution of the average characteristics of the population in the optimization process. GA and PSO instead may suffer from premature convergence, stagnation, sensitivity to problem formulation. Furthermore, GA and PSO include more internal parameters than SA, HS and BBBC, which increases the amount of heuristics in the optimization process.

Besides computational efficiency driven by (i) the good balance between exploration and exploitation phases, and (ii) ability to update population with high quality designs, simple formulation and easiness of implementation are two very important issues that may drive users towards utilizing a specific metaheuristic algorithm. In this regard, JAYA [12] probably relies on the simplest search strategy ever implemented in metaheuristic optimization: trial solutions are generated always moving towards the best design and away from the worst design of the population. Furthermore, JAYA needs only two standard control parameters such as population size and limit number of iterations. These features made JAYA become one of the most exploited metaheuristic algorithms with an exponentially growing number of applications in the last few years. However, setting of relative magnitude of perturbations for design variables to escape from the worst solution and approach the best solution remains an open issue.

The most relevant metaheuristic algorithms used in structural optimization (i.e. GA, SA, PSO, ACO, BBBC, HS and JAYA) are revised in the rest of this chapter. As mentioned before, harmony search and JAYA have been combined into the LSSO–JAYA algorithm described in this dissertation.

2.2. Genetic Algorithms (GA) and Evolution Strategies (ES)

2.2.1. *Genetic Algorithms (GA)*

Genetic Algorithms (GA) [1,93,94,110] rely on concepts of genetics and Darwinian survival of the fittest. The design is represented by a combinatorial set, called chromosomes and each component of the set is called a gene. Chromosomes and genes are generated by three predefined rules of evolution: selection/reproduction (rebirth or duplication), cross-over (generation) and mutation. Thus, one must define the population size, rate of reproduction, cross-over pattern, percentage of mutation etc. After many generations, the design represented by the most popular chromosomes indicates the optimal design. The standard formulation of GA is now recalled.

- *Generation of the initial population*. Design variables are encoded as binary substrings of finite length formed by 0 and 1. Real coding can be adopted in order to deal with continuous optimization variables. For an optimization problem with NDV design variables, each substring corresponds to an optimization variable. The NDV substrings are joined together to form a chromosome (i.e. individual), that is a trial design vector. An initial population comprised of N_{POP} individuals is randomly generated trying to sweep the whole design space the most as possible.

- *Evaluation of candidate designs and fitness*. Cost function and optimization constraints are evaluated for each individual (i.e. candidate design) included in the population. Candidate designs are hence ranked in terms of their fitness score $f_k=F_{max}/F_k$ where F_{max} is the maximum cost evaluated amongst all individuals and F_k is the cost evaluated for the k^{th} individual, respectively. If the candidate design currently evaluated is infeasible, the corresponding cost is computed by adding to the cost function a penalty term proportional to the constraint violation (see, for example, Refs. [93,94,110–112]). Individuals' fitnesses must be scaled to eliminate the dominance of highly fit individuals during the selection process.

- *Selection and reproduction*. The fittest individuals are selected and reproduced while the least fit individuals are eliminated. This leads to create an intermediate population called mating pool. The number of reproductions η_k for the k^{th} individual can be determined in principle as $\eta_k= F_k'/F_{ave}$ where F_k' and F_{ave} are respectively the scaled fitness of the k^{th} individual and the average scaled fitness. In practical implementations of GA, the roulette wheel selection approach is utilized: individuals are assigned slots on a simulated roulette wheel according to their scaled fitness. The roulette is then spun N_{POP} times to select and reproduce the elements of the mating pool.

- *Crossover*. Selected and reproduced individuals are mated randomly to form $N_{POP}/2$ pairs and crossover is performed for each pair given a value of crossover probability P_c. Crossover swaps genetic information between parents thus forming two child individuals. In the classical two-point crossover, mating individuals are cut at two randomly selected crossover sites and string portions are exchanged.

- *Mutation*. This operator is applied to the genes of child individuals in order to preserve the best characters of the population of candidate designs thus avoiding the loss of important genetic information. Furthermore, the mutation operation allows to avoid

stagnation between a limited number of dominant individuals thus preventing new individuals to always repeat the same characters of their parents. From the stand point of algorithm implementation, some genes of 1, randomly selected in a string, are switched to 0 or vice versa (from 0 to 1) according to the specified gene-wise mutation probability P_m which is usually chosen less than 0.01.

- *Termination*. The above described steps are repeated until a maximum number of structural analyses is performed or the cost function does not improve significantly over a certain number of optimization cycles.

Genetic algorithms were the first meta-heuristic optimization method utilized in design problems of skeletal structures such as spatial frames and trusses (see, for example, [113−118]). Starting from the classical Bit-string GA (BGA) formulation previously recalled in this section which can deal with discrete/integer design variables and has global optimization capability, Real-coded Genetic Algorithms (RGA) were developed in order to deal with continuous variables like those involved, for example, in layout or topology optimization problems.

Over the years, several improvements were introduced in the basic GA formulation to enhance the global search capability and the convergence behaviour of this algorithm by reducing the number of structural analyses required in the optimization process. Most of the improvements to GA were proposed by researchers working in general optimization and were then adapted to structural optimization problems. The simplest strategy is to perform a local search (i.e. gradient-based) starting from the optimized design found via GA. A similar strategy is to select the best individuals of the current population and carry out local searches in the neighbourhood of those individuals. Besides these hybridization schemes, GA can be combined with other meta-heuristic optimization algorithms. For example, hybridization with Simulated Annealing [100] can provide RGA with hill-climbing capability which instead is inherently possessed by SA. However, care should be taken in choosing the cooling schedule that governs the SA search.

Crossover and mutation operations can be speeded up by perturbing design variables according to some ad hoc probability distribution or/and by reducing the range of variability of the optimization variables from which the new value assigned to the trial design can be selected as the optimization process progresses. The generation of new trial designs can also be "directed" either by choosing a set of directions defined by some individual of the population and all individuals better than it or by considering randomly two individuals and

the direction from the worse individual to the better individual. It can be seen that any new trial design is thus "forced" to lie on a direction where cost function will in all likelihood improve. However, care must be taken in not reducing too quickly the range of variability of design variables. Furthermore, the "directing" strategy may involve constraint evaluation and continuous rating of individuals currently included in the population.

2.2.2. *Evolution Strategies (ES) and Differential Evolution (DE)*

Genetic Algorithms fall in the more general field of the so-called Evolution Strategies [2,58,119,120]. This term indicates techniques that reproduce the natural evolution mechanisms to evolve a population of individuals (i.e. designs) towards the optimum conditions over successive generations. These algorithms, originally developed for continuous optimization problems, can efficiently deal also with discrete optimization problems. Hasancebi [121] analyzed the discrete optimization approaches in evolution strategies for design optimization of steel frames and implemented a new formulation to improve design search. Design variables are mutated by means of random numbers sampled according to a geometric distribution. The basic steps, recalled also in [93,94], are now briefly summarized.

- *Generation of the initial population*. The initial population comprised of N_{POP} parent individuals is randomly generated. Each individual includes the strategy parameters p and Ψ along with the values assigned to optimization variables.
- *Evaluation of candidate designs*. Cost function and optimization constraints are evaluated for each individual (i.e. candidate design) included in the population. Candidate designs are hence ranked in terms of their fitness score.
- *Recombination*. The parent population is subject to recombination and mutation operators to generate the offspring population. Recombination consists in the exchange of information between the N_{POP} parent individuals: this leads to define N_{OFSP} new (offspring) individuals. Recombination can be applied also to strategy parameters.
- *Mutation*. This is the most important step of ES and is applied to each offspring formed through recombination. The mutation probability parameter p serves to tune the overall mutability of the individual. The strategy vector Ψ includes NDV components each of which affects the properties of the geometric distribution used for mutating the corresponding design variable. The mutation probability parameter is varied first using a normal distribution which forces the value of p to remain between 0 and 1. A random

number r is hence generated for each design variable and used as threshold to decide whether Ψ parameter and corresponding design variable value must be updated or not.

- *Selection*. The selection process has the purpose of finding which individuals included in the parent and offspring populations should survive for the next generation. The (μ,λ) strategy is utilized: all parents are left to die out and the best μ offspring with the lowest cost are selected out of λ offspring. The surviving individuals are taken as the parent individuals for the next generation.
- *Termination*. The above described steps are repeated for a given number of generations.

Another optimization algorithm based on evolution strategies is Differential Evolution [2,120] which is now briefly outlined.

- The initial population is generated randomly by perturbing design variables between lower and upper bounds. DE uses real-coding of design variables and includes three internal parameters: the crossover rate (CR), the scaling factor (F), and the population size (N_{POP}). There are three main steps of DE including mutation, crossover and selection. Mutation is executed by adding a weighted difference vector between two individuals to a third individual. After crossover and selection, mutated individuals produce offspring. The search process lasts until convergence or terminates when the limit number of structural analyses is reached or no significant improvement in cost is observed over a pre-specified number of consecutive generations. The general expression of the mutation operation in DE is:

$$\boldsymbol{v}_{p,G+1} = \boldsymbol{x}_{best,G} + F \cdot (\boldsymbol{x}_{r1,G} - \boldsymbol{x}_{r2,G}) \qquad (p=1,\ldots,N_{POP}) \qquad (1)$$

where: G is the index relative to the current generation; r_1 and r_2 are random numbers in the interval [1,N_{POP}]; $\boldsymbol{x}_{r1,G}$ and $\boldsymbol{x}_{r2,G}$ are two trial designs included in the population which were randomly selected; $\boldsymbol{x}_{best,G}$ and $\boldsymbol{v}_{p,G+1}$ are the best target vector and mutated vector of the current generation. The scaling factor F ranges between 0 and 1 and serves to enlarge or reduce the differential variation between the two randomly selected design vectors. Other mutation schemes are reported, for example, in Refs. [2,122].

- *In the crossover operation*, the trial vector $\boldsymbol{u}_{p,G+1}$ is generated by combining the mutated vector $\boldsymbol{v}_{p,G+1}$ and the target vector $\boldsymbol{x}_{p,G}$. If the random number generated is smaller than CR, the j^{th} variable of the mutation vector is set as the j^{th} variable of the trial vector.

- *In the selection operation*, the trial vector is compared with the target vector to select individuals for the next generation. The trial vector is set as target vector for the next generation only if it can improve the cost function. Otherwise, the original target vector is chosen as target vector also for the next generation.

Differential evolution has successfully been applied to size, layout and topology optimization of truss structures [122]. A multi-population DE algorithm was adopted: the population is divided in sub-populations and each sub-population is assigned a unique evolutionary purpose with different operators and parameter settings. This allows the diversity of design space to be increased and avoided premature convergence. Besides the multi-population scheme, a self-adaptive strategy operating on the value of scaling factor F was also introduced to reduce the number of infeasible solutions generated in the optimization process. DE was also the basis of the culture differential evolution algorithm previously developed in Ref. [123] to carry out sizing optimization of truss structures. Hybridization of DE with other algorithms such as big bang-big crunch and eagle strategies also were used for optimizing skeletal structures (see, for, example, Refs. [107,108]).

Sinusoidal differential evolution [124] is a DE variant where the scaling factor F and the crossover rate CR are sinusoidally adapted as the optimization progresses. This adaptive DE variant has been used very recently as a basis of comparison in the testing of novel hybrid metaheuristic formulations combining random search and gradient information obtained at low computational cost (see, for example, Refs. [53]).

2.3. Simulated Annealing (SA)

Simulated Annealing [3,60] was developed from statistical thermodynamics to simulate the behaviour of atomic arrangements in liquid or solid materials during the annealing process. The material reaches the lowest energy level (globally stable condition) as temperature decreases. SA includes a rather simple optimization strategy. A trial design is randomly generated and problem functions are evaluated at that point. If the trial point is infeasible, it is rejected and a new trial point is evaluated. If the trial point is feasible and the cost function is smaller than the current best record, then the point is accepted and the best record is updated. If the trial point is feasible but the cost function is larger than the best value, then the point is accepted or rejected based on a probabilistic criterion which estimates if design may improve in the next function evaluations. In order to compute probability, a parameter

called "temperature" is utilized. Temperature can be a target value (estimated) for the cost function corresponding to a global minimizer. Initially, a larger target value is selected. As the trials progress, the target value is reduced based on a cooling schedule. The acceptance probability steadily decreases to zero as temperature is reduced.

SA is widely utilized in structural optimization problems [125−138] because of its inherent simplicity and ability to find the global optimum even if there are many design variables. Feasibility of use of SA in structural design was demonstrated clearly in the classical papers written by Balling [125] and Bennage and Dhingra [126].

The basic formulation of the SA algorithm was often modified to improve convergence behaviour and reduce the number of exact structural analyses. For example, Shea *et al*. [127,128] developed a truss design code based on the combination of simulated annealing and shape grammar rules. Whilst in the basic SA algorithm the probability threshold for selecting any movement is set by the user and does not change throughout the optimization process, Shea *et al*. utilized dynamic rule selection where the probability of selecting any movement is based on past success rates. Quality factors are assigned to each design variable to select movements that are likely to generate trial designs which can be retained as new current best records. Pantelides and Tzan [129] included in SA a sensitivity analysis to identify which design variables should be modified to reduce global displacements. Each time a new design improved the current best record, move limits are defined to restrict the random generation of trial designs within a region close to the new best record.

Hasancebi and Erbatur [130] carried out simultaneous design optimization of truss structures with respect to sizing, shape and topology design variables. Whilst the above mentioned authors utilized a standard SA scheme, Sonmez [131] later implemented an SA-based algorithm where a set of current configurations rather is generated than just one trial design: each time a new trial design is better than the worst design included in the set of current configurations, the worst design is replaced by the new trial design. Although this SA scheme is in principle very powerful, its performance may depend on the number of trial designs included in the population as well as on the quality of initial population. Degertekin [132] found that SA is superior over GA in design optimization problems of nonlinear space steel frames formulated according to AISC-RLFD (1986) specifications. Hasancebi *et al*. [133] developed a SA scheme where the acceptance probability parameter is reformulated for non-improving trial designs and degeneration of search process by extremely poor solutions is avoided using a sigmoid function based update of the Boltzmann parameter. This

approach was very efficient and robust in the case of real-scale planar and space steel frames to be designed according to the Allowable Stress Design (ASD) specifications.

Chen and Su [134] pointed out that many trial designs may end up infeasible. For this reason, they proposed two improvements: (i) feasible regions of design space are estimated from constraint linearization and move limits are assigned to design variables; (ii) optimization starts from a largely feasible design. Although numerical results demonstrated the validity of this approach, at least three facts should be underlined. First, the optimizer may not generate trial designs always lying on descent directions. Second, constraint linearization may be computationally expensive. Third, it may be difficult to generate highly feasible designs.

Another issue is how new trial designs should be generated in SA. Chen and Su [134] and Blachut [135] pointed out that there may be two strategies: 1-directional ("local") search where variables are perturbed one at a time; multi-directional ("global") search where all variables are perturbed simultaneously.

Lamberti *et al*. [136–138] studied extensively how to improve SA formulation. In the multi-level search scheme of Ref. [136], cost function sensitivities are evaluated to locate each new trial point on a descent direction. If the design search is far enough from constraint domain boundaries, all variables are perturbed simultaneously. Conversely, local search involving perturbation of one variable at a time is performed when the current best record is located near constraint boundaries.

The SA algorithm described in [137] builds a search domain Ω, centred about each current best record, to include a set of descent directions and hence a population of candidate designs rather than just one trial point. Non-descent directions which might yield later effective improvements in design also are included in Ω.

A sophisticated SA formulation was presented in Ref. [138]. The strength points of the CMLPSA (Corrected Multi Level & Multi Point Simulated Annealing) algorithm described in Ref. [138] were: (i) since a population of candidate designs are considered rather than a single trial point, there is no need to restrict the portion of design space currently investigated as the optimum is approached; (ii) each trial point can potentially improve current best record as it lies on a descent direction; (iii) design perturbations are forced to follow the rate of change exhibited by cost function; (iv) search strategy adaptively changes based on convergence history (i.e., feasible, largely or slightly infeasible intermediate designs); (v) CMPLSA checks whether constraint domain is locally convex or not: this strategy increases further the probability of by-passing local minima yet without using any probabilistic

criterion; (vi) infeasible intermediate designs are handled by performing 4th order approximate line searches in the neighbourhood of each feasible design generated when the optimizer tries to move away from infeasible regions.

Results of numerical tests carried out on weight minimization problems of truss structures with up to 200 design variables and 3500 nonlinear constrains proved that CMLPSA is very efficient compared to other state-of-the-art SA and meta-heuristic algorithms like Heuristic Particle Swarm Optimization [101] and Harmony Search [31,32]. Convergence behaviour of CMLPSA was insensitive to initial design. With respect to other SA formulations, CMLPSA approached more quickly the portion of design space containing the optimum. With respect to HPSO and HS, CMLPSA selected more rapidly the good candidate designs from the population of trial points generated in each optimization iteration: the number of structural analyses and total CPU time required in the optimization process were drastically reduced.

The main steps of the CMLPSA algorithm [138] are now briefly recalled.

- *Perform global annealing by perturbing all design variables simultaneously. Define the search domain Ω.* Cost function gradient $\overline{\nabla}W(\boldsymbol{X}_{OPT})$ is computed at the current best record P_{OPT}. Each optimization variable x_j is randomly perturbed so to have $(\partial W/\partial x_j)\,\Delta x_j<0$. Each movement Δx_j is hence calculated as follows:

$$\begin{aligned} &\partial W/\partial x_j>0 \Rightarrow \Delta x_j = -(x_j^U - x_j^L)\cdot N_{RND,j}\cdot \mu_j \cdot W_{OPT,l-1}/\, W_{OPT,l} \\ &\partial W/\partial x_j<0 \Rightarrow \Delta x_j = (x_j^U - x_j^L)\cdot N_{RND,j}\cdot \mu_j \cdot W_{OPT,l-1}/\, W_{OPT,l} \end{aligned} \qquad (j=1,\ldots,NDV) \qquad (2)$$

The random number $N_{RND,j}$ is chosen in the interval (0,1) for each design variable. The weighting coefficient μ_j is $|\partial W/\partial x_j|/\|\overline{\nabla}W(\mathbf{X}_{OPT})\|$; $W_{OPT,l}$ and $W_{OPT,l-1}$ are the last two current best record values taken by the cost function. The ratio $W_{OPT,l-1}/W_{OPT,l}$ forces the optimizer to maintain at least the current rate of reduction in cost function.

The Δx_j movements define the first descent direction $\boldsymbol{S}_{TR}^{1}$, the diagonal of domain Ω_1 (see Figure 1 for the simple case of two design variables). Portions of design space must be added to Ω_1 to create the other descent directions containing the population of candidate designs. For example, Figure 2.1 shows that descent directions must lie below segment $\overline{A_1A_2}$ orthogonal to the gradient vector $\overline{\nabla}W(\mathbf{X}_{OPT})$. Direction $\mathbf{S}_{LIM}$' – limited by P_{OPT} and B – has one known component equal to Δx_1 and one unknown component Δx_2^{lim}. Direction

$\mathbf{S}_{LIM}$" – limited by P_{OPT} and C – has one known component, Δx_2, and one unknown component Δx_1^{lim}. Unknown movements are computed from the linear system (orthogonality conditions) $(\mathbf{S}_{LIM}';\mathbf{S}_{LIM}'')^T[\bar{\nabla}W(\mathbf{X}_{OPT})]=0$ and resized as $\xi_1\Delta x_1^{lim}$ and $\xi_2\Delta x_2^{lim}$ to transform directions $\mathbf{S}_{LIM}'$ and $\mathbf{S}_{LIM}''$ into descent directions $\mathbf{S}_{TR}^2$ and $\mathbf{S}_{TR}^3$ for which it holds $(\mathbf{S}_{TR}^{2,3})^T[\bar{\nabla}W(\mathbf{X}_{OPT})]<0$. Descent directions $\mathbf{S}_{TR}^1$, $\mathbf{S}_{TR}^2$ and $\mathbf{S}_{TR}^3$ are the diagonals of search domain Ω. The last diagonal $\overline{P_{OPT}L}$ is not a descent direction. Step size along $\mathbf{S}_{ND}$ (defined by the Δx_j^{ND} movements) is set once step sizes on descent directions are known.

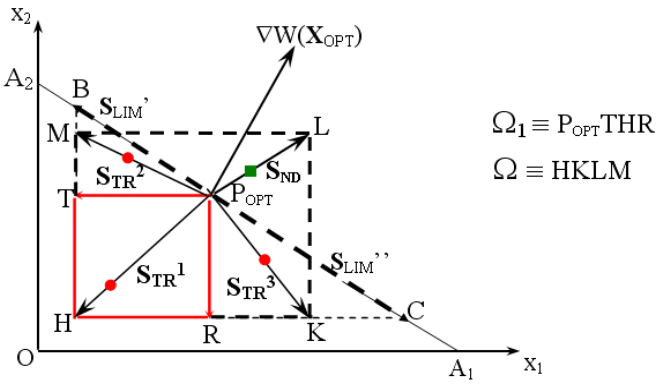

Figure 2.1. Construction of search domain Ω including pool of candidate designs

- *Refine the search domain Ω*. Since the step size δS_{TR}^q taken on each descent direction must ensure correspondence between the "linearized" change in cost $(\boldsymbol{\delta S}_{TR}^{q})^T[\nabla W(\mathbf{X}_{OPT})]$ and the actual change ΔW_q, CMLPSA utilized the following trust region scheme:

$$\frac{W(X_{OPT}) - W(X_{OPT} + \delta S_{TR}^{\ q})}{(\delta S_{TR}^{\ q})^T \bar{\nabla} W(X_{OPT})} \geq 0.75 \qquad (3)$$

Whilst the trust region model is usually used to check the quality of new intermediate designs, Eq. (3) serves instead to accept or reject potential designs defined by the δS_{TR}^q step on each descent direction. Descent directions are rescaled as follows:

$$\mathbf{S}_{TR}^{q,final} = \text{Min}\,[1, (\delta S_{TR}^q / \|\mathbf{S}_{TR}^q\|)]\,\mathbf{S}_{TR}^q \qquad (4)$$

where the step size is always reduced (by the scaling factor $\omega_q<1$) or stays the same only if the movement $||\mathbf{S_{TR}^q}||$ falls entirely within the trust region.

- *Evaluation of trial designs. Choice of the best candidate design*. Each descent direction $\mathbf{S_{TR}^{q,final}}$ is limited by P_{OPT} and a $P_{TR}^q(x_{OPT,1}+\omega_q \cdot \Delta x_1^{q,fin},..., x_{OPT,NDV}+\omega_q \cdot \Delta x_{NDV}^{q,fin})$ trial design. These trial points are represented as circle dots in Figure 2.1. Finally, $P_{ND}(x_{OPT,1}+\omega_{min}\Delta x_1^{ND},...,x_{OPT,NDV}+\omega_{min}\Delta x_{NDV}^{ND})$ (a square dot in Figure 2.1) is defined on $\mathbf{S_{ND}}$. Since ω_{min} is the smallest step size scaling factor, CMLPSA minimizes the weight penalty eventually paid if optimization variables were perturbed by moving along $\mathbf{S_{ND}}$.
  CMLPSA takes as candidate design the point P_{TR} for which there is the largest reduction in cost. Remarkably, candidate design selection dif not require any new constraint evaluation.

- *Check if trial point actually improves design*. Cost function and constraints are evaluated at P_{TR} (W_{TR} is the cost value). If $\Delta W_{TR}=W_{TR}-W_{OPT}<0$ and P_{TR} is feasible, P_{TR} is stored as new optimum. If $\Delta W_{TR}<0$ but P_{TR} is infeasible, a cubic approximation of optimization problem is built about P_{OPT}. If $\Delta W_{TR}>0$, CMLPSA shifts to local annealing search.

- *Perform local annealing. Perturb one design variable at a time*. CMLPSA perturbs design variables one by one about the current best record P_{OPT} to move away from constraint boundaries and escape from local minima. Constraints are linearized about P_{OPT}. CMLPSA checks if design space is locally non-convex. Let $P_{TR,j}$ denote the new trial design generated for the j^{th} variable. Cost change $\Delta W_j=W(\mathbf{X}_j)-W_{OPT}$ is evaluated for each $P_{TR,j}$.
  If $\Delta W_j<0$, CMLPSA stores the design $\mathbf{X}_j$ and the corresponding cost W_j in database Π_1. Nonlinear constraints are evaluated for designs $\mathbf{X}_{j,SMALL}$ and $\mathbf{X}_{j,LARGE}$ that yield respectively the smallest and largest weight changes $|\Delta W_j|$. If $\mathbf{X}_{j,LARGE}$ is feasible, CMLPSA takes it as the new current best record. Conversely, $\mathbf{X}_{j,SMALL}$ is taken as the new current best record.
  If $\Delta W_j>0$ and linearized constraints are satisfied, CMLPSA utilizes the Metropolis' probability function defined as:

$$P(\Delta W_j)=e^{\frac{-\Delta W_j}{(\sum_{r=1}^{NDW}\Delta W_r/NDW)\cdot T_K}} \tag{5}$$

where NDW is the number of trial points at which W was greater than for current best records found in the optimization process; ΔW_r are the corresponding weight penalties.

Each design $\mathbf{X}_j$ is provisionally accepted or certainly rejected according to the Metropolis' criterion reformulated as:

$$\begin{aligned} &P(\Delta W_j) > \mathrm{Max}[NRD_j, P(\Delta W_{ND})] \Rightarrow \mathrm{Accept} \\ &P(\Delta W_j) < \mathrm{Max}[NRD_j, P(\Delta W_{ND})] \Rightarrow \mathrm{Reject} \end{aligned} \tag{6}$$

where $P(\Delta W_{ND})$ is the probability computed for the cost function increment ΔW_{ND} corresponding to the P_{ND} point defined on the non-descent direction $\mathbf{S}_{ND}$.

The $\mathbf{X}_j$ designs provisionally accepted are included in the Π_2 database. If there are no trial designs for which cost function decreases, CMLPSA extracts from Π_2 the design $\mathbf{X}_j^{BEST}$ for which cost function value was the least and sets this as $\mathbf{X}_{OPT}$. Cost increase is hence minimized in the case local annealing could not improve design.

- *Check for convergence and eventually reset parameters for a new cooling cycle*. If the annealing cycles counter K>3, CMLPSA checks for convergence by checking the relative variation between the last three current best records. If the annealing cycles counter K<3 or stopping criterion is not satisfied, temperature is adaptively reduced as $T_{K+1}=\beta_K T_K$ where:

$$\beta_K = \left[\sum_{r=0}^{K-1} \beta_r / K\right] \cdot \mathrm{Max}\left[0.95 / \left(1 + \frac{N_{REJE}}{N_{TRIA}}\right); \left(1 - \frac{W_{FIN,K-1}}{W_{INIT,K-1}}\right)\right] \tag{7}$$

$W_{INIT,K-1}$ and $W_{FIN,K-1}$ are respectively the cost function values at the beginning and at the end of current annealing cycle. N_{REJE} is the number of trial designs rejected out of total number of trial designs N_{TRIA} generated in the current cooling cycle.

Couceiro *et al*. [139] recently developed an optimization framework by combining several well known features of SA: global/local approach (perturb simultaneously all design variables, or just a set of variables, or, ultimately, only one variable at a time), sensitivity analysis (to choose perturbation steps), reheating (temporary increase of temperature to allow uphill movements), penalty based constraint handling strategy, dimensionless form of cost function and optimization constraints. Compatibility module selects variables to be perturbed in each iteration while smoothing module searches for a new design by modifying only one discrete variable randomly selected. More recently, Hasancebi *et al*. [140] implemented a two-phase SA algorithm where a preliminary solution is generated by perturbing only a

specific type of design variables (i.e. layout parameters) and it is then refined by a new optimization process including all design variables.

Hence, the most important issue in SA is the search scheme utilized in the optimization process and, consequently, the number of function evaluations yielding no design improvements. Gradient information turn very useful provided that they can be obtained at low computational cost and there is enough diversity in the optimization search to bypass local minima. In view of this, the CMPLSA formulation of Ref. [138] was enhanced by Ficarella *et al*. [53] order to increase its computational efficiency. For that purpose, a global-local annealing strategy including a faster way to compute the cost function gradient $\overline{\nabla}W(\mathbf{X}_{OPT})$ regardless of the implicit/explicit form of the cost function $W(\mathbf{X})$, improved 1-D annealing and approximate line search strategies were combined in the search process.

2.4. Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO)

2.4.1. *The PSO algorithm*

The PSO algorithm [4,141,142] mimics the interactions between individuals of bird/fish swarms. If one individual sees a desirable path to go (for food, protection, etc.), the rest of swarm will be able to follow quickly even if they are not in a direct connection with the leading individuals. However, to facilitate exploration of the search space, each particle should have a certain level of "craziness" or randomness in their movement. Therefore, the problem is to determine the optimal positions or the swarm ensemble. Particles move through the search space and their positions are updated based on the best positions of individual particles in each iteration. The typical implementation of PSO (see, for example, Ref. [143]) is now recalled.

- *Generation of the initial swarm*. N_{POP} trial design vectors (i.e., the "particles" of the swarm) are randomly initialized. Each particle, denoted by the design vector $\mathbf{x}^{i}_{o}$, is a potential solution of the optimization problem. The velocity $\mathbf{v}^{i}_{o}$, also randomly generated for each particle, serves to update the position of the particle in the design space and is related to the possibility that each particle will move towards the optimal position. Positions and velocities of particles initially generated must be distributed throughout the design space. That is:

$$\mathbf{x}^{i}_{o} = x_L + r\times(x_U - x_L) \quad ; \quad \mathbf{v}^{i}_{o} = [x_L + r\times(x_U - x_L)]/\Delta t \qquad (i=1,\ldots, N_{POP}) \qquad (8)$$

where x_U and x_L are respectively the upper and lower bounds of design variables; r is a random number in the interval (0,1), Δt is a time interval usually equal to 1.

- *Evaluation of particles*. Cost function and constraints are evaluated for each candidate design included in the swarm. The best position $\mathbf{p}^i_k$ of the i^{th} particle in the k^{th} optimization iteration and the global optimum position $\mathbf{p}^g_k$ amongst all particles are thus determined.
- *Update position and velocity*. The position of each particle is updated as follows:

$$\mathbf{x}^i_{k+1} = \mathbf{x}^i_k + \mathbf{v}^i_{k+1}\,\Delta t \qquad (i=1,\ldots, N_{POP}) \qquad (9)$$

where: $\mathbf{x}^i_{k+1}$ and $\mathbf{x}^i_k$ are respectively the design vectors corresponding to the updated and original positions of the i^{th} particle; $\mathbf{v}^i_{k+1}$ is the updated velocity vector. Equation (9) indicates that the i^{th} particle has moved from its original position to the new position in the direction of the velocity $\mathbf{v}^i_{k+1}$ over the time interval Δt. The i^{th} candidate design is hence perturbed by the quantity $\mathbf{v}^i_{k+1}\Delta t$.

The velocity vector of each particle is instead updated as follows:

$$\mathbf{v}^i_{k+1} = w\,\mathbf{v}^i_k + c_1 r_1(\mathbf{p}^i_k - \mathbf{x}^i_k)/\Delta t + c_2 r_2(\mathbf{p}^g_k - \mathbf{x}^i_k)/\Delta t \qquad (i=1,\ldots, N_{POP}) \qquad (10)$$

where $\mathbf{v}^i_k$ is the velocity vector in the previous iteration; r_1 and r_2 are two random numbers in the interval (0,1): these numbers are generated anew for each component of the velocity vector in order to carry out fruitful line searches in the design space (see also Ref. [144]); c_1 and c_2 are "trust" parameters indicating respectively how much confidence the particle has in itself (c_1 is the "cognitive" parameter) and how much confidence it has in the swarm (c_2 is the "social" parameter); w is the inertia weight parameter. The inertia weight parameter w affects PSO convergence speed. If w is large, the speed of the particle increases and it is possible to explore a larger fraction of design space (i.e. global search. If w is small, the particle can move only inside a narrow region of design space (i.e. local search).

The particle best and the global best $\mathbf{p}^i_k$ e $\mathbf{p}^g_k$ are updated every time the new trial design $\mathbf{x}^i_{k+1}$ improves the particle best and global best of the previous iteration. If the trial design satisfies optimization constraints, the structural weight $W(\mathbf{X})$ is compared. Otherwise, the common penalty function approach is utilized.

- *Termination*. The above described steps are repeated over a pre-specified number of search iterations. Alternatively, the optimization process may terminate as soon as the limit number of structural analyses are performed.

Perez and Behdinan [143] pointed out that a general expression for particle position update can be obtained by combining Eqs. (9) and (10):

$$\mathbf{x}^{i}_{k+1} = \mathbf{x}^{i}_{k} + w\,\mathbf{v}^{i}_{k} + \left(c_1 r_1 + c_2 r_2\right)\left(\frac{c_1 r_1 \mathbf{p}^{i}_{k} + c_2 r_2 \mathbf{p}^{g}_{k}}{c_1 r_1 + c_2 r_2} - \mathbf{x}^{i}_{k}\right) \qquad (i=1,\ldots, N_{POP}) \qquad (11)$$

It can be seen that the particle position update rule follows the general gradient line-search form with a stochastic search step size and a stochastic search direction.

Since two decades PSO is commonly utilized in optimization of skeletal structures as primary design tool [101,145−147] or as a comparison basis with other meta-heuristic algorithms [53,93,94,138]. The original formulation of PSO has been repeatedly modified in order to improve convergence behaviour. For example, Perez and Behdinan [143] utilized an adaptive scheme where the inertia weight parameter is linearly reduced as optimization cycles progress. Furthermore, the velocity vector of a particle violating design constraints is restricted to a usable feasible direction that would reduce the objective function while pointing back towards feasible regions of the design space. The velocity of the particle is influenced only by the best position found so far for that particle and the current best position in the swarm.

Li *et al*. [101,147] developed an heuristic particle swarm optimizer (HPSO) for weight minimization of pin connected structures: the algorithm can efficiently deal with both continuous [101] and discrete sizing variables [147]. HPSO combines a PSO scheme with passive congregation (i.e. an individual is attracted at other group members but does not display social behaviour) and a Harmony Search based scheme. A fly-back mechanism is implemented to handle optimization constraints while the harmony search scheme deals with the side constraints: the particle is moved back to its previous position to guarantee a feasible solution; if some component of a newly generated trial design violates side constraints, it is replaced by the corresponding component of $\mathbf{p}^{g}_{k}$ randomly selected. In the passive congregation scheme, the velocity vector of each particle is updated as follows:

$$\mathbf{v}^{i}_{k+1} = w\,\mathbf{v}^{i}_{k} + c_1 r_1 (\mathbf{p}^{i}_{k} - \mathbf{x}^{i}_{k})/\Delta t + c_2 r_2 (\mathbf{p}^{g}_{k} - \mathbf{x}^{i}_{k}) + c_3 r_3 (\mathbf{R}^{i}_{k} - \mathbf{x}^{i}_{k}) \qquad (i=1,\ldots, N_{POP}) \qquad (12)$$

where $\mathbf{R}^{i}_{k}$ is a randomly selected particle from the swarm, c_3 is the "passive congregation" coefficient and r_3 is a random number in the interval (0,1).

Hasancebi *et al*. [94] found PSO to perform less efficiently than ES, SA and ACO in weight minimization of real size steel frames with discrete sizing variables. Since rounding off real terms of velocity vectors to nearest integers could result in zero velocity vectors, they implemented a new velocity updating scheme similar to that shown in Eq. (12). This strategy avoided particles to get stuck in suboptimal positions. The additional velocity term introduced in Eq. (10) is proportional to $\sqrt{N_s}$ where N_s is the number of commercially available cross sections. The results presented in Ref. [93] by the same authors for discrete sizing optimization problems of real truss structures indicate that with the above mentioned improvement PSO became absolutely competitive with ES, SA, ACO, HS and GA.

Several hybridization schemes were proposed in the literature to improve the performance of PSO [45,46,101–104], especially to avoid premature convergence and enhance the exploitation phase. The effective number of agents competing for the position of swarm leader and the duration of the leading role were issues dealt with to improve the PSO performance.

2.4.2. *The ACO algorithm*

The ACO algorithm [5] is a cooperative search technique that mimics the foraging behaviour of real-life ant colonies. Ants can construct the shortest path from their colony to the feeding source and back through the use of pheromone trails. When an isolated ant finds some food source in its random movement, it deposits a quantity of pheromone that can be detected by other ants which in turn deposit their own pheromone thus attracting other insects in an auto-catalytic process. The ACO implementation for structural optimization problems (see, for example, Refs. [93,94,148,149]) is recalled.

- *Initialization*. A number of ants representing possible solutions of the optimization problem are selected. Let us assume, for example, that there are N_{CS} commercially available cross-sections: hence, one ant will include NDV of these available values. Consequently, each design variable can take N_{CS} values (paths). The number of design variables usually corresponds to the number of groups in which elements of the skeletal structure are divided.

  An $N_{CS}\times NDV$ matrix termed as "trail matrix" $[T_{ji}]$ – j=1,…, N_{CS} and i=1,…,NDV – also is initialized. One can select the ant (i.e. trial design) formed by the smallest available

sections and set $T_{ji}=1/W_{min}$ where W_{min} is the corresponding weight of the structure. Each column of the trail matrix represents the entire set of pheromones accumulated in all N_{CS} paths for a design variable.

- *Selection probabilities*. Probabilities of selecting different paths (i.e. a given value of cross sectional area) for each design variable are computed as follows:

$$P_{ji} = \frac{[T_{ji}] \cdot \eta_j^{\beta}}{\sum_{k=1}^{Ncs} [T_{ji}] \cdot \eta_k^{\beta}} \tag{13}$$

where the visibility coefficient η_j is defined as $1/A_j$, i.e. the inverse of the cross sectional area of the j^{th} available section. Since visibility is inversely proportional to cross-sectional area, search is biased towards the selection of small cross-sectional area paths. Equation (13) shows that paths with higher pheromone levels are more likely to be selected. The β parameter (usually equal to 0.2) is a constant relating visibility and trail.

- *Construction of a population of candidate solutions*. A colony of N_{POP} ants is constructed based on the paths' selection probabilities for each design variable. The selection process is conducted so that each design variable is selected by all ants before passing to the next design variable. Each time a design variable is assigned to an ant the corresponding pheromone intensity T_{ji} is scaled down by a factor decreasing from 1 to 0.4÷0.5 as the optimization progresses. Path probabilities are recomputed to allow forthcoming ants choosing updated values.

- *Evaluation of a colony*. Structural analysis is carried out for each ant (i.e. candidate design) and the ants are ranked by cost function (or a penalty function approach is used) in order to select the best individual (elitist strategy).

- *Global pheromone update*. Pheromone evaporation is simulated and pheromones are added to those paths selected by the elitist and a specified number of top ranked individuals. This biases the search process towards favourable regions.

- *Termination*. The above steps are repeated for a pre-specified number of searches.

The feasibility of use of ACO in structural optimization problems was demonstrated by Camp *et al*. [150,151]. The ACO algorithm is naturally suited for design optimization of skeletal structures with discrete sizing variables. For example, trusses were optimized by Serra and Venini [148] and Hasancebi *et al*. [93]. Frames were optimized by Camp *et al*.

[151], Kaveh and Shojaee [152], Hasancebi *et al*. [94], Aydogdu and Saka [149], and Kaveh and Talatahari [153].

The basic ACO formulation was modified to improve algorithm performance. For example, Camp *et al*. [151] proposed the pheromone update rule considering both elitist and top ranked designs. Hasancebi *et al*. [93,94] introduced the concept of pheromone scaling. Furthermore, they observed that pheromone concentrates on a small number of paths for each design variable while rapidly evaporates for all other paths. In order to prevent the design process from being stuck in suboptimal regions, Hasancebi *et al*. [93] selected $\sqrt{N_s}$ top ranked paths (again, N_s is the number of commercially available cross sections) and evaluated a pheromone concentration rate for each optimization variable. This was done in purpose of obtaining a more homogeneous pheromone distribution amongst the paths of a given design variable so that paths with zero probability of selection may regain a chance of selection.

Kaveh and Talatahari [153] carried out a two-stage optimization process. In the global search phase, the sub-optimization mechanism (SOM) is utilized to rapidly reduce the search space: in fact, SOM divides the search space in sub-regions and deletes infeasible sub-domains thus speeding up the search process. In the local search phase, the position of the optimum design is refined by considering a given set of available cross-sectional areas in the neighbourhood of the optimum design found in the global search phase. The above mentioned strategy was successfully applied to the design of planar steel frames.

However, the most effective strategy seems the hybridization of ACO with other meta-heuristic optimization algorithms. For example, this approach was followed by Luh and Lin in two-stage optimization of truss structures for sizing, layout and topology [154]. Kaveh and Talatahari [45,46] combined PSO, ACO and HS to solve truss design problems with continuous or discrete variables. The heuristic particle swarm optimizer with passive congregation [101,147] is used as global optimizer while ACO works as a local search tools. Ants apply the pheromone-guided mechanism of ACO to refine the positions found by particles in the PSO stage. Therefore, ants generate solutions in the neighbourhood of the PSO global best found in the current iteration. The optimization process terminates when all design variables change by less than a threshold value defined by the user.

2.5. Harmony Search (HS)

The Harmony Search (HS) optimization method [10,31,32] mimics the process of searching for a perfect state of harmony performed by jazz players. The harmony in music is analogous to the optimum design vector in structural optimization. The musicians' improvisations are analogous to local and global search schemes in the optimization process. The correspondence between music improvisation and generation of new designs in the optimization process is clarified by Figure 2.2, taken from Ref. [32].

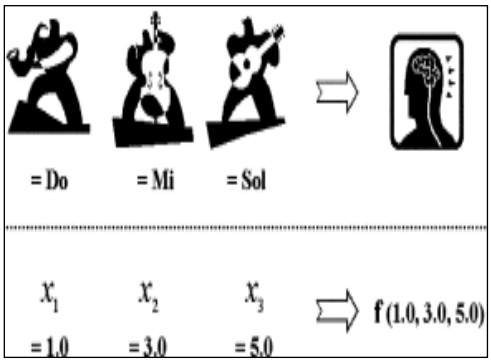

Figure 2.2. Mechanism of music improvisation reproduced by Harmony Search.

The relationship of HS to music is well summarized in Yang [27]. Three possible options are available to a skilled musician who is improvising: (1) play any famous piece of music (a series of pitches in harmony) exactly from his or her memory; (2) play something similar to a known piece (thus adjusting the pitch slightly); (3) compose new or random notes. These three options were formalized by Geem *et al*. [10] into the harmony search optimization algorithm.

The classical HS formulation entails the following steps.

1) The HS algorithm operates on N_{POP} candidate designs stored in the Harmony Memory [HM], an [N_{POP}×NDV] matrix defined by Eq. (14): each row represents a candidate design while columns include values of design variables. Designs are sorted in ascending order according to values of structural weight (for feasible designs) or values of a penalty function that accounts for constraint violation.

$$[\mathrm{HM}] = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_{NDV-1}^1 & x_{NDV}^1 \\ x_1^2 & x_2^2 & \cdots & x_{NDV-1}^2 & x_{NDV}^2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_1^{NPOP-1} & x_2^{NPOP-1} & \cdots & x_{NDV-1}^{NPOP-1} & x_{NDV}^{NPOP-1} \\ x_1^{NPOP} & x_2^{NPOP} & \cdots & x_{NDV-1}^{NPOP} & x_{NDV}^{NPOP} \end{bmatrix} \tag{14}$$

The limit number of iterations $N_{itermax}$ is specified by the user. The harmony memory considering rate (*HMCR*), pitch adjustment rate (*PAR*), and bandwidth amplitude (*bw*) parameters may be specified by the user or adaptively changed in the search process.

2) A trial design (called "harmony") is generated using three rules: (i) random selection; (ii) harmony memory consideration; (iii) design vector adjustment. In random selection, each variable is randomly chosen from [HM]. The harmony memory considering rate HMCR ranges between 0 and 1, and states the probability of selecting a value x_i' from the available set (x_i^1,x_i^2,…,x_i^{NPOP-1},x_i^{NPOP}) stored in the [HM].
The trial design $\boldsymbol{X}$'={x_1',x_2',…,x_N'} is analyzed to check if optimization variables must be adjusted or not. This process is driven by the pitch adjustment rate parameter *PAR*, which states the probability (1−*PAR*) of maintaining the previously set value x_i'.

3) A random number (*rnd*) is generated for each design variable. If *rnd*<*HMCR*, HS takes a value from the corresponding column of [HM] and checks if this value should be pitch adjusted. If *rnd*<*PAR*, the variable is modified as (x_i'± *rnd*·*bw*) where *bw* is an arbitrary distance bandwidth. Conversely, if *rnd*>*HMCR*, a new value is randomly generated for the design variable.

4) If the new harmony $\boldsymbol{X}$' is better than the worst design $\boldsymbol{X}_{\mathbf{worst}}$, it is included in [HM] replacing $\boldsymbol{X}_{\mathbf{worst}}$. In structural optimization problems, the structural weight (or the penalized weight) evaluated for the new harmony should be lower than the structural/penalized weight evaluated for the worst design.

5) Steps (1) through (4) are repeated until a pre-specified number of iterationsor function evaluations (i.e. structural analyses) are executed.The computational cost of the optimization process hence is N_{POP}×$N_{itermax}$ analyses, which may not be affordable for large-scale problems.

Harmony search optimization algorithms usually have two weak points: (i) the large number of structural analyses required in the optimization; (ii) new trial designs do not necessarily improve current designs. Besides this, HS shares with other metaheuristic algorithms the common problem of setting internal parameters. Researchers introduced many *ad hoc* features to overcome these limitations. For example, Saka [33] used an adaptive error strategy to deal with slightly infeasible designs: constraint tolerance is progressively reduced as the search process converges to the optimum. Maheri and Narimani [34] enhanced the harmony memory updating phase by considering also designs that are worse than the worst design stored in the harmony memory but are far enough from local optima. Murren and Khandelwal [35] developed a design-driven HS strategy to generate random trial solutions within intelligently specified search neighborhoods using constraint satisfaction or violation data from previous trial solutions.

Mahdavi *et al*. [36] introduced the dynamic update of pitch adjusting rate and bandwidth during optimization process. Hasancebi *et al*. [38] developed an adaptive variation scheme for HMCR and PAR: for each new harmony to be generated in the search process, HS parameters are probabilistically selected about average values observed within the current harmony memory matrix; this allowed HS to adapt itself to varying features of design space. Kaveh and Abadi [155] reprised the concept of linearly increasing PAR with the number of generations. A self-adaptive HS algorithm was utilized also by Degertekin [39]. Carbas and Saka [37] utilized yet another dynamic scheme for parameter updating where HMCR and PAR change more significantly in the current iteration as the ratio of the difference between maximum cost and average cost to the difference between maximum cost and minimum cost increases. Kaveh and Naiemi [40] developed a multi-adaptive HS algorithm where internal parameters may change linearly or exponentially in the current iteration. Geem and Sim [41] developed a parameter-setting-free technique where an additional matrix contains an operation type (random selection, memory consideration, or pitch adjustment) for each variable stored in the harmony memory. Turky and Abdullah [42] developed a multi-population approach where candidate designs are divided into several sub-populations: each sub-population explores or exploits a different region of search space; the best solutions are archived for being later used to replace redundant solutions in the harmony memory.

Finally, HS was hybridized with other metaheuristic algorithms [43–46] or gradient-based optimizers [49]. Omran and Mahdavi [47] developed a global best harmony search (GHS) algorithm by including the concept of best position typical of PSO: the new harmony is no longer generated via pitch adjustment but using the best harmony vector stored in the

harmony memory. Al-Betar *et al*. [48] attempted to replace the random selection scheme in the memory consideration phase. For that purpose, selection schemes derived from genetic algorithms and particle swarm optimization (e.g. global-best, fitness-proportional, tournament, linear rank and exponential rank) were adapted to HS. However, other authors [156,157] argued that the GHS approach is questionable either because it mixes up values of design variables which may have different scales and, more importantly, HS is a "neighborhood metaheuristic" method that compares just one trial design at a time with the rest of the population without considering any interactions between candidate designs.

From the previous discussion, adaptation of internal parameters appears to be the most effective strategy to improve convergence behavior of HS. Lamberti and Pappalettere [50] followed an alternative approach attempting to generate new harmonies that always improve the current best record. For that purpose, trial designs were defined by including information on cost function gradient and selecting descent directions. This approach was very effective for skeletal structures because gradients often are explicitly available and even constant over design space for sizing variables. The same authors developed an enhanced HS formulation in [51] trying to reduce the amount of heuristics entailed by the optimization process, increase convergence speed and enhance robustness. Further evidence of the validity of this approach was gathered by Degertekin and Lamberti [52].

Starting from the same rationale behind [50–52], a novel hybrid HS formulation was developed in [53] in order to further reduce the computational cost of the optimization and maximize robustness of optimized solutions. For that purpose, the metaheuristic search scheme was hybridized with search direction mechanisms and 1-D simulated annealing search. Unlike classical HS and the improved HS formulation of Ref. [50], parameters HMCR and PAR are not specified by the user but are randomly changed in the optimization process based on the optimization history. Furthermore, unlike Refs. [50-52], the harmony refinement process based on the bandwidth parameter is replaced by another random movement that forces HS to refine the new harmony moving along a descent direction. Finally, rather than replacing only the worst design of the population, in each iteration, the HS formulation of [53] attempts to renew the largest fraction of the population as it is feasible to approach more rapidly the global optimum.

The main steps of the hybrid HS algorithm developed in [53] are as follows.

1) Trial designs are generated by including gradient information (explicitly available in the case of skeletal structures) and a mirroring strategy corrects those designs that still do not improve the current best record.

2) Design freedom is increased in the pitch adjusting phase because the currently selected optimization variable can randomly change within an adaptively defined interval rather than be fixed to a given value (selected from the corresponding column of the harmony memory) or vary within the interval defined by the bandwidth parameter. The classical bandwidth parameter is replaced in [53] by a pitch adjusting strategy accounting for the optimization history because it depends on the ratio between the number of trial designs $NG_{pitch,adj}$ generated via pitch adjustment and the total number of trial designs NG_{tot} generated until that moment.
3) The harmony memory considering rate *HMCR* and pitch adjusting rate *PAR* parameters are randomly generated in each iteration and then adaptively changed taking into account the current trend exhibited by the optimization process. For that purpose, *HMCR* and *PAR* are scaled by factors such as: (i) the $WHS_{aver,end}/WHS_{aver,init}$ ratio between the average values of cost function for the trial designs included in the harmony memory at the end and at the beginning of the previous optimization iteration (this ratio should always be smaller than one to improve design); (ii) the $||\mathbf{X}_{\mathbf{OPT,end}}-\mathbf{X}_{\mathbf{WORST,end}}||/||\mathbf{X}_{\mathbf{OPT,init}}-\mathbf{X}_{\mathbf{WORST,init}}||$ ratio between the population "dispersion" (this feature is quantified by the difference between the best and worst designs) at the end and at the beginning of the previous iteration; (iii) the $NG_{pitch,adj}/NG_{gradient}$ ratio between the number of trial designs generated via pitch adjustment and the number of trial designs generated by including gradient information.
4) Each new trial design $\mathbf{X}_{TR}$ is very likely to improve also the current best record $\mathbf{X}_{OPT}$. However, ad hoc strategies account for all possible scenarios: (i) $\mathbf{X}_{TR}$ is feasible and $W(\mathbf{X}_{TR})<W_{OPT}$; (ii) $\mathbf{X}_{TR}$ is feasible but $W(\mathbf{X}_{TR})>W_{OPT}$; (iii) $\mathbf{X}_{TR}$ is infeasible and $W(\mathbf{X}_{TR})<W_{OPT}$; (iv) $\mathbf{X}_{TR}$ is infeasible and $W(\mathbf{X}_{TR})>W_{OPT}$. Besides updating the current best record and resorting population in terms of cost function values, the remaining designs of the population (i.e. N_{POP-2} for scenario (i); N_{POP-p} if the trial design rank p^{th} in the population for scenario (ii)) are perturbed to check if they can improve their counterparts originally stored in the population. Approximate line search with polynomial approximation is performed for scenario (iii). Mirroring strategy for scenario and scaling of perturbation step are performed for scenario (iv) in the attempt of re-entering in the feasible design space.
5) A fast 1-D simulated annealing based search is performed if the improvement routine planned for scenario (iv) cannot improve the current population. A vector joining the current best record and the trial design yielding the minimum penalty with respect to the current best record is defined. The components of this non-descent direction taken together

do not improve $\mathbf{X}_{OPT}$ but some of them taken singularly may instead improve it. A set of new descent directions corresponding to perturbations given to a single optimization variable is hence defined. Perturbation steps are properly scaled along each of these directions in order to fall within one of the scenarios (i) to (iii).

6) The convergence criterion operates on the ratios between the average and the standard deviation for both cost function values and design vectors in order to avoid premature convergence.

2.6. Big Bang–Big Crunch (BBBC)

The Big Bang-Big Crunch (BBBC) algorithm [13] reproduces the cycles of expansion and contraction of the universe. In the optimization process, N_{POP} designs are randomly generated (explosion) to form a population and the center of mass of this population is defined as a weighted average of these designs (contraction) where each weighing coefficient is the cost function for a trial design. A new population is randomly generated by perturbing design variables in the neighborhood of the center of mass (either the true center of mass is considered or the best design of the population is set as center of mass). This process is repeated until no significant improvement in design is obtained.

The inherent simplicity of BBBC favored the use of this algorithm in many fields of structural optimization. For example, Camp and his collaborators [158–160] utilized BBBC to optimize design of space trusses, retaining walls and reinforced concrete frames. Kazemzadeh Azad *et al*. [161] demonstrated the efficiency of BBBC by solving several benchmark engineering optimization examples such as the cantilever welded beam and pressure vessel problems. Rafiee *et al*. [162] optimized the design of steel frames with semi-rigid connections. Li and Pak [163] pointed out that BBBC is one of the metaheuristic algorithms used by NASA for multidisciplinary optimization of next generation aircrafts.

Kaveh and Talatahari [164–166] modified the original formulation of BBBC. The HBBBC algorithm implemented in [164] for weight minimization of truss structures not only considered the center of mass as the average point at the beginning of the big bang phase but, similarly to PSO, utilized also the best position of each particle and the best visited position of all particles; furthermore, the sub-optimization mechanism was utilized; the largest optimization problem included 59 design variables (hence, beween only 1/5 and 1/3 of the number of design variables considered in this dissertation): a 26-storey tower, about 95 m

tall, to be designed for minimum weight under 7 loading conditions; the tower was 42mprobe as a 3D truss structure including 942 elements and 244 nodes. This hybrid algorithm was also applied to discrete sizing and continuous layout optimization of Schwedler and ribbed domes [165] as well as to other skeletal structures [166].

Kaveh and Zolghadr [167] hybridized BBBC with charged system search and used the explosion phase of BBBC to escape from traps limiting the exploration of the whole design space: CSS-BBBC was successfully applied to truss weight minimization problems with frequency constraints. Kaveh and Mahdavi [168] combined HBBC with the Quasi-Newton method to solve optimization problems with multiple natural frequency constraints.

Hasancebi and his collaborators introduced several improvements to BBBC, which are reviewed in [169]. The big bang phase was enhanced by increasing the power of the random variable used to generate new trial designs from one to three [170]: this approach was very effective in discrete sizing optimization of steel frames for both normally and exponentially distributed random variables. These algorithmic variants (denoted as modified BBBC and exponential BBBC) performed well also in discrete sizing optimization of steel truss structures [171]. The upper bound strategy was developed in [57,172] to reduce the computational cost of the big bang phase in frame and truss optimizations: structural analyses were performed only for lighter trial designs than the current best record. A hybrid algorithm combining exponential BBBC and upper bound strategy [173] was applied to large-scale 3D steel frames with practical design constraints and loading conditions. Finally, Kazemzadeh Azad [174] developed a hybrid algorithm combining guided adaptive dimensional search and modified/exponential BBBC to optimize steel truss structures with numerous discrete sizing variables.

A common feature of many BBBC variants presented in literature is that each explosion about the center of mass or the fittest individual may entail many structural analyses. Furthermore, even in the case of the upper bound strategy, all of the new designs are not necessarily generated by perturbing optimization variables on descent directions. For example, Kazemzadeh Azad *et al*. [169] reported that the computational cost of the explosion phase may range between $0.1N_{\mathrm{POP}}$ and N_{POP} structural analyses based on the problem at hand.

Considering the issues pointed out above, Lamberti *et al*. [51,52,176] developed a new BBBC formulation with infrequent explosions. Approximate line searches are carried out in the neighborhood of $\boldsymbol{X}_{\mathrm{OPT}}$ in order to find a set of descent directions. This phase is enriched by gradient information. A new trial design $\boldsymbol{X}_{\mathrm{TR}}$ is generated by combining random steps taken on these descent directions. If $\boldsymbol{X}_{\mathrm{TR}}$ is better than $\boldsymbol{X}_{\mathrm{OPT}}$, it replaces the current best record.

Conversely, improvement routines including approximate line search and mirroring strategy are performed. If $\boldsymbol{X}_{\mathrm{TR}}$ remains worse than $\boldsymbol{X}_{\mathrm{OPT}}$, a new explosion about the middle point of the segment limited by the center of mass and the current best record is performed. In summary, unlike classical BBBC, no explosion is performed if the new trial design improves the current best record. This allows the number of structural analyses to be considerably reduced.

The efficient BBBC algorithm described in [51,52,176] was further developed by Ficarella *et al*. [53] in order to increase robustness and minimizing the computational cost of the optimization process. Interestingly, the new algorithm does not require any more the $\partial \mathrm{W}/\partial \mathrm{x_j}$ sensitivities to be exactly evaluated. Furthermore, descent directions from which each trial design $\boldsymbol{X}_{\mathrm{TR}}$ is generated are more accurately selected. Third, population is dynamically updated so as to simulate an explosion about the center of mass but with all new trial designs lying on potentially descent directions. The hybrid nature of the new BBBC algorithm comes from the combination of the explosion/contraction process with 1-D local annealing and line search mechanisms. The hybrid BBBC algorithm of Ref. [53] is taken as the start point of the LSSO–HBBBCJA algorithm developed in this dissertation and preliminary evaluated in shape/layout optimization problems.

2.7. The JAYA algorithm

The JAYA method [12] is a recently developed population-based method including the simplest search strategy of metaheuristic optimization: trial solutions are generated always moving towards the best design and away from the worst design of the population. This rationale well reflects the origin of the name JAYA, a Sanskrit word meaning “victory”. Remarkably, JAYA needs only two standard control parameters such as population size and limit number of iterations.

JAYA is very simple to be implemented because it has only one equation for perturbing design. Let $X_{j,k,it}$ be the value of the j$^{\mathrm{th}}$ design variable (j=1,…,*NDV*) for the k$^{\mathrm{th}}$ design of population (k=1,…,N_{POP}) at the it^{th} iteration. The $X_{j,k,it}$ value is modified as follows:

$$X_{j,k,it}^{new} = X_{\mathrm{j,k,it}} + r_{1,\mathrm{j,it}}\left(X_{\mathrm{j,best,it}} - \left|X_{\mathrm{j,k,it}}\right|\right) - r_{2,\mathrm{j,it}}\left(X_{\mathrm{j,worst,it}} - \left|X_{\mathrm{j,k,it}}\right|\right) \tag{15}$$

where: $X_{\mathrm{j,k,it}}^{new}$ is the updated value of variable $X_{\mathrm{j,k,it}}$; $r_{1,\mathrm{j,it}}$ and $r_{2,\mathrm{j,it}}$ are two random numbers in the interval [0,1] for the j^{th} variable; $X_{\mathrm{j,best,it}}$ and $X_{\mathrm{j,worst,it}}$, respectively, are the values of the

j^{th} variable for the best design $\mathbf{X}_{\text{best,it}}$ and worst design $\mathbf{X}_{\text{worst,it}}$ of the population in the current iteration.

The $r_{1,\text{j,it}}(X_{\text{j,best,it}} - |X_{\text{j,k,it}}|)$ term describes the JAYA's tendency to approach the best design $\boldsymbol{X}_{\text{best,it}}$, while the $-r_{2,\text{j,it}}(X_{\text{j,worst,it}} - |X_{\text{j,k,it}}|)$ term refers to the tendency of moving away from the worst design $\boldsymbol{X}_{\text{worst,it}}$. Random factors r_1 and r_2 allow design space to be well explored while the absolute value $|X_{\text{j,k,it}}|$ in Eq. (15) further enhances exploration [12].

The new trial solution $\boldsymbol{X}_k^{new}$ generated with Eq. (15) is compared with its counterpart $\boldsymbol{X}_k^{pre}$ stored in the population. If $\boldsymbol{X}_k^{new}$ is better than $\boldsymbol{X}_k^{pre}$, population is updated by replacing $\boldsymbol{X}_k^{pre}$ with $\boldsymbol{X}_k^{new}$. This process is repeated until the limit number of iterations/analyses is completed.

As mentioned in the Introduction section, the inherent simplicity and easiness of implementation led to utilize JAYA in a wide variety of applications such as, for example, welding process optimization, economic optimization of shell-and-tube heat exchanger, multi-objective optimization of heat exchangers, path selection in wireless networks, optimization of abrasive waterjet machining process, structural health monitoring of concrete dams, optimization of stand-alone photovoltaic, wind turbine, and battery systems, optimization of automatic voltage regulator system, recognition and classification of leaf diseases, online load frequency control of wind power systems, optimum unit sizing of hybrid renewable energy system, binary optimization problems, optimization of magnetic abrasive finishing process, and adaptive cart position control. The reader can refer to Ref. [56] for a detailed review of the applications listed above.

However, some issues have to be carefully considered in the JAYA implementation and this explains why a large number of variants and hybridization schemes are documented in the optimization literature just five years after JAYA was developed. The first question is how to choose the relative magnitude of the movements $(\boldsymbol{X}_{\mathbf{j,best,it}} - |\boldsymbol{X}_{\mathbf{j,k,it}}|)$ and $(\boldsymbol{X}_{\mathbf{j,worst,it}} - |\boldsymbol{X}_{\mathbf{j,k,it}}|)$. The random scaling used in Eq. (15) certainly is the easiest option but the effect of convergence history should be properly taken into account.

Another important aspect is the computational cost of the optimization. Similar to HS, the computational cost of the JAYA search may be quantified as $N_{POP} \times N_{itermax}$ structural analyses. Degertekin *et al*. [54–56] attempted to limit the number of structural analyses by directly rejecting heavier trial designs than those currently stored in the population. However, this strategy neither allowed to find the global optimum nor to maximize computational speed

for all structural design problems (see, for example, the numerical results reported in Ref. [53]).

The computational cost of JAYA may be a critical issue also in the case of structural optimization problems including a relevant number of discrete variables. For this reason, a simplified topology optimization approach was developed in [54]. The optimization process entailed the following steps:

(i) Set lower bound of sizing variables as 10^{-7} in^2;
(ii) Perform continuous optimization including all design variables;
(iii) Round sizing variables as they get very close to the available discrete cross-sections;
(iv) Remove sizing variables that get very close to the 10^{-7} in^2 lower bound;
(v) Perform continuous optimization with layout variables and the remaining sizing variables;
(vi) Repeat steps (iii) through (v) until no sizing variables remain to be optimized.

The case of discrete layout variables also was covered by the simplified topology optimization approach of Ref. [54]. Once step (vi) is completed, layout variables are perturbed to their nearest discrete values and the best design is set the new optimal solution. This solution is then perturbed with respect to sizing variables in order to further improve design. The process ends when no improvement is found passing from one type of variables to another.

The stiffness matrix of the truss structure is not reformulated when the cross-sectional area of some element approaches 10^{-7} in^2. The "weak" elements which are assigned the cross-sectional area of 10^{-7} in^2 practically behave as if they were killed and hence do not contribute by any extent to the global stiffness of the structure. The 10^{-7} in^2 limit was selected as it is 10^{-6} times the lower bound of 0.1 in^2 set for sizing variables. Basically, the stiffness terms (in the case of a truss structure, these terms are proportional to element cross-sectional areas) must be multiplied by a factor 10^{-6} in order to become inactive similarly to what occurs in commercial finite element programs when the "kill element" option is selected.

A fully discrete advanced JA formulation (DAJA) was developed in [55] for truss design problems. The optimization problem was solved in a single loop without the need of carrying out the preliminary continuous optimization done in [54]. Explicit gradient or pseudo-gradient information were utilized together with approximate line search in order to speed up the generation of discrete designs and reduce the number of structural analyses required by the optimization process. Basically, a set of descent directions were generated in

the neighborhood of each candidate design to increase the probability of improving population in each design cycle. This simplified the optimization process and allowed to eliminate biases towards local minima that may be originated from rounding continuous optimum solutions improperly. Although the single-stage solution strategy utilized by DAJA [55] was considerably faster than the two-stage strategy utilized by improved JA [54] for discrete truss optimization problems, test problems included at most only 59 design variables and the largest structure included less than 1000 bars.

In [56], a parameter free JA formulation was coded into the PFJA algorithm for sizing/layout of truss structures under natural frequency constraints. Unlike other improved/advanced JA variants and standard JA, PFJA uses neither common (e.g. population size and limit number of iterations usually specified by the user) nor algorithm-specific parameters. Besides using an elitist strategy to perform new structural analyses only if strictly necessary, PFJA adaptively changed population size based on the current trend of optimization history. An adaptive scheme for changing population size in the optimization process was originally proposed by Rao *et al*. [177]:

$$ps_{next} = round(ps_{current} + rand * ps_{current}) \tag{16}$$

where $ps_{current}$ and ps_{next}, respectively, are the population sizes in the current iteration and in the next iteration, *rand* is a randomly generated number in the interval [–0.5 , 0.5]. Hence, population size may increase or decrease during the optimization process because rand may be positive or negative. According to Eq. (16), the new population ps_{next} may be at most 50% larger than the current population pscurrent and at least 50% of $ps_{current}$. This variation range should keep enough diversity in the population throughout the search process without increasing too much the computational cost of optimization.

PFJA modified the adaptive scheme of Ref. [177] as follows.

1) If $ps_{next} > ps_{current}$, the current population is completely transferred into the next population and the best optimum solutions stored in the current population are used for filling the remaining ($ps_{next} - ps_{current}$) positions of the new population. In order to avoid stagnation that could be caused by the presence of duplicated designs, PFJA randomly selects one optimization variable for each of the designs, which might be duplicated in the new population. Furthermore, in order to increase probability of generating high quality trial designs, each selected variable of a design to be added into the new population is perturbed

trying to move along a descent direction. A sophisticate mirroring strategy is used each time this attempt is unsuccessful.

2) If $ps_{next}<ps_{current}$, only the best ps_{next} individuals of the current population should be transferred into the next generation. However, PFJA again attempts to improve solutions by perturbing one randomly selected variable for each design to be transferred into the new population. The new trial solutions are rejected if they cannot improve the current best record or at least the corresponding original design randomly perturbed by PFJA.
3) If $ps_{next}=ps_{current}$, the current population should remain unchanged for [177]. However, PFJA perturbs randomly designs in the attempt of improving candidate designs for the next iteration. Again, new designs are retained in the new population only if they can improve best record or at least their counterparts in the current population.
4) If $ps_{next}<NDV$, while in [177] only the best NDV individuals of current population were transferred into the new population, PFJA adopts a different strategy based on sensitivity analysis. Since the cost function (i.e. structural weight) of a skeletal structure is explicit with respect to both sizing and layout variables, it is possible to exactly compute sensitivities for the best NDV designs. For each design, PFJA randomly perturbs the variable corresponding to the largest sensitivity. A new trial solution is rejected if does not improve the current best record or the currently perturbed individual.

The above described modifications allowed computational cost of PFJA optimizations to be reduced by about 20% with respect to the PFJA version implementing the original population size adaptation scheme of Ref. [177]. This occurred because all strategies used by PFJA to adaptively refine population rely on a paradigmatic elitist principle: to attempt improving designs. The 20% reduction of computational cost still appears not to be enough for making JAYA fully suitable for large-scale optimization process. In fact, PFJA was utilized in [56] to optimize a spatial tower including 942 elements and three spatial domes respectively including 600, 1180 and 1410 elements. However, the optimization problem solved for the largest dome structure with 1410 elements included only 47 sizing variables vs. the 59 sizing variables used for the 942-bar tower and the 1180-bar dome.

The initial population of PFJA included $10\times NDV$ individuals. This made convergence behavior of the JA variant described in Ref. [56] insensitive to population size. However, robustness and insensitivity to size and composition of initial population is a fairly common feature shared by all JA variants. This confirms the primary role played by elitist strategies in the convergence process of metaheuristic algorithms.

CHAPTER 3

THE LSSO-HHSJA ALGORITHM

3.1. Initialization of the LSSO–HHSJA algorithm

The LSSO–HHSJA algorithm developed here combines the HS and JAYA methods. The main goal of the new algorithm is to efficiently explore design space, generating high quality trial designs on descent directions without complicating too much the inherently simple formulations of HS and JAYA. This allows to limit the number of structural analyses making it affordable to solve large scale structural optimization problems. The new algorithm is now described in detail and its flow chart is shown in Figure 3.1. A population of N_{POP} candidate designs is randomly generated as follows:

$$x_j^k = x_j^L + \rho_j^k \, (x_j^U - x_j^L) \qquad \begin{cases} j = 1, \dots, NDV \\ k = 1, \dots, N_{POP} \end{cases} \tag{17}$$

where NDV is the number of optimization variables and ρ_j^k is a random number in the (0,1) interval.

The N_{POP} candidate designs are stored in the Harmony Memory [HM], an [$N_{POP} \times NDV$] matrix defined by Eq. (14): each row represents a candidate design while columns include values of design variables. Designs are sorted in ascending order according to values of structural weight (for feasible designs) or values of a penalty function that accounts for constraint violation.

$$[HM] = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_{NDV-1}^1 & x_{NDV}^1 \\ x_1^2 & x_2^2 & \cdots & x_{NDV-1}^2 & x_{NDV}^2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_1^{NPOP-1} & x_2^{NPOP-1} & \cdots & x_{NDV-1}^{NPOP-1} & x_{NDV}^{NPOP-1} \\ x_1^{NPOP} & x_2^{NPOP} & \cdots & x_{NDV-1}^{NPOP} & x_{NDV}^{NPOP} \end{bmatrix} \tag{14}$$

Similar to [53], parameters HMCR and PAR are adaptively changed in the optimization process. The *bw* parameter also is not necessary as new trial designs always lie on descent directions.

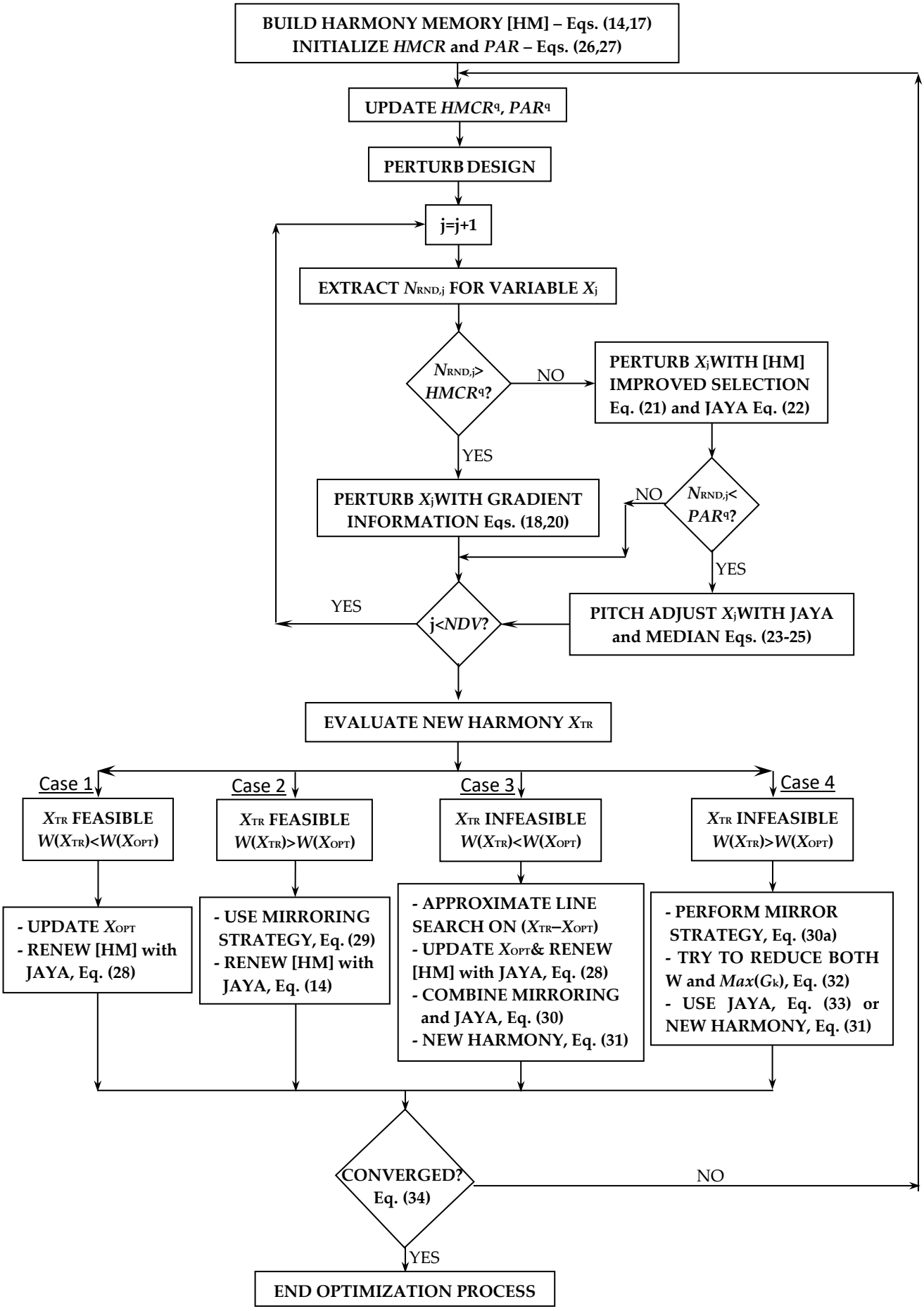
BUILD HARMONY MEMORY [HM] – Eqs. (14,17)
INITIALIZE HMCR and PAR – Eqs. (26,27)
UPDATE HMCRq, PARq
PERTURB DESIGN
j=j+1
EXTRACT NRND,j FOR VARIABLE Xj
NRND,j> HMCRq?
NO
PERTURB XjWITH [HM] IMPROVED SELECTION Eq. (21) and JAYA Eq. (22)
YES
PERTURB XjWITH GRADIENT INFORMATION Eqs. (18,20)
NRND,j< PARq?
NO
YES
PITCH ADJUST XjWITH JAYA and MEDIAN Eqs. (23-25)
j<NDV?
YES
EVALUATE NEW HARMONY XTR
Case 1
Case 2
Case 3
Case 4
XTR FEASIBLE W(XTR)<W(XOPT)
XTR FEASIBLE W(XTR)>W(XOPT)
XTR INFEASIBLE W(XTR)<W(XOPT)
XTR INFEASIBLE W(XTR)>W(XOPT)
- UPDATE XOPT
- RENEW [HM] with JAYA, Eq. (28)
- USE MIRRORING STRATEGY, Eq. (29)
- RENEW [HM] with JAYA, Eq. (14)
- APPROXIMATE LINE SEARCH ON (XTR–XOPT)
- UPDATE XOPT& RENEW [HM] with JAYA, Eq. (28)
- COMBINE MIRRORING and JAYA, Eq. (30)
- NEW HARMONY, Eq. (31)
- PERFORM MIRROR STRATEGY, Eq. (30a)
- TRY TO REDUCE BOTH W and Max(Gk), Eq. (32)
- USE JAYA, Eq. (33) or NEW HARMONY, Eq. (31)
CONVERGED? Eq. (34)
NO
YES
END OPTIMIZATION PROCESS

Figure 3.1. Flow chart of the hybrid LSSO–HHSJA algorithm developed in this research.

3.2. Step 1: generation of new trial designs

Let $\boldsymbol{X}_{\mathbf{OPT}}=\{x_{\mathrm{OPT},1},x_{\mathrm{OPT},2},\ldots,x_{\mathrm{OPT,NDV}}\}$ be the best design of population and $\overline{\nabla}W(\boldsymbol{X}_{\mathbf{OPT}})$ the cost function gradient computed at $\boldsymbol{X}_{\mathbf{OPT}}$. A random number $N_{\mathrm{RND,j}}$ in the (0,1) interval is generated for each optimization variable. A recursive cycle is performed for each design variable from 1 to *NDV*.

If $N_{\mathrm{RND,j}}>HMCR$, the new value $x_{\mathrm{TR,j}}$ randomly assigned to the currently perturbed j^{th} design variable (j=1,2,…,NDV) is:

$$\begin{cases}\left(\mathrm{x_{OPT,j}}-\mathrm{x_j^{\,L}}\right)>\left(\mathrm{x_j^{\,U}}-\mathrm{x_{OPT,j}}\right)\&\partial \mathrm{W}/\partial \mathrm{x_j}<0\Rightarrow \mathrm{x_{TR,j}}=\mathrm{x_{OPT,j}}-\mathrm{N_{RND,j}}\cdot\left(\mathrm{x_{OPT,j}}-\mathrm{x_j^{\,L}}\right)\cdot\mu_\mathrm{j}\\ \left(\mathrm{x_{OPT,j}}-\mathrm{x_j^{\,L}}\right)<\left(\mathrm{x_j^{\,U}}-\mathrm{x_{OPT,j}}\right)\&\partial \mathrm{W}/\partial \mathrm{x_j}<0\Rightarrow \mathrm{x_{TR,j}}=\mathrm{x_{OPT,j}}-\mathrm{N_{RND,j}}\cdot\left(\mathrm{x_j^{\,U}}-\mathrm{x_{OPT,j}}\right)\cdot\mu_\mathrm{j}\end{cases} \tag{18}$$

where $\partial \mathrm{W}/\partial \mathrm{x_j}$ is the sensitivity of cost function to the currently perturbed j^{th} design variable while $\mu_\mathrm{j}=(\partial \mathrm{W}/\partial \mathrm{x_j})/\|\overline{\nabla}W(\boldsymbol{X}_{\mathbf{OPT}})\|$ is the normalized sensitivity. Sensitivities $\partial \mathrm{W}/\partial \mathrm{x_j}$ are explicitly available for truss sizing optimization problems (the cost function *W* corresponds to the structural weight) and positive over the whole design space. Hence, using the '–' sign allows to generate trial points lying on descent directions based on gradient information available for the cost function.

Similar to classical HS, the new value $x_{\mathrm{TR,j}}$ is not selected from the $\mathrm{N_{POP}}$ values available in the corresponding column of the [HM] matrix if $N_{\mathrm{RND,j}}>\mathrm{HMCR}$. Eq. (18) is more likely to be used when HMCR takes a small value. Perturbations of optimization variables $(x_{\mathrm{TR,j}}-x_{\mathrm{OPT,j}})$ are weighted by sensitivities $\partial W/\partial x_\mathrm{j}$ to compute cost function variation ΔW_{TR} for the new harmony $\boldsymbol{X}_{\mathbf{TR}}$. This variation is the scalar product of the gradient vector $\overline{\nabla}W(\boldsymbol{X}_{\mathbf{OPT}})$ and the search direction $\boldsymbol{S}_{\boldsymbol{TR}}^{\mathbf{T}}=(\boldsymbol{X}_{\mathbf{TR}}-\boldsymbol{X}_{\mathbf{OPT}})$ defined by the new trial design and the current best record:

$$\Delta \mathrm{W_{TR}}=\mathbf{S}_{\mathrm{TR}}{}^{\mathrm{T}}\overline{\nabla}W(\boldsymbol{X}_{\mathbf{OPT}})=\sum_{\mathrm{j=1}}^{\mathrm{NDV}}(\boldsymbol{X}_{\mathrm{TR,j}}-\boldsymbol{X}_{\mathrm{OPT,j}})\,\partial \mathrm{W}/\partial \mathrm{x_j} \tag{19}$$

If $\Delta W_{\mathrm{TR}}<0$, the $\boldsymbol{S}_{\mathbf{TR}}^{\mathbf{T}}$ vector defines a descent direction. For that purpose, all increments $(x_{\mathrm{TR,j}}-x_{\mathrm{OPT,j}})\cdot\partial W/\partial x_\mathrm{j}$ should be negative. The strategy stated in Eq. (20) is used for retaining or adjusting variable perturbations (j=1,2,…,NDV):

$$\begin{cases}\left(\partial W/\partial x_j\right)\cdot\left(x_{TR,j}-x_{OPT,j}\right)<0\Rightarrow \text{Leave } x_{TR,j} \text{ unchanged}\\ \left(\partial W/\partial x_j\right)\cdot\left(x_{TR,j}-x_{OPT,j}\right)>0\Rightarrow \text{Reset } x_{TR,j} \text{ as } x_{TR,j}'=\left(1+N_{RND,j}\right)\cdot x_{OPT,j}-N_{RND,j}\cdot x_{TR,j}\end{cases} \quad (20)$$

The second relationship is a mirroring strategy to transform a non-descent direction $\boldsymbol{S}_{TR}$ into its opposite, the descent direction $-\boldsymbol{S}_{TR}$. Random numbers $N_{RND,j}<1$ limit variable step sizes reducing the risk that the corrected design may turn infeasible if it tends to reduce cost function too quickly. The scalar product of $\boldsymbol{S}_{TR}$ and the actual descent direction $\boldsymbol{S}_{MIRR}$ defined by the mirroring is $-N_{RND,j}\cdot(x_{TR,j}-x_{OPT,j})^2$, hence rather large if $N_{RND,j}$ tends to unity.

If $N_{RND,j}<HMCR$, regardless of the current value of *PAR*, LSSO–HHSJA defines an interval limited by the two adjacent values $\hat{x}_{TR,j}^{HM,less}$ and $\hat{x}_{TR,j}^{HM,more}$ to the $x_{OPT,j}$ value stored in the current best record $\boldsymbol{X}_{OPT}$, such that $\hat{x}_{TR,j}^{HM,less}<x_{OPT,j}<\hat{x}_{TR,j}^{HM,more}$. The j^{th} variable is updated as (j=1,2,…,NDV):

$$x_{TR,j}=x_{OPT,j}+\left(N_{RND,j}-0.5\right)\cdot \text{Max}\left[\left(x_{OPT,j}-\hat{x}_{TR,j}^{HM,less}\right);\left(\hat{x}_{TR,j}^{HM,more}-x_{OPT,j}\right)\right] \quad (21)$$

By considering the difference $(N_{RND,j}-0.5)$, the value $x_{OPT,j}$ is reduced or increased using Eq. (21). However, the latter may not be the best strategy if sensititivies $\partial W/\partial x_j$ are positive over the whole design space, such as it occurs in the weight minimization problems of truss structures considered in this study. For this reason, if $(x_{TR,j}-x_{OPT,j})\cdot\partial W/\partial x_j<0$, the trial value $x_{TR,j}$ generated with Eq. (21) is retained and checked for pitch adjustment later. Otherwise, if $(x_{TR,j}-x_{OPT,j})\cdot\partial W/\partial x_j>0$, the JAYA's characteristic equation (15) is modified as follows in order to adjust the value of $x_{TR,j}$:

$$x_{TR,j}'=x_{OPT,j}+\beta_{1,j}\left(\hat{x}_{TR,j}^{best}-x_{OPT,j}\right)-\beta_{2,j}\left(\hat{x}_{TR,j}^{worst}-x_{OPT,j}\right) \quad (22)$$

where: $\beta_{1,j}$ and $\beta_{2,j}$ are two random numbers in the interval [0,1]; $\hat{x}_{TR,j}^{worst}=\text{Min}\left[\hat{x}_{TR,j}^{HM,more};x_{TR,j}\right]$ and $\hat{x}_{TR,j}^{best}=\text{Min}\left[\hat{x}_{TR,j}^{HM,less};\left(2\cdot x_{TR,j}-x_{OPT,j}\right)\right]$. In Eq. (22), the absolute value is not necessary for $x_{OPT,j}$ as this is a sizing variable. Basically, LSSO–HHSJA attempts to escape from the "bad" value $x_{TR,j}$ of the j^{th} design variable (i.e. larger than $x_{OPT,j}$) and to approach the "good" value $x_{TR,j}$' (i.e. smaller than $x_{OPT,j}$), which satifies the $(x_{TR,j}'-x_{OPT,j})\cdot\partial W/\partial x_j<0$ condition. The $(2\cdot x_{TR,j}-x_{OPT,j})$ value is obtained by mirroring $x_{TR,j}$ with respect to $x_{OPT,j}$ to transform the nondescent direction $(x_{TR,j}-x_{OPT,j})$ into the descent direction $-(x_{TR,j}-x_{OPT,j})$. Rather than

considering the whole population stored in [HM], the JAYA-based strategy implemented by LSSO–HHSJA limits the search in the $[\hat{x}_{TR,j}^{best};\hat{x}_{TR,j}^{worst}]$ neighborhood of currently best value $x_{OPT,j}$ for the j[th] variable by exploring the descent directions $(\hat{x}_{TR,j}^{best}-x_{OPT,j})$ and $-(\hat{x}_{TR,j}^{worst}-x_{OPT,j})$.

Unlike Ref. [53] where the $\hat{x}_{TR,j}^{HM,less}$ and $\hat{x}_{TR,j}^{HM,more}$ values defining the search interval for $N_{RND,j}<HMCR$ were related to a generic value $\hat{x}_{TR,j}^{HM}$ extracted from the [HM] memory, which may even be very far from the optimum, LSSO–HHSJA always keeps searching for high quality trial designs in the neighborhood of the best design currently stored in the population. This elitist strategy is implemented in Eq. (21) and even more in Eq. (22), which defines descent directions for the JAYA operator.

If $N_{RND,j}<HMCR$ and $N_{RND,j}<PAR$, the $x_{TR,j}$ (or $x_{TR,j}$') value is pitch adjusted as:

$$\left(x_{TR,j}^{pitch,adj}\right)' = x_{TR,j} - N_{RND,j}\cdot\frac{|x_{TR,j}-x_{OPT,j}|}{NG_{tot}}\cdot NG_{pitch,adj} \qquad (j=1,2,\ldots,NDV) \qquad (23)$$

where $NG_{pitch,adj}$ is the total number of pitch adjusted values while NG_{tot} is the total number of generated trial designs. The latter is reset as $(NG_{pitch,adj}+1)$ if the number of pitch adjusted variables included in a new harmony is larger than the number of design variables perturbed with gradient information. Similar to [53], the bandwidth parameter *bw* of classical HS is not necessary. Eq. (23) accounts for optimization history by including the ratio between the number of trial designs $NG_{pitch,adj}$ generated via pitch adjustment and the total number of trial designs NG_{tot} generated until that moment. However, it is enough to consider only the reduction of design variables with respect to the corresponding values stored in $\boldsymbol{X}_{OPT}$.

Since for truss sizing problems it holds $\partial W/\partial x_j>0$ over the whole design space, all values $x_{TR,j}$ or $x_{TR,j}$' or $\left(x_{TR,j}^{pitch,adj}\right)'$ defined with Eqs. (18,20) or (21–23) are forced to be smaller than the corresponding values $x_{OPT,j}$ stored in $\boldsymbol{X}_{OPT}$ in order to lie on descent directions. While this may increase the rate of reduction of structural weight, it may however lead to generate infeasible trial designs if sizing variables are reduced too quickly. For this purpose, another pitch adjusted value is defined for $x_{TR,j}$ or $x_{TR,j}$' using a JAYA-based strategy:

$$\left(x_{TR,j}^{pitch,adj}\right)'' = x_{TR,j} + \beta_{1,j}\left(x_{OPT,j} - x_{TR,j}\right) - \beta_{2,j}\left(x_{2nd-best,j} - x_{TR,j}\right) \qquad (24)$$

where $x_{2nd\text{-}best,j}$ is the value of the j[th] variable stored in the 2[nd] best design of the population. The pitch adjusted value for $x_{TR,j}$ or $x_{TR,j}$' is finally defined as:

$$x_{TR,j}^{pitch,adj} = Median\left\{x_{TR,j} \text{ or } x_{TR,j}';\left(x_{TR,j}^{pitch,adj}\right)';\left(x_{TR,j}^{pitch,adj}\right)''\right\} \tag{25}$$

The median value expressed by Eq. (25) represents a good balance between taking large perturbations of sizing variables along descent directions to achieve a very fast reduction of structural weight (this ability turns very useful in the early optimization iterations especially if the population is dominated by conservative designs) and exploring for high quality solutions that are likely not to violate constraints. For example, the latter may occur if $x_{2nd\text{-}best,j} > x_{OPT,j}$.

Similar to [53], *HMCR* and *PAR* are automatically adapted by LSSO–HHSJA in each iteration as:

$$\begin{cases} HMCR^{q} = HMCR_{extracted}{}^{q} \cdot \dfrac{WHS_{aver,end}{}^{q-1}}{WHS_{aver,init}{}^{q-1}} \cdot \dfrac{NG_{pitch,adj}}{NG_{gradient}} \\ PAR^{q} = PAR_{extracted}{}^{q} \cdot \dfrac{WHS_{aver,end}{}^{q-1}}{WHS_{aver,init}{}^{q-1}} \cdot \dfrac{||\mathbf{X_{OPT,end}} - \mathbf{X_{WORST,end}}||^{q-1}}{||\mathbf{X_{OPT,init}} - \mathbf{X_{WORST,init}}||^{q-1}} \cdot \dfrac{NG_{pitch,adj}}{NG_{gradient}} \end{cases} \tag{26}$$

Where: q denotes the current iteration; $WHS_{aver,init}{}^{q-1}$ and $WHS_{aver,end}{}^{q-1}$, respectively, are the average costs of designs stored in [HM] at the beginning and the end of the previous iteration (rated successful if $WHS_{aver,end}{}^{q-1} / WHS_{aver,init}{}^{q-1} < 1$); $\boldsymbol{X}_{\mathbf{OPT,init}}$ and $\boldsymbol{X}_{\mathbf{WORST,init}}$, $\boldsymbol{X}_{\mathbf{OPT,end}}$ and $\boldsymbol{X}_{\mathbf{WORST,end}}$, respectively, are the best and worst designs at the beginning and the end of the previous iteration. The number of trial designs $NG_{gradient}$ generated using gradient informationis reset to ($NG_{gradient}+1$) if more than NDV/2 design variables are perturbed with Eq. (18) without using the mirroring strategy of Eq. (20).

Random values $HMCR_{extracted}{}^{q}$ and $PAR_{extracted}{}^{q}$ are defined as:

$$\begin{cases} HMCR_{extracted}{}^{q} = 0.01 + \xi_{HMCR} \cdot (0.99 - 0.01) \\ PAR_{extracted}{}^{q} = 0.01 + \xi_{PAR} \cdot (0.99 - 0.01) \end{cases} \tag{27}$$

where ξ_{HMCR} and ξ_{PAR} are two random numbers in the interval (0,1). The bounds of 0.01 and 0.99 set in Eq. (27) allow all possible values of internal parameters to be covered [37]. For q=1, it holds $HMCR^{q}=HMCR_{extracted}{}^{q}$ and $PAR^{q}=PAR_{extracted}{}^{q}$.

It can be seen that *HMCR* and *PAR* tend to decrease more sharply if the cost function is reduced by a great extent. This occurs when the defination of new trial designs is dominated by gradient information and it is easier to satisfy the condition $N_{RND,j} > HMCR$. This scenario

is consistent with small values of the $NG_{pitch,adj}/NG_{gradient}$ ratio. Pitch adjusting becomes less effective for aless sparse population characterized by small values of the $||\mathbf{X}_{OPT,end}-\mathbf{X}_{WORST,end}||/||\mathbf{X}_{OPT,init}-\mathbf{X}_{WORST,init}||$ ratio.

3.3. Step 2: evaluation of trial design $\mathbf{X}_{TR}$ and population updating

LSSO–HHSJA evaluates the new trial design $\boldsymbol{X}_{TR}$ defined in Step 1 by considering the following four cases: (1) $\boldsymbol{X}_{TR}$ feasible & $W(\boldsymbol{X}_{TR})<W_{OPT}$; (2) $\boldsymbol{X}_{TR}$ feasible but $W(\boldsymbol{X}_{TR})>W_{OPT}$; (3) $\boldsymbol{X}_{TR}$ infeasible & $W(\boldsymbol{X}_{TR})<W_{OPT}$; (4) $\boldsymbol{X}_{TR}$ infeasible & $W(\mathbf{X}_{TR})>W_{OPT}$. This is a far more general approach than classical HS where $\boldsymbol{X}_{TR}$ may at most replace only the worst design $\boldsymbol{X}_{worst}$ stored in [HM].

3.3.1. *Case 1: X_{TR} feasible and $W(X_{TR})<W_{OPT}$*

$\boldsymbol{X}_{worst}$ is removed from [HM] and the new trial design $\boldsymbol{X}_{TR}$ is reset as $\boldsymbol{X}_{OPT}$. The former optimum hence becomes the second best design of the population and is hence reset as $\boldsymbol{X}_{2ndBEST}$. The second worst design of the previous population is reset as $\boldsymbol{X}_{worst}$ in the updated population. In [53], the remaining $(N_{POP}-2)$ designs were updated by randomly combining the directions formed by $\boldsymbol{X}_{OPT}$ and the currently selected harmony, $\boldsymbol{X}_{OPT}$ and 2nd best design, $\boldsymbol{X}_{OPT}$ and the harmony corresponding to the largest approximate gradient with respect to $\boldsymbol{X}_{OPT}$.

Since in truss sizing optimization problems the gradients of cost function are constant over the whole design space, the above mentioned process may be significantly simplified and enhanced by implementing a JAYA-based approach. For that purpose, each $(\mathbf{X}_{NPOP-2})^r$ harmony is tentatively updated by LSSO–HHSJA using Eq. (28), with $r\in(N_{POP}-2)$:

$$(\boldsymbol{X}_{\mathrm{NPOP}-2})^{r,new} = (\boldsymbol{X}_{\mathrm{NPOP}-2})^{r} + \boldsymbol{\omega_1}(\boldsymbol{X}_{\mathrm{OPT}} - (\boldsymbol{X}_{\mathrm{NPOP}-2})^{r}) - \boldsymbol{\omega_2}(\boldsymbol{X}_{\mathrm{worst}} - (\boldsymbol{X}_{\mathrm{NPOP}-2})^{r}) \quad (28)$$

where $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_2$ are two vectors of *NDV* random numbers in the interval [0,1]: in particular, $\omega_{1,j}$ and $\omega_{2,j}$ are generated for the j^{th} component of the processed harmony, best and worst designs. Since the goal is to improve the $(\mathbf{X}_{NPOP-2})^r$ harmony, LSSO–HHSJA tries to search along the descent direction $(\mathbf{X}_{OPT}-(\mathbf{X}_{NPOP-2})^r)$ with respect to $(\mathbf{X}_{NPOP-2})^r$ and escape from the worst design of the population $\mathbf{X}_{worst}$, which certainly will not improve $(\mathbf{X}_{NPOP-2})^r$.

If $(\boldsymbol{X}_{NPOP-2})^{r,new}$ is feasible and $W((\boldsymbol{X}_{NPOP-2})^{r,new})<W((\boldsymbol{X}_{NPOP-2})^r)$, it replaces the old harmony $(\boldsymbol{X}_{NPOP-2})^r$ in [HM]. Otherwise, $(\boldsymbol{X}_{NPOP-2})^r$ is retained. The population is reordered

based on the cost of each designs and the next harmony is processed. LSSO–HHSJA finally checks for convergence in Step 4.

3.3.2. *Case 2: X_{TR} feasible but $W(X_{TR})>W_{OPT}$*

The trial design $\boldsymbol{X}_{TR}$ was compared with the rest of the population in [53] by modifying only the ($N_{POP}-p$) candidate designs that ranked below $\boldsymbol{X}_{TR}$. Again, the directions formed by $\boldsymbol{X}_{OPT}$ and the currently selected harmony, $\boldsymbol{X}_{OPT}$ and 2nd best design, $\boldsymbol{X}_{OPT}$ and the harmony corresponding to the largest approximate gradient with respect to $\boldsymbol{X}_{OPT}$ were combined in order to generate each new trial design. The LSSO–HHSJA algorithm developed in this study adopts a much more comprehensive approach. First, a mirroring strategy is used for transforming the non-descent direction ($\boldsymbol{X}_{TR}-\boldsymbol{X}_{OPT}$) into a descent direction. For that purpose, the trial design $\boldsymbol{X}_{\mathrm{TR}}{}^{\mathrm{mirr}}$ is defined as:

$$\boldsymbol{X}_{\mathrm{TR}}{}^{\mathrm{mirr}} = \left(1+\eta_{\mathrm{mirr}}\right)\boldsymbol{X}_{\mathrm{OPT}} - \eta_{\mathrm{mirr}}\boldsymbol{X}_{\mathrm{TR}} \tag{29}$$

where η_{MIRR} is a random number in the interval (0,1), which limits step size to reduce the probability of generating an infeasible trial design. If $\mathrm{W}\left(\boldsymbol{X}_{\mathrm{TR}}{}^{\mathrm{mirr}}\right) > W(\boldsymbol{X}_{\mathrm{TR}})$, $\boldsymbol{X}_{\mathrm{TR}}{}^{\mathrm{mirr}}$ is directly discharged; $\boldsymbol{X}_{\mathrm{TR}}$ is hence compared with the designs stored in [HM]. Conversely, if $\boldsymbol{X}_{\mathrm{TR}}{}^{\mathrm{mirr}}$ is feasible and it holds $\mathrm{W}\left(\boldsymbol{X}_{\mathrm{TR}}{}^{\mathrm{mirr}}\right) < W(\boldsymbol{X}_{\mathrm{TR}})$ (this is likely to occur because the cost function of truss sizing optimization problems is linear), $\boldsymbol{X}_{\mathrm{TR}}{}^{\mathrm{mirr}}$ is compared with the designs stored in [HM].

Similar to case (1), LSSO–HHSJA utilizes Eq. (28) to update the ($N_{POP}-p$) harmonies ranking below $\boldsymbol{X}_{TR}$ or $\boldsymbol{X}_{\mathrm{TR}}{}^{\mathrm{mirr}}$. Each new harmony is then evaluated as explained for case (1). Convergence check is done in Step 4.

3.3.3. *Case 3: X_{TR} infeasible and $W(X_{TR})<W_{OPT}$*

An approximate line search is performed on the descent direction $\boldsymbol{S}_{TR}=(\boldsymbol{X}_{TR}-\boldsymbol{X}_{OPT})$ limited by $\boldsymbol{X}_{OPT}$ and $\boldsymbol{X}_{TR}$. As explained in Ref. [53], three random numbers ξ_1, ξ_2 and ξ_3 in the interval (0,1) are generated to respectively define points $\boldsymbol{X}_{TR,appr}{}^{(1)}=\boldsymbol{X}_{OPT}+\xi_1\boldsymbol{S}_{TR}$, $\boldsymbol{X}_{TR,appr}{}^{(2)}=\boldsymbol{X}_{OPT}+\xi_2\boldsymbol{S}_{TR}$ and $\boldsymbol{X}_{TR,appr}{}^{(3)}=\boldsymbol{X}_{OPT}+\xi_3\boldsymbol{S}_{TR}$ on $\boldsymbol{S}_{TR}$. Optimization constraints are evaluated at these points; by including responses for $\boldsymbol{X}_{OPT}$ and $\boldsymbol{X}_{TR}$, it is possible to fit the 4th order polynomials $G_{k,APP}(\alpha)$ for active constraints. For sizing optimization problems of truss structures, cost function (i.e. structural weight) is linear with respect to sizing variables and it obviously takes its minimum

at α=1, that is at $\boldsymbol{X}_{\mathbf{TR}}$. The algebraic equations $G_{\mathrm{k,APP}}(\alpha)$=0 are solved for all active constraints that turn infeasible moving from $\boldsymbol{X}_{\mathbf{OPT}}$ to $\boldsymbol{X}_{\mathbf{TR}}$. A new trial design is hence defined as $\overline{\mathbf{X}}=\boldsymbol{X}_{\mathbf{OPT}}+Min\{1;\alpha_{\mathrm{MIN}}\}\boldsymbol{S}_{\mathbf{TR}}$ where α_{MIN} is the smallest root found for the $G_{\mathrm{k,APP}}(\alpha)$=0 equations.

If a better design than $\boldsymbol{X}_{\mathbf{OPT}}$ is found, it is stored as the new best record and the worst design $\boldsymbol{X}_{\mathbf{worst}}$ is removed from the population. The second worst design becomes the new worst design and the former best design becomes the second best design. The (N_{POP}−2) designs ranked below $\boldsymbol{X}_{\mathbf{OPT}}$ and $\boldsymbol{X}_{\mathbf{2ndBEST}}$ are analyzed and eventually updated with the JAYA-based strategy, Eq. (28), used for case (1). Convergence check is hence done in Step 4. Unlike Ref. [53], LSSO–HHSJA now updates the whole population each time the approximate line search provides a good trial design $\overline{\boldsymbol{X}}$. This allows to improve more rapidly the candidate designs stored in [HM].

If the approximate line search is unsuccessful, Ref. [53] re-iterated the search by increasing the order of polynomial approximation up to 10 and eventually performed a 1-D probabilistic search based on simulated annealing. However, this process may require too many structural analyses in large scale optimization. For this reason, the LSSO–HHSJA algorithm developed in this study combines a mirroring strategy with a JAYA-based strategy to turn $\boldsymbol{X}_{\mathbf{TR}}$ feasible. For that purpose, two trial designs ($\boldsymbol{X}_{\mathbf{TR}}$)' and ($\boldsymbol{X}_{\mathbf{TR}}$)'' are defined as follows:

$$\begin{cases}\left(\boldsymbol{X}_{\mathrm{TR}}{}^{\mathrm{mirr}}\right)' = \left(1+\eta_{\mathrm{mirr}}\right)\boldsymbol{X}_{\mathrm{OPT}} - \eta_{\mathrm{mirr}}\boldsymbol{X}_{\mathrm{TR}} \\ \left(\boldsymbol{X}_{\mathrm{TR}}{}^{\mathrm{mirr}}\right)'' = \boldsymbol{X}_{\mathrm{TR}} + \boldsymbol{\omega_1}(\boldsymbol{X}_{\mathrm{OPT}} - \boldsymbol{X}_{\mathrm{TR}}) - \boldsymbol{\omega_2}(\boldsymbol{X}_{\mathrm{2ndBEST}} - \boldsymbol{X}_{\mathrm{TR}})\end{cases} \tag{30}$$

where η_{MIRR} is a random number in the interval (0,1) while random vectors $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_2$ are similar to those of Eq. (28). The mirroring strategy used for generating $\left(\boldsymbol{X}_{\mathrm{TR}}{}^{\mathrm{mirr}}\right)'$, Eq. 30(a), steers the search in the opposite direction ($\boldsymbol{X}_{\mathbf{OPT}}$−$\boldsymbol{X}_{\mathbf{TR}}$) to the infeasible direction ($\boldsymbol{X}_{\mathbf{TR}}$−$X_{\mathrm{OPT}}$); however, ($X_{\mathrm{OPT}}$−$\boldsymbol{X}_{\mathbf{TR}}$) is a non-descent direction. Hence, the random number η_{MIRR} has two functions: (i) to reduce step size in order not to increase cost function too much; (ii) if $\boldsymbol{X}_{\mathrm{OPT}}$ lies on or is very close to the boundary of feasible search domain, it may be good to limit step size in order to reduce the probability of generating another infeasible trial design.

The JAYA-based strategy used for generating $\left(\boldsymbol{X}_{\mathrm{TR}}{}^{\mathrm{mirr}}\right)''$, Eq. 30(b), tries to approach $\boldsymbol{X}_{\mathbf{TR}}$ to the current best record by moving along ($\boldsymbol{X}_{\mathbf{OPT}}$−$\boldsymbol{X}_{\mathbf{TR}}$), which is opposite to the infeasible direction ($\boldsymbol{X}_{\mathbf{TR}}$−$\boldsymbol{X}_{\mathbf{OPT}}$). Furthermore, the search is "confined" between the best two candidate

solutions currently stored in [HM], where it may be easier to define high quality trial designs. This would allow at least to minimize the increase in cost function.

The new trial designs $\left(\boldsymbol{X}_{\mathrm{TR}}{}^{\mathrm{mirr}}\right)'$ and $\left(\boldsymbol{X}_{\mathrm{TR}}{}^{\mathrm{mirr}}\right)''$ are evaluated. If both designs are feasible, they are checked against $\boldsymbol{X}_{\mathbf{OPT}}$ according to case (1) and case (2) described above. The same is done if only one between $\left(\boldsymbol{X}_{\mathrm{TR}}{}^{\mathrm{mirr}}\right)'$ and $\left(\boldsymbol{X}_{\mathrm{TR}}{}^{\mathrm{mirr}}\right)''$ is feasible. If no feasible solution is obtained, a trial design between $\boldsymbol{X}_{\mathbf{OPT}}$ and $\boldsymbol{X}_{\mathbf{2ndBEST}}$ is generated as follows:

$$\boldsymbol{X}_{\mathrm{TR}} = \boldsymbol{X}_{\mathrm{OPT}} + \alpha(\boldsymbol{X}_{\mathrm{OPT}} - \boldsymbol{X}_{\mathrm{2ndBEST}}) \tag{31}$$

where α is a random number between 0 and 1. The new trial design will be in all likelihood feasible if both $\boldsymbol{X}_{\mathbf{OPT}}$ and $\boldsymbol{X}_{\mathbf{2ndBEST}}$ are feasible and hence it will be evaluated according to case (1) or case (2).

3.3.4. *Case 4: X_{TR} infeasible and $W(X_{TR})>W_{OPT}$*

This is the worst possible case, although very unlikely to occur because LSSO–HHSJA forms trial designs by perturbing variables so as to move along descent directions. Nevertheless, the present algorithm utilizes a mirroring strategy to transform the non-descent direction ($\boldsymbol{X}_{\mathbf{TR}}$–$\boldsymbol{X}_{\mathbf{OPT}}$) into the descent direction ($\boldsymbol{X}_{\mathbf{TR}}{}^{\mathbf{mirr}}$–$\boldsymbol{X}_{\mathbf{OPT}}$). Furthemore, ($\boldsymbol{X}_{\mathbf{TR}}$–$\boldsymbol{X}_{\mathbf{OPT}}$) is also an unfeasible direction and hence LSSO–HHSJA tries to move along ($\boldsymbol{X}_{\mathbf{TR}}{}^{\mathbf{mirr}}$–$\boldsymbol{X}_{\mathbf{OPT}}$) to recover such a gap. The new trial design $\boldsymbol{X}_{\mathbf{TR}}{}^{\mathbf{mirr}}$ is defined as:

$$\boldsymbol{X}_{\mathrm{TR}}{}^{\mathrm{mirr}} = (1 + \eta_{\mathrm{mirr}})\boldsymbol{X}_{\mathrm{OPT}} - \eta_{\mathrm{mirr}}\boldsymbol{X}_{\mathrm{TR}} \tag{30a rep.}$$

The new harmony $\mathbf{X}_{\mathbf{TR}}{}^{\mathbf{mirr}}$ is evaluated. The best record and the population are updated as explained above if case (1) or case (2) or case (3) occurs. Convergence check is then done in Step 4.

If $\boldsymbol{X}_{\mathbf{TR}}$ and $\boldsymbol{X}_{\mathbf{TR}}{}^{\mathbf{mirr}}$ are infeasible and $Min\{W(\boldsymbol{X}_{\mathbf{TR}}); W(\boldsymbol{X}_{\mathbf{TR}}{}^{\mathbf{mirr}})\} > W_{\mathrm{OPT}}$, their positions are updated by reducing distances from $\boldsymbol{X}_{\mathbf{OPT}}$ and in all likelihood constraint violations. That is:

$$\begin{cases} \left(\boldsymbol{X}_{\mathrm{TR}}{}^{\mathrm{mirr}}\right)' = \boldsymbol{X}_{\mathrm{OPT}} + \left(\boldsymbol{X}_{\mathrm{TR}}{}^{\mathrm{mirr}} - \boldsymbol{X}_{\mathrm{OPT}}\right)\left(\frac{G_{\mathrm{LIM}}}{Max\{G_{\mathrm{k}}\}}\right) \\ (\boldsymbol{X}_{\mathrm{TR}})' = \boldsymbol{X}_{\mathrm{OPT}} + (\boldsymbol{X}_{\mathrm{TR}} - \boldsymbol{X}_{\mathrm{OPT}})\left(\frac{G_{\mathrm{LIM}}}{Max\{G_{\mathrm{k}}\}}\right) \end{cases} \tag{32}$$

where: k=1,…,NC_{act} is the number of violated constraints; Max{G_k}is the largestnodal displacement or element stress (including buckling) computed from structural analysis and G_{LIM} is the corresponding allowable limit.

In Ref. [53], Eq. (32) was re-iterated until at leastone trial point among ($\boldsymbol{X}_{TR}$)' and ($\boldsymbol{X}_{TR}^{mirr}$)' became feasible and ultimately a 1-D simulated annealing search in the neighborhood of $\mathbf{X}_{OPT}$ was carried out to locally improve designs included in [HM]. A different strategy is adopted by LSSO–HHSJA to reduce the number of structural analyses entailed by this phase. If ($\boldsymbol{X}_{TR}$)'or ($\boldsymbol{X}_{TR}^{mirr}$)' is feasible, the new design is ranked with respect to the current population as described for cases (1) and (2), where the latter case (i.e. $W(\boldsymbol{X}_{TR})>W_{OPT}$) is much more likely to occur. If both ($\boldsymbol{X}_{TR}$)' and ($\boldsymbol{X}_{TR}^{mirr}$)' are infeasible, the best design with the lowest constraint violation amongst $\boldsymbol{X}_{TR}$, ($\boldsymbol{X}_{TR}$)' and ($\boldsymbol{X}_{TR}^{mirr}$)' is set as $\boldsymbol{X}_{TR,WORST}$ and a JAYA-based equation is used for generating yet another trial design:

$$(\boldsymbol{X}_{\mathrm{TR}})^{\mathrm{JAYA}} = \boldsymbol{X}_{\mathrm{TR}} + \boldsymbol{\omega_1}(\boldsymbol{X}_{\mathrm{OPT}} - \boldsymbol{X}_{\mathrm{TR}}) - \boldsymbol{\omega_2}\left(\boldsymbol{X}_{\mathrm{TR,WORST}} - \boldsymbol{X}_{\mathrm{TR}}\right) \tag{33}$$

where random vectors $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_2$ are similar to those of Eqs. (28,30). Basically, LSSO–HHSJA attempts first to simultaneously reduce constraint violation and cost function with Eqs. (30a,22). Should this not work, Eq. (33) moves $\boldsymbol{X}_{TR}$ towards $\boldsymbol{X}_{OPT}$ by escaping from the infeasible region of search space. If $(\boldsymbol{X}_{\mathrm{TR}})^{\mathrm{JAYA}}$ also turns infeasible and the population has at least one infeasible design, the trial solution with the lowest constraint violation amongst $\boldsymbol{X}_{TR}$, ($\boldsymbol{X}_{TR}$)', ($\boldsymbol{X}_{TR}^{mirr}$)' and $(\boldsymbol{X}_{\mathrm{TR}})^{\mathrm{JAYA}}$ is compared with the infeasible designs stored in [HM]. Conversely, if $\boldsymbol{X}_{TR}$, ($\boldsymbol{X}_{TR}$)', ($\boldsymbol{X}_{TR}^{mirr}$)' and $(\boldsymbol{X}_{\mathrm{TR}})^{\mathrm{JAYA}}$ violate constraints but [HM] has only feasible designs, the four trial points are discharged and a new trial design between $\boldsymbol{X}_{OPT}$ and $\boldsymbol{X}_{2ndBEST}$ is generated using Eq. (31); the corresponding operations described in Section 3.2.3 are then carried out.

3.4. Step 3: check for convergence

Standard deviations on variables and cost functions of the candidate designs stored in [HM] decrease as LSSO-HHSJA approaches the global optimum. Hence, the present algorithm normalizes standard deviations with respect to the average design $\boldsymbol{X}_{\mathbf{aver}} = \left(\sum_{\mathrm{k}=1}^{N_{\mathrm{POP}}} \boldsymbol{X}_{\mathbf{k}}\right)/N_{\mathrm{POP}}$ (the generic component is $x_{\mathrm{aver,j}} = \left(\sum_{\mathrm{k}=1}^{N_{\mathrm{POP}}} x_{\mathrm{j}}{}^{\mathrm{k}}\right)/N_{\mathrm{POP}}$) and the average weight $W_{\mathrm{aver}} = \left(\sum_{\mathrm{k}=1}^{N_{\mathrm{POP}}} W(\boldsymbol{X}_{\mathbf{k}})\right)/N_{\mathrm{POP}}$. The termination criterion is:

$$\text{Max}\left\{\begin{matrix}\frac{\text{STD}\{\|\boldsymbol{X_1}-\boldsymbol{X_{aver}}\|,\|\boldsymbol{X_2}-\boldsymbol{X_{aver}}\|,\ldots,\|\boldsymbol{X_{NPOP}}-\boldsymbol{X_{aver}}\|\}}{\|\boldsymbol{X_{aver}}\|} \\ \frac{\text{STD}\{W_1,W_2,\ldots,W_{\text{NPOP}}\}}{W_{\text{aver}}}\end{matrix}\right\} \leq \varepsilon_{\text{CONV}} \quad (34)$$

where the convergence limit $\varepsilon_{\text{CONV}}$ is 10^{-15}, less than the double precision limit of available computers. Steps 1 through 3 are repeated until the LSSO–HHSJA algorithm converges to the global optimum.

3.5. Step 4: terminate optimization process

LSSO-HHSJA terminates optimization process and writes output data in the results file.

CHAPTER 4

TEST PROBLEMS, RESULTS AND DISCUSSION

4.1. Statement of the optimization problem

A skeletal structure is an ensemble of one-dimensional elements connected at point nodes. Truss and frame structures are commonly modelled as skeletal structures. Design optimization of skeletal structures is an important field of engineering under continuous development. In sizing optimization problems, structures must be designed with respect to the size of members (i.e. cross-sectional area values, length and thickness of cross-section segments, etc.). In layout optimization problems, spatial coordinates of nodes are included as design variables. In topology optimization problems, the number of elements can change in the optimization process based on the local distribution of stiffness/compliance required to structures.

Whilst spatial coordinates can be coded as real design variables, member sizes to be used in real engineering applications should always be selected from a database of commercially available sections/dimensions. In the latter case, the corresponding design variables should be coded as integer/discrete variables. Therefore, the optimization problem formulated for a skeletal structure actually is a mixed problem including a set of continuous design variables and a set of discrete/integer variables.

The most common objective in design optimization of skeletal structures is to minimize the weight of the structure under design constraints on nodal displacements, member stresses, critical buckling loads, natural frequencies, etc. The cost function corresponding to the weight of the structure can be explicitly evaluated as it is represented by a summation over products between the cross-sectional area (eventually, the cross-sectional area is comprised of different segments) and the length of each element. However, optimization constraints are always stated in implicit form for a statically undetermined structure. Furthermore, non-linearity introduced by multiple loading conditions may result in the presence of local minima especially when there are many design variables.

The sizing optimization problem for a truss structure with *NOD* nodes (k=1,2,…,*NOD*) and *NEL* elements (j=1,2,…,*NEL*), subject to *NLC* independent loading conditions (*ilc*=1,2,…,*NLC*), can be stated as a weight minimization problem:

$$\text{Minimize } W(\boldsymbol{X}) = \rho g \sum_{\mathrm{j}=1}^{NEL} l_{\mathrm{j}} x_{\mathrm{j}} \tag{35}$$

$$\text{Subject to } \begin{cases} u^{\mathrm{L}}_{(\mathrm{x,y,z}),\mathrm{k}} \le u_{(\mathrm{x,y,z}),\mathrm{k,ilc}} \le u^{\mathrm{U}}_{(\mathrm{x,y,z}),\mathrm{k}} \\ \sigma^{\mathrm{L}}_{\mathrm{j}} \le \sigma_{\mathrm{j,ilc}} \le \sigma^{\mathrm{U}}_{\mathrm{j}} \\ x^{\mathrm{L}}_{\mathrm{j}} \le x_{\mathrm{j}} \le x^{\mathrm{U}}_{\mathrm{j}} \end{cases}$$

where:

- x_j is the cross-sectional area of the j^{th} element of the truss, included as a sizing design variable, ranging between its lower bound x_j^L and upper bound x_j^U;
- l_j is the length of the j^{th} element of the structure;
- g is the gravity acceleration (9.81 m/s^2) and ρ is the material density. The g term must not be considered if structural weight is expressed in kg as it was done in the present study;
- $u_{(x,y,z),k,ilc}$ are the displacements of the k^{th} node in the coordinate directions, varying between the lower limit $u^L_{(x,y,z),k}$ and the upper limit $u^U_{(x,y,z),k}$;
- $\sigma_{j,ilc}$ is the stress in the j^{th} element, varying between σ_j^L (compressive stress limit accounting also for bucking strength) and σ_j^U (allowable tension limit);
- ilc indicates the ilc^{th} loading condition acting on the structure. Constraints on nodal displacements, element stresses and buckling strengths are normalized with respect to their corresponding limits.

A large scale truss structure is comprised of hundreds or thousands of elements, which may be categorized in groups in order to reduce the number of sizing variables. Hence, a large scale optimization problem usually counts at least 200 design variables. Since the structure must withstand several independent loading conditions, the optimization problem has several thousands of nonlinear constraints on nodal displacements, element stresses (including buckling strength). Here, a planar 200-bar truss subject to five independent loading conditions (200 sizing variables and 3500 nonlinear constraints), a spatial 1938-bar tower subject to three independent loading conditions (204 sizing variables and 20700 nonlinear constraints), and a spatial 3586-bar tower subject to three independent loading conditions (280 sizing variables and 37374 nonlinear constraints) are optimized.

4.2. Implementation of the LSSO−HHSJA algorithm and rationale of the comparison with other optimizers

The proposed algorithm was implemented in the Fortran programming language. A standard Fortran compiler was utilized for this purpose.The finite element analyses of truss structures entailed by optimization runs were performed by means of another Fortran routine developed by the authors. The linear system {F}=[K]{u} formed by nodal forces, global stiffness matrix and nodal displacements for each loading condition was solved by inverting the stiffness matrix [K] with a classical matrix triangularization algorithm. The main program perturbs

design as explained in Section 3 and calls the structural analysis routine each time a new trial solution has to be evaluated.

The optimized designs obtained by LSSO–HHSJA were compared with the best solutions quoted in the literature for each test problem. In particular, the following algorithms were considered: (i) other HS variants such as the hybrid HS algorithms with line search strategy of Refs. [51,53], the adaptive HS (AHS) algorithm of Ref. [38], the self-adaptive HS (SAHS) algorithm of Ref. [39]; (ii) other JAYA variants like the improved and parameterless JAYA algorithms of Refs. [54–56]; (iii) the multi-level and multi-point simulated annealing (CMLPSA) of Ref. [51] and the hybrid fast simulated annealing (HFSA) of Ref. [53]; (iv) the hybrid big bang – big crunch (hybrid BBBC) algorithms with line search strategies of Refs. [51,53]; (v) the BBBC algorithm with upper bound search strategy (BBBC-UBS) of Ref. [173]; (vi) the sinusoidal differential evolution (SinDE) algorithm of Ref. [124]; (vii) the SQP optimization routine of MATLAB [178] and the SQP/SLP optimization routines of DOT [179]. The selected metaheuristic algorithms are similar to LSSO–HHSJA because they include parameter adaptation and strategies to generate trial designs in all likelihood better than thecurrent candidate solutions stored in the population. SQP (sequential quadratic programming) and SLP (sequential linear programming), respectively are the best and the simplest gradient-based optimization algorithmsand hence turn very useful in evaluate how fast LSSO–HHSJA reduces structural weight or constraint violation once the initial population is given.

All algorithms were coded in the Fortran programming language following the indications given in literature by their developersin order to have a homogeneous basis of comparison with LSSO–HHSJA. Commercial optimizers were executed in their recommended software environments. When SLP/SQP could not directly provide a good design for some test problem, they were alternatively executed in a cascade optimization. For each population size, twenty independent optimization runs starting from different populations were performed for LSSO–HHSJA and the other metaheuristic optimizers in order to account for their stochastic nature. Initial points selected for CMLPSA, SQP and SLP coincided with the best/average/worst designs included in the initial populations generated for each independent run of LSSO–HHSJA. Uniform initial designs with sizing variables all set to their upper or lower bounds also were considered for gradient based optimization.

Population size of LSSO–HHSJA was determined from sensitivity analysis while, for adaptive HS [38,39], BBBC-UBS [173] and sinDE [124], it was set as N_{POP}=20 or 50 to limit the theoretical computational cost of these algorithms to $N_{POP}\times ITER_{max}$ structural analyses. The latter values are fully consistent with the population size values indicated in the above mentioned references. The N_{POP}=20 value was also chosen for improved/parameterless JAYA, actually insensitive to population size in truss optimization problems (see results reported in Refs. [54–56]). It should be noted that the parameterless JAYA algorithm of Ref. [56] was not executed using the suggested initial population size of $10\times NDV$, because the generation of each population in the independent runs would have required between 2000 and 2800 structural analyses, which is up to 30% of the computational costsquoted in the literature for the selected design examples. Such a choice allowed computational cost of structural optimization to be limited by a significant extent.

Optimum designs were rated feasible if they fully satisfied design constraints. Although LSSO–HHSJA does not require penalty function (similar to the CMLPSA, HFSA, hybrid HS and hybrid BBBC with line search algorithms of Refs. [51,53]), the option of using a static penalty function strategy with a constant penalty factor throughout the optimization process was also made available to the user. Penalty factor was varied from 0 (i.e. original LSSO–HHSJA formulation with no penalty functions) to 10^{20} (i.e. $0,10^0=1,10^1,10^2,\ldots,10^{20}$). The very large values of penalty factor prevent inefficient metaheuristic search engines to converge to a feasible optimized design. Remarkably, standard deviation on optimized weight never exceeded 10^{-4} of the target structural weight for all test problems, thus proving the LSSO–HHSJA's insensitivity to constraint handling strategy. The results tables given in the rest of this section refer to the standard case without penalty function.

4.3. Planar 200-bar truss structure

The planar 200-bar truss structure shown in Figure 4.1 was optimized with 200 sizing variables corresponding to the cross-sectional areas of each element. The structure is subject to five independent loading conditions:

a) 4449.741N (i.e. 1000 lbf) in the positive X-direction at nodes 1, 6, 15, 20, 29, 34, 43, 48, 57, 62, 71;

b) 44.497 kN (i.e. 10000 lbf) in the negative Y-direction at nodes 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.

d) 4449.741 N (i.e. 1000 lbf) in the negative X-direction at nodes 5, 14, 19, 28, 33, 42, 47, 56, 61, 70, 75;

e) Loading conditions b) and d) acting together.

This test problem has 3500 non-linear constraints on nodal displacements and element stresses. The displacements of all free nodes in both coordinate directions X and Y must be less than ±1.27 cm (i.e. ±0.5 in). The allowable stress (the same in tension and compression) is 206.91 MPa (i.e. 30000 psi). All non-linear constraints were independently evaluated and no constraint grouping was adopted. The same was done for all test problems considered in this study.Cross–sectional areas vary between 0.64516 cm^2 (i.e. 0.1 in^2) and 645.16 cm^2 (i.e. 100 in^2).

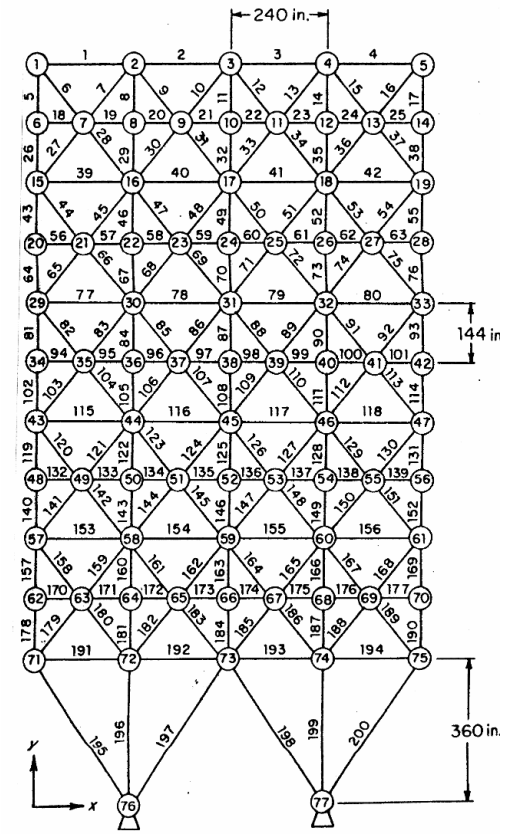

Figure 4.1. Schematic of the planar 200-bar truss structure.

In order to make a direct comparison with the hybrid HS algorithm of Ref. [51], LSSO–HHSJA optimizations were executed also for different population sizes: respectively, N_{POP}=20, 50, 100, 200, 500 and 1000. The same was done for the other hybrid HS algorithm of Ref. [53] compared with LSSO–HHSJA. Table 4.1 shows that all HS variants were practically insensitive to population size. However, only the present algorithm always converged to feasible designs while the other hybrid HS variants found optimal solutions that slightly violate displacement constraints. The robustness of LSSO–HHSJA is confirmed by the fact that it is not possible to establish any direct relationship between population size and computational cost of the optimization process in terms of required number of structural analyses.

Table 4.1. Sensitivity of LSSO–HHSJA and other hybrid HS variants' convergence behavior to population size for the 200-bar truss problem.

N_{POP}	**Structural weight** (kg)	**Structural analyses**	**Constraint tolerance** (%)
20	**12483.673**	**5562**	**Feasible**
	12490.377*	5668*	0.00735*
	12489.460♦	5716♦	0.003451♦
50	**12483.563**	**5734**	**Feasible**
	12490.482*	6604*	0.00750*
	12489.457♦	6040♦	0.003250♦
100	**12484.135**	**6096**	**Feasible**
	12490.542*	6360*	0.00750*
	12489.778♦	5621♦	0.002802♦
200	**12483.982**	**5436**	**Feasible**
	12490.332*	5679*	0.00730*
	12489.700♦	5831♦	0.003168♦
500	**12483.339**	**5637**	**Feasible**
	12490.414*	5940*	0.00643*
	12489.619♦	6372♦	0.003524♦
1000	**12484.054**	**6373**	**Feasible**
	12490.427*	5938*	0.00560*
	12489.564♦	6217♦	0.003329♦

* Results reported in Ref. [51]
♦ Results obtained using the hybrid HS algorithm of Ref. [53]

Table 4.2 shows the optimization results obtained by LSSO–HHSJA and its competitors in the 200-bar truss problem.The results of the independent optimization runs that provided the lowest and the highest structural weights for each algorithm are respectively denotedby "Best" and "Worst" in the table. Furthermore,results of the optimization runs completed within the smallest and largest number of structural analyses are respectively

denoted by “Fastest” and “Slowest” in the table. The same nomenclature has been utilized for all results tables presented in the article.

The present algorithm clearly outperformed the other metaheuristic methods because it achieved the lowest structural weight and always converged to feasible solutions. LSSO–HHSJA was the best optimizer overall because it designed the lightest structure (weighing 12483.339 kg) within only 5637 analyses. The other algorithms found heavier designs in their best optimization runs and completed those runs within at least 5679 analyses. CMLPSA also converged to feasible designs in all optimization runs but its best weight was almost 10 kg heavier than the one found by LSSO–HHSJA. The worst optimization runs of JAYA, adaptive HS variants, BBBC-UBS, sinDE and SQP-MATLAB converged to feasible solutions but the corresponding structural weights were between 18-19 kg (for BBBC-UBS, JAYA and SQP-MATLAB) and 57-293 kg (for sinDE and adaptive HS variants) heavier than the worst design found by the present algorithm.

Table 4.2. Optimization results obtained for the 200-bar truss design example.

	Optimized weight (kg)	**Number of structural analyses**	**Constraint violation** (%)
LSSO–HHSJA Present	*Best*: 12483.339 *Worst*: 12484.135 *Mean*: 12483.791 *STD*: 0.3144	*Best*: 5637 *Fastest:* 5436 *Slowest*: 6373 *Mean/STD*: 5806±357	Feasible
Hybrid HS with LS [51]	*Best*: 12490.332 *Worst*: 12490.542 *Mean*: 12490.430 *STD*: 0.07470	*Best*: 5679 *Fastest:* 5668 *Slowest*: 6604 *Mean/STD*: 6031±377	*Best*: 0.00730 *Worst*: 0.00750 *Mean*: 0.00695 *STD*: 0.000772
Hybrid HS with LS [53]	*Best*: 12489.457 *Worst*: 12489.778 *Mean*: 12489.596 *STD*: 0.1291	*Best*: 6040 *Fastest:* 5621 *Slowest*: 6372 *Mean/STD*: 5966±324	*Best*: 0.003250 *Worst*: 0.002802 *Mean*: 0.003254 *STD*: 0.0002565
Hybrid BBBC with LS [51]	*Best*: 12490.439 *Worst*: 12490.932 *Mean*: 12490.680 *STD*: 0.1773	*Best*: 7745 *Fastest: 1924* *Slowest*: 9460 *Mean/STD*: 5652±2912	*Best*: 0.00559 *Worst*: 0.00625 *Mean*: 0.00626 *STD*: 0.000347
CMLPSA [51]	*Best*: 12492.888 *Worst*: 12493.290 *Mean*: 12493.081 *STD*: 0.2014	*Best*: 11726 *Fastest:* 10338 *Slowest*: 12118 *Mean/STD*: 11394±935	Feasible
Improved/parameterless JAYA [54-56]	*Best*: 12490.603 *Worst*: 12502.175 *Mean*: 12494.460 *STD*: 3.056	*Best*: 12869 *Fastest:* 12316 *Slowest*: 14326 *Mean/STD*: 12818±601	*Best*: 0.03490 *Worst*: Feasible *Mean*: 0.01888 *STD*: 0.01644
AHS [38]	*Best*: 12497.475 *Worst*: 12777.339 *Mean*: 12567.441 *STD*: 174.716	*Best*: 16981 *Fastest:* 15063 *Slowest*: 21412 *Mean/STD*: 17130±2996	*Best*: 0.1570 *Worst*: Feasible *Mean*: 0.08363 *STD*: 0.06836
SAHS	*Best*: 12495.939 *Worst*: 12669.384	*Best*: 15812 *Fastest:* 13384	*Best*: 0.1824 *Worst*: Feasible

[39]	*Mean*: 12553.754 *STD*: 144.817	*Slowest*: 17464 *Mean/STD*: 15618±1681	*Mean*: 0.09718 *STD*: 0.07942
BBBC-UBS [173]	*Best*: 12490.035 *Worst*: 12502.346 *Mean*: 12497.562 *STD*: 4.170	*Best*: 15250 *Fastest:* 13865 *Slowest*: 16198 *Mean/STD*: 14795±1141	*Best*: 0.05540 *Worst*: Feasible *Mean*: 0.03048 *STD*: 0.02837
sinDE [124]	*Best*: 12502.536 *Worst*: 12541.640 *Mean*: 12520.999 *STD*: 16.369	*Best*: 21653 *Fastest:* 17124 *Slowest*: 22635 *Mean/STD*: 20766±2472	*Best*: 0.05880 *Worst*: Feasible *Mean*: 0.03030 *STD*: 0.03017
SQP-MATLAB	*Best*: 12491.400 *Worst*: 12503.300 *Mean*: 12498.034 *STD*: 5.661	*Best*: 28198 *Fastest:* 24619 *Slowest*: 38410 *Mean/STD*: 30990±5957	*Best*: 0.05131 *Worst*: Feasible *Mean*: 0.03298 *STD*: 0.03969

The weight reduction achieved by LSSO–HHSJA with respect to the hybrid HS algorithm described in [51] was only 0.0560% but the present algorithm completed the optimization process within less structural analyses and, as mentioned above, it converged to a feasible solution. The other hybrid HS variant described in [53] also was implemented for this test problem and achieved a better design than the one quoted in [51]: however, the optimized weight was practically the same (i.e. only 0.007% reduction) and the corresponding optimal solution remained infeasible although constraint violation was reduced by about 50%. The computational cost of the optimization did not change much passing from the hybrid HS variants of Refs. [51,53] to the present algorithm. In fact, Table 4.2 shows that LSSO–HHSJA saved on average only 225 structural analyses with respect to Ref. [51] and only 160 analyses with respect to Ref. [53], that is about 3.5% of the computational cost of the optimization. However, it must be remarked once again that only LSSO–HHSJA could find feasible solutions in all optimization runs.

As far as it concerns the adaptive HS variants compared with LSSO–HHSJA, Table 4.2 confirms that using line search strategies which generate trial designs lying on descent directions is far more effective than adapting only the *HMCR* and *PAR* parameters in the context of a classical HS formulation. In fact, the best designs found by AHS [38] and SAHS [39] exhibited the largest constraint violations amongst the different methods as well as the largest structural weights for the worst optimization runs. A very important point is to limit the number of descent directions utilized to form high quality trial designs and update population in the current iteration. The JAYA-based equations (14,16,19) combined by LLSO–HHSJA with the line search based HS architecture greatly help to accomplish these tasks. Furthermore, JAYA naturally tries to avoid low quality designs and this reduces the probability of dealing with infeasible regions of search space. Finally, Eqs. (14,16,19) make it unnecessary to perform 1-D probalistic searches based on SA if trial designs are worse than

the current best record. Indeed, these searches were significantly reduced passing from the hybrid HS formulation [51] to its enhanced version [53]. However, they still played some role in the optimization process thus complicating the practical implementation of the algorithm.

Interestingly, improved/parameterless JAYA [54-56] ranked right after LSSO–HHSJA, hybrid HS and BBBC variants [51,53] and CMLPSA [51]. In fact, while most of the algorithms listed in Table 4.2 converged to structural weights between 12483.3 and 12492.9 kg, JAYA's average constraint violation was only 0.01888%, much less than for BBBC-UBS [173] (0.03048%), sinDE [124] (0.03030%) and adaptive HS [38,39] (between 0.08363 and 0.09718%). This confirms the inherent ability of JAYA to move towards good regions of search space but also that a sufficient number of descent directions must be considered in order to form new trial designs of very high quality in each iteration.

LSSO–HHSJA ranked 2nd overall after hybrid BBBC [51] in terms of average number of required structural analyses but BBBC's computational cost increased by almost five times passing from N_{POP}=20 to 1000: in particular, the slowest optimization run of hybrid BBBC, corresponding to N_{POP}=1000, took 9460 structural analyses vs. only 6373 analyses of LSSO–HHSJA (just one more analysis than the hybrid HS variant of Ref. [53]). CMLPSA [51], JAYA [54–56] and BBBC-UBS [173] ranked after the hybrid HS and hybrid BBBC algorithms and were on average two times slower than LSSO–HHSJA. Adaptive HS [38,39] variants and sinDE [124] were the slowest metaheuristic methods and required on average from 15618 to 20766 structural analyses vs. only 5806 analyses of the present algorithm. SQP-MATLAB was considerably slower than the other optimizers and required five times more analyses than LSSO–HHSJA. Such a behavior was seen regardless of starting SQP optimizations from very conservative or unconservative designs and can be explained with the inherent complexity of the 200-bar truss problem that makes it difficult for SQP to solve the approximate sub-problems built in each iteration.

The present algorithm was slightly less robust in terms of optimized weight than the other HS, BBBC and SA methods analyzed in Refs. [51,53] but considerably less sensitive to initial population/design than all other optimizers including SQP-MATLAB. However, the standard deviation on optimized weight was always less than 1.4% of the best design for all algorithms compared in this study. LSSO–HHSJA obtained the 2nd lowest standard deviation on the required number of structural analyses after the hybrid HS variant of Ref. [53] but the referenced algorithm obtained slightly heavier designs that violate displacement constraints

on average by 0.003254%. Hence, the present algorithm is also the most robust optimizer overall for the 200-bar truss design problem.

Figure 4.2 compares the convergence curves recorded for the best optimization runs of LSSO–HHSJA and its competitors. For the sake of clarity, the plot is limited to the first 13000 structural analyses, and the 8000-128000 kg structural weight range is selected for the Y-axis. Second, the curves relative to hybrid HS variants [51,53] are averaged. Last, the figure includes only the plot relative to the best adaptive HS method [38,39].

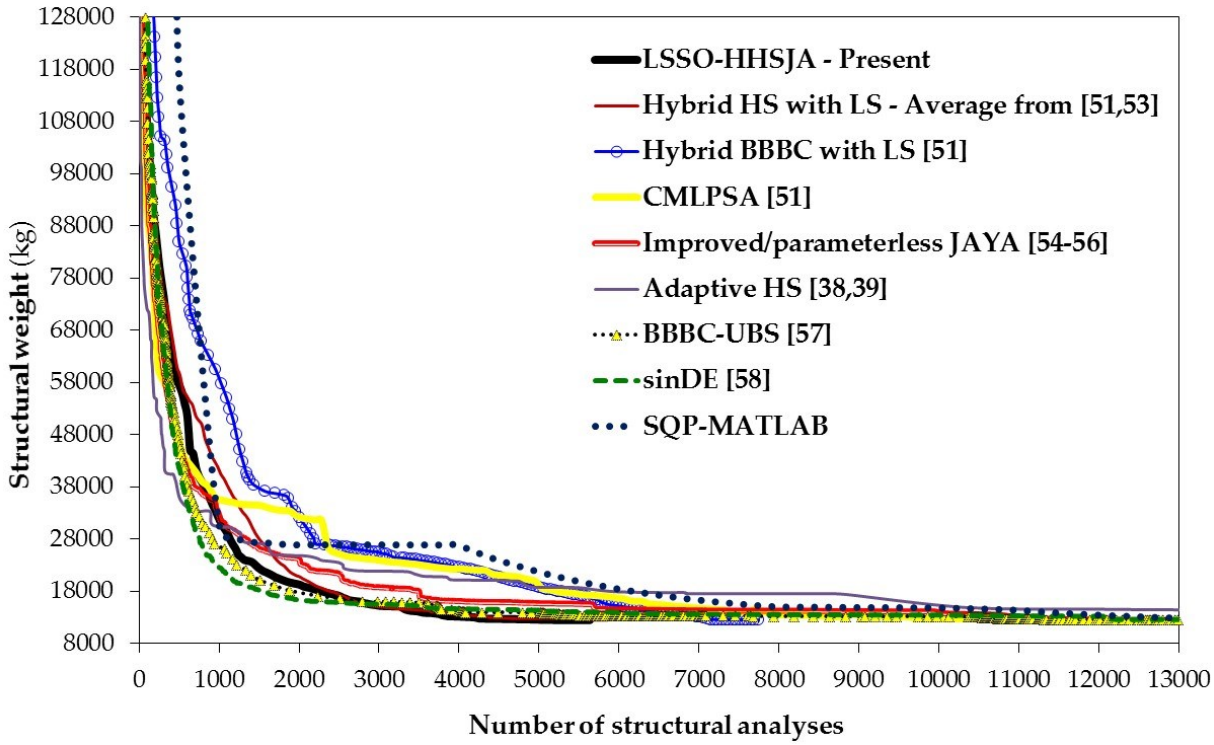

Figure 4.2. Convergence curves obtained for the 200-bar truss design example.

It can be seen from the figure that all methods started their optimizations from initial populations including best individual/designs about 16 times heavier than the target optimization weight of about 12490 kg. However, structural weight was drastically reduced (by about 150,000 kg) within the first 400 structural analyses by all optimizers except hybrid BBBC with line search strategy [51]. CMLPSA [51] also utilizes line search to perturb design but the set of descent directions considered for this purpose delivers only one trial design at a time while LSSO–HHSJA, hybrid HS and hybrid BBBC [51,53] operate on a population of candidate solutions. Adaptive HS [38,39], BBBC-UBS [173] and improved/parameterless JAYA [54–56] were even faster than LSSO–HHSJA between 400 and 3200 analyses but then had to penalize weight for recovering constraint violations while the present algorithm

always conducted its search in the feasible design space and generated the best intermediate designs amongst all methods until the end of the optimization process.

In summary, the results presented for the 200-bar problem demonstrate with no shadow of doubt that the LSSO–HHSJA's selection of good descent directions is much more effective than (i) considering a larger set of search directions that however explore the fraction of design space covered by the current population in less detail (i.e. LSSO–HHSJA vs. hybrid HS/BBBC with line search [51,53]) or (ii) simply escaping from the worst design and approaching the best design stored in the current population (i.e. LSSO–HHSJA vs. improved/parameterless JAYA [54–56]). Hence, the hybrid algorithmic formulation implemented by LSSO–HHSJA is very effective.

4.4. Spatial 1938-bar tower

The second design example considered in this study is the weight minimization of the spatial 1938-bar truss tower with 481 nodes shown in Figure 4.3. The tower is 285 m tall; its layout section is a regular dodecagon at the ground level and a square at the top segment.The numbering of nodes proceeds from structure's top to bottom as follows: (i) Segment 1 ends into the top node 1; (ii) 25 sets of 4 nodes each, from 2-3-4-5 to 98-99-100-101, belonging to segments 1 and 2; (iii) 16 sets of 8 nodes each, from 102-103-104-105-106-107-108-109 to 222-223-224-225-226-227-228-229, belonging to Segment 3; (iv) 21 sets of 12 nodes each, from 230-231-232-233-234-235-236-237-238-239-240-241 to 470-471-472-473-474-475-476-477-478-479-480-481, belonging to Segment 4. The nomenclature of segments included in the tower is detailed in the appendix.This large-scale optimization problem includes 204 sizing variables and was originally presented in [53,54]. The Young's modulus E of the material is 68.971 GPa while the mass density is 2767.991 kg/m^3. Structural symmetry allows to categorize bars into 204 groups (see Table A1 of the appendix) each of which includes elements with the same cross-sectional area taken as a sizing variable.

The tower must carry the three independent loading conditions listed in Table 4.3. The first loading condition presents concentrated forces at all free nodes acting downward; force magnitude increases from top to bottom of the structure: the total load acting on the tower is more than 20 times larger than the gravity load corresponding to the target structural weight. The second loading condition includes concentrated forces acting in the X-direction; forces applied to the left side of the tower (for example, at nodes 98, 101, 228 etc.) act in the positive

X-direction while forces applied to the right side of the tower (for example, at nodes 99, 100, 224 etc.) act in the negative X-direction: hence, the resultant forces of this loading condition bend the tower rightwards. The third loading condition includes concentrated forces acting in the Y-direction; forces applied to the front side of the tower (for example, at nodes 100, 101, 226 etc.) act in the positive Y-direction while forces applied to the rear side of the tower (for example, at nodes 98, 99, 222 etc.) act in the negative Y-direction: hence, the resultant forces of this loading condition tend to compress the tower about the XZ symmetry plane of the structure.

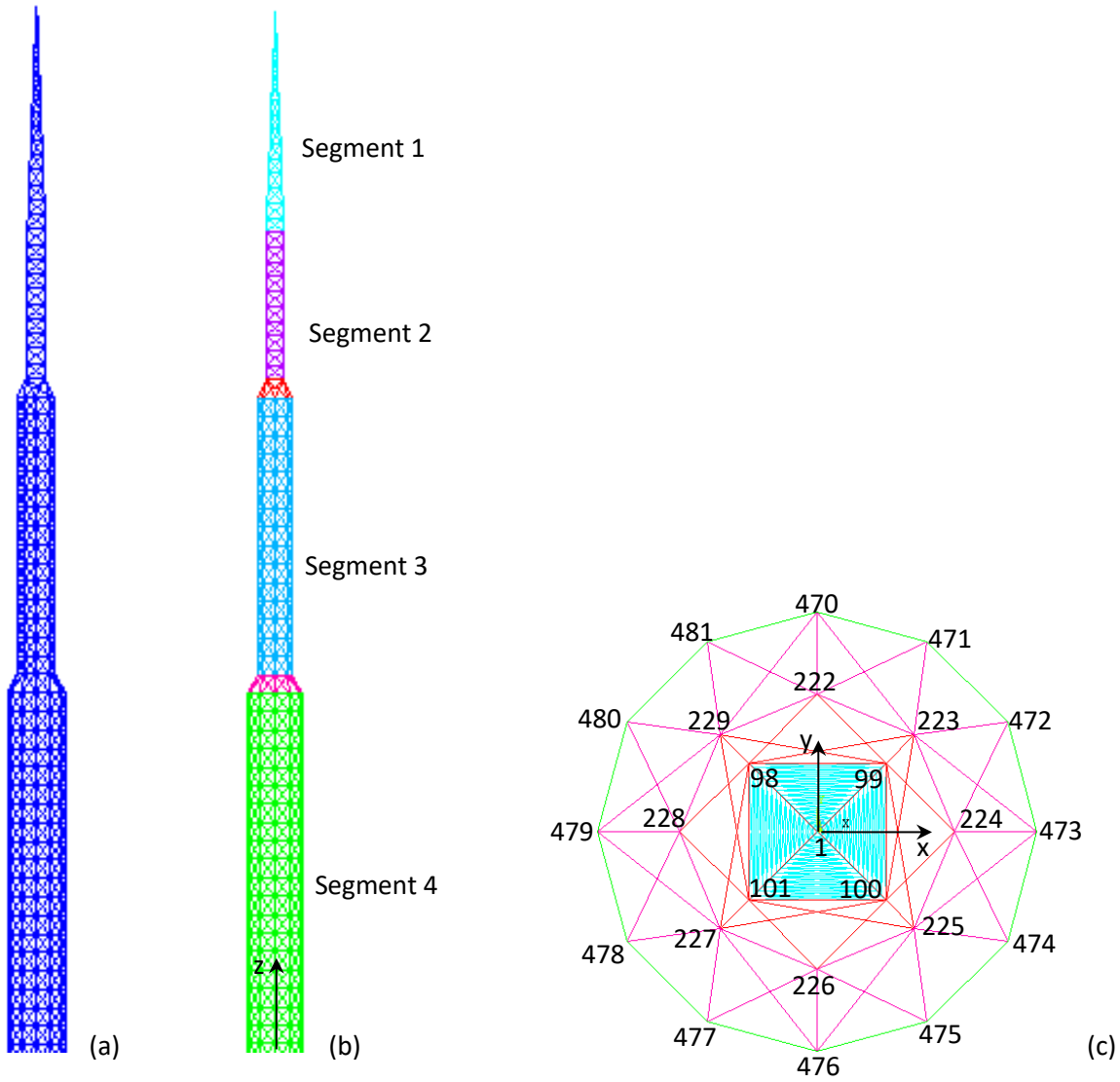

Figure 4.3. Schematic of the spatial 1938-bar tower: (a) Assembly view; (b) Color view of the four segments and two junction modules included in the structure; (c) Layout view indicating key-nodes.

The optimization problem has 20070 non-linear constraints on nodal displacements, member stresses and critical buckling loads. Displacements of free nodes in the X,Y,Z directions must not exceed ±40.64 cm (i.e. ±16 in). The allowable tensile stress is 275.9 MPa (i.e. 40000 psi). The buckling strength of the j^{th} element of the structure is $-100.01\pi EA_j/8l_j^2$ as the tower includes tubular elements with a nominal diameter to thickness ratio of 100. Cross-sectional areas vary between 0.64516 and and 1290.32 cm^2 (i.e. between 0.1 and 200 in^2).

Table 4.3. Loading conditions acting on the 1938-bar tower.

Loading condition 1	
X	None
Y	None
Z	**–13.5 kN** @ nodes 1 through 61 (*the "–" sign indicates that concentrated forces act in the negative Z-direction)*; **–27 kN** @ nodes 62 through 101; **–40.5 kN** @ nodes 102 through 229; **–54 kN** @ nodes 230 through 469.
Loading condition 2	
X	**+6.672 kN** @ nodes 2, 5, 6, 9, 10, 13, 14, 17, 18, 21, 22, 25, 26, 29, 30, 33, 34, 37, 38, 41, 42, 45, 46, 49, 50, 53, 54, 57, 58, 61, 62, 65, 66, 69,70, 81, 82, 85, 86, 89, 90, 93, 94, 97, 98, 101, 108, 116, 124, 132, 140, 148, 156, 164, 172, 180, 188, 196, 204, 220, 228, 239, 251, 263, 275, 287, 299, 311, 323, 335, 347, 359, 371, 383, 395, 407, 419, 431, 443, 455, 467; **–4.448 kN** @ nodes 3, 4, 7, 8, 11, 12, 15, 16, 19, 20, 23, 24, 27, 28, 31, 32, 35, 36, 39, 40, 43, 44, 47, 48, 51, 52, 55, 56, 59,60, 63, 64, 67, 68, 71, 72, 75, 76, 79, 80, 83, 84, 87, 88, 91, 92, 95, 96, 99, 100, 104, 112, 120, 128, 136, 144, 152, 160, 168, 176, 184, 192, 200, 208, 216, 224, 233, 245, 257, 269, 281, 293, 305, 317, 329, 341, 353, 365, 377, 389, 401, 413, 425, 437, 449, 461 (*the "–" sign indicates that concentrated forces act in the negative X-direction)*.
Y	None
Z	None
Loading condition 3	
X	None
Y	**–4.448 kN** @ nodes 2, 3, 6, 7, 10, 11, 14, 15, 18, 19, 22, 23, 26, 27, 30, 31, 34, 35, 38, 39, 42, 43, 46, 47, 50, 51, 54, 55, 58, 59, 62, 63, 66, 67, 70, 71, 74, 75, 78, 79, 82, 83, 86, 87, 90, 91, 94, 95, 98, 99, 102, 110, 118, 126, 134, 142, 150, 158, 166, 174, 182, 190, 198, 206, 214, 222, 230, 242, 266, 278, 290, 302, 314, 326, 338, 350, 362, 374, 386, 398, 410, 422, 434, 446, 458 (*the "–" sign indicates that concentrated forces act in the negative Y-direction)*; **+4.448 kN** @ nodes 4, 5, 8, 9, 12, 13, 16, 17, 20, 21, 24, 25, 28, 29, 32, 33, 36, 37, 40, 41, 44, 45, 48, 49, 52, 53, 56, 57, 60, 61, 64, 65, 68, 69, 72, 73, 76, 77, 80, 81, 84, 85, 88, 89, 92, 93, 96, 97, 100, 101, 106, 114, 122, 130, 138, 146, 154, 162, 170, 178, 186, 194, 202, 210, 218, 226, 236, 248, 260, 272, 284, 296, 308, 320, 332, 344, 356, 368, 380, 392, 404, 416, 428, 440, 452, 464.
Z	None

The optimization results obtained for the 1938-bar tower design example are presented in Table 4.4. LSSO–HHSJA's optimization runs were carried out for N_{POP}=20 and 500 in order to have a fair comparison with the hybrid HS, hybrid BBBC and HFSA algorithms of Ref. [53] and the improved JAYA algorithm of Ref. [54]. No data are listed in the table for

sinDE [124] and SQP-Matlab as those algorithms could not find satisfactory designs. In particular, sinDE's best record after 104,000 structural analyses still weighted about 111.4 ton (i.e. more than 10 ton heavier than the optimized designs achieved by its competitors) and violated displacement constraints by about 1%. SQP-Matlab could not complete a single loop optimization successfully and was hence alternated with SQP-DOT. However, after about 25000 structural analyses, the SQP's current best record was still heavier than the feasible solutions obtained by all metaheuristic algorithms compared in Table 4.4 (i.e. about 101.5 kg vs. at most 101.120 kg) and violated constraints by 0.258%.

Table 4.4. Optimization results obtained for the 1938-bar tower design example.

	Optimized weight (ton)	**Number of structural analyses**	**Constraint violation** (%)
LSSO-HHSJA Present	*Best*: 98.822 *Worst*: 98.860 *Mean/STD:* 98.832±0.01980	*Best*: 7529 *Worst*: 7780 *Mean/STD:* 7680±284	Feasible
Hybrid HS with LS [53]	*Best*: 100.008 *Worst*: 100.240 *Mean/STD:* 100.147±0.467	*Best*: 7931 *Worst*: 8694 *Mean/STD:* 8395±657	Feasible
Hybrid BBBC with LS [53]	*Best*: 99.164 *Worst*: 99.225 *Mean/STD:* 99.179±0.02687	*Best*: 8167 *Worst*: 9655 *Mean/STD:* 8861±526	Feasible
HFSA [53]	*Best*: 99.794 *Worst*: 101.251 *Mean/STD:* 100.523±0.516	13201±598	Feasible
Improved JA [54]	*Best*: 99.255 *Worst*: 99.265 *Mean/STD:* 99.263±0.003536	*Best*: 20051 *Worst*: 21980 *Mean/STD:* 21136±843	Feasible
AHS [38]	*Best*: 100.750 *Worst*: 103.421 *Mean/STD:* 101.919±1.653	*Best*: 19139 *Worst*: 17184 *Mean/STD:* 18394±1134	Feasible
SAHS [39]	*Best*: 100.120 *Worst*: 104.368 *Mean/STD:* 102.623±2.188	*Best*: 15437 *Worst*: 14297 *Mean/STD:* 15201±1383	Feasible
BBBC-UBS [173]	*Best*: 101.120 *Worst*: 102.628 *Mean/STD:* 101.335±0.3033	*Best*: 17461 *Worst*: 19980 *Mean/STD:* 18930±666	Feasible
SLP-DOT	102.789	12310	Feasible

It can be seen from Table 4.4 that LSSO–HHSJA was the best algorithm also in this design example because it converged to the global minimum weight of 98.822 kg within only 7529 structural analyses. All metaheuristic algorithms found a feasible solution rankinginthe following order in terms of optimized weight: LSSO–HHSJA, hybrid BBBC with line search [53], improved JAYA [54], HFSA [53], hybrid HS with line search [53], SAHS [39], AHS

[38] and BBBC-UBS [173]. In particular, structural weight increased by about 2.33% passing from the present algorithm to BBBC-UBS.

The computational cost of optimization process varied much more significanty than structural weight: from the 7529 structural analyses required by the present algorithm to the 20051 analyses required by improved JAYA [54]. In general, the optimizers that implemented more sophisticated line search strategies such as LSSO–HHSJA, hybrid HS and hybrid BBBC [53], HFSA [53] and improved JAYA [54] outperformed their competitors. Interestingly, improved JAYA was about 2.7 times slower than LSSO–HHSJA, while hybrid HS and hybrid BBBC were at most 15% slower than the present algorithm. This is a direct consequence of the fact that improved JAYA actually worked with a simplified line search strategy comparing each trial design $\boldsymbol{X}_k^{new}$ only with its counterpart design $\boldsymbol{X}_k^{pre}$ of the current population. Conversely, hybrid HS and hybrid BBBC always considered a rather large set of descent directions to form a new trial design $\boldsymbol{X}_{\mathrm{TR}}$. Parameter adaptation confirmed itself definitely less effective than line search strategy. In fact, the AHS [38] and SAHS [39] algorithms exhibited the largest average weights and the heaviest worst designs over the independent optimization runs.

LSSO–HHSJA was also the most robust optimizer overall with the lowest standard deviations on optimized weight and required number of structural analyses. As expected, adaptive HS variants [38,39] were characterized by the largest statistical dispersions on structural weight and computational cost. This is fully consistent with the fact that AHS [38] and SAHS [39] are inherently more sensitive than LSSO–HHSJA to the sequence of generated random numbers in the search process. In fact, while SAHS and AHS, respectively, need NDV or (NDV+2) random numbers to form a new trial design in the current iteration, LSSO–HHSJA may need up to (NDV+2)+2×NDV×(N_{POP}−2) random numbers because the new trial design must also be evaluated according to the four cases described in Section 4.2. Hence, LSSO–HHSJA may dispose of a much larger number of optimal combinations of random numbers than SAHS and AHS and this makes the present algorithm less sensitive to independent optimization runs.

Convergence curves of the best optimization runs excuted for the algorithms listed in Table 4.4 are compared in Figure 4.5. Again the plot is limited to the first 13000 structural analyses for the sake of clarity while the 50-650 ton weight range is represented for the Y-axis; the figure includes only the plot relative to the best adaptive HS method [38,39].

Interestingly, LSSO–HHSJA generated better intermediate designs than hybrid HS with line search [53] throughout optimization process. Adaptive HS was initially faster than the present algorithm but such a fast weight reduction resulted in the presence of several steps in the convergence curve with marginal improvements in design. LSSO–HHSJA recovered the weight gap with respect to adaptive HS within only 3000 structural analyses. Hybrid fast SA [53] and improved JAYA [54] were the most competitive algorithms with LSSO–HHSJA but they soon reduced their structural weight reduction rates crossing the LSSO–HHSJA's best run convergence curve after about 4850 and 5150 analyses, respectively. SLP-DOT soon generated infeasible designs and had to penalize weight in order to recover constraint violation.

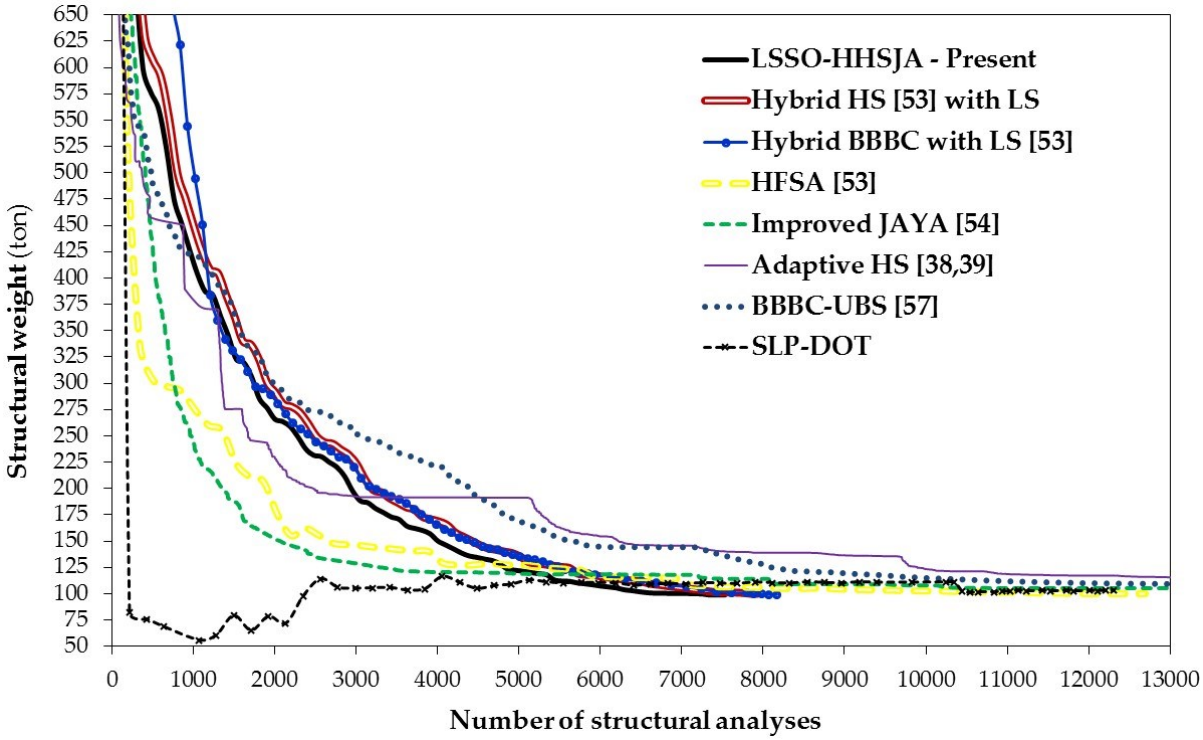

Figure 4.5. Convergence curves obtained for the 1938-bar tower design example.

4.5. Spatial 3586-bar truss tower

The last design example is the weight minimization of the spatial 3586-bar tower with 897 nodes shown in Figure 4.6. This large-scale optimization problem, originally presented in [51], has 280 sizing variables. Material properties are the same as in the 1938-bar tower problem. Structural symmetry allows to categorize bars into 280 groups (see Figure A1 and

Table A1 of the appendix) each of which includes elements with the same cross-sectional area taken as a sizing variable.

The tower is 415 m tall, includes five segments and three junction modules; its layout section is a regular hexadecagon at the ground level and a square at the top segment. In Figure 4.6(b), the segments of the structure are represented in different colors. Figures 4.6(c-f) clarify storey numbering progressing from the top to the bottom of the structure. The layout view of Figure 4.6(g) indicates the nodes limiting the tower cross-sections for the different segments. Node and element group numbering increases from the top to the bottom of the structure.

The square-based pyramid “Segment 1” includes element groups 1 through 57; the square-based prismatic “Segment 2” groups 58 through 97; group 98 connects segments 1 and 2; the octagon-based prismatic “Segment 3” includes groups 99 through 143; group 144 connects segments 3 and 4; the dodecagon-based prismatic “Segment 4” includes groups 145 through 204; group 205 connects segments 4 and 5; the hexadecagon-based prismatic “Segment 5” includes groups 206 through 280.

It can be seen from Figure 4.6(g) that node numbering is the same as for the 1938-bar tower up to node 481. The tower is then completed by 25 sets of 16 nodes each, from 482-483-484-485-486-487-488-489-490-491-492-493-494-495-496-497 to 882-883-884-885-886-887-888-889-890-891-892-893-894-895-896-897, belonging to Segment 5.

The tower must carry the three independent loading conditions listed in Table 4.5. The explanation given in Section 4.4 holds true also for the 3586-bar structure. However, the total load acting on the tower (i.e. about 4862 ton) now is about 15 times larger than the gravity load corresponding to the target structural weight. In the second loading condition, forces acting in the positive X-direction on the left side of the tower are applied, for example, at nodes 98, 101, 228, 479 etc; forces acting in the negative X-direction on the right side of the tower are applied, for example, at nodes 99, 100, 224, 473 etc. In the third loading condition, forces acting in the positive Y-direction on the front side of the tower are applied, for example, at nodes 100, 101, 226, 476 etc; forces acting in the negative Y-direction on the rear side of the tower are applied, for example, at nodes 98, 99, 222, 470 etc.

The optimization problem includes 37374 non-linear constraints on nodal displacements, member stresses and critical buckling loads using the same limits defined for the 1938-bar tower problem. Side constraints on sizing variables also coincide with those of the previous design example.

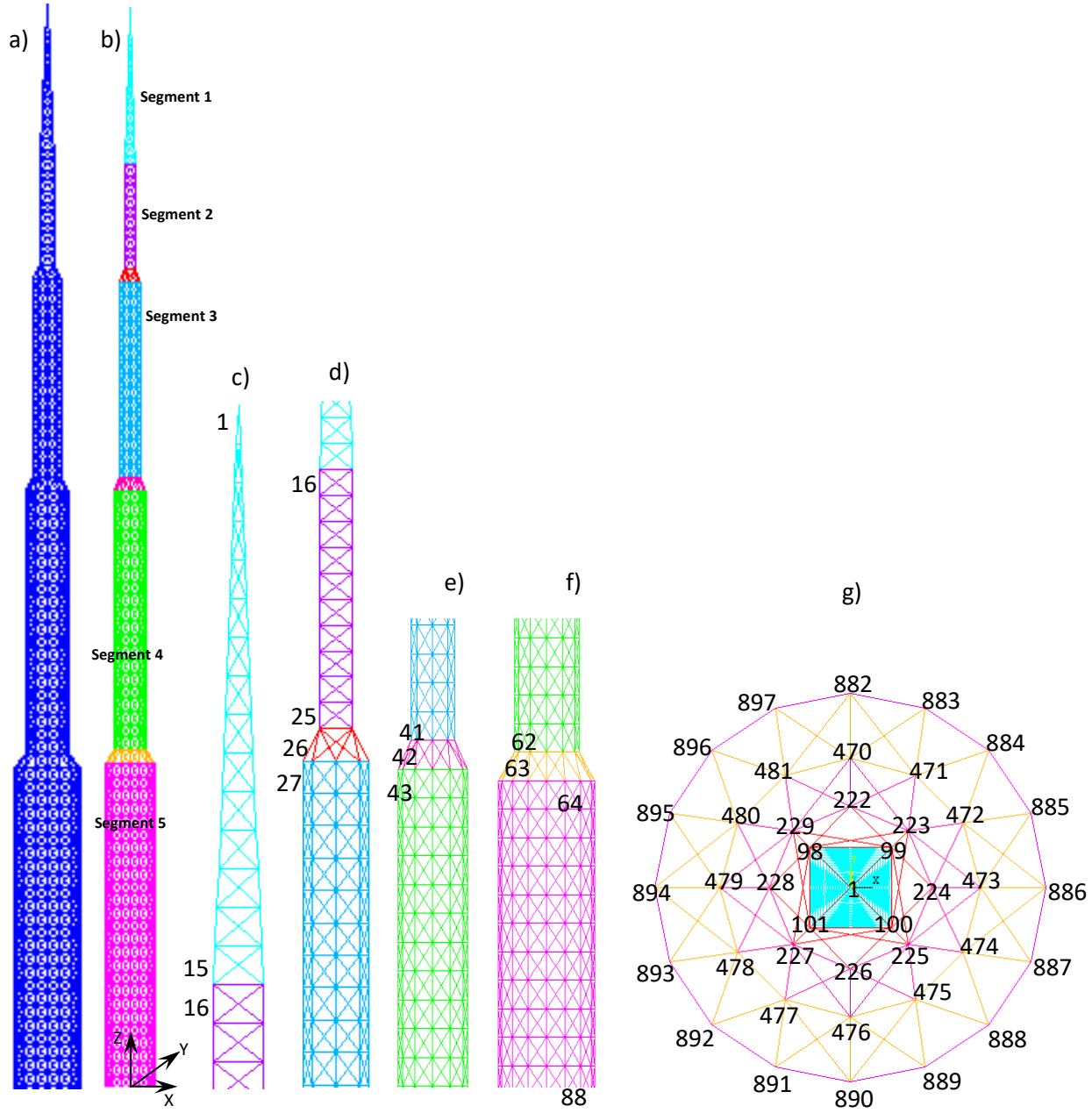

Figure 4.6. Schematic of the spatial 3586-bar tower (representative story and junction numbers also are shown): (a) Assembly view; (b) Color view of the five segments and three junction modules included in the structure; c) Transition from the top of the tower to segment 1 (top of the structure); d-e) Transitions betweensegments 2 and 3, and between segments 3 and 4 (center of the structure); f) Transition between segments4 and 5 (bottom of the structure); g) Layout view indicating key-nodes.

Table 4.5. Loading conditions acting on the 3586-bar tower.

Loading condition 1	
X	None
Y	None
Z	**–13.5 kN** @ nodes 1 through 61 (*the "–" sign indicates that concentrated forces act in the negative Z-direction)*; **–27 kN** @ nodes 62 through 101; **–40.5 kN** @ nodes 102 through 229; **–54 kN** @ nodes 230 through 481; **–67.5 kN** @ nodes 482 through 881.
Loading condition 2	
X	**+6.672 kN** @ nodes 2, 5, 6, 9, 10, 13, 14, 17, 18, 21, 22, 25, 26, 29, 30, 33, 34, 37, 38, 41, 42, 45, 46, 49, 50, 53, 54, 57, 58, 61, 62, 65, 66, 69,70, 81, 82, 85, 86, 89, 90, 93, 94, 97, 98, 101, 108, 116, 124, 132, 140, 148, 156, 164, 172, 180, 188, 196, 204, 220, 228, 239, 251, 263, 275, 287, 299, 311, 323, 335, 347, 359, 371, 383, 395, 407, 419, 431, 443, 455, 467, 479, 494, 510, 526, 542, 558, 574, 590, 606, 622, 638, 654, 670, 686, 702, 718, 734, 750, 766, 782, 798, 814, 830, 846, 862, 878; **–4.448 kN** @ nodes 3, 4, 7, 8, 11, 12, 15, 16, 19, 20, 23, 24, 27, 28, 31, 32, 35, 36, 39, 40, 43, 44, 47, 48, 51, 52, 55, 56, 59,60, 63, 64, 67, 68, 71, 72, 75, 76, 79, 80, 83, 84, 87, 88, 91, 92, 95, 96, 99, 100, 104, 112, 120, 128, 136, 144, 152, 160, 168, 176, 184, 192, 200, 208, 216, 224, 233, 245, 257, 269, 281, 293, 305, 317, 329, 341, 353, 365, 377, 389, 401, 413, 425, 437, 449, 461, 473, 486, 502, 518, 534, 550, 566, 582, 598, 606, 622, 638, 654, 670, 686, 702, 718, 734, 750, 766, 782, 798, 814, 830, 846, 862, 878 (*the "–" sign indicates that concentrated forces act in the negative X-direction)*.
Y	None
Z	None
Loading condition 3	
X	None
Y	**–4.448 kN** @ nodes 2, 3, 6, 7, 10, 11, 14, 15, 18, 19, 22, 23, 26, 27, 30, 31, 34, 35, 38, 39, 42, 43, 46, 47, 50, 51, 54, 55, 58, 59, 62, 63, 66, 67, 70, 71, 74, 75, 78, 79, 82, 83, 86, 87, 90, 91, 94, 95, 98, 99, 102, 110, 118, 126, 134, 142, 150, 158, 166, 174, 182, 190, 198, 206, 214, 222, 230, 242, 266, 278, 290, 302, 314, 326, 338, 350, 362, 374, 386, 398, 410, 422, 434, 446, 458, 470, 482, 498, 514, 530, 546, 562, 578, 594, 610, 626, 642, 658, 674, 690, 706, 722, 738, 754, 770, 786, 802, 818, 834, 850, 866 (*the "–" sign indicates that concentrated forces act in the negative Y-direction)*; **+4.448 kN** @ nodes 4, 5, 8, 9, 12, 13, 16, 17, 20, 21, 24, 25, 28, 29, 32, 33, 36, 37, 40, 41, 44, 45, 48, 49, 52, 53, 56, 57, 60, 61, 64, 65, 68, 69, 72, 73, 76, 77, 80, 81, 84, 85, 88, 89, 92, 93, 96, 97, 100, 101, 106, 114, 122, 130, 138, 146, 154, 162, 170, 178, 186, 194, 202, 210, 218, 226, 236, 248, 260, 272, 284, 296, 308, 320, 332, 344, 356, 368, 380, 392, 404, 416, 428, 440, 452, 464, 476, 490, 506, 522, 538, 554, 570, 586, 602, 618, 634, 650, 666, 682, 698, 714, 730, 746, 762, 778, 794, 810, 826, 842, 858, 874.
Z	None

Table 4.6 presents the results obtained for the 3586-bar tower design example. LSSO–HHSJA's optimization runs were carried out for N_{POP}=20 and N_{POP}=500 for two reasons: (i) to have statistically more significant results than Ref. [51], which illustrated only the case N_{POP}=500; (ii) to make optimization runs of adaptive HS variants [38,39], improved/parameterless JAYA [54-56], BBBC-UBS [173] and sinDE [124] computationally affordable as the computational cost of those algorithms is proportional to the population size N_{POP}. In regard to issue (i), the hyrid HS, hybrid BBBC and HFSA algorithms of [53] were

used to solve this test problem as none of the metaheuristic algorithms described in Ref. [51] could find a feasible solution. In regard to issue (ii), by setting $N_{POP}=20$ and a computational budget of 20000 structural analyses, it was possible to complete up to 1000 optimization iterations for adaptive HS and sinDE, and about 2000 iterations for the improved/parameterless JAYA and BBBC-UBS algorithms. The selected 20000 analyses computational budget is consistent with the data listed in Table 4.4 for the 1938-bar tower, a similar structure yet with about one half nodes/elements than the 3586-bar tower. The largest fraction of computation time entailed by a single structural analysis of a 3D truss structure is associated to the inversion of the global stiffness matrix [K] to solve the linear system {F}=[K]{u}: this operation was found to be four times more expensive for the 3586-bar tower.

No data are listed in Table 4.6 for adaptive HS variants [38,39] and sinDE [58] because the best candidate designs included in the population after 20000 structural analyses still were between 10 and 15% heavier than the optimum design found by LSSO–HHSJA and violated displacement constraints by an amount between 3% and 5%. Once again, using parameter adaptation without any line search strategy does not allow to efficiently solve large structural optimization problems.

Table 4.6. Optimization results obtained for the 3586-bar tower design example.

	Optimized weight (ton)	**Number of structural analyses**	**Constraint violation** (%)
LSSO-HHSJA Present	*Best*: 323.175 *Worst*: 323.722 *Mean/STD:* 323.287±0.158	*Best*: 10997 *Worst*: 11753 *Mean/STD:* 11262±534	Feasible
Hybrid HS with LS[51]	325.381	11312	0.06110
Hybrid HS with LS [53]	*Best*: 323.611 *Worst*: 324.794 *Mean/STD:* 324.202±0.4358	*Best*: 11504 *Worst*: 12046 *Mean/STD:* 11904±628	*Best*: 0.03679 *Worst*: 0.02170 *Mean/STD:* 0.0317±0.00986
Hybrid BBBC with LS [51]	325.980	14616	0.06180
Hybrid BBBC with LS [53]	*Best*: 324.246 *Worst*: 325.752 *Mean/STD:* 325.299±0.5607	*Best*: 12356 *Worst*: 13403 *Mean/STD:* 13295±789	*Best*: 0.02376 *Worst*: 0.03750 *Mean/STD:* 0.0269±0.0104
CMLPSA [51]	326.185	16240	0.105
HFSA [53]	*Best*: 323.567 *Worst*: 325.329 *Mean/STD:* 324.385±0.4283	14466±565	Feasible
Improved/parameterless JA [54-56]	*Best*: 323.977 *Worst*: 324.431 *Mean/STD:* 324.130±0.287	Computational budget: 20000 structural analyses	Feasible
BBBC-UBS [173]	*Best*: 325.097 *Worst*: 327.681 *Mean/STD:* 325.541±1.083	Computational budget: 20000 structural analyses	Feasible
SLP-MATLAB & DOT	326.278	16480	Feasible

LSSO–HHSJA was once again the most efficient optimizer overall converging to the lowest structural weight of 323.175 ton, within the smallest number of structural analyses, 10997. The hybrid HS and hybrid BBBC variants including line search strategies and 1-D probabilistic search developed in [53] found lighter designs than those reported in [51] also reducing constraint violations. HFSA [53] totally outperformed CMLPSA [51] converging to a lighter feasible design, very close to the global optimum found by the present algorithm (i.e. 323.567 vs 323.175 ton). Hence, HFSA should be considered the 2nd best optimizer overall after LSSO–HHSJA.

Improved/parameterless JAYA [54–56] and BBBC-UBS [173] also found feasible solutions, lighter than those quoted in [51] for hybrid HS, hybrid BBBC and CMPLSA, but their optimization runs were always stopped before reaching a fully convergenced solution in the sense of Eq. (20). However, improved/parameterless JAYA's design was only 0.248% heavier than the global optimum found by LSSO–HHSJA. This confirms the utility of integrating JAYA-based operators in a harmony search architecture. BBBC-UBS [173] also reached a rather satisfactory performance (i.e. feasible solution with only 0.595% weight penalty with respect to LSSO–HHSJA) but was outperformed by improved/parameterless JAYA [54–56], which adopts a very similar elitist strategy as BBBC-UBS to accept/reject new trial designs but has the inherent capability to move towards the best design of the population and move away from the worst design.

SQP also converged to a feasible solution which is about 1% heavier than the best design found by LSSO–HHSJA. However, the rate of success of this gradient based algorithm was rather low and less than 10% of the different optimization runs started from each of the 500 designs included in the LSSO–HHSJA's initial population converged to feasible designs ligher than 330 ton. The average constraint violation was of the order of 0.15%, similar to that reported in [51] for SQP.

Similar to that seen for the 200-bar truss and 1938-bar tower problems, the number of structural analyses varied more significantly than the optimized weight passing from one algorithm to another. HFSA [53] was on average about 30% slower than LSSO–HHSJA while improved/parameterless JAYA [54–56] and BBBC-UBS [173] were about two times slower than the present algorithm. Multiple line search put in the context of a population-based search appears to be the most efficient approach to handle the complex design spaces

with many design variables and nonlinear constraints that characterize large scale structural optimization problems.

LSSO–HHSJA presented the lowest standard deviations on optimized weight and required structural analyses. All of the algorithms compared in Table 4.6 were actually robust and the ratio between standard deviation on optimized weight and average optimized weight never exceeded 0.34%. This confirms the validity of the selected competitors of LSSO–HHSJA that provide effective information on the real performance of the proposed algorithm.

Optimal values of sizing variables determined for the three design examples were not listed in the results tables for the sake of brevity. In the spatial towers examples, cross-sectional areas of bars increased from top to bottom of each segment thus realizing uniform stiffness designs. Each segment contributed to the total structural weight by the same extent regardless of having 3586 or 1938 elements in the tower. The bottom segment (i.e. respectively, “5” and “4” for the 3586-bar and 1938-bar towers) counted by almost 50% of the total weight. For example, Figure 4.7 compares the optimized cross-sections by LSSO–HHSJA and its competitors for the 3586-bar tower problem. Distributions relative to hybrid HS/BBBC variants with line search of [51,53] are averaged for the sake of clarity.

It can be seen from Figure 4.7 that cross-sectional areas of each segment are distributed in fashion of “oscillating waves” between very small and very large areas.Values of large areas tend to increase from the top to the bottom of the segment while values of small areas tend to be constant. In particular, the SQP’s design is characterized by linearly increasingelement areas towards the bottom of each segment while the other members are sized at their minimum gage. Interestingly, optimal cross-sectional areas of LSSO–HHSJA and HFSA [53] are much closer to the SQP’s area distribution than those of the hybrid HS/BBBC variants [51,53]. The linearly increasing areas realize the uniform stiffness profile required by the second loading condition acting on the tower.

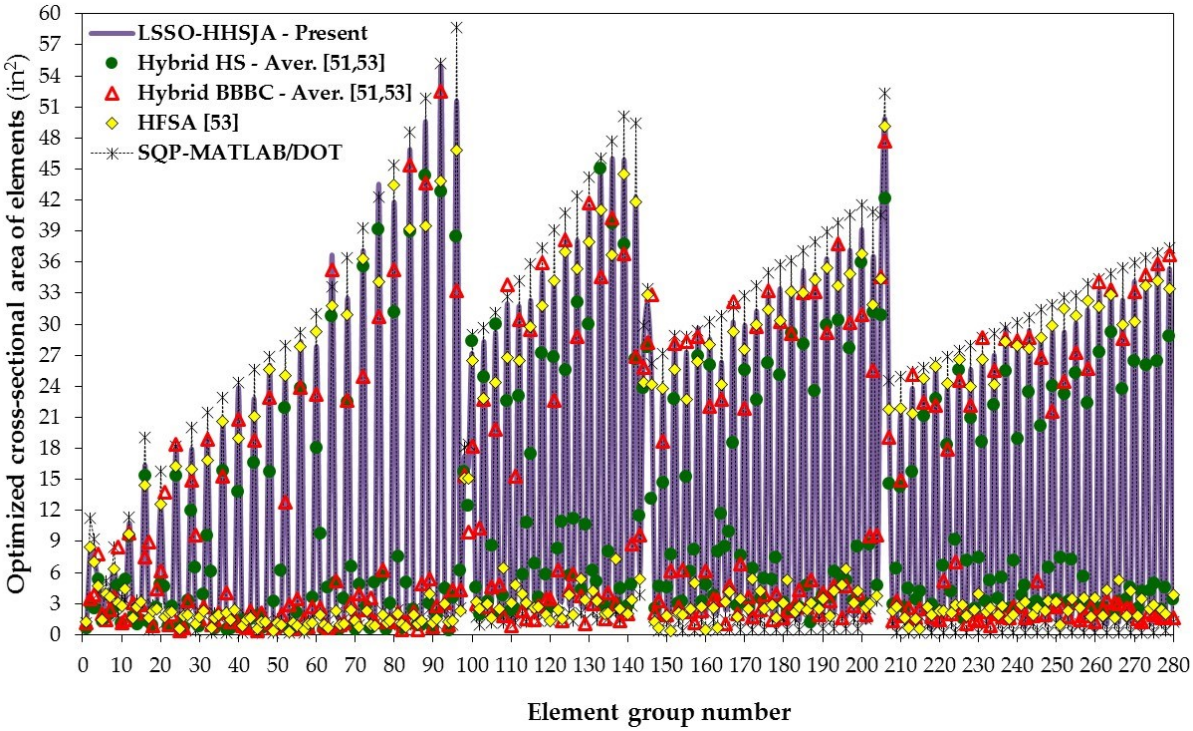

Figure 4.7. Optimized cross-sectional areas of the 3586-bar tower.

Convergence curves for this optimization problem are shown in Figure 4.8, where the plot is limited to the first 16000 structural analyses and the 200-3000 ton weight range is represented for the Y-axis. The figure does not include the optimization histories of improved/parameterless JAYA [54–56] and BBBC-UBS [173] algorithms as these curves lie well above the one relative to hybrid BBBC with line search of Ref. [51]. This is a direct consequence of the low ratio (i.e. less than 0.1) between population size and number of optimization variables determined for N_{POP}=20. Although convergence behavior of improved/parameterless JAYA was proven to be insensitive to population size in many studies including the results presented here, it is a matter of fact that this algorithm (as well as BBBC-UBS) attempts to replace candidate designs one by one without discharging $\boldsymbol{X}_{\mathbf{worst}}$ before all of the N_{POP}=20 designs were perturbed in the current iteration. Such a strategy may lead to perform too many analyses especially when there is a large set of optimization variables that drive the search process. For example, the bottom segments"4" or "5"include between 27% and 30% of the total number of design variables and count by 50% of the structural weights of the two towers.

It can be seen that LSSO–HHSJA again generated better intermediate designs than hybrid HS with line search [51,53] throughout optimization process. The significant improvement in convergence behavior seen for the hybrid HS and hybrid BBBC variants of

Ref. [53] over the formulations of [51] was due both to having enhanced the line search strategy and carried outindepent optimization runs to have statistically significant results.

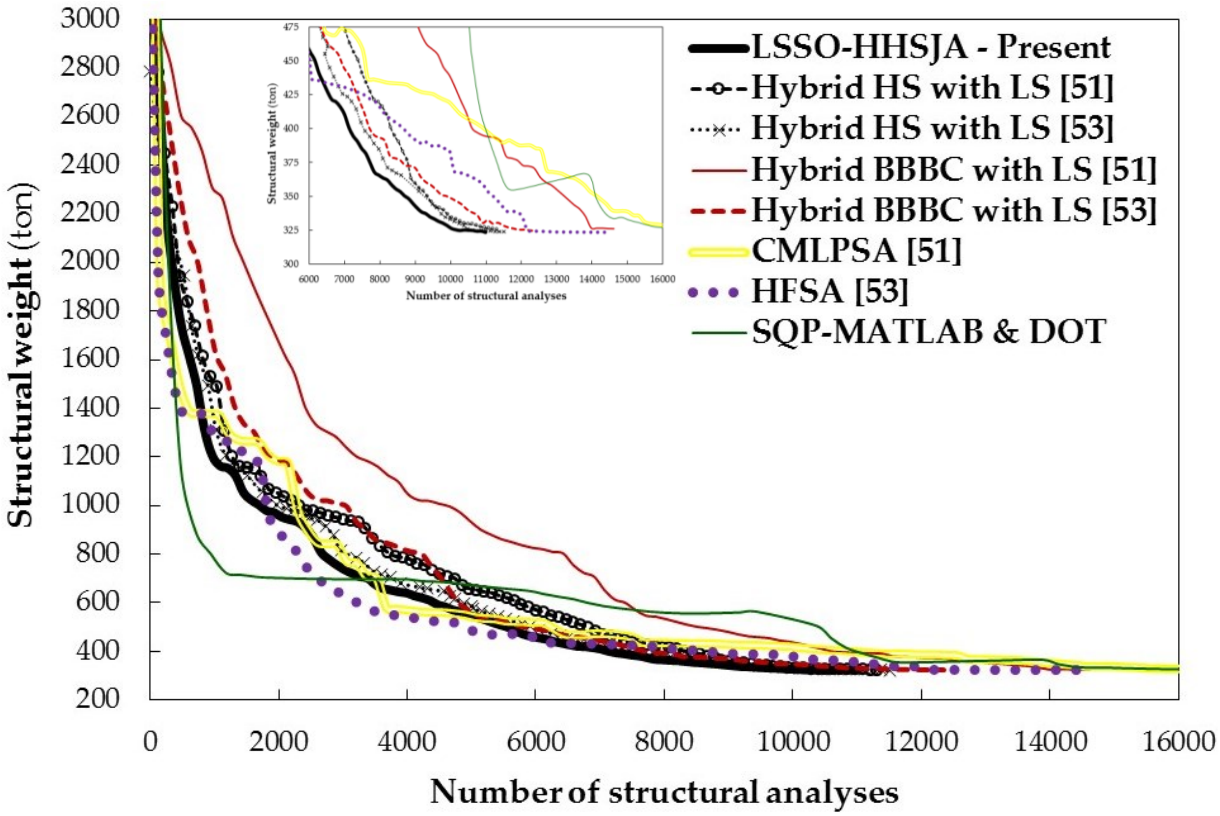

Figure 4.8. Convergence curves obtained for the 3586-bar tower design example.

HFSA [53] was competitive with LSSO–HHSJA over the first 6400 structural analyses (see the inset of Figure 4.8 showing the detail of convergence curves in the weight range from 300 to 475 ton) and the convergence curves of the two algorithms repeatedly crossed each other before this turning point. Interestingly, the convergence trends of SA-based variants (i.e. CMLPSA and HFSA) almost overlapped with the SQP's trend in the early optimization iterations. This can be explained with the informal argument that simulated annealing develops one design at a time like SLP/SQP and CMLPSA/HFSA form their trial designs by perturbing the current best record in the fashion $\boldsymbol{X}_{\mathrm{TR}}=\boldsymbol{X}_{\mathrm{OPT}}+\bar{\nabla}W^{\mathrm{T}}(\boldsymbol{X}_{\mathrm{OPT}})(\boldsymbol{\rho}*\boldsymbol{\delta X})$, which practically corresponds to a linearization with random coefficients; the "$*$" symbol denotes a new vector is defined by multiplying term by term one vector of random numbers and one perturbation vector with respect to $\boldsymbol{X}_{\mathrm{OPT}}$. This similarity becomes more evident as the initial design of the gradient-based optimizer is close to the starting point or the best design included in the initial population of the metaheuristic algorithm.

The SQP's convergence curve shown in Figure 4.8 presents the typical steps corresponding to the transition from MATLAB to DOT optimization routines, which made it possible to reach a monotonic convergence behavior. As usual, the gradient-based optimizer reduced the structural weight by a great extent in the early optimization cycles and then stepped in order to recover the constraint violation. Conversely, LSSO–HHSJA operates on a set of descent directions and can select the best path to remain always in the feasible search space.

CHAPTER 5

CONCLUSIONS

The dissertation presented a new hybrid metaheuristic optimization algorithm, LSSO–HHSJA, combining the harmony search optimization (HS) and JAYA methods. LSSO–HHSJA forms trial designs enhancing the HS architecture with multiple line searches based on explicitly available gradient information. These line searches are then augmented by JAYA's based operators, whichfinally allow to minimize the number of structural analyses required in the optimization process. All stages of LSSO–HHSJA attempt to generate trial designs lying on descent directions with respect to the best individual(s) stored in the population of the current iteration. The new algorithm developed here was successfully tested in three large scale sizing optimization problems of truss structures (i.e. planar 200-bar truss, spatial 1938-bar tower, spatial 3586-bar tower) including up to 280 sizing variables and 37374 nonlinear constraints. LSSO–HHSJA was very competitive with other HS and JAYA variants, other state-of-the art metaheuristic methods (i.e. simulated annealing, big bang-big crunch and sinusoidal differential evolution), and commercially available gradient-based optimizers (i.e. sequential quadratic programmingand sequential linear programming). Remarkably, the proposed algorithm always converged to the lowest structural weight, obtained feasible designs, and required less structural analyses than other HS variants that implemented line search and/or parameter adaptation strategies.

An interesting question that may arise looking at the formulation of LSSO–HHSJA is the following. Would it be possible to "capture" the effect of each strategy implemented in LSSO–HHSJA (i.e. the role played by each "decision" made by LSSO–HHSJA to activate one or another available option in its formulation) on final results? A careful analysis of the LSSO–HHSJA algorithm reveals that all "decisions" actually are of two types: (i) to use gradient information, mirroring strategies and line searches to generate new trial designs on descent directions with respect to the current best record or currently selected individuals of population; (ii) to push the search towards the best designs currently stored in the population and at the same time escape from the worst designs. LSSO–HHSJA has a highly dynamic character that efficiently integrates types (i) and (ii) throughout optimization process. It can be concluded that separating effects of each search strategy implemented in LSSO–HHSJA on final results is definitely less important than how these strategies may concur to determine the optimal solution.

The results presented in the dissertation confirmed the utility of using trial descent directions to form new candidate solutions. This approach is more effective than simply updating the internal parameters of HS with more or less sophisticate strategies. The JAYA's

rationale appears very suited for the purpose as it enhances the search of descent directions by avoiding the worst regions of design space. The generation of new trial designs hence relies only on the best quality directions without performing unnecessary analyses.

The possibility of using the same hybridization strategy for another metaheuristic algorithm such as Big Bang–Big Crunch was also investigated by solving two highly nonlinear design problems including up to 84 variables, (i) shape optimization of a concrete dam and (ii) discrete layout optimization of a planar steel frame, to prove the feasibility of the proposed approach. For that purpose, the LSSO–HBBBCJA algorithm was developed. Remarkably, the JAYA-based operators boosted the search ability of BBBC and allowed the new algorithm to obtain better solutions than those reported in the literature even operating on a smaller computational budget.

REFERENCES

1. Goldberg, D.E. *Genetic Algorithms in Search, Operation and Machine Learning*. Addison-Wesley, Reading, MA, USA, 1989.

2. Storn, R. Price, K. *Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces*. Technical Report No. TR-95-012, International Computer Science Institute, Berkley, CA, USA, 1995.

3. Van Laarhoven, P.J.M.; Aarts, E.H.L. *Simulated Annealing: Theory and Applications*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1987.

4. Clerc, M. *Particle Swarm Optimization*. ISTE Publishing Company, London, UK, 2006.

5. Dorigo, M.; Stutzle, T. *Ant Colony Optimization*. MIT Press, Cambridge, MA, USA, 2004.

6. Gandomi AH, Yang XS, Alavi AH. Mixed variable structural optimization using firefly algorithm. *Computers and Structures* **2011**; 89(23–24):2325–2336.

7. Gandomi AH, Yang XS, Alavi AH. Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. *Engineering with Computers* **2013**; 29(1):17–35.

8. Mirjalili S. The ant lion optimizer. *Advances in Engineering Software* **2015**; 83:80–98.

9. Glover F, Laguna M. *Tabu Search*. Kluwer Academic Publishers, Boston, MA, USA, 1997.

10. Geem ZW, Kim JH, Loganathan G. A new heuristic optimization algorithm: harmony search. *Simulation* **2001**; 76(2):60–68.

11. Rao RV, Savsani VJ, Vakharia DP. Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems. *Computer Aided Design* **2011**; 43(3):303–315.

12. Rao RV. Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. *International Journal of Industrial Engineering Computations* **2016**; 7:19–34.

13. Erol OK, Eksin I. A new optimization method: big bang-big crunch. *Advances in Engineering Software* **2006**; 37(2):106–111.

14. Kaveh A, Talatahari S. A novel heuristic optimization method: charged system search. *Acta Mechanica* **2010**; 213(3–4):267–289.

15. Kaveh A, Khayat Azad M. A new meta-heuristic method: ray optimization. *Computers and Structures* **2012**; 112–113:283–294.

16. Kaveh A, Mahdavi VR. Colliding bodies optimization: A novel meta-heuristic method. *Computers and Structures* **2014**; 139:18–27.

17. Kaveh A, Bakhshpoori T. A new metaheuristic for continuous structural optimization: water evaporation optimization. *Structural and Multidisciplinary Optimization* **2016**; 54:23–43.

18. Kaveh A, Zolghadr A. Cyclical parthenogenesis algorithm for guided modal strain energy based structural damage detection. *Applied Soft Computing* **2017**; 57:250–264.

19. Pierezan J, Coelho LS. Coyote optimization algorithm: a new metaheuristic for global optimization problems. In: *Proceedings of the IEEE World Conference on Computational Intelligence, Congress on Evolutionary Computation*; Rio de Janeiro, Brazil, 2018; pp. 2633–2640.

20. Lamberti L, Pappalettere C. Metaheuristic design optimization of skeletal structures: a review. *Computational Technology Reviews* **2011**; 4:1–32.

21. Saka MP, Dogan E. Recent developments in metaheuristic algorithms: a review. *Computational Technology Reviews* **2012**; 5:31–78.

22. Kaveh A. *Advances in Metaheuristic Algorithms for Optimal Design of Structures*. Springer International Publishing, Switzerland, 2014.

23. Kaveh A. *Applications of Metaheuristic Optimization Algorithms in Civil Engineering*. Springer International Publishing, Switzerland, 2017.

24. Kaveh A, Ilchi Ghazaan M. *Meta-heuristic Algorithms for Optimal Design of Real-Size Structures*. Springer International Publishing, Switzerland, 2018.

25. Wolpert, D.H.; Macready, W.G. No Free Lunch Theorems for optimization. *IEEE Transactions on Evolutionary Computation* **1997**; 1(1):67–82.

26. Ho YC, Pepyne DL. Simple explanation of the No-Free-Lunch Theorem and its implications. *Journal of Optimization Theory and Applications* **2002**; 115:549–570.

27. Yang XS. Harmony Search as a metaheuristic algorithm. In: *Music-Inspired Harmony Search Algorithm: Theory and Applications*; Geem ZW, Ed.; Springer: Berlin, Germany, 2009; Chapter 1, pp. 1-14.

28. Haftka RT, Gurdal Z. *Elements of Structural Optimization*, 3rd ed. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1992.

29. Vanderplaats GN. *Numerical Optimization Techniques for Engineering Design*. VR&D Inc., Colorado Springs, CO, USA, 1998.

30. Arora JS. *Introduction to Optimum Design*. McGraw-Hill Book Company, New York, USA, 1989.

31. Lee KS, Geem ZW. A new structural optimization method based on the harmony search algorithm. *Computers and Structures* **2004**; 82(9–10):781–798.

32. Lee KS, Geem ZW. A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. *Computer Methods in Applied Mechanics and Engineering* **2005**; 194(36–38); 3902–3933.

33. Saka MP. Optimum design of steel sway frames to BS5950 using harmony search algorithm. *Journal of Constructional Steel Research* **2009**; 65:36–43.

34. Maheri MR, Narimani MN. An enhanced harmony search algorithm for optimum design of side sway steel frames. *Computers and Structures* **2014**; 136:78–89.

35. Murren P, Khandelwal K. Design-driven harmony search (DDHS) in steel frame optimization. *Engineering Structures* **2014**; 59:798–808.

36. Mahdavi M, Fesanghary M, Damangir E. An improved harmony search algorithm for solving optimization problems. *Applied Mathematics and Computation* **2007**; 188(2):1567–1579.

37. Carbas S, Saka MP. Optimum topology design of various geometrically nonlinear latticed domes using improved harmony search method. *Structural and Multidisciplinary Optimization* **2012**; 45(3):377–399.

38. Hasancebi O, Erdal F, Saka MP. Adaptive harmony search method for structural optimization. *ASCE Journal of Structural Engineering* **2010**; 136(4):419–431.

39. Degertekin SO. Improved harmony search algorithms for sizing optimization of truss structures. *Computers and Structures* **2012**; 92-93:229–241.

40. Kaveh A, Naiemi M. Sizing optimization of skeletal structures with a multi-adaptive Harmony Search algorithm. *Scientia Iranica Transactions on Civil Engineering* **2015**; 22(2):345–366.

41. Geem ZW, Sim KB. Parameter-setting-free harmony search algorithm. *Applied Mathematics and Computation* **2010**; 217:3881–3889.

42. Turky AM, Abdullah S. A multi-population harmony search algorithm with external archive for dynamic optimization problems. *Information Sciences* **2014**; 272:84–95.

43. Kaveh A, Ahangaran M. Discrete cost optimization of composite floor system using social harmony search model. *Applied Soft Computing* **2012**; 12(1):372–381.

44. Cheng MY, Prayogo D, Wu YW, Lukito MM. A Hybrid Harmony Search algorithm for discrete sizing optimization of truss structure. *Automation in Construction* **2016**; 69:21–33.

45. Kaveh A, Talatahari S. A particle swarm ant colony optimization for truss structures with discrete variables. *Journal of Constructional Steel Research* **2009**; 65(8–9):1558–1568.

46. Kaveh A, Talatahari S. Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures. *Computers and Structures* **2009**; 87(5–6):267–283.

47. Omran MGH, Mahdavi M. Global best harmony search. *Applied Mathematics and Computation* **2008**; 198(2):643–656.

48. Al-Betar MA, Abu Doush I, Khader AT, Awadallah MA. Novel selection schemes for harmony search. *Applied Mathematics and Computation* **2012**; 218(10):6095–6117.

49. Fesanghary M, Mahdavi M, Minary-Jolandan M, Alizadeh Y. Hybridizing harmony search algorithm with sequential quadratic programming for engineering optimization problems. *Computer Methods in Applied Mechanics and Engineering* **2008**; 197(33–40):3080–3091.

50. Lamberti, L.; Pappalettere, C. An improved harmony-search algorithm for truss structure optimization. In *Proceedings of The Twelfth International Conference on Civil, Structural and Environmental Engineering Computing*; Topping, B.H.V., Costa Neves, L.F., Barros, R.C., Eds.; Funchal, Portugal, September 2009.

51. Lamberti L, Pappalettere C. Truss weight minimization using hybrid Harmony Search and Big Bang-Big Crunch algorithms. In *Metaheuristic Applications in Structures and Infrastructures*; Gandomi, A.H., Yang, X.S., Talatahari, S., Alavi, A.H., Eds.; Elsevier: Waltham, MA, USA, 2013; Chapter 9, pp. 207–240.

52. Degertekin SO, Lamberti L. Comparison of hybrid metaheuristic algorithms for truss weight optimization. In *Proceedings of the Third International Conference on Soft Computing Technology in Civil, Structural and Environmental Engineering*; Tsompanakis, Y., Ed.; Cagliari, Italy, September 2013.

53. Ficarella E, Lamberti L, Degertekin SO. Comparison of three novel hybrid metaheuristic algorithms for structural optimization problems. *Computers and Structures* **2021**; 244:106395.

54. Degertekin SO, Lamberti L, Ugur IB. Sizing, layout and topology design optimization of truss structures using the Jaya algorithm. *Appl. Soft Computing* **2018**; 70:903–928.

55. Degertekin SO, Lamberti L, Ugur IB. Discrete sizing/layout/topology optimization of truss structures with an advanced Jaya algorithm. *Applied Soft Computing* **2019**; 79:363–390.

56. Degertekin SO, Yalcin Bayar G, Lamberti L. Parameter free Jaya algorithm for truss sizing-layout optimization under natural frequency constraints. *Computers and Struct*ures **2021**; 245:106461.

57. Holland JH. Adaptation *in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor (MI), USA, 1975.

58. Rechenberg I. *Evolutions strategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog, Stuttgart, Germany, 1973. (In German)

59. Beyer HG, Schwefel HP. Evolution strategies – a comprehensive introduction. *Natural Computing* **2002**; 1(1):3–52.

60. Kirkpatrick S, Gelatt CD, Vecchi MP. Optimization by simulated annealing. *Science* **1983**; 220(4598):671–680.

61. Eberhart RC, Kennedy J. A new optimizer using particle swarm theory. In: *Proceedings of The Sixth International Symposium on Micro Machine and Human Science*, Nagoya, Japan, 1995.

62. Dorigo M, Maniezzo V, Colorni A. The ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems Man and Cybernetics B* **1996**; 26(1):29–41.

63. Bonabeau E, Dorigo M, Theraulaz G. Inspiration for optimization from social insect behavior. *Nature* **2000**; 406(6791):39–42.

64. Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of Global Optimization* **2007**; 39(3):459–471.

65. Yang XS. Engineering Optimization: An Introduction with Metaheuristic Applications, John Wiley & Sons, 2010.

66. Oftadeh R, Mahjoob MJ, Shariatpanahi M. A novel meta-heuristic optimization algorithm inspired by group hunting of animals: Hunting search. *Computers and Mathematics with Applications* **2010**; 60(7):2087–2098.

67. Yang XS, Deb S. Eagle strategy using Levy walk and firefly algorithms for stochastic optimization, In: C. Cruz, J.R. González et al. (Eds.), *Nature Inspired Cooperative Strategies for*

Optimization (NICSO2010), Studies in Computational Intelligence, Vol. 284, pp. 101–111, Springer, 2010.

68. Yang XS, Gandomi AH. Bat algorithm: a novel approach for global engineering optimization. *Engineering with Computers* **2012**; 29(5):464–483.

69. Gandomi AH, Yang XS, Alavi AH, Talatahari S. Bat algorithm for constrained optimization tasks. *Neural Computing and Applications* **2012**; 22(6):1239–1255.

70. Kaveh A, Farhoudi N. A new optimization method: dolphin echolocation. *Advances in Engineering Software* **2013**; 59:53–70.

71. Passino KM. *Biomimicry of Bacterial Foraging for Distributed Optimization*, University Press, Princeton (NJ), USA, 2001.

72. Gazi V, Passino KM. Stability analysis of social foraging swarms. *IEEE Transactions on Systems Man and Cybernetics B* **2004**; 34(1):539–557.

73. Askarzadeh A. A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm. *Computers and Structures* **2016**;169:1–12.

74. Kanarachos S, Griffin J, Fitzpatrick ME. Efficient truss optimization using the contrast-based fruit fly optimization algorithm. *Computers and Structures* **2017**; 182:137–148.

75. Tschida CE, Silverberg LM. Cellular growth algorithms for shape design of truss structures. *Computers and Structures* **2013**; 116:1–6.

76. Luh GC, Chueh CH. Multi-objective optimal design of truss structure with immune algorithm. *Computers and Structures* **2004**; 82(11−12):829–844.

77. Kaveh A, Talatahari S. Optimum design of skeletal structures using imperialist competitive algorithm. *Computers and Structures* **2010**; 88(21−22):1220–1229.

78. Degertekin SO, Hayalioglu MS. Sizing truss structures using teaching-learning-based optimization. *Computers and Structures* **2013**; 119:177−188.

79. Camp CV, Farshchin M. Design of space trusses using modified teaching-learning based optimization. *Engineering Structures* **2014**; 62:87−97.

80. Sadollah A, Bahreininejad A, Eskandar H, Hamdi M. Mine blast algorithm for optimization of truss structures with discrete variables. *Computers and Structures* **2012**; 102−103:49−63.

81. Gonçalves MS, Lopez RH, Miguel LFF. Search group algorithm: A new metaheuristic method for the optimization of truss structures. *Computers and Structures* **2015**; 153:165−184.

82. Rashedi E, Nezamabadipour H, Saryazdi S. GSA: a gravitational search algorithm. *Information Sciences* **2009**; 179(13):2232−2248.

83. Kaveh A, Motie Share MA, Moslehi M. A new meta-heuristic algorithm for optimization: magnetic charged system search. *Acta Mechanica* **2013**; 224(1):85−107.

84. Alatas B, Akin E. Chaotically encoded particle swarm optimization algorithm and its applications. *Chaos Solitons and Fractals* **2009**; 41(2):939–950.

85. Alatas B. Chaotic harmony search algorithms. *Applied Mathematics and Computation* **2010**; 216(9):2687–2699.

86. Alatas B. Uniform Big Bang–chaotic Big Crunch optimization. *Communications in Nonlinear Science and Numerical Simulation* **2011**; 16(9):3696–3703.

87. Gandomi AH, Yang XS, Talatahari S, Alavi AH. Firefly algorithm with chaos. *Communications in Nonlinear Science and Numerical Simulation* **2013**; 18(1):89–98.

88. Eskandar H, Sadollah A, Bahreininejad A, Hamdi M. Water cycle algorithm - a novel metaheuristic optimization method for solving constrained engineering optimization problems, *Computers and Structures* **2012**; 110–111:151–166.

89. Bekdaş G, Nigdeli SM, Yang XS. Sizing optimization of truss structures using flower pollination algorithm. *Applied Soft Computing* **2015**; 37:322–331.

90. Kaveh A, Dadras A. A novel meta-heuristic optimization algorithm: thermal exchange optimization. *Advances in Engineering Software* **2017**; 110:69–84.

91. Yang XS, Cui Z, Xiao R, Gandomi AH, Karamanoglu M (Eds). *Swarm Intelligence and Bio-inspired Computation: Theory and Applications*, Elsevier, 2013.

92. Hasancebi O. Adaptive evolution strategies in structural optimization: enhancing their computational performance with applications to large-scale structures. *Computers and Structures* **2008**; 86(1–2):119–132.

93. Hasancebi O, Çarbaş S, Doğan E, Erdal F, Saka MP. Performance evaluation of metaheuristic search techniques in the optimum design of real size pin jointed structures. *Computers and Structures* **2009**; 87(5–6):284–302.

94. Hasancebi O, Çarbaş S, Doğan E, Erdal F, Saka MP. Comparison of non-deterministic search techniques in the optimum design of real size steel frames. *Computers and Structures* **2010** 88(17–18):1033–1048.

95. Kaveh A, Sabzi O. A comparative study of two meta-heuristic algorithms for optimum design of reinforced concrete frames. *International Journal of Civil Engineering* **2011**; 9(3):193–206.

96. Erdal F, Doğan E, Saka MP. Optimum design of cellular beams using harmony search and particle swarm optimizers. *Journal of Constructional Steel Research* **2011**; 67(2):237–247.

97. Kaveh A, Farhoudi N. A unified approach to parameter selection in meta-heuristic algorithms for layout optimization. *Journal of Constructional Steel Research* **2011**; 67(10):1453–1462.

98. Kaveh A, Zakian P. Optimal design of steel frames under seismic loading using two meta-heuristic algorithms, *Journal of Constructional Steel Research* **2013**; 82:111–130.

99. Kaveh A, Zolghadr A. Comparison of nine meta-heuristic algorithms for optimal design of truss structures with frequency constraints. *Advances in Engineering Software* **2014**; 76:9–30.

100. Hwang SF, He RS. Improving real-parameter genetic algorithm with simulated annealing for engineering problems. *Advances in Engineering Software* **2006**; 37(6):406–418.

101. Li JL, Huang ZB, Liu F, Wu QH. A heuristic particle swarm optimizer for optimization of pin connected structures. *Computers and Structures* **2007**; 85(7–8):340–349.

102. Kaveh A, Bakhshpoori T, Afshari E. An efficient hybrid particle swarm and swallow swarm optimization algorithm. *Computers and Structures* **2014**; 143:40–59.

103. Gholizadeh S. Layout optimization of truss structures by hybridizing cellular automata and particle swarm optimization. *Computers and Structures* **2013**; 125:86–99.

104. Kaveh A, Javadi SM. Shape and size optimization of trusses with multiple frequency constraints using harmony search and ray optimizer for enhancing the particle swarm optimization algorithm. *Acta Mechanica* **2014**; 225:1595–1605.

105. Kaveh A, Sheikholeslami R, Talatahari S, Keshvari-Ilkhichi M. Chaotic swarming of particles: a new method for size optimization of truss structures. *Advances in Engineering Software* **2014**; 67:136–147.

106. Kaveh A, Ilchi Ghazaan M. Hybridized optimization algorithms for design of trusses with multiple natural frequency constraints. *Advances in Engineering Software* **2015**; 79:137–147.

107. Talatahari S, Gandomi AH, Yang XS, Deb S. Optimum design of frame structures using the Eagle Strategy with Differential Evolution. *Engineering Structures* **2015**; 91:16–25.

108. Prayogo D, Cheng MY, Wu YW, Herdany AA, Prayogo, H. Differential Big Bang - Big Crunch algorithm for construction-engineering design optimization. *Automation in Construction* **2018***;* 85:290–304.

109. Pholdee N, Bureerat S. Performance enhancement of multiobjective evolutionary optimisers for truss design using an approximate gradient. *Computers and Structures* **2012**; 106–107:115–124.

110. Saka MP. Optimum design of steel frames using stochastic search techniques based on natural phenomena: a review. In: *Civil Engineering Computations: Tools and Techniques* (B.H.V. Topping, Ed.), Chapter 6, pp. 105-147. Saxe-Coburg Publications, Stirling, UK, 2007.

111. Deb K. An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering* **2000**; 186:311–338.

112. Coello Coello CA. 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* **2002**; 191:1245–1287.

113. Rajeev S, Krishnamoorthy CS. Discrete optimization of trusses using genetic algorithms. *ASCE Journal of Structural Engineering* **1992**; 118:1233–1250.

114. Hajela P, Lee E. Genetic algorithm in truss topological optimization. *International Journal of Solids and Structures* **1995**; 32:3341–3357.

115. Galante M. Genetic algorithms as an approach to optimize real-world trusses. *International Journal for Numerical Methods in Engineering* **1996**; 39:361–382.

116. Erbatur F, Hasancebi O, Tutuncu I, Kilic H. Optimal design of planar and space structures with genetic algorithms. *Computers and Structures* **2000**; 75:209–224.

117. Ali N, Behdinan K, Fawaz Z. Applicability and viability of a GA based finite element analysis architecture for structural design optimization. *Computers and Structures* **2003**; 81: 2259–2271.

118. Rahami H, Kaveh A, Gholipour Y. Sizing, geometry and topology optimization of trusses via force method and genetic algorithm. *Engineering Structures* **2008**; 30:2360–2369.

119. Schwefel HP. *Kybernetische evolution als strategie der experimentellen forschung in der strömungstechnik diplomarbeit*. Technische Universität, Berlin, 1965. (In German)

120. Storn R, Price K. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization* **1997**; 11:341–359, 1997.

121. Hasancebi O. Discrete approaches in evolution strategies based optimum design of steel frames. *Structural Engineering and Mechanics* **2007**; 26:191–210.

122. Wu CY, Tseng KY. Truss structure optimization using adaptive multi-population differential evolution. *Structural and Multidisciplinary Optimization* **2010**; 42:575–590.

123. Becerra RL, Coello Coello CA. Cultured differential evolution for constrained optimization. *Computer Methods in Applied Mechanics and Engineering* **2006**; 195:4303–4322.

124. Draa A, Bouzoubia S, Boukhalfa I. A sinusoidal differential evolution algorithm for numerical optimization. *Applied Soft Comp*uting **2015**; 27:99–126.

125. Balling RJ. Optimal steel frame design by simulated annealing. *Journal of Structural Engineering* **1991**; 117:1780–1795.

126. Bennage WA, Dhingra AK. Single and multi-objective structural optimization in discrete-continuous variables using simulated annealing. *International Journal for Numerical Methods in Engineering* **1995**; 38:2553–2573.

127. Shea K, Cagan J, Fenves SJ. A shape annealing approach to optimal truss design with dynamic grouping of members. *Journal of Mechanical Design* **2007**; 119:388–394.

128. Shea K, Cagan J. The design of novel roof trusses with shape annealing: assessing the ability of a computational method in aiding structural designers with varying design intent. *Design Studies* **1999**; 20:3–23.

129. Pantelides CP, Tzan SR. Modified iterated simulated annealing algorithm for structural synthesis. *Advances in Engineering Software* **2000; 31**:391–400.

130. Hasancebi O, Erbatur F. Layout optimization of trusses using simulated annealing. *Advances in Engineering Software* **2002**; 33:681–696.

131. Sonmez FO. Shape optimization of 2D structures using simulated annealing. *Computer Methods in Applied Mechanics and Engineering* **2007**; 196:3279–3299.

132. Degertekin SO. A comparison of simulated annealing and genetic algorithm for optimum design of nonlinear steel space frames. *Structural and Multidisciplinary Optimization* **2007**; 34:347–359.

133. Hasancebi O, Carbas S, Saka MP. Improving the performance of simulated annealing in structural optimization. *Structural and Multidisciplinary Optimization* 2010; **41**:189–203.

134. Chen TY, Su JJ. Efficiency improvement of simulated annealing in optimal structural designs. *Advances in Engineering Software* **2002**; 33:675–680.

135. Blachut J. Optimal barreling of steel shells via simulated annealing algorithm. *Computers and Structures* **2003**; 81:1941-1956.

136. Genovese K, Lamberti L, Pappalettere C. Improved global-local simulated annealing formulation for solving non-smooth engineering optimization problems. *International Journal of Solids and Structures* **2005**; 42:203–237.

137. Lamberti L, Pappalettere C. Weight optimization of skeletal structures with multi-point simulated annealing. *Computer Modeling in Engineering & Sciences* **2007**; 18:183–221.

138 Lamberti L. An efficient simulated annealing algorithm for design optimization of truss structures. *Computers and Structures* **2008**; 86:1936–1953.

139. Couceiro I, Paris J, Martinez S, Colominas I, Navarrina F, Casteleiro M. Structural optimization of lattice steel transmission towers. *Engineering Structures* **2016**; 117:274–286.

140. Hasancebi O, Tort C, Sahin S. Optimum design of steel lattice transmission line towers using simulated annealing and PLS-TOWER. *Computers and Structures* **2017**; 179:75–94.

141. J. Kennedy, R.C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann, San Francisco, USA, 2001.

142. A.P. Engelbrecht. *Fundamentals of Computational Swarm Intelligence*. John Wiley & Sons, Chichester, UK, 2005.

143. Perez RE, Behdinan K. Particle swarm approach for structural design optimization. *Computers and Structures* **2007**; 85:1579–1588.

144. Wilke DN, Kok S, Groenwold A. Comparison of linear and classical velocity update rules in particle swarm optimization: notes on diversity. *International Journal for Numerical Methods in Engineering* **2007**; 70:962–984.

145. Fourie P, Groenwold A. The particle swarm optimization algorithm in size and shape optimization. *Structural and Multidisciplinary Optimization* **2002**; 23:259–267.

146. Schutte JF, Groenwold A. Sizing design of truss structures using particle swarms. *Structural and Multidisciplinary Optimization* **2003**; 25:261–269.

147. Li JL, Huang ZB, Liu F. A heuristic particle swarm optimization method for truss structures with discrete variables. *Computers and Structures* **2009**; 87:435–443.

148. Serra M, Venini P. On some applications of ant colony to plane truss optimization. *Structural and Multidisciplinary Optimization* **2006**; 32:499-506.

149. Aydogdu I, Saka MP. Ant colony optimization of irregular steel space frames including the effect of warping. In: *Proceedings of the Twelfth International Conference on Civil, Structural and Environmental Engineering Computing* (B.V.H. Topping, L.F. Costa Neves, and R.C. Barros, Eds.), Madeira (Portugal), September 2009. Civil Comp Press, Dun Eaglais, UK, 2009.

150. Camp CV, Bichon J. Design of space trusses using ant colony optimization. *ASCE Journal of Structural Engineering* **2004**; 130:741–751.

151. Camp CV, Bichon J, Stovall SP. Design of steel frames using ant colony optimization. *ASCE Journal of Structural Engineering* **2005**; 131:369–379.

152. Kaveh A, Shojaee S. Optimal design of skeletal structures using ant colony optimization. *International Journal for Numerical Methods in Engineering* **2007**; 70:563–581.

153. Kaveh A, Talatahari S. An improved ant colony optimization for the design of planar steel frames. *Engineering Structures* **2010**; 32:864–873.

154. Luh GG, Lin CY. Optimum design of truss structures using ant algorithm. *Structural and Multidisciplinary Optimization* **2008**; 36:365-379.

155. Kaveh A, Abadi ASM. Cost optimization of a composite floor system using an improved harmony search algorithm. *Journal of Constructional Steel Research* **2010**; 66:664–669.

156. Pan QK, Suganthan PN, Tasgetiren MF, Liang JJ. A self-adaptive global best harmony search algorithm for continuous optimization problems. *Applied Mathematics and Computation* **2010**; 216(3):830–848.

157. Wang CM, Huang YF, Self-adaptive harmony search algorithm for optimization. *Expert Systems with Applications* **2010**; 37(4):2826–2837.

158. Camp CV. Design of space trusses using Big Bang-Big Crunch optimization. *Journal of Structural Engineering* **2007**; 133(7):999–1008.

159. Camp CV, Akin A. Design of retaining walls using big bang–big crunch optimization. *Journal of Structural Engineering* **2012**; 138(3):438–448.

160. Camp CV, Huq F. CO_2 and cost optimization of reinforced concrete frames using a big bang-big crunch algorithm. *Engineering Structures* **2013**; 48:363–372.

161. Kazemzadeh Azad S, Hasançebi O, Erol OK. Evaluating efficiency of big bang-big crunch algorithm in benchmark engineering optimization problems. *International Journal of Optimization in Civil Engineering* **2011**; 1(3):495–505.

162. Rafiee A, Talatahari S, Hadidi A. Optimum design of steel frames with semi-rigid connections using Big Bang-Big Crunch method. *Steel and Composite Structures* **2013**; 14(5):431–451.

163. Li WW, Pak C. Aeroelastic optimization study based on the X-56A model. In: *Proceedings of the AIAA Atmospheric Flight Mechanics Conference*, Atlanta (GA), USA, June 2014.

164. Kaveh A, Talatahari S. Size optimization of space trusses using Big Bang-Big Crunch algorithm. *Computers and Structures* **2009**; 87(17–18):1129–1140.

165. Kaveh A, Talatahari S. Optimal design of Schwedler and ribbed domes via hybrid Big Bang-Big Crunch algorithm. *Journal of Constructional Steel Research* **2010**; 66(3):412–419.

166. Kaveh A, Talatahari S. A discrete big bang-big crunch algorithm for optimal design of skeletal structures. *Asian Journal of Civil Engineering* **2010**; 11:103–122.

167. Kaveh A, Zolghadr A. Truss optimization with natural frequency constraints using a hybridized CSS-BBBC algorithm with trap recognition capability. *Computers and Structures* **2012**; 102–103:14–27.

168. Kaveh A, Mahdavi VR. Optimal design of structures with multiple natural frequency constraints using a hybridized BB-BC/Quasi-Newton algorithm. *Periodica Polytechnica Civil Engineering* **2013**; 57(1):27–38.

169. Kazemzadeh Azad S, Kazemzadeh Azad S, Hasancebi O. Structural optimization using big bang-big crunch algorithm: a review. *International Journal of Optimization in Civil Engineering* **2016**; 6(3):433–445.

170. Hasancebi O, Kazemzadeh Azad S. An exponential big bang-big crunch algorithm for discrete design optimization of steel frames. *Computers and Structures* **2012**; 110–111:167–179.

171. Hasancebi O, Kazemzadeh Azad S. Discrete size optimization of steel trusses using a refined Big Bang-Big Crunch algorithm. *Engineering Optimization* **2014**; 46(1):61–83.

172. Kazemzadeh Azad S, Hasancebi O, Kazemzadeh Azad S. Upper bound strategy for metaheuristic based design optimization of steel frames. *Advances in Engineering Software* **2013**; 57:19–32.

173. Kazemzadeh Azad S, Hasancebi O, Kazemzadeh Azad S, Erol OK. Upper bound strategy in optimum design of truss structures: a big bang-big crunch algorithm based application. *Advances in Structural Engineering* **2013**; 16(6):1035–1046.

174. Kazemzadeh Azad S, Hasancebi O, Kazemzadeh Azad S. Computationally efficient optimum design of large scale steel frames. *International Journal of Optimization in Civil Engineering* **2014**; 4(2):233–259.

175. Kazemzadeh Azad S. Enhanced hybrid metaheuristic algorithms for optimal sizing of steel truss structures with numerous discrete variables. *Structural and Multidisciplinary Optimization* **2017**; 55(6):2159–2180.

176. Lamberti L, Pappalettere C. A fast Big Bang-Big Crunch optimization algorithm for weight minimization of truss structures. In: Y. Tsompanakis, B.H.V. Topping (Eds.), *Proceedings of the Second International Conference on Soft Computing Technology in Civil, Structural and Environmental Engineering*. Civil Comp Press, Dun Eaglais, UK, 2011.

177. Rao RV, More KV. Design optimization and analysis of selected thermal devices using self-adaptive Jaya algorithm. *Energy Conversion and Management* **2017**; 140:24–35.

178. The MathWorks, *MATLAB® Release 2018b*, Austin, TX, USA, 2018.

179. G.N. Vanderplaats, *DOTs Users Manual, Version 4.20*. VR&D Inc.: Colorado Springs, CO, USA, 1995.

Appendix A

Details of geometry for the spatial 1938 and 3586-bar towers

Table A1. Element grouping for the two space towers optimized in this study. Group element numbers are indicated between parentheses. Elements shared by the two towers are typed in bold.

(1) 1-4	**(41) 177-184**	**(81) 357-364**	**(121) 695-702**	**(161) 1231-1242**	**(201) 1867-1890**	(241) 2723-2754
(2) 5-6	**(42) 185-186**	**(82) 365-366**	**(122) 703-718**	**(162) 1243-1266**	**(202) 1891-1902**	(242) 2755-2770
(3) 7-10	**(43) 187-190**	**(83) 367-370**	**(123) 719-726**	**(163) 1267-1278**	**(203) 1903-1914**	(243) 2771-2786
(4) 11-14	**(44) 191-194**	**(84) 371-374**	**(124) 727-734**	**(164) 1279-1290**	**(204) 1915-1938**	(244) 2787-2818
(5) 15-22	**(45) 195-202**	**(85) 375-382**	**(125) 735-750**	**(165) 1291-1314**	(205) 1939-1986	(245) 2819-2834
(6) 23-24	**(46) 203-204**	**(86) 383-384**	**(126) 751-758**	**(166) 1315-1326**	(206) 1987-2002	(246) 2835-2850
(7) 25-28	**(47) 205-208**	**(87) 385-388**	**(127) 759-766**	**(167) 1327-1338**	(207) 2003-2018	(247) 2851-2882
(8) 29-32	**(48) 209-212**	**(88) 389-392**	**(128) 767-782**	**(168) 1339-1362**	(208) 2019-2050	(248) 2883-2898
(9) 33-40	**(49) 213-220**	**(89) 393-400**	**(129) 783-790**	**(169) 1363-1374**	(209) 2051-2066	(249) 2899-2914
(10) 41-42	**(50) 221-222**	**(90) 401-402**	**(130) 791-798**	**(170) 1375-1386**	(210) 2067-2082	(250) 2915-2946
(11) 43-46	**(51) 223-226**	**(91) 403-406**	**(131) 799-814**	**(171) 1387-1410**	(211) 2083-2114	(251) 2947-2962
(12) 47-50	**(52) 227-230**	**(92) 407-410**	**(132) 815-822**	**(172) 1411-1422**	(212) 2115-2130	(252) 2963-2978
(13) 51-58	**(53) 231-238**	**(93) 411-418**	**(133) 823-830**	**(173) 1423-1434**	(213) 2131-2146	(253) 2979-3010
(14) 59-60	**(54) 239-240**	**(94) 419-420**	**(134) 831-846**	**(174) 1435-1458**	(214) 2147-2178	(254) 3011-3026
(15) 61-64	**(55) 241-244**	**(95) 421-424**	**(135) 847-854**	**(175) 1459-1470**	(215) 2179-2194	(255) 3027-3042
(16) 65-68	**(56) 245-248**	**(96) 425-428**	**(136) 855-862**	**(176) 1471-1482**	(216) 2195-2210	(256) 3043-3074
(17) 69-76	**(57) 249-256**	**(97) 429-436**	**(137) 863-878**	**(177) 1483-1506**	(217) 2211-2242	(257) 3075-3090
(18) 77-78	**(58) 257-258**	**(98) 437-462**	**(138) 879-886**	**(178) 1507-1518**	(218) 2243-2258	(258) 3091-3106
(19) 79-82	**(59) 259-262**	**(99) 463-470**	**(139) 887-894**	**(179) 1519-1530**	(219) 2259-2274	(259) 3107-3138
(20) 83-86	**(60) 263-266**	**(100) 471-478**	**(140) 895-910**	**(180) 1531-1554**	(220) 2275-2306	(260) 3139-3154
(21) 87-94	**(61) 267-274**	**(101) 479-494**	**(141) 911-918**	**(181) 1555-1566**	(221) 2307-2322	(261) 3155-3170
(22) 95-96	**(62) 275-276**	**(102) 495-502**	**(142) 919-926**	**(182) 1567-1578**	(222) 2323-2338	(262) 3171-3202
(23) 97-100	**(63) 277-280**	**(103) 503-510**	**(143) 927-942**	**(183) 1579-1602**	(223) 2339-2370	(263) 3203-3218
(24) 101-104	**(64) 281-284**	**(104) 511-526**	**(144) 943-978**	**(184) 1603-1614**	(224) 2371-2386	(264) 3219-3234
(25) 105-112	**(65) 285-292**	**(105) 527-534**	**(145) 979-990**	**(185) 1615-1626**	(225) 2387-2402	(265) 3235-3266
(26) 113-114	**(66) 293-294**	**(106) 535-542**	**(146) 991-1002**	**(186) 1627-1650**	(226) 2403-2434	(266) 3267-3282
(27) 115-118	**(67) 295-298**	**(107) 543-558**	**(147) 1003-1026**	**(187) 1651-1662**	(227) 2435-2450	(267) 3283-3298
(28) 119-122	**(68) 299-302**	**(108) 559-566**	**(148) 1027-1038**	**(188) 1663-1674**	(228) 2451-2466	(268) 3299-3330
(29) 123-130	**(69) 303-310**	**(109) 567-574**	**(149) 1039-1050**	**(189) 1675-1698**	(229) 2467-2498	(269) 3331-3346
(30) 131-132	**(70) 311-312**	**(110) 575-590**	**(150) 1051-1074**	**(190) 1699-1710**	(230) 2499-2514	(270) 3347-3362
(31) 133-136	**(71) 313-316**	**(111) 591-598**	**(151) 1075-1086**	**(191) 1711-1722**	(231) 2515-2530	(271) 3363-3394
(32) 137-140	**(72) 317-320**	**(112) 599-606**	**(152) 1087-1098**	**(192) 1723-1746**	(232) 2531-2562	(272) 3395-3410
(33) 141-148	**(73) 321-328**	**(113) 607-622**	**(153) 1099-1122**	**(193) 1747-1758**	(233) 2563-2578	(273) 3411-3426
(34) 149-150	**(74) 329-330**	**(114) 623-630**	**(154) 1123-1134**	**(194) 1759-1770**	(234) 2579-2594	(274) 3427-3458
(35) 151-154	**(75) 331-334**	**(115) 631-638**	**(155) 1135-1146**	**(195) 1771-1794**	(235) 2595-2626	(275) 3459-3474
(36) 155-158	**(76) 335-338**	**(116) 639-654**	**(156) 1147-1170**	**(196) 1795-1806**	(236) 2627-2642	(276) 3475-3490
(37) 159-166	**(77) 339-346**	**(117) 655-662**	**(157) 1171-1182**	**(197) 1807-1818**	(237) 2643-2658	(277) 3491-3522
(38) 167-168	**(78) 347-348**	**(118) 663-670**	**(158) 1183-1194**	**(198) 1819-1842**	(238) 2659-2690	(278) 3523-3538
(39) 169-172	**(79) 349-352**	**(119) 671-686**	**(159) 1195-1218**	**(199) 1843-1854**	(239) 2691-2706	(279) 3539-3554
(40) 173-176	**(80) 353-356**	**(120) 687-694**	**(160) 1219-1230**	**(200) 1855-1866**	(240) 2707-2722	(280) 3555-3586

As mentioned in Section 4.5, the 3586-bar tower was obtained by adding the bottom segment to the 1938-bar tower. Hence, from the top of the spire to the ground level, the 3586-bar tower includes: (a) a 15-storey square-based pyramid segment of height 60 m; b) a 10-storey square-based prismatic segment of height 40 m and side length 5 m; c) a 15-storey octagon-based prismatic segment of height 75 m and radius 5 m; d) a 20-storey dodecagon-based prismatic segment of height 100 m and radius 8 m; e) a 25-storey hexadecagon-based

prismatic segment of height 125 m and radius 12 m. The three intermediate modules (each is 5 m tall) of the tower, connect adjacent segmentsof different profile. The 1938-bar tower includes instead only segments (a), (b), (c) and (d) as well as two intermediate modules of height 5 m connecting segments (b) and (c), and (c) and (d).

Figure A1 illustrates in detail the segments and the connecting modules of the 3586-bar tower. Figures A1(a) through (g) cover also the case of the 1938-bar tower. Element group numbering progresses from the top to the bottom of the structure.

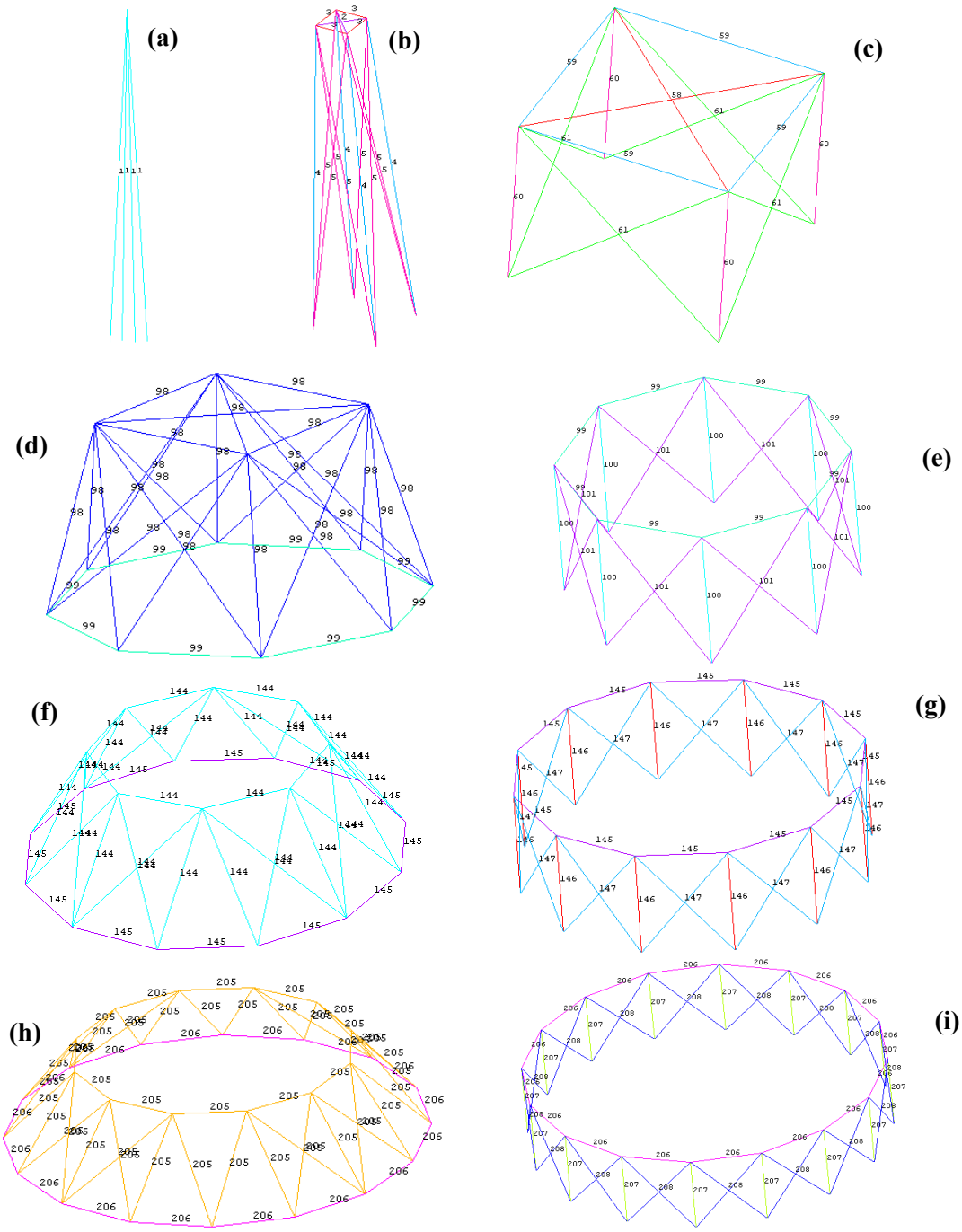

Figure A1. Details of element group numbering for the spatial 3586-bar truss tower: a) Top-level storey of thesquare-based pyramid segment; b) Second storey from the top of the tower; c) Top-level storey of the square-based prismatic segment; d) Transition from the bottom level-storey of the square-based prismatic segment to the top-level storey of the octagon-based prismatic segment; e) Top-level storey of the octagon-based prismatic segment; f) Transition from the bottom-level storey of the octagon-based prismatic segment to the top-level storey of the dodecagon-based prismatic segment; g) Top-level storey of the dodecagon-based prismatic segment; h) Transition from the bottom-level storey of the dodecagon-based prismatic segment to the top-level storey of the hexadecagon-based prismatic segment; i) Top-level storey of the hexadecagon-based prismatic segment.

Appendix B

Preliminary results for LSSO–HBBBCJA

B1. The LSSO–BBBCJA algorithm

The Big Bang-Big Crunch (BBBC) algorithm [13] reproduces the cycles of expansion and contraction of the universe. In the optimization process, N_{POP} designs are randomly generated (explosion) to form a population and the center of mass of this population is defined as a weighted average of these designs (contraction) where each weighing coefficient is the cost function for a trial design. A new population is randomly generated by perturbing design variables in the neighborhood of the center of mass (either the true center of mass is considered or the best design of the population is set as center of mass). This process is repeated until no significant improvement in design is obtained.

The possibility of using the same hybridization strategy for another metaheuristic algorithm such as Big Bang–Big Crunch (BBBC) is also investigated by solving two highly nonlinear design problems including up to 84 variables, (i) shape optimization of a concrete dam and (ii) discrete layout optimization of a planar steel frame, to prove the feasibility of the proposed approach. BBBC has been selected for hybridization with line search and JAYA-based strategies because it is a well established metaheuristic algorithm widely used in structural optimization and has a simpler formulation than the harmony search method used for developing the LSSO–HHSJA algorithm described in this dissertation. However, computational efficiency of BBBC is affected by the considerably high computational cost of the explosion phase. This may be a considerable drawback in large scale structural optimization.

The new algorithm, Large Scale Structural Optimization – Hybrid Big Bang–Big Crunch JAYA (LSSO–HBBBCJA) is now outlined. The initial population of N_{POP} designs is generated in the same way as for LSSO–HHSJA (see Eq. (12) of Section 3.1)

$$x_j^k = x_j^L + \rho_j^{\,k}\,(x_j^U - x_j^L) \qquad \begin{cases} j = 1, \dots, NDV \\ k = 1, \dots, N_{POP} \end{cases} \tag{B1}$$

where *NDV* is the number of optimization variables and ρ_j^k is a random number in the (0,1) interval. The algorithm does not require any setting of internal parameters except for the population size N_{POP}. Cost function and optimization constraints are evaluated for all designs. The best design $\mathbf{X_{OPT}}$ is determined.

- Step 1. *Definition of the center of mass*.

  The coordinates of the center of mass of the population $\mathbf{X_{CM}}(x_{CM,1},x_{CM,2},\ldots,x_{CM,NDV})$ are defined as:

$$x_{CM,j} = \left(\sum_{k=1}^{NPOP} \frac{x_j^k}{W^k} \right) / \left(\sum_{k=1}^{NPOP} \frac{1}{W^k} \right) \qquad (j=1,\ldots,NDV) \qquad (B2)$$

  where x_j^k is the value of the j^{th} optimization variable stored in the k^{th} trial design, W^k is the cost function value for the k^{th} trial design. Penalty functions can be used to sort designs. The weighting coefficients $1/W^k$ make position of center mass be more sensitive to the best designs stored in the population.

- Step 2. *Evaluation of the center of mass and progressive update of* $\mathbf{X_{CM}}$ *as current best record*.

  Cost function and optimization constraints are evaluated at $\mathbf{X_{CM}}$. As mentioned above, BBBC formulations usually converge to the optimum design by updating the position of $\mathbf{X_{CM}}$. However, there is no guarantee that the new $\mathbf{X_{CM}}$ can always improve the current best record or may be the center of a better population. Since $\mathbf{X_{CM}}$ represents a weighted average of candidate designs, its quality will be in between the worst and best individuals included in the population. Similar to the powerful hybrid BBBC algorithm of Ref. [53], LSSO–HBBBCJA accounts for two possible scenarios: (i) $\mathbf{X_{CM}}$ is better than $\mathbf{X_{OPT}}$; (ii) $\mathbf{X_{CM}}$ is worse than $\mathbf{X_{OPT}}$.

  In order to make scenario (i) more likely to occur, LSSO–HBBBCJA defines another center of mass $\mathbf{X_{CM,JAYA}}$ using a JAYA-based equation where $\mathbf{X_{CM}}$ is taken as the base point while the current best record $\boldsymbol{X_{OPT}}$ and the second best design included in the current population $\mathbf{X_{2nd\text{-}best}}$ are taken as the best and worst designs. Each design variable is hence updated as follows:

$$x_{CM-JAYA,j} = x_{CM,j} + \beta_{1,j}(x_{OPT,j} - x_{TR,j}) - \beta_{2,j}(x_{2nd-best,j} - x_{TR,j}) \qquad (B3)$$

where $x_{2nd\text{-}best,j}$ is the value of the j[th] variable stored in the 2[nd] best design of the population while $\beta_{1,j}$ and $\beta_{2,j}$ are two random numbers in the interval (0,1).

Basically, LSSO–HBBBCJA tries to perturb the position of the center of mass by combining movements in the neighborhood of the best two solutions currently available in the population.

If $\boldsymbol{X}_{CM}$ or $\boldsymbol{X}_{CM,JAYA}$ are better than $\boldsymbol{X}_{OPT}$, it is reset as $\boldsymbol{X}_{OPT}$. The former best record becomes the second best design while the worst design is removed from the population.

The same JAYA-based strategy used for case (1) of LSSO–HHSJA (see Section 3.2.1 of the main part of this dissertation) is utilized to update the rest of the population thus trying to improve the quality of the new designs that will be used for determining the position of the new center of mass. Each $(\mathbf{X}_{NPOP-2})^r$ design stored in the population is tentatively updated by LSSO–HBBBCJA using Eq. (B4), with $r \in (N_{POP}-2)$:

$$(\boldsymbol{X}_{\mathrm{NPOP-2}})^{r,new} = (\boldsymbol{X}_{\mathrm{NPOP-2}})^{r} + \boldsymbol{\omega_1}(\boldsymbol{X}_{\mathrm{OPT}} - (\boldsymbol{X}_{\mathrm{NPOP-2}})^{r}) - \boldsymbol{\omega_2}(\boldsymbol{X}_{\mathrm{worst}} - (\boldsymbol{X}_{\mathrm{NPOP-2}})^{r}) \quad \text{(B4)}$$

where $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_2$ are two vectors of *NDV* random numbers in the interval [0,1]: in particular, $\omega_{1,j}$ and $\omega_{2,j}$ are generated for the j[th] component of the processed harmony, best and worst designs. Since the goal is to improve the $(\mathbf{X}_{NPOP-2})^r$ design, LSSO–HBBBCJA tries to search along the descent direction $(\mathbf{X}_{OPT}-(\mathbf{X}_{NPOP-2})^r)$ with respect to $(\mathbf{X}_{NPOP-2})^r$ and escape from the worst design of the population $\mathbf{X}_{worst}$, which certainly will not improve $(\mathbf{X}_{NPOP-2})^r$.

If $(\mathbf{X}_{NPOP-2})^{r,new}$ is feasible and $W((\mathbf{X}_{NPOP-2})^{r,new}) < W((\mathbf{X}_{NPOP-2})^r)$, it replaces the old design $(\mathbf{X}_{NPOP-2})^r$ of population. Otherwise, $(\mathbf{X}_{NPOP-2})^r$ is retained. Population is reordered based on the cost of each design and LSSO–HBBBCJA checks for convergence (Step 4).

It should be noted that LSSO–HHSJA updates the current best record with a trial design $\mathbf{X}_{TR}$, which may be totally unrelated to the rest of the population. Conversely, LSSO–HBBBCJA tries to update the position of $\mathbf{X}_{OPT}$ with the new center of mass of the population thus including information on the overall characteristics of the population.

- Step 3. *Evaluation of* $\mathbf{X}_{CM}$ *and generation of different trial designs*.

If $\mathbf{X}_{CM}$ and $\mathbf{X}_{CM\text{-}JAYA}$ could not improve $\mathbf{X}_{OPT}$ (i.e. if $W(\mathbf{X}_{CM})>W(\mathbf{X}_{OPT})$), LSSO–HBBBCJA uses a much simpler strategy than the hybrid BBBC algorithm of Ref. [53]. A new trial design $\mathbf{X}_{TR}^{mirr}$ is still defined with the mirroring strategy:

$$\mathbf{X}_{TR}^{mirr}=(1+\eta_{mirr})\cdot\mathbf{X}_{OPT}-\eta_{mirr}\cdot\mathbf{X}_{CM} \tag{B5}$$

where η_{mirr} is a random number in the interval (0,1). The mirroring strategy attempts to turn the non-descent direction ($\mathbf{X}_{CM}-\mathbf{X}_{OPT}$) into the descent direction ($\mathbf{X}_{TR}^{mirr}-\mathbf{X}_{OPT}$). Using a random number smaller than one reduces the risk of violating optimization constraints. The $\mathbf{X}_{CM}$ vector corresponds to best among the center of mass defined by Eq. (B2) and the JAYA-corrected center of mass $\boldsymbol{X}_{CM\text{-}JAYA}$ defined by Eq. (B3).

If $W(\mathbf{X}_{TR}^{mirr})<W(\mathbf{X}_{OPT})$ and $\mathbf{X}_{TR}^{mirr}$ is feasible, Step 2 is executed.

If $W(\mathbf{X}_{TR}^{mirr})<W(\mathbf{X}_{OPT})$ but $\mathbf{X}_{TR}^{mirr}$ is infeasible, an approximate line search is performed on the descent direction $\boldsymbol{S}_{TR}=(\mathbf{X}_{TR}^{mirr}-\mathbf{X}_{OPT})$ limited by $\mathbf{X}_{OPT}$ and $\mathbf{X}_{TR}^{mirr}$. This process is similar to that described for LSSO–HHSJA. If a better design than $\mathbf{X}_{OPT}$ is obtained, Step 2 is executed. If the approximate line search is unsuccessful, LSSO–HBBBCJA algorithm uses the same strategy of LSSO–HHSJA to turn $\mathbf{X}_{TR}$ feasible. For that purpose, two trial designs $(\mathbf{X}_{TR})'$ and $(\mathbf{X}_{TR})''$ are defined as follows:

$$\begin{cases}\left(\boldsymbol{X}_{\mathrm{TR}}{}^{\mathrm{mirr}}\right)' = (1+\eta_{\mathrm{mirr}})\boldsymbol{X}_{\mathrm{OPT}} - \eta_{\mathrm{mirr}}\boldsymbol{X}_{\mathrm{TR}} \\ \left(\boldsymbol{X}_{\mathrm{TR}}{}^{\mathrm{mirr}}\right)'' = \boldsymbol{X}_{\mathrm{TR}} + \boldsymbol{\omega_1}(\boldsymbol{X}_{\mathrm{OPT}} - \boldsymbol{X}_{\mathrm{TR}}) - \boldsymbol{\omega_2}(\boldsymbol{X}_{2\mathrm{ndBEST}} - \boldsymbol{X}_{\mathrm{TR}})\end{cases} \tag{B6}$$

where η_{mirr} is a random number in the interval (0,1) while random vectors $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_2$ are similar to those of Eq. (B4). The best among trial designs $\left(\boldsymbol{X}_{\mathrm{TR}}{}^{\mathrm{mirr}}\right)'$ and $\left(\boldsymbol{X}_{\mathrm{TR}}{}^{\mathrm{mirr}}\right)''$ is compared with $\boldsymbol{X}_{OPT}$: if it improves $\boldsymbol{X}_{OPT}$, Step 2 is executed. Conversely, should $\left(\boldsymbol{X}_{\mathrm{TR}}{}^{\mathrm{mirr}}\right)'$ and $\left(\boldsymbol{X}_{\mathrm{TR}}{}^{\mathrm{mirr}}\right)''$ remain unfeasible, a trial design between $\mathbf{X}_{OPT}$ and $\mathbf{X}_{2ndBEST}$ is generated as follows:

$$\boldsymbol{X}_{\mathrm{TR}} = \boldsymbol{X}_{\mathrm{OPT}} + \alpha(\boldsymbol{X}_{\mathrm{OPT}} - \boldsymbol{X}_{2\mathrm{ndBEST}}) \tag{B7}$$

where α is a random number between 0 and 1. The new trial design will be in all likelihood feasible if both $\mathbf{X}_{\mathbf{OPT}}$ and $\mathbf{X}_{\mathbf{2ndBEST}}$ are feasible.

Should the best among designs $\mathbf{X}_{\mathbf{TR}}^{\mathbf{mirr}}$ and $(\mathbf{X}_{\mathbf{TR}}^{\mathbf{mirr}})$" still yield a larger cost function value or a larger penalized cost than $W(\mathbf{X}_{\mathbf{OPT}})$, it is ranked with respect to population. LSSO–HBBBCJA utilizes the JAYA-based Eq. (B4) to update the (N_{POP}–p) designs ranking below $\boldsymbol{X}_{\mathrm{TR}}$ or $\boldsymbol{X}_{\mathrm{TR}}^{\mathrm{mirr}}$. Each new design is then evaluated as explained for Step 2. Convergence check is done in Step 4.

- *Step 4. Check for convergence and perform a new explosion if necessary.*

  LSSO–HBBBCJA checks if the best design of the population has been improved in the current optimization cycle. Convergence check is performed after operations planned in Steps 2 and 3 are completed. Let be $(\mathbf{X}_{\mathbf{OPT}})^{\mathrm{init}}$ and $(\mathbf{X}_{\mathbf{OPT}})^{\mathrm{final}}$ the best designs at the beginning and at the end of the current optimization cycle. If $(\mathbf{X}_{\mathbf{OPT}})^{\mathrm{final}}$ is better than $(\mathbf{X}_{\mathbf{OPT}})^{\mathrm{init}}$, LSSO–HBBBCJA checks for convergence using the same criterion adopted for LSSO–HHSJA. Standard deviations are normalized with respect to the average design $\boldsymbol{X}_{\mathbf{aver}} = \left(\sum_{\mathrm{k}=1}^{N_{\mathrm{POP}}} \boldsymbol{X}_{\mathbf{k}}\right)/N_{\mathrm{POP}}$ and the average weight $W_{\mathrm{aver}} = \left(\sum_{\mathrm{k}=1}^{N_{\mathrm{POP}}} W(\boldsymbol{X}_{\mathbf{k}})\right)/N_{\mathrm{POP}}$.
  The termination criterion is:

$$\mathrm{Max}\left\{\begin{array}{c}\dfrac{\mathrm{STD}\{\|\boldsymbol{X}_{\mathbf{1}}-\boldsymbol{X}_{\mathbf{aver}}\|,\|\boldsymbol{X}_{\mathbf{2}}-\boldsymbol{X}_{\mathbf{aver}}\|,\ldots,\|\boldsymbol{X}_{\mathbf{NPOP}}-\boldsymbol{X}_{\mathbf{aver}}\|\}}{\|\boldsymbol{X}_{\mathbf{aver}}\|};\\ \dfrac{\mathrm{STD}\{W_1,W_2,\ldots,W_{\mathrm{NPOP}}\}}{W_{\mathrm{aver}}}\end{array}\right\} \leq \varepsilon_{\mathrm{CONV}} \qquad \text{(B8)}$$

  where the convergence limit $\varepsilon_{\mathrm{CONV}}$ is 10^{-15}, less than the double precision limit of available computers. Steps 1 through 3 are repeated until the LSSO–HBBBCJA algorithm converges to the global optimum.
  If design is improved in the current iteration, a new explosion is performed about $\mathbf{X}_{\mathbf{OPT}}$ trying to generate a higher quality population. The following equation is utilized:

$$\mathrm{x}_{\mathrm{j}}^{\mathrm{k}} = \mathrm{x}_{\mathrm{OPT,j}} - \rho_{\mathrm{j}}^{\mathrm{k}}\left(\mathrm{x}_{\mathrm{OPT,j}} - \mathrm{x}_{\mathrm{CM,j}}\right) \qquad \begin{cases} j = 1,\ldots,NDV \\ k = 1,\ldots,N_{POP} \end{cases} \qquad \text{(B9)}$$

  where $\rho_{\mathrm{j}}^{\mathrm{k}}$ is a random number in the interval (0,2) to generate the j^{th} variable of the k^{th} design. The interval (0,2) is large enough to avoid stagnation near the current best record.

If $x_j^k < x_j^L$ or $x_j^k > x_j^U$, x_j^k is reset to $x_j^k = (x_j^L + x_{OPT,j})/2$ or $x_j^k = (x_{OPT,j} + x_j^U)/2$, respectively.

The new population is generated by perturbing the current best record $\mathbf{X}_{OPT}$ along the direction $-(\mathbf{X}_{OPT}-\mathbf{X}_{CM})$, opposite to the non-descent direction $(\mathbf{X}_{CM}-\mathbf{X}_{OPT})$. The rationale of Eq. (B9) is to search for descent directions with respect to $\mathbf{X}_{OPT}$ by decomposing a potentially descent direction in its components. The new designs are compared with the previous population and only the best N_{POP} designs are retained in the new population. This elitist strategy allows to keep the $\mathbf{X}_{OPT}$ design in the population passed into the next optimization iteration should all of the new N_{POP} designs generated with Eq. (B9) be worse than $\mathbf{X}_{OPT}$. The optimization process is continued from Step 1.

- *Step 5. End optimization process.*

  The present LSSO–HBBBCJA algorithm terminates the optimization process and writes the output data in the results file.

B2. Shape optimization of a concrete gravity dam

The first benchmark problem for testing LSSO–HBBBCJA regards the shape optimization of a concrete gravity dam located in the Shafaroud region of Northern Iran. The cross-section of the dam to be optimized is schematized in Figure B1. The objective is to minimize the concrete volume V_{dam} per unit width using the nine shape variables (x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9) indicated in the figure. The height of the dam is H_{dam}=150 m, the water elevation in the normal state in upstream of dam is h_{water}=145 m, the height of the sediments is h_{sed}=7 m.

The optimization problem includes 8 constraints and is formulated as follows:

$$\begin{cases} \text{Min } V_{dam} = H_{dam}X_1 + \frac{1}{2}X_2X_3 + (X_4 + X_6)X_3 + \frac{1}{2}X_4X_5 + X_5X_6 + \frac{1}{2}X_6X_7 + \frac{1}{2}X_8X_9 \\ X_2 + X_4 + X_6 = H_{dam} \\ X_4 + X_6 \le X_8 \\ SFS \ge 4 \\ SFO \ge 1.5 \\ 0 \le \sigma_U \le \sigma_{max} \\ 0 \le \sigma_D \le \sigma_{max} \end{cases} \quad \text{(B10)}$$

where: SFS is the safety factor against dam sliding; SFO is the safety factor against dam overturning; σ_U, σ_D and σ_{max} are the vertical fatigue specific loads in upstream and

downstream, and the corresponding fatigue limit, respectively. Side constraints of shape variables are set as follows:

$1 \leq X_1, X_3 \leq 20$ m; $1 \leq X_2 \leq 50$ m; $5 \leq X_4, X_5, X_6, X_7 \leq 100$ m; $50 \leq X_8, X_9 \leq 150$ m.

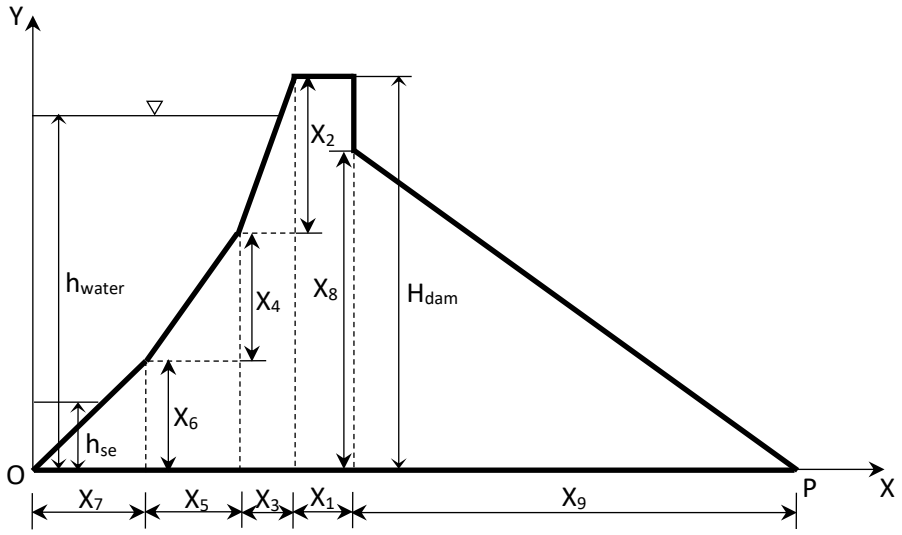

Figure B1. Schematic of the small concrete gravity dam.

The SFS factor in Eq. (B10) is defined as $(f \cdot \Sigma F_v + \sigma \cdot b)/\Sigma F_H$, where f is the static friction coefficient, σ is the allowable shear tension in the cutting surface and b is the dam base defined as $b=(X_1+X_3+X_5+X_7+X_9)$. The vertical forces F_V acting on dam are the concrete weight W, the vertical components of water pressure Fh_v and sediment pressure P_{SV}, the uplift force generated by the dam basement F_{uplift}, and the earthquake force F_E: forces W, Fh_v and P_{SV} are directed downward while forces F_{uplift} and F_E are directed upward. The horizontal forces F_H acting on the dam in the positive X-direction are the corresponding components of water pressure Fh_h and sediment pressure P_{SO}.

The SFO factor in Eq. (B10) is defined as $\Sigma M_R/\Sigma M_o$ where M_R and M_o, respectively, are the torque of resisting forces (W, Fh_v, P_{SV}) and the torque of driving forces (F_{uplift}, Fh_h, P_{SO}) on the dam.

The vertical fatigue loads σ_U and σ_D in Eq. (B10) are respectively defined as $(\Sigma F_v/b - 6\Sigma M_o/b^2)$ and $(\Sigma F_v/b + 6\Sigma M_o/b^2)$.

More details on this optimization problem can be found in [53] and in the references cited therein. The total height of the dam must remain equal to 150 m. Furthermore, the

earthquake force is subtracted from the dam gravity load thus reducing the resultant resisting moment ΣM_R and making the constraint on overturning (i.e. SFO≥0) harder to be satisfied. The problem formulation includes an additional horizontal earthquake force in the reservoir $2 \cdot F_E$ that amplifies overturning. This increased severity of design requirements.

The best solutions obtained in [53] for this test problem ranked as follows: 7108.772 m^3 for hybrid BBBC, 7109.608 m^3 for hybrid HS, 7113.456 m^3 for HFSA and 7117.437 m^3 for improved/parameterless JAYA. All designs are considerably more efficient than the actual structural configuration of the dam, which has a concrete volume unit width of 10502.1 m^3. The number of structural analyses required by the optimization was as follows: 17708 for hybrid HS, 18128 for hybrid BBBC, 18232 for HFSA and 19917 for improved/parameterless JAYA.

In this study, LSSO–HBBBCJA was forced to complete the optimization process within only 16000 structural analyses, a 10% smaller computational budget than for the fastest optimizer of the literature. The population size was set equal to 20, consistently with Ref. [53]. Thirty independent optimization runs were carried out starting from different populations randomly generated. Remarkably, LSSO–HBBBCJA found an average optimized volume of 7102.318 m^3 with a standard deviation of 3.976 m^3, thus improving the results reported in the literature. The optimized shape obtained by LSSO–HBBBCJA is fully consistent with the optimized designs, as is shown in Figure B2.

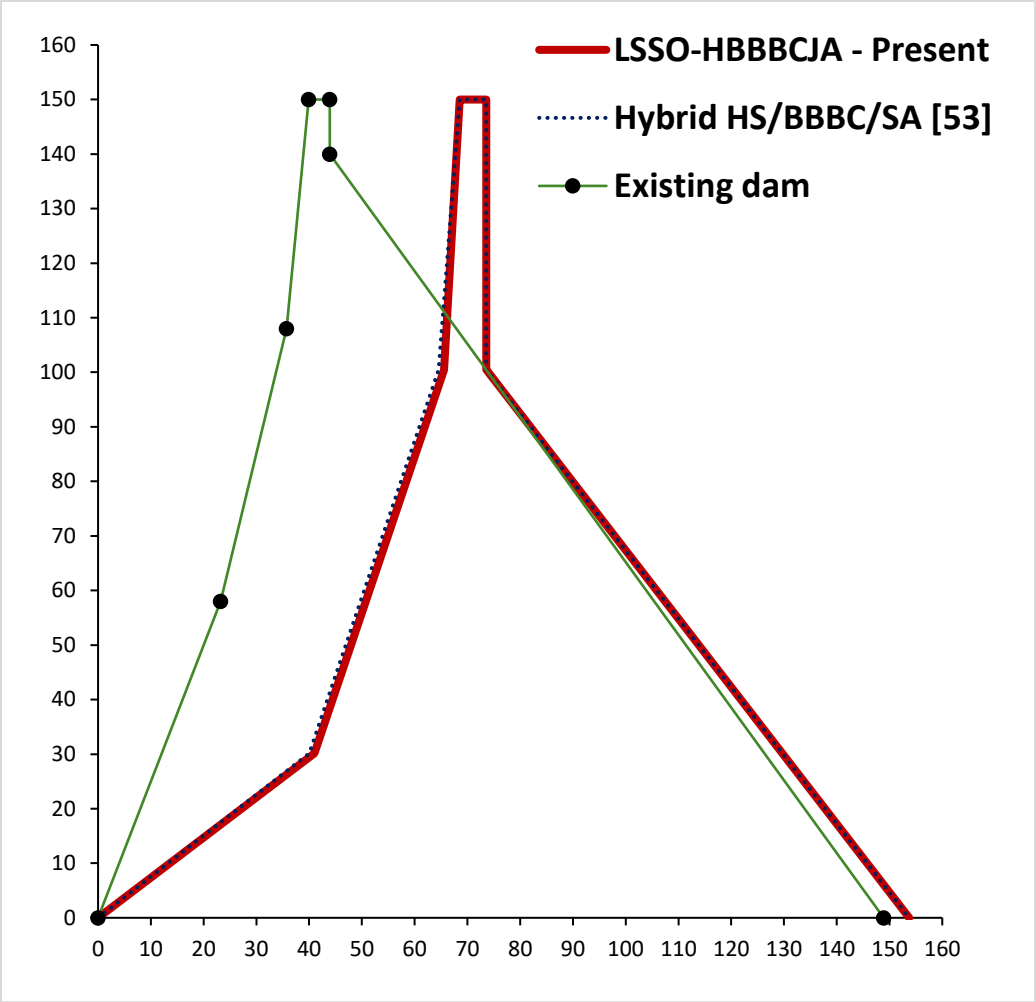

Figure B2. Comparison of optimal shapes found for the concrete gravity dam problem.

B3. Discrete sizing/layout optimization of a planar 2-bay 9-story frame

The second benchmark problem for testing LSSO–HBBBCJA the planar 2-bay 9-story frame shown in Figure B3. The frame is made of steel (density is 8303.971 kg/m^3, Young's modulus is 206.9 GPa) and has 45 I-beam elements connected by 30 nodes. Beams are grouped into twelve groups: the first three groups include, respectively, elements 1 through 9, elements 19 through 27, and elements 37 through 45. The other nine groups include, respectively, elements 10 and 28, elements 11 and 29, elements 12 and 30, elements 13 and 31, elements 14 and 32, elements 15 and 33, elements 16 and 34, elements 17 and 35, elements 18 and 36. The schematic of element cross-section and its local reference system also are shown in Fig. B3. The geometric dimensions (b_f,h_w,t_w,t_f) of each cross-section group are included as sizing variables. Since there are four design variables for each group, forty-eight sizing variables

are considered in the optimization process. In addition, the coordinates of nodes 5, 8, 11, 14, 17, 20, 23, 26, 29, 6, 9, 12, 15, 18, 21, 24, 27 and 30 are included as thirty-six layout variables. Therefore, the frame structure is optimized with 84 design variables.

The structural weight of the frame can be expressed as:

$$W(\mathbf{X}) = \rho g \sum_{j=1}^{84} \left(2b_f^j t_f^j + h_w^j t_w^j\right) \sqrt{(x_{j1} - x_{j2})^2 + (y_{j1} - y_{j2})^2} \tag{B11}$$

where $x_{j1,2}$, $y_{j1,2}$ are the coordinates of the nodes limiting the j^{th} element of the structure, that are included as layout variables in the optimization process.

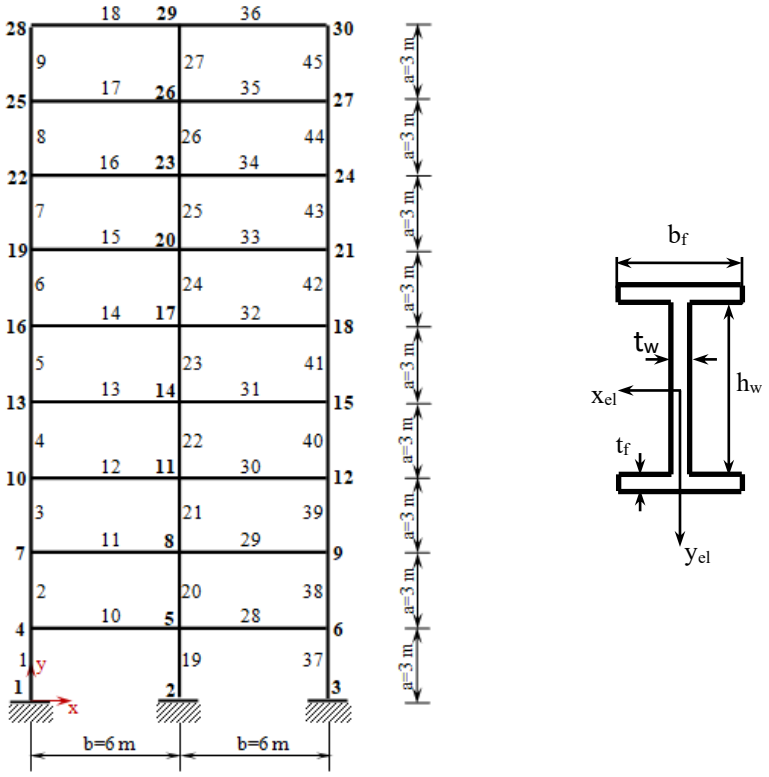

Figure B3. Schematic of the planar 2-bay 9-story frame and I-beam element cross-section nomenclature.

Sizing variables can be selected from the following sets of discrete values: (i) for b_f and h_w, $D_1 \equiv [1,2,3,\ldots,28,29,30]$ cm with the additional constraints $0.05 \leq b_f/h_w \leq 1$; (ii) for t_f and t_w, $D_2 \equiv [1,2,3,\ldots,98,99,100]$ mm with the additional constraints $0.005 \leq t_f/h_w \leq 0.1$ and $0.005 \leq t_w/h_w \leq 0.1$. Coordinates of nodes 1, 2, 3, 4, 7, 10, 13, 16, 19, 22, 25 and 28 are fixed to the initial configuration shown in Figure B3. The other coordinates can take discrete values

with a resolution of 0.1 m: (ii) for X-coordinates, $D_3 \equiv [0.1,0.2,0.3\ldots,11.8,11.9,12]$ m; (ii) for Y-coordinates, $D_4 \equiv [0.1,0.2,0.3\ldots,26.8,26.9,27]$ m. Additional constraints on nodal coordinates are set as follows: $X_4<X_5<X_6$, $X_7<X_8<X_9$, $X_{10}<X_{11}<X_{12}$, $X_{13}<X_{14}<X_{15}$, $X_{16}<X_{17}<X_{18}$, $X_{19}<X_{20}<X_{21}$, $X_{22}<X_{23}<X_{24}$, $X_{25}<X_{26}<X_{27}$ and $X_{28}<X_{29}<X_{30}$; $Y_1<Y_4<Y_7<Y_{10}<Y_{13}<Y_{16}<Y_{19}<Y_{22}<Y_{23}<Y_{25}<Y_{28}$, $Y_2<Y_5<Y_8<Y_{11}<Y_{14}<Y_{17}<Y_{20}<Y_{23}<Y_{26}<Y_{29}$ and $Y_3<Y_6<Y_9<Y_{12}<Y_{15}<Y_{18}<Y_{21}<Y_{24}<Y_{27}<Y_{30}$.

The number of available discrete combinations of optimization variables for this design example is in principle very large (i.e. $3.6 \cdot 10^{489}$). Furthermore, discrete values of sizing variables are chosen so as to make additional constraints harder to satisfy thus increasing computational complexity of this test problem.

The frame is designed for minimum weight to carry the following loads: (i) horizontal forces P=80 kN acting in the positive X-direction at nodes 4, 7, 10, 13, 16, 19, 22, 25 and 28; (ii) vertical forces Q=30 kN acting in the negative Y-direction at all free nodes. A total of 243 nonlinear constraints are imposed on nodal displacements (limited to be less than 2.54 cm, i.e. 1/1063 of the total frame height), Von Mises equivalent stresses at nodes (stress limit is 172.43 MPa, i.e. 25,000 psi) and Euler buckling. Besides the 243 nonlinear constraints, there are 113 linear constraints on cross-section aspect ratio and frame layout. Hence, the design problem includes 356 constraints. The initial configuration of the structure is shown in Figure B3.

Similar to the dam problem, 30 independent runs were carried out starting from different initial populations. No special strategy was used for updating discrete design variables. Updated values of design variables stored in any trial design $\boldsymbol{X}_{\mathrm{TR}}$ that could not be directly included in the discrete available sets of the problem at hand were rounded to their nearest discrete values. However, this operation was performed always checking if rounding keeps the modified trial design $\boldsymbol{X}_{\mathrm{TR}}$' on a descent direction. When this did not occur, mirroring strategy was utilized to form a new design $\boldsymbol{X}_{\mathrm{TR}}$" and the best design between $\boldsymbol{X}_{\mathrm{TR}}$' and $\boldsymbol{X}_{\mathrm{TR}}$" was further developed in the optimization search. Mirroring strategy was reiterated until the new trial design improved population. Interestingly, such a goal was always accomplished within 5 recursive cycles.

The best solutions obtained in [53] for this test problem ranked as follows: 43700.260 kg for HFSA, 43700.814 kg for hybrid HS, 43700.836 kg for hybrid BBBC, 44674.120 kg for improved discrete JAYA [55], and 45319.334 kg for BBBC-UBS [173]. The number of structural analyses required by the optimization was as follows: 37748 for hybrid HS, 37964

for hybrid BBBC, 38521 for HFSA, 39487 for improved/parameterless discrete JAYA, and 42194 for BBBC-UBS.

In this study, LSSO–HBBBCJA was forced to complete the optimization process within only 35000 structural analyses, about 10% less than the computational cost quoted in [53] for the fastest optimizer. Thirty independent optimization runs were carried out starting from different populations randomly generated; each population included only 20 designs, that is between 1/25 and 1/10 of the population sizes reported in [53]. Remarkably, LSSO–HBBBCJA found an average optimized weight of 43695.636 m^3 with a standard deviation of 9.549 kg, thus improving the results reported in the literature. Constraint violations evaluated for all optimized designs obtained in the independent runs never exceed 0.0035%, regarding only geometric relationships between nodal coordinates or cross-section segments.

LSSO-HBBBCJA designed an uniform strength configuration for the frame by increasing length of horizontal elements from the top to the bottom of the structure. The optimized layout of the frame hence resembles a triangle with a vertical side subjected to concentrated loads. Heavier elements are placed near the loaded side and the clamped side of the frame. Buckling constraints are satisfied by reducing length of critical elements or locally reinforcing cross-sections. Such a design is fully consistent with Ref. [53].