Optimization approaches in distributed systems

Politecnico di Bari

LIBERATORIA PER L'ARCHIVIAZIONE DELLA TESI DI DOTTORATO

Al Magnifico Rettore
del Politecnico di Bari

Il/la sottoscritto VOLPE GAETANO nato a TERLIZZI (BA) il 10/05/1983

residente a BISCEGLIE (BT) in via G. BOVIO, 459 e-mail: gaetano.volpe@poliba.it

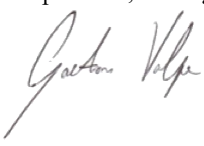
iscritto al 3° anno di Corso di Dottorato di Ricerca in INGEGNERIA ELETTRICA E DELL'INFORMAZIONE ciclo 36

ed essendo stato ammesso a sostenere l'esame finale con la prevista discussione della tesi dal titolo:

OPTIMIZATION APPROACHES IN DISTRIBUTED SYSTEMS

DICHIARA

1) di essere consapevole che, ai sensi del D.P.R. n. 445 del 28.12.2000, le dichiarazioni mendaci, la falsità negli atti e l'uso di atti falsi sono puniti ai sensi del Codice penale e delle Leggi speciali in materia, e che nel caso ricorressero dette ipotesi, decade fin dall'inizio e senza necessità di nessuna formalità dai benefici conseguenti al provvedimento emanato sulla base di tali dichiarazioni;
2) di essere iscritto al Corso di Dottorato di ricerca in INGEGNERIA ELETTRICA E DELL'INFORMAZIONE ciclo 36, corso attivato ai sensi del "*Regolamento dei Corsi di Dottorato di ricerca del Politecnico di Bari"*, emanato con D.R. n.286 del 01.07.2013;
3) di essere pienamente a conoscenza delle disposizioni contenute nel predetto Regolamento in merito alla procedura di deposito, pubblicazione e autoarchiviazione della tesi di dottorato nell'Archivio Istituzionale ad accesso aperto alla letteratura scientifica;
4) di essere consapevole che attraverso l'autoarchiviazione delle tesi nell'Archivio Istituzionale ad accesso aperto alla letteratura scientifica del Politecnico di Bari (IRIS-POLIBA), l'Ateneo archivierà e renderà consultabile in rete (nel rispetto della Policy di Ateneo di cui al D.R. 642 del 13.11.2015) il testo completo della tesi di dottorato, fatta salva la possibilità di sottoscrizione di apposite licenze per le relative condizioni di utilizzo (di cui al sito http://www.creativecommons.it/Licenze), e fatte salve, altresì, le eventuali esigenze di "embargo", legate a strette considerazioni sulla tutelabilità e sfruttamento industriale/commerciale dei contenuti della tesi, da rappresentarsi mediante compilazione e sottoscrizione del modulo in calce (Richiesta di embargo);
5) che la tesi da depositare in IRIS-POLIBA, in formato digitale (PDF/A) sarà del tutto identica a quelle **consegnate**/inviate/da inviarsi ai componenti della commissione per l'esame finale e a qualsiasi altra copia depositata presso gli Uffici del Politecnico di Bari in forma cartacea o digitale, ovvero a quella da discutere in sede di esame finale, a quella da depositare, a cura dell'Ateneo, presso le Biblioteche Nazionali Centrali di Roma e Firenze e presso tutti gli Uffici competenti per legge al momento del deposito stesso, e che di conseguenza va esclusa qualsiasi responsabilità del Politecnico di Bari per quanto riguarda eventuali errori, imprecisioni o omissioni nei contenuti della tesi;
6) che il contenuto e l'organizzazione della tesi è opera originale realizzata dal sottoscritto e non compromette in alcun modo i diritti di terzi, ivi compresi quelli relativi alla sicurezza dei dati personali; che pertanto il Politecnico di Bari ed i suoi funzionari sono in ogni caso esenti da responsabilità di qualsivoglia natura: civile, amministrativa e penale e saranno dal sottoscritto tenuti indenni da qualsiasi richiesta o rivendicazione da parte di terzi;
7) che il contenuto della tesi non infrange in alcun modo il diritto d'Autore né gli obblighi connessi alla salvaguardia di diritti morali od economici di altri autori o di altri aventi diritto, sia per testi, immagini, foto, tabelle, o altre parti di cui la tesi è composta.

BARI, 15 dicembre 2023 Firma 

Il/La sottoscritto, con l'autoarchiviazione della propria tesi di dottorato nell'Archivio Istituzionale ad accesso aperto del Politecnico di Bari (POLIBA-IRIS), pur mantenendo su di essa tutti i diritti d'autore, morali ed economici, ai sensi della normativa vigente (Legge 633/1941 e ss.mm.ii.),

CONCEDE

- al Politecnico di Bari il permesso di trasferire l'opera su qualsiasi supporto e di convertirla in qualsiasi formato al fine di una corretta conservazione nel tempo. Il Politecnico di Bari garantisce che non verrà effettuata alcuna modifica al contenuto e alla struttura dell'opera.
- al Politecnico di Bari la possibilità di riprodurre l'opera in più di una copia per fini di sicurezza, back-up e conservazione.

BARI, 15 dicembre 2023 Firma 

Politecnico di Bari

DEPARTMENT OF
ELECTRICAL AND INFORMATION ENGINEERING

PH.D. PROGRAM IN
ELECTRICAL AND INFORMATION ENGINEERING

SSD: ING-INF/04 - SYSTEMS AND CONTROL ENGINEERING

Final Dissertation

OPTIMIZATION APPROACHES IN DISTRIBUTED SYSTEMS

by
Gaetano Volpe

A thesis submitted for the degree of
Doctor of Philosophy

External Reviewers:
Prof. Giuseppe Franzè
Prof. Antonio Ferramosca

Supervisor:
Prof. Agostino M. Mangini

Ph.D. Program Coordinator:
Prof. Mario Carpentieri

Cycle XXXVI - November, 1st 2020 - October 31st 2023

To my beloved wife Francesca
and my beautiful kids
Giuseppe and Alessandro.

Acknowledgements

The last three years have been a life changing experience for me. What I have learned in undertaking this Ph.D. at the *Polytechnic University of Bari* is of inestimable value. Doing a research in any topic is not a trivial job and learning a good scientific methodology is the main key to achieve results.

My journey to become a good researcher is still long but the people I have encountered during my Ph.D. have helped me to do the first steps towards the right direction. To this, I wish first to thank my supervisor, Prof. Agostino Marcello Mangini. His suggestions and corrections in conducting my research and writing manuscripts, together with his incredible knowledge and skills as both researcher and professor, have led me to publish my papers and value my research.

At the same time, the mentorship of Prof. Maria Pia Fanti, head of the *Laboratory of Control and Automation*, has been unvaluable. Having been supported by a so well reputed and long experience professor in Automation and Control has been an honour for me. I have admired her continuous effort in research and her dedication to every member of the laboratory.

I wish to thank my beloved wife Francesca ad my two beautiful kids Giuseppe and Alessandro. Every single moment dedicated to research, especially overnight or in the weekend, is a moment not spent with them and finding a good balance between family and work is not always easy. To this, without the strong support and patience of my family, I would have not been ever able to achieve my goals.

I wish to thank also my parents and my brother Francesco. They always pushed me to complete my studies and supported me in hard times.

Last but not least, I'm always grateful to my two additional brothers: Roberto De Gennaro and Nino Barile. Since almost twenty years, their suggestions and the continuous exchange of opinions are enriching me and pushing to improve more and more.

Contents

Preface

This thesis is submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy in Electrical and Information Engineering at the *Polytechnic University of Bari*. The research presented in this dissertation was conducted at the *Laboratory of Control and Automation* of the *Polytechnic University of Bari* under the supervision of Professors Agostino Marcello Mangini and Maria Pia Fanti, between November 2020 and October 2023. Most parts of this thesis have been published in International Journals and Conference Proceedings over the last three years.

My main research activity, described in Part I, has been focused on the implementation of the optimal task assignment problem in manufacturing environments, in distributed and decentralized systems and, in particular, in Blockchain contexts. To this purpose, I have designed two decentralized collaborative platforms to deliver and orchestrate complex software tasks, equipped with machine learning based runtime prediction algorithms. The original results of this work, including a concrete case study conducted at *DAI Optical Industries*, an ophthalmic lenses production company based in Molfetta (Bari), Italy, are reported in Chapters 3 and 4 and have been published in [1], [2] and [3], whose I am the first author.

During my Blockchain studies, I have also been involved in the development of an incentive platform for energy management, described in Chapter 5 and published in [4], and in the design of an original modular architecture for Quantum-Safe Blockchain [5].

In the second-half of my research, described in Parts II and III, I have explored the use of Timed Coloured Petri Nets (TCPN) to simulate mass production systems and I have designed a Swarm-based algorithm to solve the well known Flexible Job Shop Scheduling Problem (FJSSP). The results of this work are reported in Chapter 6 and have been published in [6].

Moreover, thanks to my involvement in the *IN2CCAM project*, funded by the EU Horizon 2020 programme, I had the opportunity to approach the autonomous driving domain and, in particular, the problem of autonomous intersection management at unsignalized intersections. The results of this

still on-going work are partially reported in Chapter 7 and have been published in [7] and [8].

The findings of this research activity are summarized in Chapter 8 together with an outlook on future developments and research directions. The full list of publications written by the author is reported hereafter.

List of Publications

International Journals

[1] Gaetano Volpe, Agostino Marcello Mangini, and Maria Pia Fanti. An architecture combining blockchain, docker and cloud storage for improving digital processes in cloud manufacturing. *IEEE Access*, 10:79141–79151, 2022. doi: 10.1109/ACCESS.2022.3194264.

[2] Gaetano Volpe, Agostino Marcello Mangini, and Maria Pia Fanti. A deep reinforcement learning approach for competitive task assignment in enterprise blockchain. *IEEE Access*, 11:48236–48247, 2023. doi: 10.1109/ACCESS.2023.3276859.

International Conferences

[1] Gaetano Volpe, Agostino Marcello Mangini, and Maria Pia Fanti. An architecture for digital processes in manufacturing with blockchain, docker and cloud storage. In *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pages 39–44, 2021. doi: 10.1109/CASE49439.2021.9551633.

[2] Gaetano Volpe, Agostino Marcello Mangini, and Maria Pia Fanti. Job shop sequencing in manufacturing plants by timed coloured petri nets and particle swarm optimization. *IFAC-PapersOnLine*, 55(28):350–355, 2022. ISSN 2405-8963. doi: https://doi.org/10.1016/j.ifacol.2022.10.365. URL `https://www.sciencedirect.com/science/article/pii/S2405896322024028`. 16th IFAC Workshop on Discrete Event Systems WODES 2022.

[3] Giuseppe Olivieri, Gaetano Volpe, Agostino Marcello Mangini, and Maria Pia Fanti. A district energy management approach based on internet of things and blockchain. In *2022 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl*

Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), pages 1–6, 2022. doi: 10.1109/DASC/PiCom/CBDCom/Cy55231.2022.9927910.

[4] Gaetano Volpe, Agostino Marcello Mangini, and Maria Pia Fanti. A cooperative drl approach for autonomous traffic prioritization in mixed vehicles scenarios. In *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*, pages 1–6, 2023. doi: 10.1109/CASE56687.2023.10260615.

[5] Marco Fiore, Federico Carrozzino, Marina Mongiello, Gaetano Volpe, and Agostino Marcello Mangini. A blockchain-based modular architecture for managing multiple and quantum-safe encryption algorithms. In *2023 9th International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 598–601, 2023. doi: 10.1109/CoDIT58514.2023.10284090.

[6] Francesco Paparella, Gaetano Volpe, Agostino Marcello Mangini, and Maria Pia Fanti. Collision avoidance strategy for autonomous intersection management by a central optimizer algorithm. In *2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2023 (to appear).

1 Introduction

The Industry 4.0 paradigm has paved the way for the development of efficient computational and storage infrastructures in manufacturing plants. In fact, the ability of perform multiple complex tasks in short time in a production flow and provide efficient communications between machines and servers is crucial to improve throughput, quality and provide constant monitoring to prevent faults and ensure business continuity and resilience [9].

In this context, the contribution of *Cloud Computing* is essential. In *Cloud Computer* paradigm, computational and storage resources are located in multiple geographical positions and are consumed by the users in a variety of business models, such as *Software-as-a-Service* (SaaS), *Platform-as-a-Service* (PaaS) and *Infrastructure-as-a-Service* (IaaS) [10]. In SaaS, the user simply utilizes and manage a one or more applications, while the provider takes care of bug fixes and maintenance of the underlying infrastructure. On the contrary, in PaaS and IaaS, the user is involved in managing a part of the infrastructure such as administering virtualization and storage resources (IaaS) or deploy code to pre-set environments (PaaS).

Cloud Computing suppliers often provide their services from their data centres in a distributed fashion. A distributed system is simply any environment where multiple computational units are working on a variety of tasks and components, all spread across a network [11]. Components within distributed systems split up the work, coordinating efforts to complete a given job more efficiently than if only a single device ran it. Although the computational effort is split across multiple nodes, in distributed systems there is often a centralized leader that acts as a coordinator and takes the decisions. On the contrary, in decentralized systems, a subset of distributed systems, there is no single point where the decision is made. Every node makes a decision for its own behaviour and the resulting system behaviour is the aggregate response.

A variety of well known optimization problems can be performed in distributed and decentralized systems using classical and *Artificial Intelligence*

(AI) based algorithms. Classical approaches pertain mostly on linear and non-linear minimization problems whereas, in AI, both machine-learning approaches, such as *Artificial Neural Networks* (ANN) and *Deep Reinforcement Learning* (DRL), and swarm intelligence algorithms such as *Particle Swarm Optimization* (PSO) are widely employed.

The main research direction of this thesis is to investigate the implementation of optimization algorithms in distributed systems in a variety of fields. In Part I, the applications of such algorithms in different Blockchain-based environments is presented. A Blockchain is a special case of a decentralized system whose main purpose is to record transactions, such as in a ledger, and allow their verification by all nodes joining a peer-to-peer network, without the need of a central authority [12]. Transactions are grouped in blocks and all blocks are connected each other in a tamper-proof chain. Moreover, each node holds a copy of the ledger and all nodes must reach a sort of agreement to add a new block to the chain.

The first investigated problem addressed in Chapters 3 and 4 is the optimal task assignment in Manufacturing as part of a production workflow. Some examples of such tasks are optimization problems, data mining algorithms, heavy file format conversion (e.g. video and audio file) and 3D rendering. In more detail, in Chapter 3 a collaborative decentralized *Ethereum*-based platform to deliver such complex software tasks is proposed. In the introduced scheme, software tasks are delivered as *Docker* images, requested by manufacturers, orchestrated by a *Smart Contract* and, finally, assigned to *Docker*-based process runners. The task assignment is performed by a ANN that predicts the runtime of each requested task for each agent and selects the one with the shortest predicted execution time. The main advantage of the proposed solution is the combination of Blockchain with other cloud-based technologies and *Deep Learning* techniques to standardize process delivery in a secure, scalable and tamper-proof context. Preliminaries on Blockchain, *Docker* and *Smart Contracts* are provided in Chapter 2.

In Chapter 4, the same concept is deeply extended and the platform is implemented in *HyperLedger Fabric*, an enterprise oriented Blockchain, as a optimal auction and bidding scheme. While the task assignment procedure is still driven by a *Smart Contract*, the main advancement compared to the previous work is that the performance prediction algorithm is online rather than offline. In detail, the proposed algorithm is based on DRL and allows agents to incrementally learn how to predict runtime of software tasks in their current load state as they collect new experiences. The algorithm is able to learn theoretically every kind of task. Moreover, compared to previous approaches in literature, it does not assume any preliminary information in advance.

To conclude Part I, an incentive platform for energy management based on Blockchain is presented in Chapter 5. The proposed Ethereum-based platform stores real-time consumption data of users in a district and performs user clustering and rating by a k-means algorithm. Then, based on each user's assigned class, a penalty-reward scheme based on a minimization problem is applied to calculate users invoices amount. The entire process is performed on the Blockchain in a secure and tamper-proof environment.

In Part II, the use of Swarm algorithms in Manufacturing is investigated. More in detail, in Chapter 6, a framework based on Timed Coloured Petri Nets (TCPN) to model mass production systems is introduced. In this context, the problem of job scheduling optimization, that is known in literature as Flexible Job Shop Scheduling Problem (FJSSP), is solved by the implementation of a Particle Swarm Optimization (PSO) algorithm. The contribution of this work is the complementary use of TCPN, for system simulation, and PSO to solve the NP-hard FJSSP problem, maximize the throughput and minimize machine waiting time. Due to the intrinsic nature of PSO, the proposed algorithm can be implemented in a parallel fashion in a distributed environment, for example in a GPU-based environment, to improve efficiency and speed up convergence time.

Part III deals with autonomous vehicles and the problem of autonomous intersection management. More specifically, in Chapter 7, a novel Multi-agent Cooperative DRL approach for autonomous intersection management at unsignalized intersections under prioritized mixed traffic scenarios is proposed. The proposed scheme includes three different vehicles categories: Connected Automated Vehicles, i.e. CAVs, that act as agents in the DRL algorithm implementation, Connected Priority Vehicles (CPVs), such as ambulances or police cars, that are connected but driven by humans and Regular Vehicles (RVs), that are neither connected or automated. Each agent updates its state at each time step and receives a global reward. The main goals of the system are to optimize traffic flow, ensure emergency vehicles priority, prevent collisions and reduce crossing times. The main contribution of this work are the novel state representation, that is only partially observable by the agents due to the presence of unconnected vehicles, the structure of the global reward function and the approach based on Proximal Policy Optimization (PPO) to determine the best policy. Also in this case, a parallel DRL training of the agents can be performed in a distributed system context.

Summing up, in this thesis, Deep Reinforcement Learning as optimization and control technique, shows great flexibility in different domains as well as the possibility to be easily implemented in a distributed infrastructure such as a Blockchain-based system. On the contrary, it is shown that

the use of traditional Artificial Neural Networks is limited by their offline fashion that normally requires a full dataset to perform the training. In addition, Swarm Intelligence Algorithms, such as PSO, represent a good alternative for NP-hard optimization problems in some cases and, compared to traditional iterative methods, can be trained in a parallel fashion in a distributed system.

Chapter 8 concludes the thesis and gives an outlook for future research directions.

Part I

Optimal Algorithms and Blockchain Applications in Manufacturing and Power Systems

2 Blockchain, Docker & DRL Preliminaries

2.1 Blockchain and Smart Contracts

Blockchain is essentially a ledger whose architecture is distributed rather than centralized. Its main purpose is to record transactions, signed at least by the issuer, and allow their verification by all nodes joining the peer-to-peer network, without the need of a central authority [12].

As the name itself suggests, the ledger is made by a chain of blocks, in which each block stores a certain number of transactions. The chain is guaranteed to be *tamper-proof* as each block is connected to the previous one by including its Secure Hash Algorithm (SHA) *SHA-256* in the header [13].

As the order in which the blocks are connected is relevant for the transactions history, all nodes must reach a sort of agreement whenever a new block has to be added to the chain. For this purpose, many different *consensus* mechanisms have been studied in recent years [14].

In a Blockchain, there is also the possibility to enforce a specific agreement between two or more parties during a transaction. In this case, one or more *Smart Contracts* are employed [15]. A Smart Contract is a small algorithm coded in the Blockchain in languages such as *Solidity* [16] and *Vyper* [17] that is executed when specific trggering conditions are met.

Moreover, Blockchains can be public or private. Public blockchains are also called *permissionless*, in the sense that an authorization is not required to join the network as a node or as a user. On the one hand, these blockchains have a very high resilience and trust rate because they are maintained by a great number of stakeholders in the world. However, due to the high level of decentralization and the high computing power required to reach consensus, they have some limits in scalability and performance [18]. Conversely, private blockchains require an authorization to join the network. Although the level of decentralization is reduced, the intrinsic

trust between the nodes in this case allows to avoid mining in reaching consensus and, thus, prevents scalability issues.

2.2 Ethereum & Hyperledger Fabric

As of today, many different software solutions exist to implement a Blockchain. In this regard, Ethereum platform has represented the first real paradigm change as it introduced the concept of Smart Contract, allowing applications to be executed in the Blockchain [19]. In regard to the Ethereum platform, Smart Contracts are executed in the so-called *Ethereum Virtual Machine (EVM)*, which is one its core components.

Hyperledger Fabric is an enterprise-grade Blockchain platform which has great advantages, compared to other platforms such as Ethereum, in terms of transactional privacy, flexibility and data-query capabilities [19]. Since it is mainly designed for permissioned networks, a new node must be authorized to join the network by another node with appropriate permissions. In the same way, a user can be granted access to all of part of the data according to its role. Moreover, in order to ease deployment, it leverages on a series of single micro-services based on Docker [20] containers. The list of services includes *peer*, *Certification Authority (CA)* and *orderer*, each with a specific role. A CouchDB [21] no-sql database is often used to deploy the ledger, while *Smart Contracts* are called *Chaincodes*. Some of the most popular languages are supported to develop chaincodes including Java, NodeJs and GoLang. Finally, on consensus layer, Hyperledger Fabric supports practical leader-based algorithms, such as RAFT [22], in which a recognized leader node publishes the blocks while all other nodes verify and validate transactions.

2.3 Dockers, Containers and Cloud Storage

Docker is a client-server platform on which multiple applications can be run in isolated environments called *containers* on the same host [23]. Containers are different from regular virtual machines in the sense that they do not need an hypervisor, but they leverage the kernel of the host operating system. However, a full life-cycle management is provided within the platform, including actions such as create, start, stop and destroy and features such as collecting metrics.

Each container is run on the top of a Docker *image*, which is a template built according to a list of instructions included in a *Dockerfile*. Each of those instructions is a single *layer* that is added to the image filesystem. For each image a unique fingerprint can be indicated by calculating a single *SHA-256 hash* that considers all of its layers.

Docker images are typically stored in a *registry*, which is essentially a repository that allows users to pull or push images according to their permissions and to store multiple versions of a single image identified by different *tags*.

Cloud Storage is one of the most popular cloud technologies and it is offered by some of the major providers, such as *Amazon* and Microsoft by the *S3*[1] service and the *Azure Blob Storage*[2] platform respectively. It is essentially a data storage service [24] that enables users to store their files in the Cloud.

[1] `https://aws.amazon.com/s3/`

[2] `https://azure.microsoft.com/en-us/services/storage/blobs/`

2.4 Deep Reinforcement Learning

DRL is a technique that combines traditional RL and deep learning. One of the first approaches of DRL was DQN for Atari games [25], in which a deep neural network (DNN) was used as the function approximator in the place of traditional Q-learning [26]. A more recent application is Deep Deterministic Policy Gradient (DDPG) [27], which is designed for scenarios with exponentially large continuous action space.

Two specific applications of DRL for optimal task assignment in Blockchain and Autonomous Intersection Management for Autonomous Vehicles are extensively described in Chapters 4 and 7 respectively.

3 Design of a Tasks Orchestration Platform in Ethereum with ANN

3.1 Introduction

In this chapter, the first of two works dedicated to the problem of optimal software tasks assignment in Blockchain environments in Manufacturing is presented. In the proposed approach, the Blockchain is the base platform for tasks orchestration while the optimal assignment of tasks is supported by the prediction of task runtime performed by an Artificial Neural Network (ANN). The ANN is trained using a dataset of past tasks executions in different time frames and various load conditions.

Cloud Computing significantly changed the way industries do their business today [10]. In 2020, enterprise spending on cloud infrastructure services amounted to almost 130 billion U.S. dollars, while in 2011 it was only 3.5 [28]. In manufacturing industry, the adoption of cloud technologies lead to the new paradigm of *Cloud Manufacturing* [29], which enables companies to provide and receive services over Internet via an intelligent service-oriented architecture.

Decentralization is a key factor both in Cloud Computing and Cloud Manufacturing as distributed architectures improve services reliability, scalability and performance when compared to traditional on-premise centralized approaches [29]. Similarly, security is also a key concern in these contexts as data elaborated in the cloud may be compromised, altered or stolen and the integrity of services could be severely affected [30].

Blockchain, as a secure ledger for storing transactions, plays a strategic role in modern distributed architectures and the recent literature has shown a great interest about this revolutionary technology over the last decade. A

systematic review provided in [12], shows that Blockchain based application have been adopted in many business domains: from popular financial applications, such as cryptocurrencies, to completely different fields, such as *Business and Industry*, *IoT*, *Governance*, *Education* , *Health*, and *Privacy & Security*.

Smart Contracts are typically the core component on which most of these applications are based. A comprehensive summary of smart contracts and their most popular applications is provided in [31].

Among all business domains, a specific attention in recent literature has been given to the application of Blockchain in *Business & Industry* [12]. In detail, as described in the taxonomy provided in [32], a well structured class of Industry 4.0 applications are already implemented. Moreover, the authors describe the adoption of Smart Contracts in fields such as *Internet-Of-Things*, *Supply Chain*, *Manufacturing* and *Energy*. However, in [33], the challenges and the current status of Blockchain in manufacturing are surveyed and the relative research paths for the near future are suggested by the authors.

Some specific solutions for *Engineering and Manufacturing* are detailed in [34]. The reviewed applications are designed for specific purposes such as increasing the efficiency of the manufacturing process, protecting data validity and enhancing communications inside and outside the organization.

The Blockchain, as a tool that could be employed by all the management stakeholders of a product life-cycle is argued in [9] by describing different purposes, such as making deals and sharing information about products. In their work, the authors review the state-of-the-art literature in the field from both the manufacturing system perspective and product life-cycle management perspective. In general, when sustainable manufacturing in Industry 4.0 is concerned, the Blockchain is considered as a concrete enabler for current Enterprise Resource Planning and Manufacturing Execution Systems solutions.

As an example of such application, in [35], a decentralized task execution model for Industrial-Internet-of-Things (IIoT) environments named *ManuChain* is suggested. The proposed architecture has a dual layer architecture and it is based on a permissioned Blockchain and specifically designed Smart Contracts. More recently, in [36], the authors introduce a Blockchain based security service architecture for Cloud Manufacturing environments, that facilitates authentication and privacy protection in decentralized 5G IIoT contexts.

Collaborative *Business Process Management* (BPM) in Blockchain domain is discussed in [37]. In their systematic review, the authors show that, among others, papers about *process implementation & execution* and

process modeling phases are the most representative.

In this context, a common research topic is the translation in Smart Contracts of processes defined according to the well known Business Process Model and Notation (BPMN) [38]. In [39], an open source BPM system named *Caterpillar*, is suggested. The solution includes a BPMN to Smart Contract compiler to perform the translation in automatic fashion.

Concerning the Cloud Manufacturing, the existing Blockchain based solutions already satisfy some of the most critical requirements such as decentralization and security. Furthermore, compared with other distributed architectures, the Blockchain is known to have a greater performance when those features are concerned. In addition, in several business domains, a Blockchain peculiar feature such as the immutability of transactions clearly enhances their credibility while, at the same time, the unnecessity of trusted third parties makes easier to verify contents and improves the reliability of a given platform [12].

Moreover, in [40], the advantages of using Blockchain in such environments and, specifically, for Additive Manufacturing and 3D printing is noticeable. In that respect, the authors observe that preserving the intellectual property to secure CAD models is a critical issue in such context and that the Blockchain technology can be efficiently implemented to license the models and secure the whole manufacturing process. In addition, in [41], a concrete example of a Blockchain and Smart Contract based architecture for Additive Manufacturing is proposed with the specific purpose of enforcing agreements between design and 3D printing companies and preserve the intellectual property of design files.

However, it is often required that modern platforms satisfy other important requirements in the context of collaboration. In particular, they need to be standardized and flexible in order to be easily implemented. As suggested in [29] and [10], most of the existing solutions about the development of virtualization and servitization technology are still weak and, consequently, fail to fit those requirements.

This research focuses on particular manufacturing processes that produce pieces in small lots or in single unit, such as ophthalmic lenses or 3D printed products. Such types of manufacturing procedures require specific digital processes, computational intensive tasks and raise standardization, cybersecurity and intellectual property preservation issues.

Cloud solutions and standardized frameworks can enable different actors to effectively provide, update and consume any kind of process delivered as a service. The absence of such standard framework in current related literature makes it hard to flexibly integrate a generic software logic in such environments and impacts on servitization process.

Motivated by this open issue, this work proposes a decentralized, distributed, cybersecure and collaborative platform to improve digital processes in Cloud Manufacturing environments. The proposed platform is based on Ethereum [42], which is one of the most popular Blockchain implementations, and Smart Contracts. In addition, different actors such as *Process Owner*, *Process Consumer* and *Process Runner* cooperate in the platform to handle process life cycle and effectively execute digital business processes.

Differently from previous solutions, such as [41], that consider only Blockchain as enabling technology, the main contribution of the research is using Docker, Cloud Storage and Deep Learning in the Manufacturing Sector, integrated with Blockchain, in order to solve the problem of standardizing the processes, ensuring flexibility and security, preserving intellectual property, and enabling scalability. Such requirements are addressed by the following technologies in an innovative approach.

First, we integrate in the proposed Blockchain platform two popular cloud technologies: *Docker* [23] and *Cloud Storage* [24]. Docker is a container-based platform to run applications in isolated environments from lightweight and self-contained images and it has great advantages in terms of performance, scalability, and portability over traditional virtual machines [20]. While Docker in a typical Blockchain architecture is only employed to run the components of the Blockchain itself, in the proposed platform, the process logic is delivered to the runners right in terms of a Docker image, which includes both the application and its dependencies. To the best of our knowledge, it is the first time that Docker is used in Cloud Manufacturing and in a Blockchain environment in this specific role to provide a flexible and distributed computational environment.

Second, each Docker image is identified by a unique *digest* that is stored and updated in the chain by the *Process Owner* through a Smart Contract. Since the Blockchain is tamper-proof in nature, by storing the *digest* in the chain, consumers and runners are always guaranteed to use the latest version of the process as deployed by the *Process Owner*, thus preserving the integrity of the execution. Moreover, in order to keep the Blockchain not too heavy, we rely on traditional Cloud Storage platforms to store and retrieve instances related input and output files.

Third, since in Cloud Manufacturing task scheduling plays a fundamental role [43] and the allocation of physical resources such CPU and memory in a shared Cloud Computing environment is generally variable, we introduce a task assignment problem with a *deep learning* approach to predict process run-time that is based on past Docker containers metrics. In this way, given an instance request in a specified time band, we are able to

choose the fastest runner.

The proposed blockchain platform including Docker and Cloud storage is implemented in a real case study of ophthalmic lenses manufacturing. Thanks to the flexibility of Docker technology, we show that the proposed platform combines the great advantages of existing Blockchain based solutions in Cloud Manufacturing in terms of reliability, performance, and cybersecurity with a completely different service deployment approach. In fact, in this environment any process that can be packed in a Docker image can be easily and safely implemented. Hence, a more effective collaboration between owners, runners and consumers can be achieved with no integration overhead for new processes.

The results of this research have been published firstly in [3] in their embryonic form and, successively, in [1], in which we provide a detailed description of the platform architecture and large implementation details about the case study regarding ophthalmic lenses manufacturing. In particular, the task assignment problem and the related deep learning approach are described by focusing on the lens surface calculation step. Moreover, we discuss the specifications of the process implementation on the proposed platform highlighting the related benefits in terms of cybersecurity, reliability and performance compared to the traditional solutions.

In Chapter 2, some basic concepts related to Blockchain, Smart Contracts, Docker, Cloud Storage have been introduced. The remainder of this chapter is organized as follows: In Section 3.2, the concept of Digital Process and *Time Predicition Algorithm* is explained. In Section 3.3, we introduce the platform architecture describing in detail its principal components, use cases, smart contracts and client applications involved. In section 3.4, we present a real case study regarding ophthalmic lenses manufacturing. Finally, section 3.5 concludes the chapter.

3.2 Digital Processes and Time Prediction Algorithm

In this work, a digital process is defined as a software task that is typically made of a computationally intensive algorithm. The task requires one or more input files and produces a certain number of output files to be used in the following steps of a manufacturing process.

In addition, we refer to a *Time Prediction Algorithm* as a function to predict the amount of time needed to complete tasks on a specified environment that we call *Process Runner*. For this purpose, a deep learning approach based on an *Artificial Neural Network* (ANN) is introduced. The proposed ANN is trained by considering the past metrics collected from process runners after each task execution.

3.3 The Proposed Platform Structure

In this section, we introduce a novel practical approach for implementing BPM in the Blockchain by leveraging on the features of two of the most popular cloud technologies: *Docker* and *Cloud Storage*. This platform is specifically designed for manufacturing environments, nevertheless other implementations are viable.

Based on Dumas BPM life-cycle [44], we suggest the use of Docker in *process execution* phase, supposed that the process to be executed is a digital process. Moreover, we store process inputs and outputs in a traditional cloud storage and we use Blockchain and Smart Contracts for *process implementation and monitoring*. We also introduce a basic task assignment problem based on execution time prediction performed through an ANN trained with past process runs metrics.

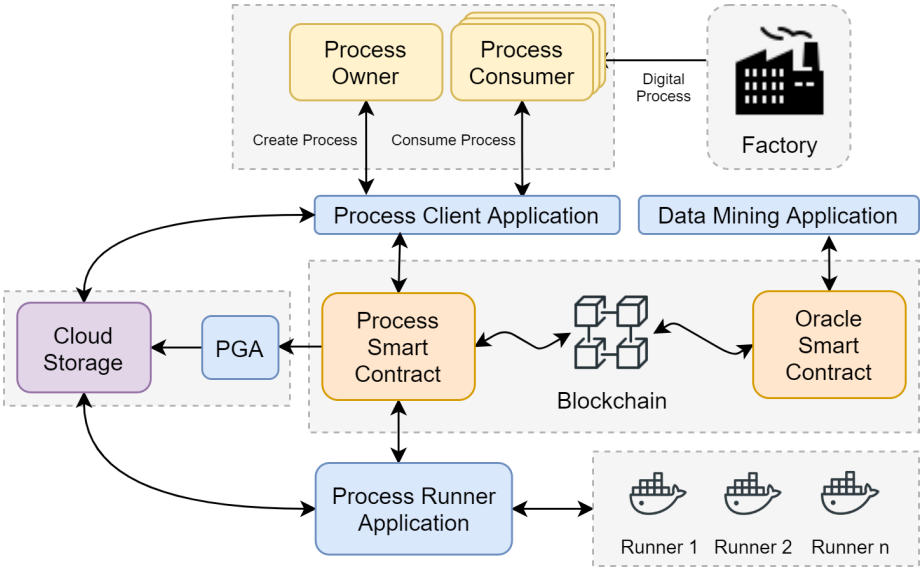


Figure 3.1: Platform Architecture & Use Cases

The proposed platform architecture is described in Fig. 3.1. The core of the platform, in the central part of the scheme, is a *Consortium Blockchain* built on *Ethereum*. It implements two main use cases: *Create Process* and *Consume Process*, that represent the process life-cycle which is orchestrated by the *Process Smart Contract*.

Given the consortium nature of the Blockchain, we decided to use *Proof-of-Authority* consensus algorithm [45]. More in detail, *Create Process* is related to the creation of a new process and *Consume Process* describes a single instance life-cycle.

In addition, four client applications are also provided for the purpose of interacting with the Blockchain: the *Process Client Application* that implements the main use cases, the *Process Runner Application* that interacts with the Docker based process runners, the *Permission Granting Application* that listens for events on the *Process Smart Contract* and orchestrates the permissions on the Cloud Storage and, finally, the *Data Mining Application* that collects metrics about past instances and updates the ANN weights on a dedicated *Oracle Smart Contract*.

Components and Use Cases

In this subsection, the main components of the platform are described in detail. We identify five main entities in the platform related to the process life-cycle:

1. **Process Owner**: It represents the entity that created the process core logic and coded the related algorithm. An owner usually delivers a process to the platform that can be requested by multiple consumers.

   This entity is involved in the following use case:

   > **Create Process**: It is the action of creating the process in the chain. First, the *Process Owner* packs the algorithm files in a Docker image along with all dependencies, then it sends the image to a Docker registry and finally executes the *CreateProcess* function.

2. **Process Consumer**: It represents the entity that needs to perform an instance of the digital process for its business purposes.

3. **Process Runner**: It provides the computational resources to execute the process instance by running the related image in a container.

4. **Permissions Granting Application** (PGA): this component assigns and revokes permissions on files in the Cloud Storage.

   These last three components are involved in the following use case:

   > **Consume Process**: First, the *Process Consumer* requests a new instance of the process by invoking the *ProcessInstance* function, then a *Process Runner* is assigned by the Smart Contract and the instance is finally executed. Both the *Process Consumer* and the *Process Runner* need to interact with the Cloud Storage for storing and retrieving instance inputs and outputs, therefore

the PGA plays its role by assigning and revoking the related permissions.

5. **Data Mining Algorithm**: It is the external component that collects past instances metrics from the chain and trains the execution time prediction ANN for each registered runner and for each known process on a regular time basis. At the end of each training, it pushes updated weights for each runner to the *Oracle Smart Contract* in the Blockchain.

Smart Contracts

The whole process life-cycle relies on two Smart Contracts developed in Solidity programming language:

Process Smart Contract

It stores data of processes, runners and instances. In addition, it provides *CreateProcess* and *ProcessInstance* functions that, in the proposed platform, represent the two main transactions of the Blockchain. As shown in Listing 3.1, the Process model includes the relevant Docker image name (optionally with its tag) and the docker SHA-256 *digest* that ensures the integrity of the execution, effectively encoding the process logic in the Smart Contract and in the Blockchain. Consequently, when the Process Owner invokes the *CreateProcess* function in the Smart Contract, it creates a new unique process identified by the related Docker image *digest*. The *Process Runner* is therefore allowed to use only that image when it is assigned a new instance. Should it retain a different version in its local repository, it must preliminary pull the required image from the remote registry before executing the instance.

```
pragma solidity 0.8.1;

contract Process {
    enum ProcessInstanceState
    {Placed, Assigned, Completed, Refused}
    uint public totalProcesses;
    uint public totalProcessInstances;
    Process[] public processes;
    ProcessInstance[]
    public processInstances;

    struct Account {
```

```
        int256 balance;
        bool accountType;
    }
    struct Process {
        string dockerImage;
        // Docker SHA-256 Hash
        bytes dockerId;
        uint clickFee;
        address owner;
        ProcessRunner[] runners;
    }
    struct ProcessRunner {
        address runner
    }
    struct ProcessInstance {
        uint processId;
        address runner;
        address consumer;
        uint executionTime;
        uint executionFee;
        uint totalFee;
        bytes inputId;
        bytes outputId;
        ProcessInstanceState state;
    }
    function createProcess(...)
    function processInstance(...)
    ...
```

Listing 3.1: Solidity code snippet for Process Smart Contract

The *ProcessInstance* function is depicted in the *Unified Modeling Language* (UML) [46] activity diagram in Fig. 3.2. When the consumer requests a new process instance, it stores the inputs in its area on the Cloud Storage. At the same time, the *ProcessInstance* function selects the fastest *Process Runner* by predicting the execution time for each registered available runner based on the weights stored in the *Oracle Smart Contract*. Since each runner is set for events listening on the Smart Contract, it becomes aware of the assignment and decides to accept or refuse the task by executing an auxiliary function. If it accepts, it is immediately granted the needed permissions on the cloud Storage by the PGA to retrieve the instance inputs. Once the inputs are available, the process instance is executed in a Docker container with the requested image. At the end of the execution, the Smart Contract is notified and the outputs are stored on the Cloud Storage to

be finally retrieved by the consumer. On the contrary, if the first assigned runner refuses, the Smart Contract is notified and next fastest runners are attempted until one of them accepts or there are no more available runners. In the latter case, the instance ends and the consumer is notified so that it can try again at a later time.

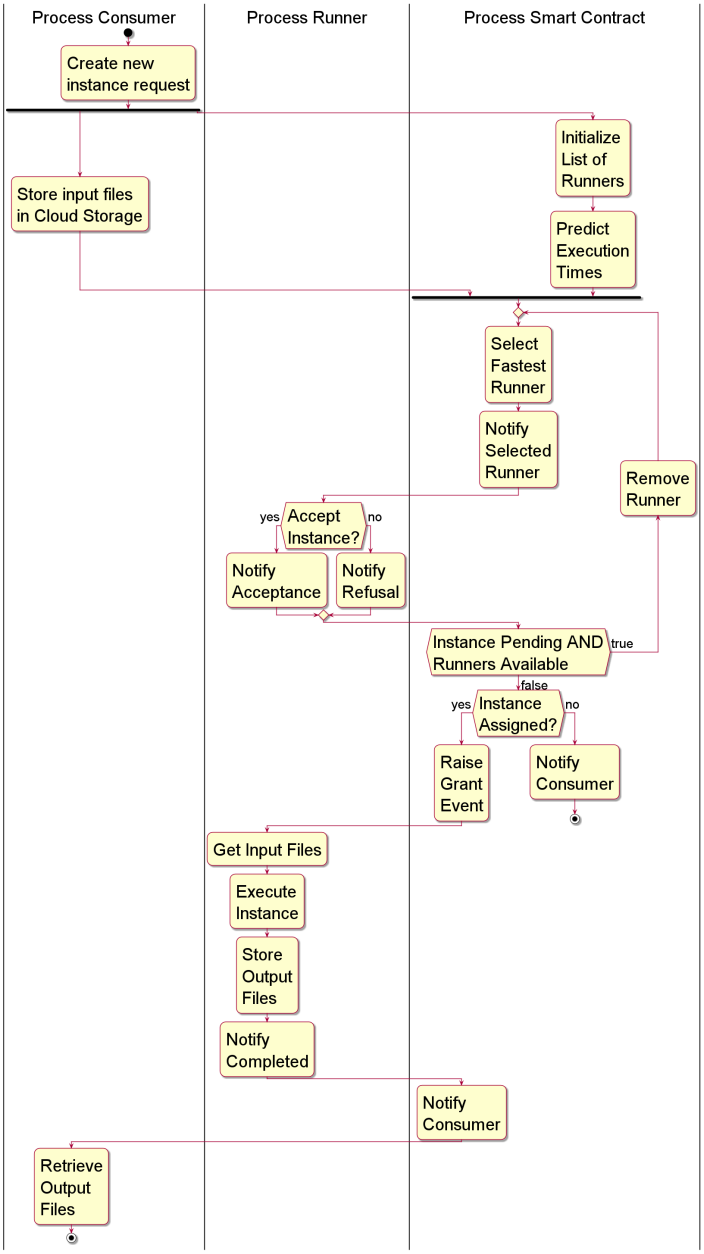


Figure 3.2: ProcessInstance Function UML Activity Diagram

Oracle Smart Contract

The *Oracle Smart Contract* is used to store the calculated weights of each ANN trained on past instance metrics. The weights are updated on a regular time basis. The whole process is implemented in the *Data Mining Application* described in Section 3.3.

Client Applications

The interaction of the components with the Blockchain and the Smart Contracts is carried out by using client applications that invoke functions in the Smart Contract and listen to the chain events.

In this work, we introduce four applications:

1. **Process Client Application**: it implements *Create Process* and *Consume Process* use cases and interacts with the Cloud Storage to store instance inputs and retrieve instance outputs. It could be also connected with third-party applications to exchange files and data.

2. **Process Runner Application**: it implements *Consume Process* use case and interacts with the Docker host to pull requested images from the registry and run the instance container for process execution. It also interacts with the Cloud Storage to retrieve instance inputs and store instance outputs.

3. **Permission Granting Application - PGA**: it listens for chain events and assigns and revokes permission on instance inputs and outputs for consumers and runners.

4. **Data Mining Application**: it collects past instances metrics by querying the *Process Smart Contract* on a regular time basis. Each observation in the training dataset includes the day of the week and the time band (hour of day) as input, while the actual running time in seconds is collected as output. Those features are depicted in Table 3.1. After building the dataset, a single multilayer perceptron ANN (MLP) for each runner and for each process is trained by using standard backpropagation. MLP is a class of feedforward artificial neural network composed of multiple layers of perceptrons with a threshold activation and it is widely used for regression problems where, given a set of inputs, a real-valued quantity is to be predicted. After training, the calculated weights are pushed to the *Oracle Smart Contract* to be used for future instances running time prediction. In this way, although a machine learning approach is adopted, the task assignment

process is deterministic and can be easily integrated in a Blockchain based architecture.

Table 3.1: Time Prediction ANN Dataset Details

Feature	Domain	Description
dayOfWeek	$1 \leq x \geq 7, x \in \mathbb{Z}$	Day of the week, Sun-Sat
hourOfDay	$0 \leq x \geq 23, x \in \mathbb{Z}$	Hour in which instance has been requested
runningTime	$\mathbb{R}_{>0}$	Actual execution time (Output)

3.4 Case Study: Ophthalmic Lenses Manufacturing

In this section, we present an implementation of the proposed architecture in ophthalmic lenses manufacturing. Indeed, the manufacture of ophthalmic lenses is a typical production of single, specific pieces, each one made on the basis of a unique medical prescription. Moreover, the lenses production requires high amount of data and complex optimization. On the other hand, cybersecurity issues must be considered to secure the ownership of the lenses design and implementation software. For this reason, the selected case study reflects the requirements to apply the proposed technology and allows us to enlighten the benefits of the Blockchain architecture.

The process of lens manufacturing is typically made of five steps in sequence: *Calculation*, in which the lens surface to be machined is computed, *Surfacing*, *Polishing*, *Coating* and, finally, *Edging*.

In this work, we focus on *Calculation*, which is a pure digital process. A typical lens calculation software is called *Lens Design System* (LDS).

Lens Design System

A LDS takes the patient's ophthalmic prescription as input, which is typically called a *job*, and performs a mathematical calculation that produces the data to be sent to a CNC machine to realize a surface on the back of a semi-finished lens. We depicted the process using BPMN notation [38] as shown in Fig. 3.3.

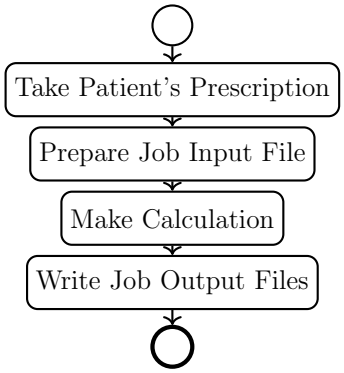


Figure 3.3: Lens Design System Process

In the eye care industry the exchange of data between machines and systems is regulated by a standard published by the *Vision Council of*

America, an US based organization that includes almost all the relevant players. The name of this standard is *Data Communication Standard* and the current version is 3.12 [47].

Listing 3.2 shows a typical job input file including prescription data along with other relevant manufacturing parameters.

```
JOB=1111
DO=B
REQ=LDS
LNAM=LensDesignName;LensDesignName
SPH=+1.00;+1.00
CYL=+0.0;+0.00
AX=0.0;0.0
ADD=2.0;2.0
CRIB=60;60
LIND=1.5;1.5
FRNT=5.00;5.00
MINEDG=1.0;1.0
```

Listing 3.2: LDS Job Input File

Each parameter is identified by a record label followed by a record separator, which is invariably the "=" sign, and then by actual field values for right and left eyes, separated by a semicolon ";". Table 3.2 shows the meaning of the most relevant labels included in the presented case study.

Table 3.2: DCS Record labels description

Record Label	Description
LNAM	Name of lens style
SPH	Rx Sphere power (diopters)
CYL	Rx cylinder power (diopters)
AX	Prescribed cylinder axis
ADD	Addition power for multifocal lenses (diopters)
CRIB	Lens diameter (mm)
LIND	Refraction index of lens material
FRNT	Blank true front curve for power calculations (diopters)
MINEDG	Minimum edge thickness of finished lens cut to shape (mm)
GBASEX	Generator base curve (diopters)
GCROSX	Generator cross curve (diopters)

The type of lens surface to be calculated is set by the LNAM label value and in some cases, such as for progressive lenses, it heavily affects

the calculation time. In fact, while conventional single vision lenses have a spherical or toric shape and can be easily computed, when it comes to progressive lenses, a differential geometry problem has to be solved to compute the resulting *freeform* surface. These problems are often computationally intensive and thus require several seconds of processing.

In the case of conventional geometric shapes the output file, which is called the LMS file, simply includes the radius of the sphere or of the torus. On the contrary, in case of freeform surfaces, an additional file is needed which describes the whole surface through a square matrix and it's called the SDF file. In Listing 3.3 and 3.4 an example of both files is provided.

```
JOB=1111
DO=B
REQ=LMS
LNAM=LensDesignName;LensDesignName
CRIB=60;60
LIND=1.5;1.5
GBASEX=-4.43;-4.43
GCROSX=-4.60;-4.60
```

Listing 3.3: LMS Job Output File

```
REQ=SDF
JOB=1111
SURFMT=1;R;B;85;85;84;84
...
ZZ=16.688;16.176;15.682;15.206; ...
ZZ=16.310;15.800;15.308;14.833; ...
...
```

Listing 3.4: SDF Job Matrix File - 84x84 mm - step 1mm

Implementation

This subsection proposes an implementation of a LDS system based on the presented architecture. Firstly, we identify the actors of the case study as follows: *Process Owner* is the Lens Designer, *Process Consumer* is the Lens Manufacturer and *Process Runner* is the Surface Calculator.

In this case study, a Lens Designer implements its calculation algorithm with the programming language of his choice and packs the compiled application along with all libraries and dependencies in a Docker image to be pushed to a container registry. Although the *Process Smart Contract* encodes the calculation logic by including the Docker *digest* in the model, it is in charge only of the process flow, therefore no particular changes are needed to support the specific case study. Consequently, the implementation of *CreateProcess* and *ConsumeProcess* functions for respectively adding new lens design and requesting new calculation is straightforward.

Moreover, even the implementation of client applications does not require any particular change from proposed architecture as we only need to collect input and output files both on client and runner edges. However, the Process Client Application needs to be connected to the lens manufacturer's information system to allow the CNC machine to retrieve the calculated surface. We use *Amazon S3* service as our cloud storage, therefore both consumers and runners need to have an account on those platforms to store and retrieve files.

Running Time Prediction ANN

Data Mining Application collects metrics from past lens calculation instances and trains a model for each runner and for each process to predict the running time of a single lens design instance on the runner at a specified time.

In this particular context, calculation time is pretty stable for a design unless other simultaneous processes are running on the node and the available resources, such as cpu and memory, must be shared. In order to simulate a real environment and evaluate the efficacy of the prediction approach, we use one *c5d.large* node on Amazon Web Services equipped with two *Intel Xeon 3.6Ghz* vCPUs and 4GB of RAM. In addition, we choose a progressive lens design and we run manually a set of 9143 calculations to get a reference value of running time in seconds in both *busy* and *normal* state to evaluate the performance of the sample node under different load conditions. In the first *busy* case, namely when there are multiple intensive processes running at the same time on the node, the calculation mean time

is 15s with a standard deviation of 1.0s. Conversely, in the second *normal* case, when only one process is running at a time, the mean of calculation time is 7s with a standard deviation of 0.5s. We use calculated values and two normal distributions to build multiple random datasets of 200 samples/hour, from 8:00 to 20:00, from Monday to Sunday, collecting a total of 18200 samples.

In particular, for each day of the week, after randomly picking a two hours period for busy state condition, we build a first normal distribution:

$$X_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$$

where X_1 is a random variable that represents the calculation time in busy state condition with mean value $\mu_1 = 15s$ and standard deviation $\sigma_1 = 1s$.

Then, we build a second normal distribution:

$$X_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$$

where X_2 is a random variable that represents the calculation time in normal state condition with mean value $\mu_2 = 7s$ and standard deviation $\sigma_2 = 0.5s$.

Finally, we randomly sample 200 values/hour either from X_1 or X_2 according to the corresponding state. Since we use a normal distribution, almost 99.7% of the sampled values fall in the interval $[\mu - 3\sigma, \mu + 3\sigma]$.

After generating the dataset, we build the ANN described in Table 3.1. Since a multilayer perceptron network requires numerical values as input, *One-Hot* encoding technique [48] is used to transform *day of week* and *hour of day* features in a single vector of only 0 and 1 for each sample. The resulting network is depicted in Fig. 3.4 and consists of 20 neurons in the input layer, 21 neurons in the hidden layer and a single neuron in the output layer which represents the calculation time.

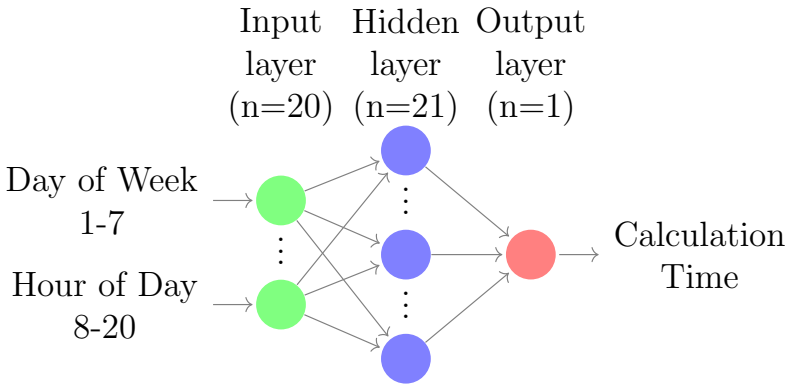


Figure 3.4: MLP Network Infrastructure

We split the original dataset into 80% training and 20% test sets. Given that we deal with a regression problem, the network is trained with Mean Squared Error (MSE) as loss function to minimize the average squared difference between the collected calculation time values and the predicted values, whereas Rectified Linear Unit (ReLU), which is nowadays the default choice for many types of neural networks as it often achieves better performance than other methods [49], is used as activation function. Lastly, *Adam* algorithm [50] is used as optimizer. The training process converges to MSE=0.38, for both training and test sets, in less than 10 epochs, as depicted in Fig. 3.5.

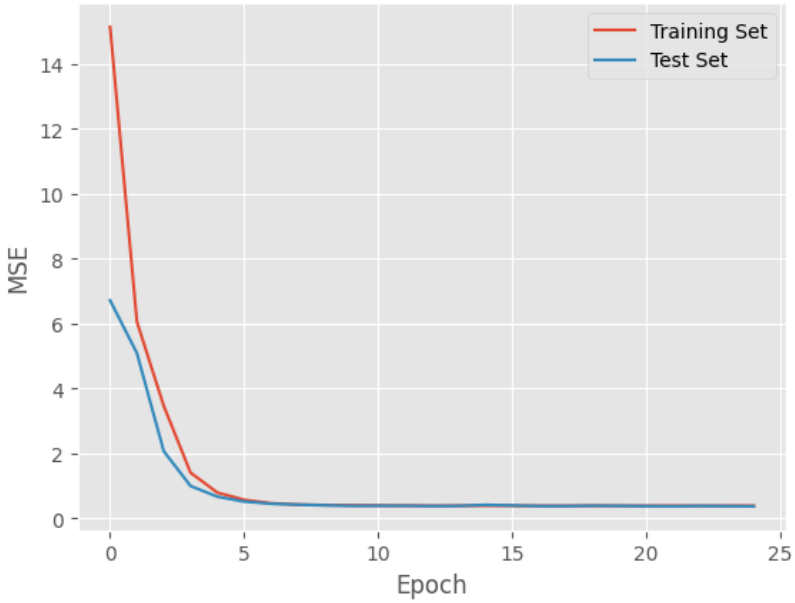


Figure 3.5: Training Epochs vs. MSE

The residual MSE=0.38 yields to RMSE=0.61s that allows us to predict the calculation time with sufficient accuracy for the purpose of this platform. In order to clearly show the effectiveness of the network after training, we predict the calculation time of the last 100 samples of the test set and we plot both actual and predicted values in Fig. 3.6. In particular, Fig. 3.6 shows that the proposed platform is able to fully capture the time band in which a specific node is busy and, consequently, to select an alternative free node to calculate a specific lens design instance in the least possible time.

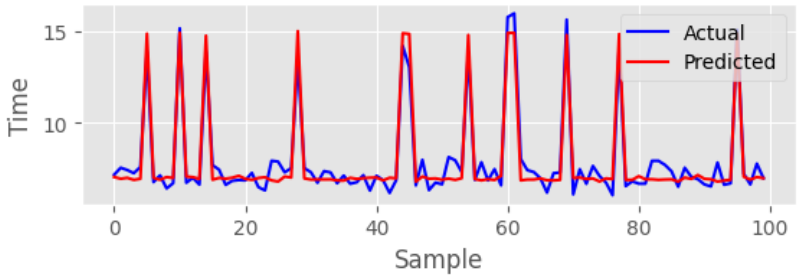


Figure 3.6: Effectiveness of the Time Prediction ANN

Discussion

In this subsection, we compare the proposed BPM platform with two existing LDS design software: IOT[1] that is a monolithic software package to be installed in a lens manufacturer calculation server; ProCrea[2] that is a client-server architecture providing calculation service through a centralized platform.

The mentioned solutions have the following limitations.

- **Lack of flexibility:** The two vendors provide ten different lens designs for single vision and progressive lenses. The availability of new designs for the lens manufacturer only depends on their updates. It is not easy for a lens designer to make a new design available to the market without relying on one of these legacy solutions.
- **The instances are executed one by one in a queue:** therefore, it is not possible to parallelize calculations. Huge queues require a long time to be processed and product lead time is inevitably affected.
- **Lack of reliability:** in the case of IOT, any fault of the local server would result in a production stop; in the case of ProCrea, the fault of the centralized platform would prevent on-premise clients to make new calculations.
- **Lack of security:** The details and the accounting of calculated jobs are stored either in the local on-premise server or in the centralized platform database. There's no guarantee that these data cannot be damaged, lost or altered and therefore the integrity of economic transactions could be affected.

[1] `https://www.iot.es/`
[2] `https://www.procreatech.com/`

- **Constrained requirements:** ProCrea client software is *Java* based and it requires a specific version of *Java Runtime Environment*. IOT, which is *Microsoft .NET* based, makes calculations locally; thus, calculation time strictly depends on the amount of CPU and memory on the on-premise server.

By implementing a LDS system in the proposed Blockchain platform, the following improvements are achieved.

- **Collaboration**: we provide a shared and decentralized platform on which multiple Lens Designers can make their algorithms available in form of self-contained Docker images. Consequently, computational resources providers can register as calculators and provide their service. The manufacturer uses always the same client for all lens designs. The integration of new or updated designs is therefore easier, as it does not require any structural change or any significant reconfiguration of the environment.

- **Performance**: the provided solution is distributed and fault tolerant. Indeed, the manufacturer can run multiple calculations, that are executed simultaneously on different runners. Hence, the performances in term of computational time and reliability improve.

- **Cybersecurity**: since the Blockchain is tamper-proof, the economic transactions of the fees charged to the manufacturer and paid to lens designers and surface calculators cannot be altered. The use of Ethereum platform gives the additional option to finalize transactions right on the chain by implementing a dedicated ERC-20 token [51].

- **Costs**: since the cloud storage service has typically low fares, a long-term storage is guaranteed for input and output files. This feature also prevents data loss on manufacturer's side. In addition, when the Blockchain is implemented in *Consortium* fashion, such as in the proposed platform, the cost of executing transactions in Ethereum is controllable and can be even zeroed.

However, every Blockchain based solution has some limitations that need to be investigated. For example, in terms of scalability, the number of transactions per second is strictly dependent on the consensus algorithm that is implemented in the system. In that respect, since the adopted Blockchain is implemented in *Consortium* fashion, the consensus algorithm is scalable

and does not require the time consuming operations of mining-based algorithms. In addition, from the transactional privacy perspective, although the core of the intellectual property is in the Docker images and in the file stored in the Cloud Storage area, the content of transactions is visible to all users who have access to the Blockchain. Hence, the implementation of a complex Blockchain architecture such as *HyperLedger Fabric*, which can limit the user access to the transactions, would benefit the proposed system.

3.5 Conclusion

The Blockchain is a consolidated paradigm shift in software architectures. The increasing interest in this technology over the last decade has shown that it can be adopted not only for financial applications such as cryptocurrencies, but also in industry environments and specifically in Cloud Manufacturing to increase the efficiency of production process and take the benefits of a decentralized approach and a tamper-proof ledger.

This work proposes a novel platform for improving digital processes in an Ethereum based Blockchain enviroment in which we integrate two of the most popular cloud technologies: Docker and Cloud Storage. In detail, Docker, which in a typical Blockchain architecture is usually employed to run the software components of the Blokchain, in this platform runs process instances using input files stored in Cloud Storage in a distributed and flexible computational platform. We identify and describe the roles of *Process Owner*, *Process Consumer* and *Process Runner*, whose differences and cooperation are the fundamentals of the proposed platform.

Process life-cycle is implemented in the main Smart Contract, while the individual process core logic is packed in a Docker image enabling any generic host to perform an heterogeneous set of tasks. A simple task assignment problem, implemented through an ANN approach, is also introduced for the purpose of improving the performance by decreasing instances execution times.

Finally, we provide a case study regarding the implementation of this platform in Ophthalmic Lenses Manufacturing environment and specifically in Lens Design Systems. Two results are enlightened. First, we show that no significant changes are needed on the industrial process to implement the new platform. Second, we underline the benefits of moving from a legacy static and centralized approach to a flexible decentralized, distributed, secure and collaborative platform on which multiple lens designers can provide their calculation algorithms while lens manufacturers can run different calculations simultaneously on registered calculators.

In the next chapter, the same architecture is proposed in *Hyperledger Fabric* blockchain and the task assignment problem is implemented by a Deep Reinforcement Algorithm algorithm.

4 Optimal Task Assignment Problem with DRL in Hyperledger Fabric

4.1 Introduction

In this chapter, the second work dedicated to the problem of optimal software tasks assignment in Blockchain environments in Manufacturing is presented. Differently from the first work, in which the runtime prediction algorithm, based on a classical ANN, is offline, in this second approach an online method based on Deep Reinforcement Learning is suggested that gradually learns how to make correct predictions. Moreover, a more structured auction-based tasks orchestration framework designed on Hyperledger Fabric is suggested.

The migration from on-premise to Cloud-based platforms has been rising over the past years. The public Cloud *infrastructure as a service (IaaS)* spending forecast for 2023 is of 156 billions U.S. dollars, whereas in 2019 it was only 45 [52]. Consequently, due to the high requirements of several application categories, the rapid increase of company investments in the Cloud leads to a continuous growing of computing power, memory and storage demands.

In this context, scientific workflows, data mining algorithms, file format conversion (e.g. video and audio file), 3D rendering and cryptography-based solutions are only some examples. All of those applications are resource consumption intensive in terms of CPU and memory and it is often required in enterprise environments that they are completed in the least time possible, especially when they are part of a production flow.

However, due to the high volatility of resources availability in the Cloud, application runtime is not always static and strictly depends on the assigned environment current workload, therefore it needs to be somehow estimated.

This problem is well known in literature as performance prediction and it has been addressed since 2005 with different techniques based on the continuous collection of resource consumption data such as CPU and memory usage. In [53], a systematic review of performance prediction of parallel applications is provided. The authors observe that in the 81.7% of considered papers, the estimation is based on analytic methods in which the corresponding equations are either manually or automatically derived mostly by stochastic linear and non-linear approaches. On the other hand, in the remaining 18.29%, non-analytic methods, such as *Support Vector Machines* or *Artificial Neural Networks* are used. Although most of these approaches are quite accurate, they cannot be easily used in modern Cloud environments for at least three reasons:

i) they are too application-specific or platform-dependent and, therefore, they have a limited field of application;

ii) they are intended for single-core processors or for a specific class of multi-core architectures and they do not consider the vast class of available platforms;

iii) they assume a free workload context in which the underlying resources are not being used by other concurrent processes, which is unrealistic in a typical Cloud environment.

Indeed, these limitations are reflected in some recent works. For example, in [54], the proposed performance prediction approach is restricted to scientific applications on *HPC Cluster* platforms. In contrast, the method proposed in [55] is not application-specific but it is restricted to *Grid* environments. Moreover, many of the existing works, such as [56] and [57], suggest an offline approach, whereas all modern applications require an online continuous incremental learning.

As a result, due also to the heterogeneity of the applications and the volatility of configuration and availability of the underlying resources, the existing methods are not generally applicable in a Cloud environment.

In alternative, the adoption of Reinforcement Learning-based (RL) techniques is a valid approach. In [58, 59], the authors propose RL-based methods for solving specific resource management and task mapping problems and minimizing application runtime by prediction. However, [58] is restricted to the problem of optimizing the performance of communication bound applications on parallel computing systems. In contrast, [59] proposes a mapping problem for generic tasks in a multi-resource cluster but, in its prediction approach, it assumes that the resource demand for each

task is known in advance. When it comes to Cloud environments, this is certainly a strong assumption as the task demands are not necessarily known upon arrival on the system.

In addition, decentralization is another key factor in Cloud environments. In fact, in a typical scenario multiple resources are available from different locations on a variable time-based fee. The resulting resources trading problem among nodes poses several security and integrity issues for transactions that have been addressed in recent literature with auction systems implemented on Blockchain, a popular secure distributed ledger technology, and combined with RL-based techniques for various optimization objectives such as maximizing participants payoff, minimizing energy consumption or adjusting block size [60].

In this work, we overcome the limitations of the existing methods and we formulate a task assignment problem by performance prediction for Cloud applications in a multi-agent environment that is based on an incremental online learning process. Diversely from the approaches proposed in [54] and [55], which are restricted to only some classes of platforms and applications, the novel method is neither application-specific or platform-dependent and is able to work in a generic cloud environment in which, for each agent, a certain number of concurrent different processes are running. The main objective of the proposed approach is to find, according to the client preferences, the agent that completes the execution in least time or, in alternative, the cheapest agent.

Differently from [60], which is focused mostly on the use of Blockchain for the implementation of optimal auction and bidding strategies, we leverage on Hyperledger Fabric to manage both the agent selection process by auction and the task execution. In addition, we adopt a DRL incremental learning approach [25] to enable each agent to predict task runtime and therefore place a bid for a submitted task. The proposed DRL-based algorithm overcomes the limitations of [59] as it does not assume any preliminary consumption-related information in advance.

In more detail, the research contributions are listed as follows:

1. A Blockchain-based trading platform is designed on top of Hyperledger Fabric to orchestrate clients requests, agents bids and resulting task assignment. We formulate a double bidding strategy according to client preferences. On the one hand, agents provide both a runtime estimation and a price. On the other hand, clients choice between the least price and the least estimated time according to their current setting.

2. A model-free DRL framework for task runtime estimation to support the agent selection process. The proposed algorithm enables the agent to incrementally learn how to do predictions for generic tasks represented by only two parameters, considering its current load in terms of resources consumption and already running tasks.

To the best of our knowledge, it is the first time that Hyperledger Fabric Blockchain and DRL are combined together for task assignment orchestration and performance prediction. In this regard, in our previous works [1] and [3], resumed in Chapter 3, we propose a similar framework, which is based on Ethereum [61] and implements a traditional offline deep learning-based prediction strategy, rather than an online approach.

We conduct extensive experiments to evaluate the performance of the proposed DRL algorithm with different training factors. In particular, we show how it works by varying discount factor, number of episodes and policy exploration vs. exploitation probability.

In Chapter 2, some basic concepts related to Blockchain, HyperLedger, Smart Contracts and DRL have been introduced. The rest of the chapter is organized as follows: Section 4.2 describes some works about performance prediction, DRL-based task assignment algorithms and resources trading systems in Blockchain environments combined with DRL. Section 4.3 introduces the proposed system model. The DRL-based task assignment approach is described in section 4.4, whereas in Section 4.5 experimental results are presented in detail. Finally, Section 4.6 concludes the chapter.

4.2 Related Work

The prediction of performance for a vast class of tasks with machine learning (ML) techniques has been extensively studied over the past years. Some of the earliest works suggest how to predict resource consumption over time, such as CPU utilization, amount of used memory, disk space and network bandwidth, given a set of environment attributes.

In [62], the suitability of several machine learning techniques for predicting spatio-temporal utilization of resources by two bioinformatics applications, BLAST and RAxML, is studied. Some of the investigated methods are: k-nearest neighbor, linear regression, decision table, Radial Basis Function network, Predicting Query Runtime and Support Vector Machine. The authors conclude that different algorithms perform better in different situations and that including as many attributes as available, even from monitoring systems, can improve prediction performance.

In [63], a novel approach for learning ensemble prediction models of tasks runtime is presented. The authors use bagging techniques to produce ensembles of regression trees to be used in scientific workflows, such as gene expression analysis, made of different sub-tasks, and show that their method leads to significant prediction-error reductions when compared with standalone models.

Still referring to scientific workflows, an online incremental learning approach to predict the runtime of tasks in cloud environments is suggested in [64]. The incremental approach enables to capture some typical cloud features, such as the continuous environmental changes and the heterogeneity of the different platforms whereas. Recurrent Neural Network (RNN) and K-Nearest Neighbors (KNN) are used for estimation. To improve predictions, fine-grained time-series resources monitoring data related to CPU utilization, memory usage, and I/O activities are collected for each unique task in the form of time-series records. The authors show that their approach significantly outperforms state-of-the-art solutions in terms of estimation error.

A similar approach, called two-stage prediction, is proposed in [65]. In this work, first the resources consumption in terms of CPU utilization, memory, storage, bandwidth, I/O is estimated for a single task instance in a cloud environment and then the result is used for runtime prediction. However, that is an offline machine learning approach and has some limitations compared to online approach in which data are processed as soon as they arrive in a near real-time fashion and it does not reflect the streaming nature of workloads in cloud environments.

Furthermore, in [53] a systematic literature review of performance prediction methods for parallel applications is presented that includes analytic and non-analytic methods and indicates future research trends and some unsolved issues. In particular, this work shows that performance prediction has been applied on a wide range of domains and reviews 82 different approaches developed between 2005 and 2020. However, the authors conclude that most of these methods focus on a specific application type and platform and therefore there's a lack of independent solutions able to work in unknown environments.

Resource management and task assignment are two other problems that have received great attention in the related literature. In this context, the DRL approach, that is often used for intelligent-robot related problems such as optimal path planning and obstacle avoidance [66], has recently become very popular. For instance, in [59], a multi-resource cluster scheduler named *DeepRM* is presented that is able to learn how to manage resources directly from experience in an online fashion and to optimize various objectives such as minimizing average job slowdown or completion time. The authors show that their method performs comparably to state-of-the-art heuristics, adapts to different conditions and converges relatively quickly.

In [58], a DRL approach for solving task mapping problems with dynamic traffic on parallel systems is discussed. The algorithm explores better task mappings by using a network simulator that predicts performance and runtime communication behaviors. Since communication pattern are often changing and unknown, network performance is difficult to be accurately estimated, therefore the authors claim that DRL is an efficient solution in this dynamic context and show that their method performs comparably or better than previous approaches.

Diversely from the aforementioned works, the DRL approach proposed in this work does not rely on a specific class of application, but it is intended to be an abstract framework combined with modern container based technologies, such as Docker, with the purpose to learn the behavior and estimate the execution time of a generic software task that can be packed in a container image and can be monitored by Docker resources metrics in a competitive environment.

Blockchain-enabled solutions for a vast range of problems have been recently arising. For example, in our previous works [1] and [3], a Smart Contracts-based platform is proposed for improving digital processes in a Cloud manufacturing environment. In detail, we combine Blockchain with Docker and Cloud Storage and introduce a deep learning approach in a task mapping framework. In [67], a blockchain-based two-stage secure spectrum intelligent sensing and sharing auction mechanism for mobile devices is

designed in a consortium blockchain to guarantee secure and efficient spectrum auction with low complexity. In [68], the authors introduce a novel scalable and multi-layer blockchain-based energy trading framework for cooperative microgrid systems that considers the issue of of perceiving the status of block generation over temporary network disruption and improves the consensus and the reliability of energy trading.

Moreover, several recent works have been proposed addressing security, data integrity and optimization problems with a combination of Blockchain and DRL. For instance, in [69], a reliable data collection and a secure sharing scheme for smart mobile terminals are proposed. The authors introduce an Ethereum-based Blockchain to safely manage data sharing with a DRL approach to achieve the maximum amount of collected data, geographic fairness and minimize energy consumption. Several simulations experiments show that their method outperforms traditional database based approaches in terms of reliability and security.

In [70], a smart grid blockchain combined with fog computing is suggested. This work includes a Hyperledger Fabric Blockchain in which the nodes are part of a fog computing environment. A verifiable random function is proposed to ensure randomness and increase safety in the selection of the primary node while keeping the probability proportional to the computing power provided by each member. Based on storage cost and security constraints, a DRL scheme is implemented to adjust the block size and the block interval in the proposed Blockchain. By conducting extensive simulations, the authors show the superiority of their scheme in terms of throughput and latency. A similar approach, though intended to decrease energy consumption and to improve the efficiency of the consensus process in Blockchain-enabled Industrial Internet of Things systems by adjusting block size and offloading some tasks to computing servers, is suggested in [71]. The problem is formulated as a high-dynamic and high-dimensional Markov Decision Process for whom a DRL approach is used to converge to an optimal solution.

A peer-to-peer energy trading problem among microgrids is investigated in [72]. In this work a multi-agent deep deterministic policy gradient, based on energy trading algorithm, is proposed to enable each microgrid to maximize its own utility in a local market. Given the uncertainties and the constraints in renewable energy and power demand, the authors claim that the DRL-based approach is suitable to help each microgrid to find its optimal policy. An Ethereum Blockchain is adopted to ensure the integrity of transaction data.

A Blockchain-enabled computing resource trading system is proposed in [60]. This system takes into account pricing and bidding strategies to

enable providers and customers to trade computing resources on a safe and tamper-proof environment. A decision-making problem in the continuous double auction is formulated with the goal of maximizing each participant payoffs, while a DRL approach is adopted to help them building their optimal bidding strategies. The authors conduct extensive simulations and show that their scheme outperforms other existing methods.

Finally, in [73], a DRL approach is used to solve a joint optimization problem to enhance adaptivity and scalability in Blockchain environments. The proposed approach considers the optimal selection of consensus protocols and the allocation of computation and bandwidth resources. The authors show through extensive simulations the effectiveness of their scheme.

4.3 System Model

In this section, we introduce a new system that combines a novel runtime estimation algorithm based on DRL in a competitive task assignment framework safely managed by a Hyperledger-based Enterprise Blockchain. The proposed architecture is depicted in Fig. 4.1. The system is made up of three layers: the Blockchain layer in the middle, and the Agent and Client layers on the sides.

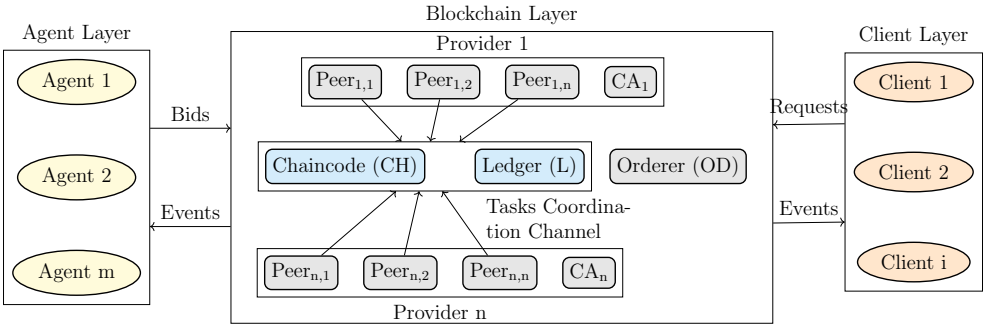


Figure 4.1: System Architecture

Blockchain Layer

Let us consider a set of pre-authorized organizations $O = \{o_1, o_2, o_n, \ldots, o_{|O|}\}$, called providers, that constitute the nodes of the permissioned Hyperledger Blockchain. An organization in Hyperledger Fabric is simply a firm that decides to join a network and that is authorized by other existing members. Note that symbol $|A|$ denotes the cardinality of the generic set A. Moreover, each provider o_n delivers a set of peers $P_n = \{p_{n,1}, \ldots, p_{n,i}, \ldots, p_{n,|P_n|}\}$ joining the network, where $p_{n,i}$ is the *i-th* peer of provider o_n. Each provider o_n has one Certification Authority CA_n to provide and renew certificates for Agents, Clients and other components. All peers in the set P_n hold a copy of the chaincode CH and of the ledger L. The chaincode CH is the main Smart Contract that regulates the requests of the clients, the bids received by the agents and coordinates the task assignment process. The ledger L stores the current state of the tasks and the related transactions. Finally, the Orderer node OD is responsible for packaging transactions into blocks and distributing them to the peers in the sets P_n across the network.

Client Layer

In the proposed system, we define the set of clients $C = \{c_1, \ldots, , c_i, \ldots, c_{|C|}\}$, where c_i implements a step of a generic process flow, for which it is required to run computationally intensive tasks. Since c_i may not have enough resources to run tasks locally, it interacts with the Blockchain to submit execution requests. For each successful task execution, a variable fee is charged.

For each c_i, we define the set

$$R_i = \{r_{i,1}, \ldots, r_{i,j}, \ldots, r_{i,|R_i|}\}$$

where $r_{i,j}$ represents the *j-th* execution request of client c_i. In addition, for each request $r_{i,j}$, the client c_i has to specify the strategy and the task details. Considering the used strategy, we define two alternative options: *time-sensitive* and *price-sensitive*. In the first case, we assume that the client objective is to complete the task in the least time possible, regardless of the charged fee. In the second case, the requirement is to pay the lowest fee, regardless of the execution time. In order to specify task details, we define the set $T = \{t_1, \ldots, t_k, ..., t_{|T|}\}$, where t_k represents a generic task that can be requested by the clients with an arbitrary input parameter.

Now, each request $r_{i,j}$ can be defined by a triple $r_{i,j} = \langle g_{i,j}, t_{i,j}, p_{i,j} \rangle$, where $g_{i,j} \in [0, 1]$ is the strategy, with 0 representing *time-sensitive* and 1 *price-sensitive*, $t_{i,j} = t_k \in T$ is the required task and $p_{i,j}$ is the input argument.

Agent Layer

Each request $r_{i,j}$ submitted by a client requires an Agent to execute the associated task. In this fully distributed environment, we define the set of agents $A = \{a_1, \ldots, a_m, \ldots, a_{|A|}\}$, where a_m is a node that joins the network and provides its computational resources, such as CPU and memory, to the system. For the sake of generalization, we use Docker container-based technology to embed the tasks in self-dependent images [23]. In this way, since everything is packed in the Docker image, the nodes do not have any constraint in terms of operating system and dependencies.

Now, at each request $r_{i,j}$, we associate an agent $a_m \in A$ that is denoted $a_{i,j}$. Such an agent is paid a variable fee upon successful execution of the related task. Moreover, since for each agent a_m we allow the execution of concurrent processes, the execution time strictly depends on its current load state.

As illustrated in section 4.3, the client can choose between *time-sensitive* and *price-sensitive* strategies. More specifically, when the former strategy is chosen, the client goal is to minimize the run-time of its processes and thus each agent a_m is required to provide a reliable prediction of the execution time for each request $r_{n,i}$ based on its current state. Conversely, in case the latter strategy is selected, only the proposed price is relevant for the assignment. The main scope of the *price-sensitive* option is to allow an agent that has recently joined the network and it is not able to provide a reliable execution time prediction to compete in price rather than in time and collect new experiences to progressively improve its predictions.

4.4 DRL-Based Task Assignment Process

In this section, we introduce a competitive task assignment process by expected runtime prediction that implements the two client strategies: *time-sensitive* and *price-sensitive*. The agents leverage on a novel DRL approach to provide predictions.

Task Assignment Process

The main process is designed in the chaincode CH and is depicted in the UML sequence diagram in Fig. 4.2. Both the clients C and the agents A continuously interact with the Blockchain for requests and tasks management. More specifically, in the first part of the diagram, the agent selection is performed and in the second part, post-assignment actions are implemented.

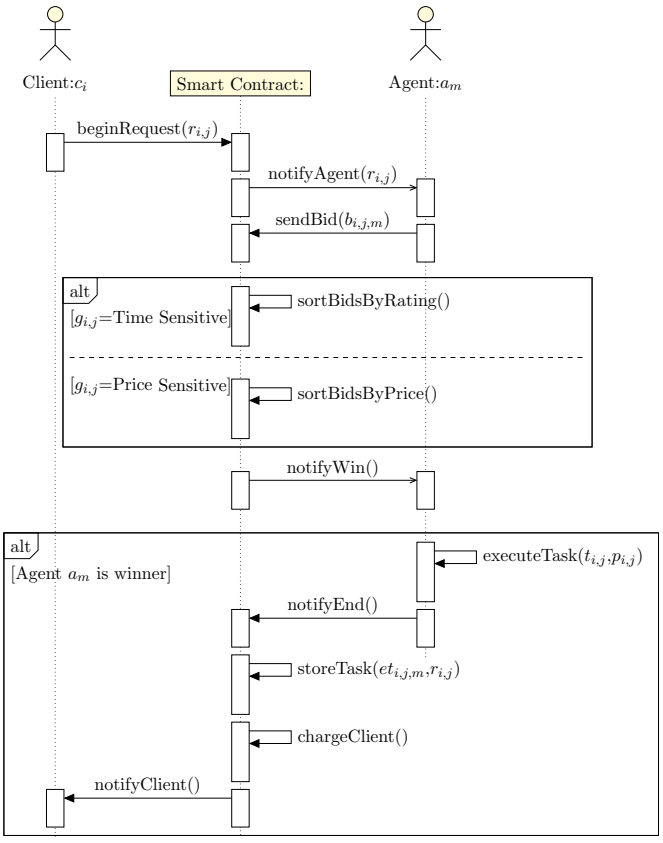


Figure 4.2: Agent selection UML sequence diagram

The agent selection process is detailed in Algorithm 1, which is implemented in the chaincode in *GO* [74] programming language. The input is the client request $r_{i,j}$ and the output is the winner agent $a_{i,j} \in A$.

Algorithm 1 Agent selection algorithm

Input: A Client Request $r_{i,j} = \langle g_{i,j}, t_{i,j}, p_{i,j} \rangle$
Output: Winner Agent $a_{i,j} \in A$
Step 1:
foreach $a_m \in A$ **do**
Forward request $r_{i,j}$ to Agent a_m Collect bid $b_{i,j,m} = \langle et_{i,j,m}, pr_{i,j,m} \rangle$ $B \leftarrow B \cup \{b_{i,j,m}\}$
end
Step 2:
if $g_{i,j}$ *is time-sensitive* **then**
Sort set B by estimated time $et_{i,j,m}$
else if s *is price-sensitive* **then**
Sort set B by bid price $pr_{i,j,m}$
Step 3:
Set $h \leftarrow 1$ **while** $B \neq \varnothing$ **do**
Take bid $b_{i,j,m}(h) \in B$ **if** *Agent* $\overline{a_m}$ *is available* **then**
Agent $\overline{a_m}$ is the winner $a_{i,j} \leftarrow \overline{a_m}$ **return** $a_{i,j}$;
else
$B \leftarrow B \setminus \{b_{i,j,m}\}$ Set $h = h + 1$
end

In Step 1, a request $r_{i,j}$ for a $b_{i,j,m}$ is sent to each available agent $a_m \in A$, where $b_{i,j,m}$ is the bid of agent m for the request j of client c_i. In addition, $b_{i,j,m}$ is defined as $b_{i,j,m} = \langle et_{i,j,m}, pr_{i,j,m} \rangle$ where $et_{i,j,m}$ is the estimated completion time provided by the agent a_m and $pr_{i,j,m}$ is the proposed price. All received bids are collected in the set $B = \{b_{i,j,1}, \ldots, b_{i,j,m}, \ldots, b_{i,j,|A|}\}$. In Step 2, given the strategy $g_{i,j}$ in $r_{i,j}$, the members of B are sorted by $et_{i,j,m}$ or $pr_{i,j,m}$ accordingly. Finally, in Step 3, the first available agent $\overline{a_m}$ associated to $b_{i,j,m}$ in the sorted rank is elected as the winner and therefore $a_{i,j} = \overline{a_m}$.

The estimated time $et_{i,j,m}$ leverages on the DRL approach that is described in Section 4.4. On the contrary, the bid price $pr_{i,j,m}$ is determined autonomously by the agent to compete in the current race.

After a request $r_{i,j}$ has been successfully assigned, the winner agent $a_{i,j}$ executes the task $t_{i,j}$ with the parameter $p_{i,j}$ and notifies the completion to the chaincode CH. Finally, the instance metrics, such as the effective runtime, are stored in the tamper-proof Blockchain and the client c_i is charged according to the offset between the actual execution time and the prediction.

DRL approach for running-time predictions

In this section, we propose a DRL approach for task runtime prediction. In particular, the Deep-Q-Network (DQN) algorithm proposed in [25] is applied to the considered problem. The triple (S, A, R) defines a RL deterministic model, where S is the State Space, A is the Action Space and R is the Reward. In the considered application the triple (S, A, R) is specified in the following.

1. **State Space (S):** We assume that each agent is able to run a fixed number k of concurrent tasks as individual containers. As a consequence, execution time is potentially impacted by three combined parameters to be considered on a new incoming request $r_{i,j}$:

   I) the type of task $t_{i,j} \in T$, its associated input parameter $p_{i,j}$ and submission time $ts_{i,j}$;

   II) total available resources for each agent a_m: cpu_speed_m, number of cores n_cores_m, maximum memory max_memory_m, hdd_type_m (rotational or high-speed SSD), available network bandwidth bw_m;

   III) current resources consumption of previous $k-1$ processes running on a_m: $cpu\_time_{m,n}$, $mem_{m,n}$, $net\_i/o_{m,n}$, $block\_i/o_{m,n}$, number of processes $n\_pids_{m,n}$ where $n = 1, 2, \ldots, k-1$.

   Now, we introduce the following three tuples:

   I) $s_{m,1} = \langle t_{i,j}, p_{i,j}, ts_{i,j} \rangle$;

   II) $s_{m,2} = \langle cpu\_speed_m, n\_cores_m, max\_memory_m, hdd\_type_m \rangle$;

   III) $s_{m,3} = \langle cpu\_time_{m,n}, mem_{m,n}, net\_i/o_{m,n}, block\_i/o_{m,n}, n\_pids_{m,n} \rangle$, where $n = 1, 2, \ldots, k-1$.

   Moreover, we define $S_m = \{s_{m,1}, s_{m,2}, s_{m,3}\}$ that represents the current state of agent a_m in the proposed DQN framework. The structure of $s_{m,2}$ reflects the current node configuration. Those features are assumed to be static and are described in Table 4.1. In addition, $s_{m,3}$ denotes the current running containers resources usage and is continuously updated. Since in the proposed system, each task is represented by a Docker image and, consequently, each instance is essentially a container based on that image, we can leverage on live metrics exposed by Docker to collect live resources usage data. In Table 4.2, all those metrics are described.

2. **Action Space (A):** Given the current state of the candidate agent a_m, the target is to predict the expected runtime in seconds for the incoming client request $r_{i,j}$. In the proposed DRL approach, the bid estimated value $et_{i,j,m}$ constitutes the action. In order to ease the training, we consider a discrete set of execution times in steps of 1 second with a predefined upper bound value that constitutes a global time out for each instance processed by the system.

3. **Reward (R):** Once task $t_{i,j}$ has been executed, each state S_m and action $et_{i,j,m}$ produce the actual elapsed time as observation and contribute to the reward by assessing the level of prediction accuracy. Thus, if we set a constant token TK_{max} as a maximum reward for each successful calculation, we can calibrate the reward value $RD_{i,j,m}$ by considering the relative error between actual execution time $at_{i,j,m}$ and estimated time $et_{i,j,m}$. In detail, $RD_{i,j,m}$ is defined by the following formula:

$$RD_{i,j,m} = TK_{max} \begin{cases} \frac{at_{i,j,m}}{et_{i,j,m}} & at_{i,j,m} \leq et_{i,j,m} \\ \frac{et_{i,j,m}}{at_{i,j,m}} & otherwise. \end{cases} \tag{4.1}$$

Table 4.1: Node Configuration Features

Feature	Description	UM
cpu_{speed}	CPU speed	Mhz
n_{cores}	Number of cores	n/a
max_{memory}	Memory	MB
hdd_{type}	Storage type	(0=spinning,1=solid state)
bw	Network bandwidth	Kbps

The adopted DQN scheme is described by Algorithm 2. Firstly, according to the approach in [25], a number of episodes E for training and the condition for a state S_m to be terminal are defined. In the proposed approach, the number of episodes is arbitrary and is strictly related to the number and types of tasks in the set T. Basically, it should be as long as to guarantee an accurate prediction of execution runtime $et_{i,j,m}$ in every agent load condition. Similarly, a maximum number of iterations I is set after which the state S_m is terminal and single episode is completed.

Step 1 initializes the replay memory set D with capacity N and the Q network that will be trained. Moreover, in Step 2, for each iteration of each

Table 4.2: Resources Usage Metrics

Metrics	Description	UM
cpu_{time}	The amount of cpu usage accumulated by the process	1/100th seconds
mem	The total memory the container is using	MiB
$net_{i/o}$	The amount of data the container has sent and received over its network interface	bytes
$block_{i/o}$	The amount of data the container has read to and written from block devices on the host	bytes
n_{pids}	The number of processes or threads the container has created	n/a

episode, whenever a new request $r_{i,j}$ is submitted to agent a_m, a new state S_m is built and a value for $et_{i,j,m}$ is randomly selected or predicted with probability ϵ. Subsequently, price $pr_{i,j,m}$ is arbitrarily determined by agent a_m and the bid $b_{i,j,m}$ is definitely set. As shown in Fig. 4.2, $b_{i,j,m}$ is sent to the Smart Contract on which the Algorithm 1 is executed after all bids from all agents are collected.

In Step 3, if the agent a_m is the winner, $t_{i,j}$ is run locally and the actual execution time $at_{i,j,m}$ is observed from Docker metrics to calculate the step reward $RD_{i,j,m}$. At the same time, a new training step begins.

In Step 4, on next incoming request $r'_{i,j}$, the future state S'_m is determined. Then, the whole transition tr_h is stored in experience replay memory set D. Subsequently, M transitions are randomly selected from set D in the subset D_h. Then, a new value y_n that considers maximum future reward as predicted from target network Q and discounted by γ is calculated as updated reward for each transition $tb_n \in D_h$.

Finally, in Step 5, a new Q network is trained by performing a gradient descent step on $(y_n - Q(S_{m,n}, et_{i,j,m,n}; \Theta))^2$, in order to get new θ for next iteration $h+1$. The target network is updated every w iterations to stabilize learning, where w must be preliminary assigned.

Algorithm 2 Deep Q Learning

Step 1:
Initialize replay memory set D with capacity N
Initialize $Q(S_m, \widehat{et})$ arbitrarily
Step 2:
Set $e = 1$
while $e \leq E$ **do**
  Set $\hat{i} = 1$
  while $\hat{i} \leq I$ **do**
    Wait for request $r_{i,j}$
    With probability ϵ select a random action $\widehat{et}$
    otherwise select $\widehat{et} = \max_{\widehat{et}} Q(s, \widehat{et}; \theta)$
    Set $et_{i,j,m} = \widehat{et}$
    Set arbitrary price $pr_{i,j,m}$
    Set $b_{i,j,m} = \langle et_{i,j,m}, pr_{i,j,m} \rangle$
    Send $b_{i,j,m}$ to Smart Contract and wait for Algorithm 1
    Step 3:
    if *agent* a_m *is the winner* **then**
      Take action $et_{i,j,m} = \widehat{et}$, observe $RD_{i,j,m}, S'_m$
      Step 4:
      Wait for next request $r'_{i,j}$
      Store transition $tr_h = \langle S_m, et_{i,j,m}, RD_{i,j,m}, S'_m \rangle$ in D
      Sample random minibatch of M transitions $D_h = \{tb_1, tb_2, tb_n, \ldots, tb_M\}$ where $tb_n \in D$
      foreach $tb_n \in D_h$ **do**

$$\text{Set } y_n \leftarrow \begin{cases} RD_{i,j,m,n} \\ \text{for terminal } S'_{m,n} \\ RD_{i,j,m,n} + \\ \gamma \max_{et_{i,j,m}} Q(S'_{m,n}, et'_{i,j,m}; \theta) \\ \text{for non-terminal } S'_{m,n} \end{cases}$$

      end
      Step 5:
      Perform gradient descent step on $(y_n - Q(S_{m,n}, et_{i,j,m,n}; \theta))^2$ where $n = 1, 2, \ldots, M$
      $S_m \leftarrow S'_m$
      Set $\hat{i} = \hat{i} + 1$
    end
    Set $e = e + 1$
  end
end

4.5 Performance Evaluation

In this section, we evaluate the performance of the proposed task runtime prediction DQN driven algorithm in a simulated environment.

Simulation Settings

In the proposed case study, we identify three common software algorithms with different complexities to evaluate the proposed system, therefore we set $T = \{t_1, t_2, t_3\}$. We code the tasks using *Python* language in a single Docker image that requires two parameters on launch: task $t_{i,j} \in T$ and parameter $p_{i,j}$. In the considered case, the value of $p_{i,j}$ is restricted to the members of the set $P = \{1, ..., 5\}$ with $p_{i,j} \in P$.

In particular, we consider the following three well known algorithms:

1. **Standard Array Sorting** (t_1): it builds a random big *Python* integer list whose number of elements is based on $p_{i,j}$. After building the list, the *sort* method is called that implements the *Timsort* algorithm [75]. This algorithm has a runtime complexity of $\mathcal{O}(n \log n)$ in the worst case;

2. **Fast Array Sorting** (t_2): instead of constructing a list, a random *NumPy* [76] integer array is built that also implements the *sort* method. However, in *NumPy* library the *quicksort* algorithm [77] is adopted which has a runtime complexity of $\mathcal{O}(n^2)$ in the worst case;

3. **Dijkstra Shortest Path Search** (t_3): The *Dijkstra*'s algorithm [78] is an algorithm for finding the shortest paths between nodes in a graph. In this implementation, firstly we build a graph with a large number of vertices V determined by parameter $p_{i,j}$. Secondly, we find the shortest path from the first vertex to all other vertices. The *Dijkstra* standard implementation has a complexity of $\mathcal{O}((|V| + |E|) \log |V|)$ in the worst case.

The proposed algorithms are currently implemented in scientific and industrial applications. For example, sorting tasks are used in operations research to implement both the *Shortest Processing Time First* and the *Longest Processing Time First* rules for optimal jobs scheduling and load balancing [79]. Indeed, *Dijkstra*'s algorithm is currently been used for a vast class of problems including vehicle path planning [80] and optimal packet routing in software defined network environments [81].

In Table 4.3, we summarize the three tasks, their runtime complexities and the minimum and maximum value of elements as determined by their linear combination with parameter $p_{i,j}$.

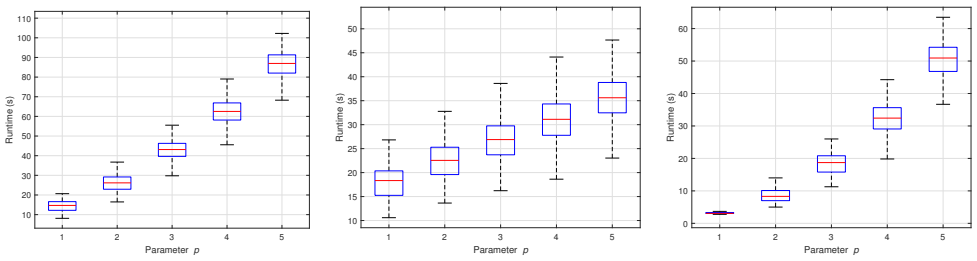
Table 4.3: Tasks Details

ID	Algorithm	Complexity	Min	Max
1	Sort	$\mathcal{O}(n \log n)$	2×10^4	5×10^4
2	NumPy sort	$\mathcal{O}(n^2)$	10^6	2×10^6
3	Dijkstra SPS	$\mathcal{O}((\|V\| + \|E\|) \log \|V\|)$	10^3	5×10^3

We set maximum concurrent tasks $k = 4$ and we run a random sequence of simultaneous instances on a designated agent. On each new submitted instance, we collect current Docker metrics from the node as described in section 4.4.

As agent, we use a *t2.medium* cloud instance from *Amazon AWS* equipped with 2x Intel Xeon 3.3Ghz vCPU and 4GB of memory. Since the number of available CPUs is less than k, we simulate an elevated competition between tasks during runtime and therefore we observe high variability in execution time, which is typical of real cloud environments.

In Fig. 4.3, the average task $t_{i,j}$ runtime, observed for each $p_{i,j}$ value over a total of 9995 metrics collected in two days simulation, is depicted.



(a) Sort t_1 (b) NumPy sort t_2 (c) Dijkstra SPS t_3

Figure 4.3: (a), (b), (c), Tasks Average Runtime for different values of parameter $p_{i,j}$

We use the metrics set for testing the performance of the DRL approach described in Section 4.4. In particular, on each iteration a new state from current item is built, then either a random time is selected or the expected runtime is predicted from current Q-function according to the exploration vs. exploitation probability. Finally, the reward is calculated against the actual runtime as described in 4.1. The current transition is stored in

experience replay, then the rest of the steps in Algorithm 2 update the current Q-function.

In this experiment, the Q-function is approximated by a dense DNN made up of three layers:

1. The input layer represents the state and consists of 21 neurons. In this layer, we combine some elements of the state to reduce the dimension and we use One Hot Encoding techniques [48] to represent some categorical variables, such as the tasks IDs.

2. The intermediate layer consists of 70 neurons, as the average between input and output dimensions

3. The output layer represents the discretized action space and consists of 120 neurons, where 1 second is the minimum expected runtime and 120 seconds is the last accepted value before raising a timeout error.

The deep neural network approximating the Q-function is depicted in Fig. 4.4. To evaluate the performance of our DRL strategy, we use a *t3.2xlarge* cloud instance from *Amazon AWS* equipped with 8 x Intel Xeon 3.1Ghz vCPU and 32GB of memory.

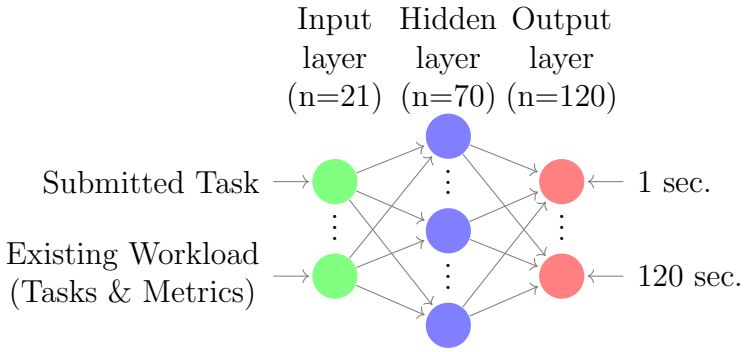


Figure 4.4: Deep Neural Network for Q-function

Performance Comparison

This section reports a performance comparison between two different scenarios tested in the performed experiment. The first scenario consists of 5000 episodes of 150 iterations each, whereas the second scenario consists of 3000 episodes of 300 iterations each. Since we are using a DRL approach, we have two more critical hyper-parameters to consider:

1. *Discount Factor* γ: in DRL environments, setting the discount factor is part of the problem [82]. The discount factor essentially determines how much the RL agent cares about rewards in the distant future relative to those in the immediate future. If $\gamma = 0$, the agent will be completely myopic and only learn about actions that produce an immediate reward. If $\gamma = 1$, the agent will evaluate each of its actions based on the sum total of all of its future rewards. In the simulations, we continuously feed the system with tasks and we are interested in evaluating how the DRL approach performs with different values of γ. Therefore, we test the algorithm for both scenarios with $\gamma = 0.2$, $\gamma = 0.5$ and $\gamma = 0.8$.

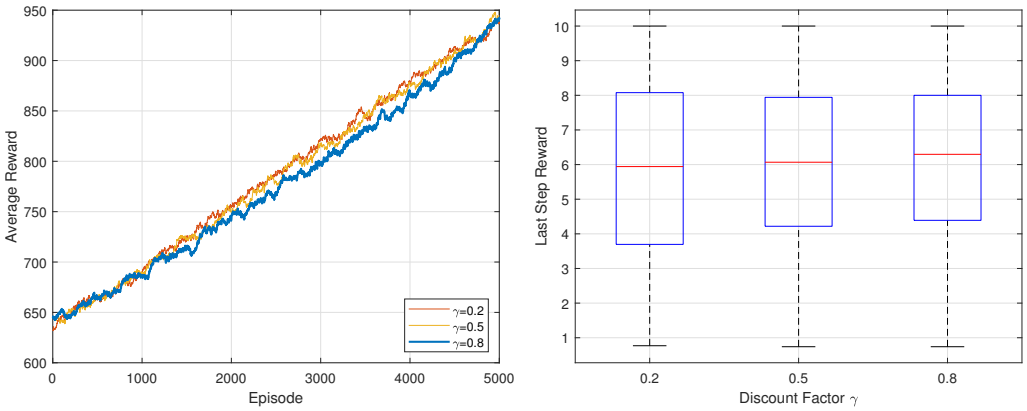
2. *Exploration vs. Exploitation Probability* ϵ: in RL, exploration means that the agent explores randomly the whole action space to improve its knowledge about each action for a long-term benefit. On the other hand, exploitation means that the agent uses only its current knowledge to get the most reward. In Algorithm 2, the choice between exploration and exploitation is made with a probability coefficient ϵ that usually varies at each episode. Normally, this value starts from 1, meaning that, since the agent doesn't know anything at the beginning about the actions, it must explore all available actions for each state. Successively, it slowly decays over future episodes until the training end, when it is very close to 0, meaning that it fully levarages on its knowledge. For the performed evaluations, we start from $\epsilon = 1$ and we implement a linear decaying strategy over episodes till a minimum value of $\epsilon = 0.001$.

Considering the aforementioned hyper-parameters, we are interested in comparing the following two metrics for both scenarios.

1. **Cumulative Reward** CR is defined for single episode of n steps as:

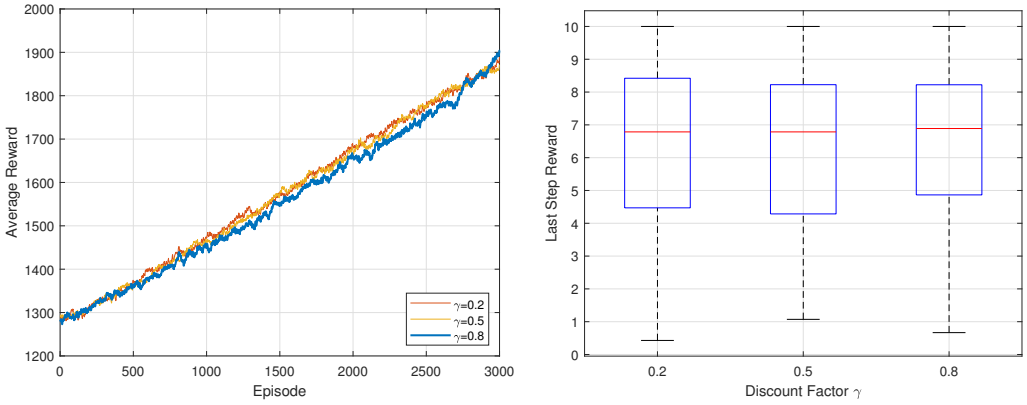
$$CR = \sum_{n} R(n);\ CR \in [0, C_{max}n]. \tag{4.2}$$

Figures 4.5a and 4.6a compare the average values CR vs. episodes for all three considered discount factors in both scenarios. For all values of γ, as the agent explores new actions and trains its Q-function, the CR value increases almost linearly and results in 46% performance improvement at the end of the training, compared to the first exploration-only episode.



(a) Average Reward (b) Prediction Accuracy

Figure 4.5: First scenario performance evaluated for different values of γ: (a) Average reward over episodes, (b) Runtime prediction accuracy summary statistics



(a) Average Reward (b) Prediction Accuracy

Figure 4.6: Second scenario performance evaluated for different values of γ: (a) Average reward over episodes, (b) Runtime prediction accuracy summary statistics

Moreover, though $\gamma = 0.2$ appears to learn faster than $\gamma = 0.5$ and $\gamma = 0.8$, in the end the performance of the highest discount factors

outperforms the smallest one. In the last part of the training, the value of ϵ becomes very small and let the algorithm leverage only on exploitation for task runtime prediction. In this case, the comparison shows that for higher values of γ, the accuracy of the Q-function slightly improves proving that the proposed DRL approach is able to catch a sort of correlation between successive submitted tasks. There are no major differences between the performance of the two scenarios. It can only be observed that in the second scenario, the learning speed for $\gamma = 0.8$ is higher than in the first scenario, compared to other values of γ. This metric can be influenced by the different number of steps in the episode and the resulting value of exploration vs. exploitation probability coefficient ϵ in each step. In fact, in the first scenario, ϵ decreases linearly every 150 steps, whereas in the second scenario every 300 steps. As a consequence, although the evaluated performance shows that a higher number of steps per episode improves the algorithm, it is evident that episode length also affects the training time. However, the simulations point out that different values of γ and ϵ may help in minimizing total computational effort.

2. **Runtime Prediction Accuracy** is measured in the last training step, in which the algorithm only exploits the approximated Q-function. We set $C_{max} = 10$ as highest step reward in case of exact runtime prediction, otherwise the value is determined in a linear fashion as defined in (4.1). Figures 4.5b and 4.6b show summary step reward statistics for both scenarios and for different values of discount factor γ. As for the previous metric, in all the cases a similar performance is observed among the compared schemes, having $\gamma = 0.8$ case the best median $\tilde{x} = 6.30$ and $\tilde{x} = 6.89$ for the first and second scenario, respectively. The latter case with $\gamma = 0.8$ shows the overall best performance, having the 50% of the steps a prediction accuracy $\geq \approx 69\%$, whereas only a residual 25% of the steps show an accuracy $\leq \approx 49\%$. Finally, the upper quartile shows a prediction accuracy $\geq \approx 82\%$ for 25% of the steps.

Given the dimensions of state space and action space, the evaluated performance is acceptable and confirms the effectiveness of the proposed DRL approach.

4.6 Conclusion

In this work, we introduce a Hyperledger Fabric Blockchain-based resources trading platform to orchestrate clients requests, agents bids and provide a solution to a common task assignment problem.

A double bidding strategy is proposed that enables client to choose either the least time among estimations provided by the agents or the cheapest price. To support node selection and task assignment, we propose a model-free DRL framework for task runtime estimation in agent current load state. In the proposed algorithm, both the submitted and the existing tasks are represented by only two parameters combined with some resources usage metrics collected at runtime. As the agent receives new tasks requests, it incrementally learns how to do predictions in an online fashion.

Simulations show that DRL is suitable for task runtime prediction and provides similar or better performance, compared to other more complex existing solutions. Moreover, it is important to enlighten that the proposed approach is not tied to a specific application and can be adopted on any type of tasks in a modern cloud environment. In future work, we plan to integrate this solution on *Kubernetes*, an open-source container-orchestration system, and evaluate the performance of the DRL approach for workload operations such as deployment and scaling.

5 An Incentive Platform in Ethereum for Energy Management

5.1 Introduction

This chapter concludes the part dedicated to the implementation of optimal algorithms in Blockchain. In this work an incentive platform for energy communities is suggested. A user ranking algorithm and a penalty-reward scheme are designed in the Blockchain. Since sensistive users consumption data are concerned, Blockchain is chosen for its intrinsic tamper-proof capabilities. The penalty-reward scheme is formulated as a classical minimization problem.

Given the continuous increase in temperatures globally, over the past 40 years, the problem of energy consumption due to buildings conditioning systems has become increasingly important [83], [84], [85].

In recent years, smart and green building management is approached by considering District Energy Management (DEM) systems. More precisely, the district (building network) has to be managed for purchasing energy and optimally balancing real-time energy consumption while satisfying user comfort [84].

One method to promote virtuous user behaviour is to provide to the consumers monetary incentives rewarding or penalizing them according to their energy management.

In [86], a novel approach is proposed based on the energy credits allocation to reduce electricity costs. The customers can use such credits during periods with high energy prices for saving considerable costs.

A holistic model has been studied [87] to optimize power scheduling within individual microgrids and energy trading among interconnected microgrids.

In [88], the authors use low-cost hardware such as the Raspberry Pi to act as the central hub of a smart home air conditioning system. However, Raspberry Pi is not the only hardware that can be used; lots of other raw sensors can be implemented with a small and cheap dedicated IoT hardware, as depicted in [89].

IoT devices are increasingly widespread [90], and this involves a series of problems related to the use of data, their privacy and storage. In [91], the Ethereum Blockchain is used to create an integrated system between the IoT and the Blockchain.

In [92], the Blockchain technology is introduced alongside the IoT sensors. This configuration for air monitoring in smart cities sends data directly to the Blockchain service; in doing so, the Blockchain platform adds a security level to the certainty of the data, avoiding their manipulation.

A more specific case of remote condition monitoring efficiency has been studied in [93], by exploring the use of Blockchains and Smart Contracts in situations where it is essential to ensure commercial and economic agreements in the data processing.

In [94], artificial neural networks are used to predict the thermal parameters of air conditioning based on historical time-series data. This solution uses a set of IoT sensors that communicate with a Wireless LAN access point. Once again, the use of a RaspberryPi as an IoT gateway is preferred.

However, all the cited papers contemplate a passive use of the Blockchains, employing them as mere register providing a higher degree of reliability.

This work proposes an innovative system for improving the user behaviour on the basis of IoT components, Blockchain, Smart Contract and a reward and penalty strategy. In particular, the IoT components provide data about the energy consumption in real time. Such data are stored in a specifically designed Smart Contract that allows adding a basic feature of the Blockchain: the payment management. Moreover, the payments are subject to a reward and penalty strategy: the users are classified in consumption classes and each class is characterized by multiplier coefficient that increases (penalty) or decreases (reward) the user energy costs.

Compared with credit-based incentive systems proposed in the related literature[86], we enlighten that the centralized approach means that users have to trust the outcomes of the process blindly. On the contrary, the application of Blockchain and Smart Contracts allows us to obtains two benefits: i) since multiple nodes are available, the decentralized approach increases the system reliability; ii) the users do not have to trust on a central authority; iii) the data stored in the Smart Contract are immutable thanks to the intrinsic nature of the Blockchain. In addition, the complete process

breaks away from the cryptocurrencies intrinsic volatility, by using a stable coin as a means of exchange, making the whole process, up to the payment, more stable, transparent and secure.

In Chapter 2, some basic concepts related to Blockchain and Smart Contracts have been introduced. The remainder of this chapter is organized as follows. Section 5.2 introduces some basic concepts about stable coins. Section 5.3 describes the proposed system and the Smart Contract architecture. Section 5.4 presents the reward and penalty procedure whereas Section 5.5 discusses a case study. Finally, Section 5.6 draws the conclusions.

5.2 Stable Coins

One crucial feature of Smart Contracts is the ability to create specific cryptocurrencies, namely tokens, to be used for financial purposes. A popular token in Ethereum is Tether ($USDT), a token with the sole and primary purpose of maintaining its value pegged to the US dollar and, therefore, to prevent fluctuations typical of other cryptocurrencies, such as Bitcoin or Ether. These tokens fall in the set of *Stable Coins*. However, Tether is not the only token of that kind available.

In this work, among the stablecoins, we use Maker's DAO ($DAI), which is a token generated by MakerDAO, a *"peer-to-peer organization created on the Ethereum network to allow people to lend and borrow using cryptocurrencies"* [95]. The value of $DAI is pegged to the US dollar through a series of smart contracts that automatically manage the liquidation of debt positions which fall below their loan/value ratio.

The rationality of using this token is due to its resilience to price fluctuations, as it remains stable at $\pm$\$0.004 from the value of the peg in the year-to-date period. Thanks to the use of this token, it is possible to generate a set of application scenarios that are not subject to the price fluctuations of Ether or other cryptocurrencies.

5.3 The Proposed System

This section describes the overall architecture of the proposed system that is shown in Fig. 5.1.

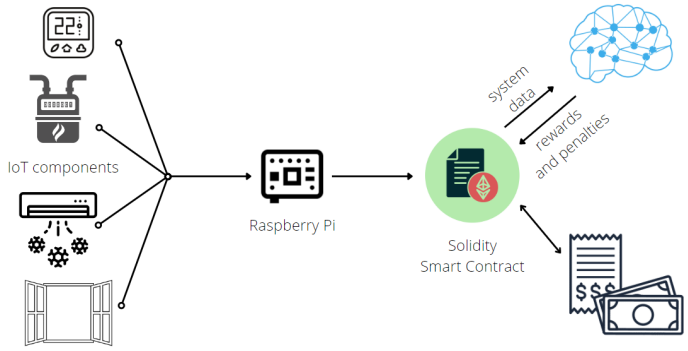


Figure 5.1: System Architecture

The system is composed of a set of IoT elements that collect consumption data from air-conditioning system and environmental data from sensors. Such IoT elements are then connected to a central data collecting unit by WiFi, which then invokes a Smart Contract in a Blockchain-based infrastructure and delivers the data collected in a specified time period. Moreover, an external module computes rewards and penalties on the basis of optimization tools. Finally, in this work, we propose to process payments on chain by $DAI stable token in order to guarantee safety and transparency.

System Architecture

The local infrastructure system consists of a Raspberry Pi (RasPi), which is a single-board computer running Linux OS and providing a set of General Purpose Input/Output (GPIO) pins. Moreover, RasPi allows the interaction through the GPIO interface with humidity and temperature sensors. The RasPi is connected by internet to the smart meter, the IoT devices and sensors that monitor energy consumptions, temperature and humidity. The data are sampled at a given time interval T and are collected during a predetermined period T_{period}. The data are sent via Internet to the Smart Contract that computes the penalties and the rewards to be assigned to the users for the considered time period T_{period}.

The implementation of a RasPi on the hardware side allows improving the efficiency of air conditioning systems: the temperature and humidity

sensors detect the environmental condition at time intervals T; basic rules for exceeding threshold intervals trigger the switching on or off of air conditioning systems.

If the air conditioning and infrastructure systems are already equipped with IoT devices connected to the network, then it is possible to avoid the local coordination and data collection systems like the RasPi. A remote data collection system can be envisaged utilizing APIs. In this case, the role of RasPi can be performed by a remote virtual server. However, the RasPi acts as a centralized hub of the air conditioning system, allowing the triggering of more advanced logic than those that can be implemented on individual appliances.

Smart Contract Architecture

RasPi locally collects (directly or via API) and preprocesses the data and ensures that they are stored into a set of variables specifically designed in the Smart Contract. More specifically, the Smart Contract stores the values related to the operation of the air conditioning systems along with environmental data, such as the indoor and outdoor temperatures, coming from the IoT devices. Being the RasPi based on Linux OS, a command-line based Ethereum wallet is installed to interact with the Smart Contract.

At the end of each billing period T_{period}, collected data are preprocessed and the total consumption for each user along with average temperatures is delivered to the Smart Contract and, subsequently made available to the optimization module.

Figure 5.2 depicts the steps that are carried out in the Smart Contract. Initially, in order to allow payments once the invoices are generated, the user's wallet must authorize the implemented Smart Contract to spend $DAI on its behalf. Then, once a new billing cycle begins, the Smart Contract receives and stores data preprocessed by the local units. If the user is in a public Blockchain context and has to pay a fee for each transaction, he/she can choose to store only the data *hashes*. On the contrary, in a private Blockchain context, the whole data can be stored at no charge.

After choosing the storage mode, the local unit starts broadcasting a set of transactions to the Smart Contract containing all data. This process continues until all data of the current billing cycle have been delivered and the last block has been reached.

Having collected all data of the current billing cycle, the last step consist in determining the rewards and penalties on the basis of the users behavior. The data integrity are verified by the optimization module that applies the reward and penality scheme. The optimization module writes back to

the Smart Contract the final amounts to invoice to each user and process payments.

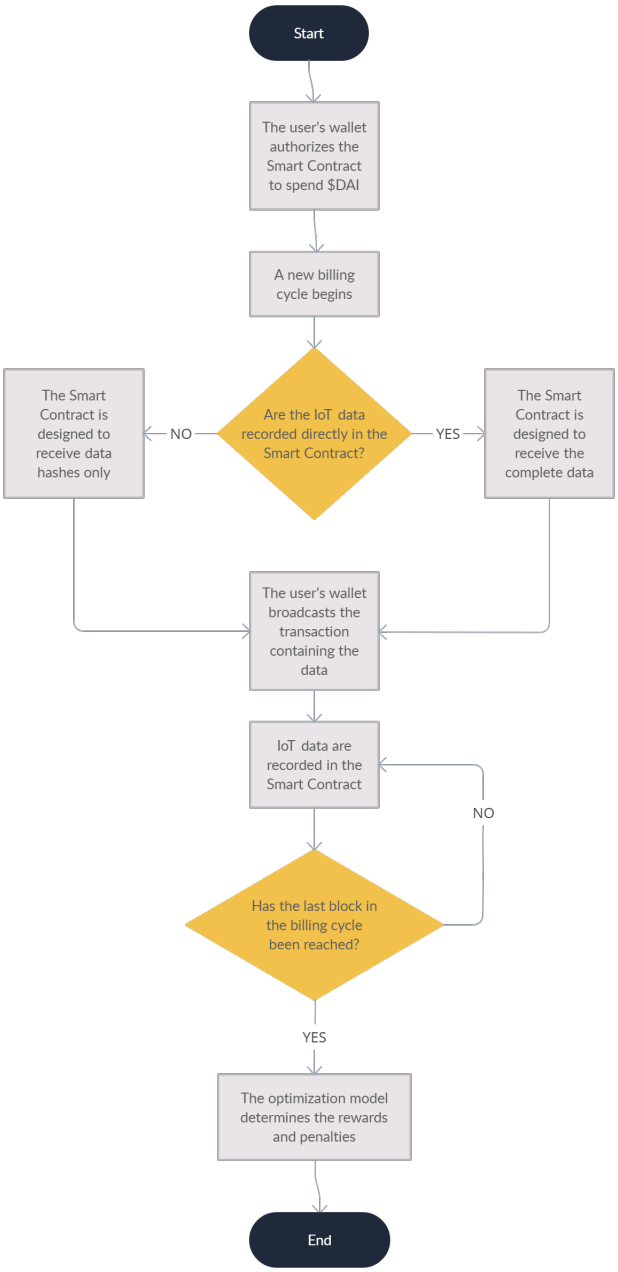


Figure 5.2: Smart Contract Flow Chart

5.4 The Reward and Penalty Scheme

The proposed reward and penalty scheme consists of the following three-steps procedure:

1. The set of m users $U = \{i | i = 1, 2, .., m\}$ are classified in n *Consumption classes* belonging to a set $C = \{j | j = 1, ..., n\}$. To this aim the k-means classifier [96] and the data recorded in the Smart Contract are used.

2. A target $\hat{k}_j \in [\hat{k}_{min}, \hat{k}_{min}]$ multiplier coefficient is assigned to each class $j \in C$. The purpose of the calculated $\hat{k}_j$ values is to apply discount (if $\hat{k}_j > 1$) or penalties ($\hat{k}_j \leq 1$) to the users on the basis on their consumption profiles.

3. The values of $\hat{k}_j$ for $j = 1, ..., m$ are revised in order to guarantee that the values of the assigned rewards do not exceed the values of the penalties with an economic loss of the energy provider. The computation of the new values of k_j for $j = 1, ..., m$ are obtained by solving a simple optimization problem.

The k-means Based Users Classification

The first step of the proposed reward and penalty procedure consists of classifying the n users belonging to the set $U = \{1, 2, .., m\}$ in the n *Consumption classes*. We assume that n users are registered in the system and some data are recorded in the Smart Contract for each user. Then, a feature vector v_i with $i = 1, ..., m$ is associated to each user $i \in U$. The set of feature vectors is denoted by $V = \{v_1, v_2, v_i, \ldots, v_m\}$ and is used to partition the user set by the k-means classifier. The components of each v_i vector are related to the consumption and behavior of the users in the considered time frame.

Furthermore, the behavior profile is driven by the level of compliance of the user to the law and to the best practices in terms of energy saving and environment preservation. For the considered time period T_{period}, the following data are recorded in the Smart Contract for each user along with the consumption: the average indoor temperature $\overline{T_i}$, the average outdoor temperature $\overline{T_e}$, the average seasonal outdoor temperature $\overline{T_s}$, the seasonal indoor temperature threshold enforced by the law $\overline{T_l}$ and the total number of hours $HoursOp_i$ of air conditioning system operation.

Now, we define the generic vector $v_i = [c_i, \alpha_{1i}, \alpha_{2i}, \alpha_{3i}, \alpha_{4i}]^T$ where:

- c_i is the total consumption in kWh;
- $\alpha_{1i} = \overline{T_i}/\overline{T_e}$;
- $\alpha_{2i} = \overline{T_i}/\overline{T_s}$;
- $\alpha_{3i} = \overline{T_i}/\overline{T_l}$;
- $\alpha_{4i} = HoursOp_i/HoursTot$.

Here, $\alpha_{1i}, \alpha_{2i}, \alpha_{3i}$ represent the average indoor temperature in the considered time frame compared to the average outdoor temperature in the same time interval, the seasonal average and the threshold enforced by the law, respectively. In addition, α_{4i} considers the level of operation of the air conditioning system by comparing the total numbers of hour of operation to the total number of hours $HoursTot$ in the same time frame.

Now, the purpose of the k-means algorithm is to assign each vector to the cluster with the nearest mean, e.g., the least squared Euclidean distance. The process is iterative and converges when the assignments no longer change. At the end of the procedure the set of users is partitioned in n disjoints set. To this aim the following parameter is defined:

$y_{ij} = 1$ if user i belongs to the class j;

$y_{ij} = 0$ otherwise.

If user i belongs to the class j then the multiplier $\hat{k}_j$ is associated to the user with the related remuneration or penalty, determined for each class according to the governance policy.

Target $\hat{k}$ Coefficient Assignment

If we consider a unitary price p per kWh in the billing period T_{period}, the total billed energy cost for each user is $EB_i = c_i p$. The purpose of the second step of the proposed system is to assign a desired $\hat{k_1}, \ldots, \hat{k_j}, \ldots, \hat{k_n}$ to each of the classes determined in the previous step. More precisely, k_j is then used to calculate a new $EB'_i = EB_i \hat{k}_j$ where $\hat{k}_j$ is the target coefficient chosen for the class assigned to user i.

If $\hat{k_j} > 1$ the behavior of the users in the related class is considered not compliant and, therefore, each user is charged an increased amount proportionally to its consumption. On the contrary, if $\hat{k_j} \leq 1$ the behavior is considered normal or virtuous and, consequently, a discount applies for all users in the relative class.

As mentioned above, the choice of assigning a certain value of k_j to a class is strictly related to the governance policy and to external factors

such as the current season, the energy saving, law compliant enforcement and environment preservation. This analysis is driven by the values of the feature vectors v_i in each class.

For example, in winter, a high consumption value c_i along with high values of coefficients α_{1i}, α_{2i} and α_{3i} and a low value of α_{4i} denotes a very bad behavior of the relative user in the sense that the system operates for a few hours with too high temperatures and considerably over the threshold enforced by law and, therefore, it is very inefficient. Similarly, in summer, when the average indoor temperature is less than the corresponding outdoor temperature, high values for c_i and low values for the other coefficients indicate the same bad condition.

Conversely, a virtuous behavior is denoted, in general, by low values of c_i along with a full operation condition, e.g. $\alpha_{4i} \approx 1$ and $\alpha_{1i}, \alpha_{2i} \leq 1.7$ or ≥ 0.7 in winter and summer respectively. The same thresholds can be considered for α_{3i} although it should be more strict given that they represent a compliance to the law.

Finally, mixed cases can also be considered. For example, in winter, $\alpha_{2i} \gg \alpha_{3i}$ denotes an exceptional case in which, in the considered time frame, the average outdoor temperature is much less than the seasonal average and, consequently, an increased value of c_i should not be penalized.

Optimal Values of Target Coefficients

In order to collect all the amount of money to pay the energy market operator, the sum of the recalculated energy costs for all users must be equal to the sum of the original costs.

Due to the heterogeneity of users consumption, it is not possible to use directly the target $\hat{k}_j$ values, but it is necessary to calculate the new k_j values such that the penalties compensate the rewards, while each k_j value is as close as possible to the corresponding target $\hat{k}_j$ value in order to comply to the governance policy.

The decision variables are $k_1, \ldots, k_j, \ldots, k_n$ and the optimal values can be determined by solving the following optimization problem:

$$\min \sum_{j=1}^{n} (k_j - \hat{k_j})^2 \tag{5.1}$$

s.t.

$$\sum_{i=1}^{m} EB_i = \sum_{i=1}^{m} \sum_{j=1}^{n} EB_i y_{ij} k_j \tag{5.2}$$

$$\hat{k}_{min} \leq k_j \leq \hat{k}_{max} \quad j = 1, 2, \ldots, n \tag{5.3}$$

Equation (5.1) is the objective function to be minimized: it is the sum of the squared differences of the k_j values with respect to the corresponding decided target $\hat{k}_j$ values. Constraint (5.2) is the *balance* constraint and ensures that the sum of the recalculated costs is equal to the sum of the original costs.

Finally, constraints (5.3) impose that the values of k_j are in the interval of values $\hat{k}_{min}$ and $\hat{k}_{max}$. Such constraints are important to comply to the governance policy and prevent both excessive penalties and discounts.

5.5 Case Study

In order to test the feasibility and the performance of the proposed approach, we consider the case study of a district of $m = 2000$ users. We implement the Blockchain infrastructure in a simulated Ethereum environment based on *Ganache* and we write random sensor data in the Smart Contract for all users considering a time period of one week in the month of June, in Italy.

We set $\overline{T_s} = 23°C$, which is the average temperature in June and $\overline{T_l} = 26°C$, that is considered as the minimum temperatures prescribed by the law for public offices and institutions to save energy. For all users, we consider a range of weekly consumption between 40 kWh and 180 kWh, a daily continuous operation between 2 and 21 hours, an outdoor average temperature between $23°C$ and $31°C$, and an indoor average temperature between $18°C$ and $30°C$.

We decide to split the users in $n = 5$ *Consumption classes* and run the first-step of our approach, e.g., the k-means algorithm, accordingly. At the end of the classification step, we analyze the average data of the resulting classes and we identify five different behaviors.

The paritioned users set is depicted in Table 5.1.

Table 5.1: Partioned Users Set

				Temperature [$°C$]		
	Class	No.	kWh	Indoor	Outdoor	Daily Usage [Hrs]
1	Small	501	49.72	27.00	27.00	4.01
2	Good	317	93.56	24.00	27.00	14.76
3	Normal	291	113.18	23.56	27.50	15.48
4	Nearly Bad	516	134.22	22.53	28.00	14.71
5	Bad	375	154.47	21.17	27.87	12.10
	Total	2000				

We observe that users in class 1 have a relative low consumption along with a low daily usage and the same average indoor and outdoor temperatures. On the basis of this data, we classify those users as *Small*.

Users in class 2 and 3 have higher consumptions and daily usages than those in class 1 and they set their indoor thermostat to an average value of $24°C$ and $23.56°C$, respectively. Although the indoor temperatures of users in both class 2 and 3 are lower than the respective average outdoor temperatures and the threshold prescribed by the law, we have to consider that the average outdoor temperature is higher than the seasonal reference

and that indicates the need to cool down the buildings. As a result, we classify those users as *Good* and *Normal* respectively. Users in class 2 are considered *Good* due to the slightly higher indoor temperature and the consequential lower consumption compared to class 3.

Moreover, due to the lower indoor temperatures found for users in class 4 and 5 compared to both outdoor average temperature and the threshold prescribed by the law and the relative high consumptions, we classify the last two classes as *Nearly Bad* and *Bad* respectively.

In order to visualize the partitions, we apply Principal Component Analysis (PCA) [97] to the classified users set by projecting the data on the first two principal components as shown in Fig. 5.3. We observe that, as a consequence of the k-means algorithm, all similar users are grouped together and the boundaries of different regions are clearly visible. Moreover, it is shown that in some classes the users are more similar than in others. For example, points relative to users in *Small* class are more concentrated compared to those of the other classes.

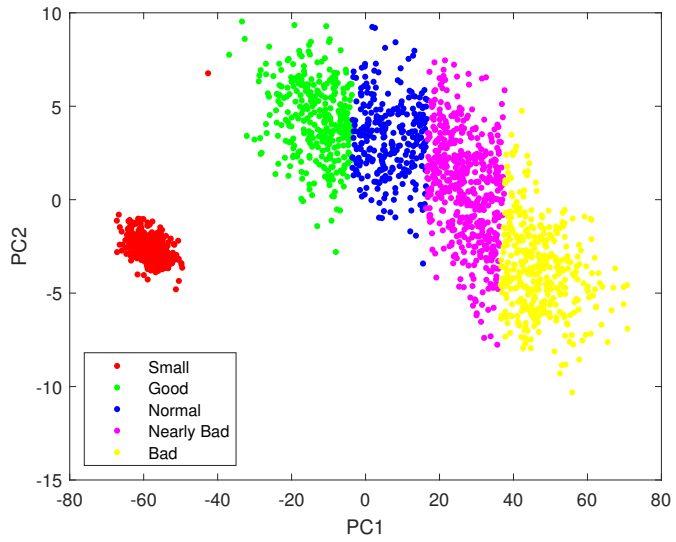


Figure 5.3: Partitioned users set projected on first two components after PCA reduction

At this point, we assign a $\hat{k_j}$ target coefficient to each of the classes. As shown in Table 5.2, we decide to apply a discount to the first three classes and a penalization to the remaining two classes. Now, if we set p=\$0.20, the total billed energy cost for all users, on the basis of on their consumptions, is \$42.940. If we apply the assigned $\hat{k_j}$ target coefficient to

each user's consumption, the total billed energy cost results in \$41.298, that is not enough to pay the energy provider.

Then, we solve the optimization problem described in Section 5.4 to calculate the final k_j coefficients. The problem is solved by the Matlab *fmincon* solver [98], which in this case converges to an acceptable local minimum in 15 iterations. The final k_j coefficients are shown in Table 5.2 and they exhibit a percentage relative error less than 10% compared with the corresponding values of $\hat{k_j}$.

Table 5.2: Optimized k_j coefficients vs. $\hat{k_j}$ target coefficients

	Class	$\hat{k_j}$	k_j	Relative error [%]
1	Small	0.90	0.93	3.3%
2	Good	0.70	0.74	5.7%
3	Normal	0.85	0.93	9.4%
4	Nearly Bad	1.10	1.13	2.7%
5	Bad	1.20	1.20	0%

Now, the total energy cost for all users with the new coefficients applied is again \$42.940, that is the amount required to pay the energy provider. At the end of the procedure, the final k_j coefficients are written to the Smart Contract to invoice the users and process the payments.

5.6 Conclusion

In this work, a district energy management approach based on IoT and Blockchains is investigated. The proposed architecture is based on a decentralized Ethereum Blockchain and a local set of sensors connected to a Raspberry Pi device for each user that acts both as data collector and Blockchain client. Each device delivers its data to the Blockchain through a Smart Contract. Thanks to the features of the Blockchain, all the data stored in the Smart Contract are safe and immutable.

Having collected the data, the purpose of this work is to classify the users to reward or penalize them on the basis of the analysis of the collected data. In particular, specific attention is given to the air conditioning system regulation to reward virtuous users and penalize not virtuous ones.

Note that in the proposed system, the Blockchain plays an active role, not just being a means of notarization but managing the district's payments. Moreover, the user that receives a penalty can directly check the reasons of the penalties, compared with the other users of the district and correct his behaviour in the next billing cycle.

The feasibility and the performance of the proposed approach is investigated in a case study. In future work, we plan to extend the number of features used for classification and study the scalability of the proposed approach over the number of considered district users.

Part II

Swarm Algorithms in Manufacturing

6 Flexible Job Shop Sequencing Problem with TCPN and PSO

6.1 Introduction

This chapter explores the use of Timed Coloured Petri Nets (TCPN) in mass production system modelling. The objective of the work is to solve the well known Flexible Job Shop Scheduling problem (FJSSP) to determine the production sequence that maximizes the throughput. Since the FJSSP problem is NP-hard, a method based on Particle Swarm Optimization is suggested. Case study results show that the method is effective and converges after a few iterations. PSO has been chosen as it can be implemented in parallel fashion in distributed systems thus further reducing convergence time.

The increasing demand of diversified products in the market poses major challenges to producers. On the one hand, in order to be competitive, each company is required to offer the widest possible product range. On the other hand, that has to be achieved with as few machines as possible to optimize spaces and costs.

In many industries, the differences among products consist in raw materials and tools selection throughout the production. Hence, modern Computerized Numerical Control (CNC) machines have the ability to work with different materials and they are often equipped with different tools. As a result, we are in the age of smart and flexible factory, where multiple diversified jobs can be produced in a single plant made of a few connected machines.

However, the optimization of job scheduling in such field, that is known in literature as Flexible Job Shop Scheduling Problem (FJSSP), is one of the most difficult combinatorial optimization problems [99] and it is NP-

hard in general [100]. Some of the typical objectives of that problem are the throughput maximization and the consequent minimization of job and machine waiting time. In addition, the scheduling has to be responsive against external disturbances such as failures or interruptions of the supply chain, allowing an immediate reconfiguration based on the actual conditions.

Due to the usual non-linearity of FJSSP, meta-heuristics based methods have been proven to be a very efficient approach to this scheduling problem in recent years. Among them, evolution-based, such as genetic algorithm (GA), and swarm-based, such as Particle Swarm Optimization (PSO), dominate the research area. However, regardless of the specific technique, the choice of a suitable modeling and simulation algorithm is essential to represent a specific plant throughout the optimization process.

This work considers manufacturing systems for the mass production of different types of jobs through a sequence of operations performed by a set of flexible working machines. Moreover, each machine can perform simultaneously multiple operations on different job types. The problem is determining the optimal sequence of job types and the number of jobs for each job type to be processed, in order to maximize the throughput of the system.

First, the production system is described in a framework of Timed Coloured Petri Nets (TCPN) [101] to model flexible CNC machines enabled to simultaneously work with multiple materials and multiple tools. TCPN is a variant of classic Coloured Petri Nets (CPN) in which time is also associated to tokens. Secondly, we implement a simulation algorithm for computing the system throughput by the TCPN model. Finally, we apply the PSO by the TCPN simulation to maximize the system throughput by optimizing the job type sequencing and the amount of units of each job type that enters the system.

A case study describing and simulating a real manufacturing system producing ophthalmic lenses is studied to demonstrate the efficiency of the proposed method.

The new contribution of the work consists of the complementary use of the TCPN to model and simulate the production system and the implementation of the PSO for solving this complex sequencing problem.

The remainder of this chapter is organized as follows. In Section 6.2, we provide some basic concepts about Petri Nets and TCPN. In Sections 6.3 and 6.4, we introduce the TCPN-based modeling technique and the related PSO optimization respectively. Section 6.5 delivers some experimental results. Finally, Section 6.6 concludes the chapter.

6.2 Basics of Timed Coloured Petri Nets

A Petri Net (PN) is a bipartite graph described by non-empty sets of places P, transitions T and arcs with associated weights. A place $p \in P$ is called an input place of a transition $t \in T$ if there exists a directed arc from p to t. On the contrary, a place p is called an output place of a transition t if there exists a directed arc from t to p. A transition t is called enabled if each of its input places contains at least a number of tokens equal to the corresponding arcs weights. An enabled transition can fire. Firing a transition means consuming tokens from the input place and producing tokens for the output places according to the corresponding arcs weights.

Diversely from generic PN, in a Coloured Petri Net (CPN), places have a color set (a data type) associated with them that specifies the set of allowed token colors at this place. Each token in a place of a CPN has an attached data value (color) to it that matches the corresponding color set of the place. In a Timed Coloured Petri Net (TCPN) a timestamp is added to each token indicating the time at which they will be available for the firing of transitions. The system model includes a global clock that represents the total elapsed time.

Based on the formalism proposed by [101] and [102], a TCPN is defined as a tuple $(P, T, A, \Sigma, V, G, E)$ where:

- P is a finite set of places
- T is a finite set of transitions such that $P \cap T = \emptyset$
- A is a set of directed arcs where $A \subseteq P \times T \cup T \times P$
- Σ is a finite set of non-empty color sets
- V is a finite set of typed variables in Σ, i.e., $Type(v) \in \Sigma$, for all $v \in V$.
- G is the *guard function* where $G : T \rightarrow EXPR_V$, which assigns a Boolean expression to each transition, i.e. $Type(G(t))=Bool$, where $Bool \in \{enabled, disabled\}$
- $E : A \rightarrow EXPR_V$ is the *arc expression function*, which assigns an expression to each arc that determines the color set of the generated tokens in the output places when a transition is fired.

In this definition, $EXPR_V$ denotes the expressions constructed using the variables in V according to a given syntax.

The state of the TCPN is represented by a marking $\mathcal{M}$ that is a mapping defined as $\mathcal{M} : P \rightarrow \mathcal{B}(\Sigma)$, i.e., $\mathcal{M}(p)$ is a set of elements of Σ, also with repeated elements (i.e., a multiset of Σ denoted by $\mathcal{B}(\Sigma)$) corresponding to the token colors in place p.

The dynamics of a TCPN is regulated by guard functions and arc expressions. In particular, given an input place p with a certain number of tokens and a transition t, t is set to enabled by G only if certain preconditions on its current tokens colors are met and the corresponding tokens in the output places are generated according to the function E. When a transition is enabled, firing can only occur once the timestamp attached to each token has been reached.

6.3 TCPN Model of a Manufacturing Plant

In this section, we describe the TCPN model of a production system consisting of CNC machines connected in a production line and capable of working, at the same time, a finite number of jobs. The jobs are of different types characterized by different raw material and by the required different processing tools.

Production System Model

Let us consider a set of N CNC machines $M = \{M_1, M_2, M_n, \ldots, M_N\}$, connected in a line production system. Let us also consider a set of features $F = \{f_1, f_2, f_i, \ldots, f_{|F|}\}$, where each feature f_i can represent a raw material or a tool in the CNC machine devoted to a particular operation. Note that symbol $|A|$ denotes the cardinality of the generic set A.

Figure 6.1 shows an example of the TCPN model of a system composed by $N = 3$ CNC machines. Each CNC machine n is modeled by a TCPN with three places $p_{n,i}$ with $i = 1, 2, 3$ and two transitions. More precisely, place $p_{n,1}$ is the jobs input buffer where jobs wait to be processed, $p_{n,2}$ is the storage area that provides a limited set of processing tools, $p_{n,3}$ is the working cell in which jobs are processed.

Each machine is modelled also by two transitions $t_{n,i}$ with $i = 1, 2$: $t_{n,1}$ denotes the beginning of the operation of the job in the CNC machine n and $t_{n,2}$ denotes the end of the operation and the entering in the subsequent machine. In addition, t_{start} is a source transition that feeds the system with a new job by firing at predetermined time intervals and p_{end} is the place collecting the completed jobs.

In the model, a token can be either a feature (i.e., a tool or a raw material) or a job depending whether the token is in a storage area or in a buffer. Moreover, let be $J = \{J_1, J_2, \ldots, J_i, \ldots, J_m\}$ the job type set and each job type $J_i \in J$ is described by a tuple $F_{\alpha i}$ of distinct elements of F, characterizing specific products to be machined. We assume that the routing of the job types is fixed and the same for all the jobs.

Each machine can use simultaneously more tools with different predefined processing times and each tool can process only one job at a time. Furthermore, we consider that the job type has an attached timestamp ts and therefore is described by $J_i =< F_{\alpha i}, ts >$.

In the considered system, the features are described by two sets: the set $R_c = \{r_c, r_{c1}, \ldots, r_{ci} \ldots, r_{c|R_c|}\}$ of the raw materials where $r_{ci} \in R$ with $i = 1, .., |Rc|$ are the raw materials and r_c is a neutral material, and the

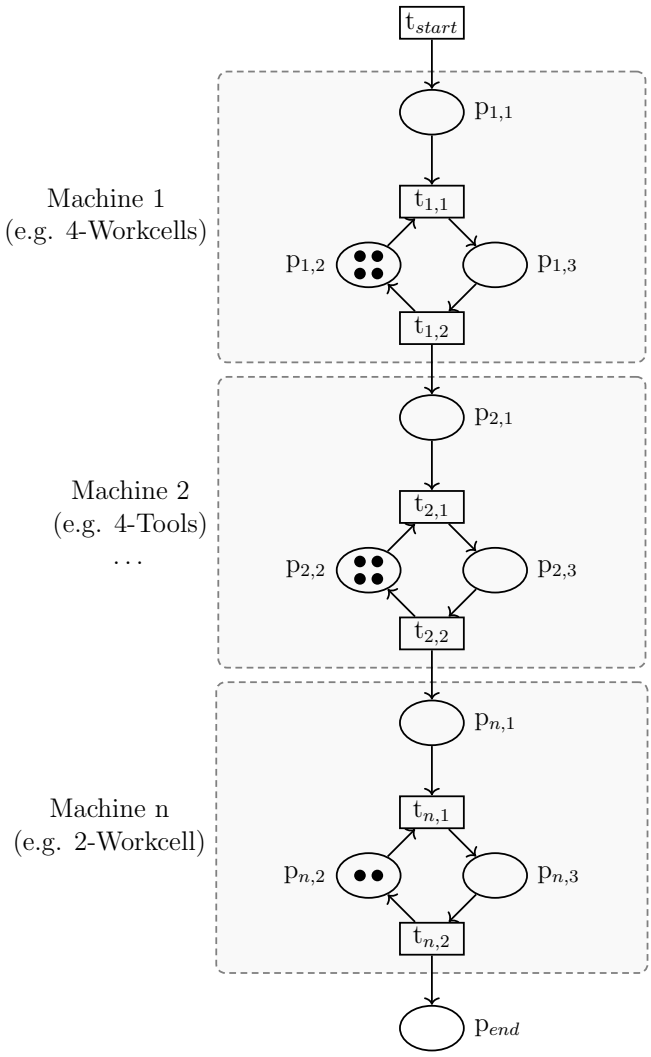


Figure 6.1: Model of a line production system with 3 CNC machines

set $S_c = \{s_{c1}, \ldots, s_{ci}, \ldots, s_{c|S_c|}\}$ of the available tools s_{ci} with $i = 1, .., |S_c|$. Then, it holds $F = R_c \cup S_c$. In this case, each job type $J_i \in J$ is described by a couple of features: a raw material $r_{ci} \in R_c \setminus \{r_c\}$ and the associated tool $s_{ci} \in S$. Then the colors of the $J_i \in J$ are $F_{\alpha i} \in J_c = R_c \setminus \{r_c\} \times S_c$. The sets of features and the set of the couples of features J_c denote the color sets associated to the token of the described TCPN, i.e., $\Sigma = R_c \cup S_c \cup J_c$.

The guard function G is described by Algorithm 3 and the arc expression E is described by Algorithm 4.

Algorithm 3 Guard function G

Require: Transition $t \in T$, job $J_i =< F_{\alpha i}, ts >$
Ensure: t_{state}
  $P_t \leftarrow$ Set of input places of t **foreach** $p \in P_t$ **do**
    if $\mathcal{M}(p) \in \mathcal{B}(S_c \cup \{r_c\})$ **then**
    $F_p \leftarrow \mathcal{M}(p)$ **if** $F_p \cap F_{\alpha i} = \emptyset \wedge r_c \notin F_p$ **then**
      $t_{state} \leftarrow$ *disabled*
      return else
        end
        $t_{state} \leftarrow$ *enabled*
      end
    end
  end

Algorithm 4 Arc Expression function E

Require: Transition $t \in T$, job $J_i =< F_{\alpha i}, ts >$
  $P_t \leftarrow$ Set of output places of t **foreach** $p \in P_t$ **do**
    if $\mathcal{M}(p) \in \mathcal{B}(\{r_c\})$ **then**
    Add r_c to $\mathcal{M}(p)$ **else if** $\mathcal{M}(p) \in \mathcal{B}(S_c)$ **then**
      end
      $s_c \leftarrow F_{\alpha i} \cap S_c$
    Add s_c to $\mathcal{M}(p)$ **else**
      end
      Add J_i to $\mathcal{M}(p)$
    end
  end

Now, the marking of the places are characterized as follows:

- $\mathcal{M}(p_{n,1}) \in \mathcal{B}(J_c)$
- $\mathcal{M}(p_{n,2}) \in \mathcal{B}(S_c \cup \{r_c\})$
- $\mathcal{M}(p_{n,3}) \in \mathcal{B}(J_c)$.

In particular, if transition t_{start} fires, then a new job of type J_i enters the input buffer $p_{1,1}$. Moreover, as a job token enters a generic input buffer place $p_{n,1}$, the guard function G of $t_{n,1}$ checks the marking of the storage place $p_{n,2}$ for raw materials or tool tokens matching the features in $F_{\alpha i}$ and sets the t_{state} of transition $t_{n,1}$ accordingly, where $t_{state} \in \{enabled, disabled\}$. Note that if $\mathcal{M}(p_{n,2}) \in \mathcal{B}(\{r_c\})$, then $p_{n,2}$ matches with every raw material in $F_{\alpha i}$.

If $t_{n,1}$ is enabled and the timestamp ts of the job has been reached, the $t_{n,1}$ fires and, according to the arc expression function E, the job token

is moved from $p_{n,1}$ to the working cell $p_{n,3}$, while a corresponding feature token is removed from $p_{n,2}$ to indicate that the resource is temporarily not available.

In addition, when the job enters the working cell, its timestamp is updated according to a function $TS(p_{n,3}, J_i)$ that, given a job J_i and a working place $p_{n,3}$, returns the processing time required in that place for the corresponding feature. Once the new timestamp has been reached and the processing has been completed in $p_{n,3}$, the transition $t_{n,2}$ is enabled and fires. Then, according to arc expression function E, the job is moved to the input buffer of the subsequent machine where the process starts again and proceeds until the job reaches the final output buffer place p_{end}.

To complete the description of the dynamics, there are two final considerations:

- the initial markings $\mathcal{M}_0(p_{n,2})$ of places $p_{n,2}$ with $n = 1, ..., N$ indicate the available features of machine n and the corresponding maximum number of jobs that can be simultaneously processed;

- if the initial marking of a storage place is $\mathcal{M}(p_{n,2}) \in \mathcal{B}(\{r_c\})$ then it allows each token color $F_{\alpha i}$ to enable transition $t_{n,1}$, on the contrary if $\mathcal{M}(p_{n,2}) \in \mathcal{B}(J_c)$ then the corresponding color in marking $\mathcal{M}(p_{n,2})$ can enable transition $t_{n,1}$. By this way, we represent the capability of a CNC machine of processing all raw materials with different times determined by the TS function and the capability of the tools of working only particular jobs.

6.4 Job Sequencing by Particle Swarm Optimization

In this section, we define the job sequencing problem and we propose the optimization approach based on the PSO applied to the CTPN simulation.

Throughput Maximization Problem

We assume that in the considered manufacturing system, the daily production is continuous and consists of a predetermined repeated sequence of job types.

The objective of the optimization problem is to find the sequence of the job types and the number of the jobs pertaining to each job type that have to enter the system to maximize the output throughput TH. Denoting by Ω the set of the possible permutations of the elements in J, we define the following decision variables:

- $\omega_h \in \Omega$ is a permutation of the elements of J that indicates the sequence in which the job types enter the production system;
- δ_i is the number of jobs of type $J_i \in J$ for $i = 1, ..., m$ in the sequence ω_h.

We assume that for each $i = 1, ..., m$, the number of jobs δ_i is in an interval $[\delta_{mini}, \delta_{maxi}]$ with $\delta_{mini} > 0$.

The problem is formulated as follows:

$$TH_{MAX} = \max_{X} TH(X) \tag{6.1}$$

with $X = \{\omega_h, \delta_1, \ldots, \delta_i, \ldots, \delta_m\}$, $\omega_h \in \Omega$ and $0 < \delta_{mini} \leq \delta_i \leq \delta_{maxi}$.

Different techniques are applied in the related literature for optimizing objective functions obtained by simulations [103]. Since such strategies consist of searching variations iteratively in the domain of decision variables, this approach requires a connection between the optimization algorithm and the simulation model [104]. In the considered application context, we implement a PSO technique because the search space has a straightforward representation using PSO particles, thus avoiding complex encoding and decoding operations. Moreover, the evaluation of the objective function requires a limited number of simulations that are typically time consuming.

TCPN Simulation Algorithm

The throughput of the system is computed by the simulation Algorithm 5 that implements the dynamic behaviour of the $TCPN$. Firstly, at each valid timestamp as determined by the given time interval, the source transition t_{start} feeds the system with a new job in $p_{1,1}$ according to the current sequence. Secondly, for each outbound transition of each place, the corresponding guard function G is evaluated against all jobs to check whether the transition has to be enabled. Thirdly, for each enabled transition, in case the timestamp has been reached for a job, a fire event occurs and the corresponding arc expression E is executed that moves the job to next place and eventually restores the tool or the raw material token. Finally, the job timestamp is updated in the new place according to the TS function.

Algorithm 5 Simulation Function

Require: Jobs sequence X
Ensure: Throughput TH
$t_{max} \leftarrow 3600$
$\tau \leftarrow$ time units interval for new jobs (e.g. 5s)
$TT \leftarrow$ empty array for job timestamps
$t \leftarrow 0$ **while** $t \leq t_{max}$ **do**
if $t \bmod \tau = 0$ **then**
Add next job in sequence to $\mathcal{M}(p_{11})$
Add $t + \tau$ value to array TT
end
foreach $p_i \in P$ **do**
$T_{out} \leftarrow$ outbound transitions from p_i **foreach** *Job* $J_i \in \mathcal{M}(p_i)$ **do**
foreach $T_j \in T_{out}$ **do**
T_j enabled $\leftarrow G(J_i, T_j)$ **if** T_j *is enabled* **then**
if *job timestamp* $ts \leq t$ **then**
Fire transition T_j for job J_i
Execute $E(J_i, T_j)$
Take new p_k place of job j_i
Update J_i timestamp
$ts \mathrel{+}= TS(P_k, J_i)$
Add new ts of J_i to array TT
end
end
end
end
end
$TH \leftarrow$ number of tokens in last place p_{end}
Sort TT array
Remove from TT array all values $\leq t$ **if** TT *has remaining values* **then**
Set next $t \leftarrow TT\{0\}$ **else**
end
return
end
end

Particle Swarm Optimization

This subsection specifies the application of the PSO algorithm. PSO is a stochastic meta-heuristic optimization algorithm, which simulates the flocking behavior of birds [103].

In the PSO a number of simple entities, called particles, is used for optimization purposes: the particles represent candidate solutions with respect to the problem being optimized. In particular, each particle of the swarm is composed of three D-dimensional vectors, where D is the dimension of the search space: the current position x_i, the previous its best position p_i, and the velocity v_i. The particles are placed in the search space of some problem or function, and each of them evaluates the objective function at its current location. Each particle then determines its movement through the search space by combining some aspect of the history of its own current and best (best-fitness) locations with those of one or more members of the swarm, with some random perturbations. The next iteration takes place after all particles have been moved [105].

In detail, the current position x_i can be considered as a set of coordinates describing a point in the space and is evaluated as a possible problem solution. If such position results to be better than the previous ones, then its coordinates are stored in the vector p_i. The value of the resulted best function is stored in a variable called previous best $pbest_i$, for comparison on the later iterations. The objective of each particle is to find better positions and update p_i and $pbest_i$ vectors. For this reason, the algorithm iteratively updates the velocity vector v_i of each particle and calculates new positions x_i, also considering the best location of all particles ($gbest$), in accordance with the following two-update equations:

$$\begin{aligned} v_i(k+1) = w \cdot v_i(k) + c_1 \cdot r_1^{(k)} \cdot [pbest_i(k) - x_i(k)] + \\ \ldots + c_2 \cdot r_2^{(k)} \cdot [gbest(k) - x_i(k)] \end{aligned} \tag{6.2}$$

$$x_i(k+1) = x_i(k) + v_i(k+1) \tag{6.3}$$

where w is the inertia weight, k is the iteration number, c_1 and c_2 are respectively the cognitive and social weight, r_1 and r_2 are vectors of random numbers sampled from a uniform distribution in the range $[0, 1]$.

In the considered problem, the position $x_i(k)$ at iteration k is a vector of $m+1$ components associated to the elements of $X = \{\omega_h, \delta_1, ..., \delta_j, \ldots, \delta_m\}$. In addition, the parameters w, c_1, c_2 and the particle numbers have to be appropriately chosen, depending on the problem to be solved. In the case study we dynamically adjust the inertia w at each iteration in the range $[0.1, 1]$, we fix $c_1 = c_2 = 1.49$ and set the size of the population equal

to 30 particles [106]. Finally, the optimization process is completed if the best location *gbest* does not change for a fixed number of 10 consecutive iterations. The corresponding throughput TH is the optimal value of the objective function determined by the PSO.

6.5 Case Study

We consider as a case study an ophthalmic lenses mass production system in a firm consisting of three CNC machines: the first one is a lens-blocker with four work-cells and can process up to four jobs at the same time, regardless of the raw material. The second one is a lens-generator characterized by four different finishing tools and can process up to to four jobs simultaneously, supposed that each job requires a different tool. The third machine is a lens polisher having two work-cells that concludes the lens generation process and can process up to two jobs at a time. The production system works three raw materials: 1.5 - CR39 (r_{c1}), 1.53 - MR-8 (r_{c2}) and 1.6 Polycarbonate (r_{c3}). Moreover, four types of finishing tools are used by the second machine. The model of the production line is shown in Figure 6.1.

Hence, the set of the TCPN are the following:

$$P = \{p_{11}, p_{12}, p_{13}, p_{21}, p_{22}, p_{23}, p_{31}, p_{32}, p_{33}, p_{end}\}$$

$$T = \{t_{start}, t_{11}, t_{12}, t_{21}, t_{22}, t_{31}, t_{32}\}$$

$$R_c = \{r_c, r_{c1}, r_{c2}, r_{c3}\}$$

$$S_c = \{s_{c1}, s_{c2}, s_{c3}, s_{c4}\}$$

We set the initial marking as follows: $\mathcal{M}_0(p_{12}) = \{r_c, r_c, r_c, r_c\}$, $\mathcal{M}_0(p_{22}) = \{s_{c1}, s_{c2}, s_{c3}, s_{c4}\}$, $\mathcal{M}_0(p_{32}) = \{r_c, r_c\}$, $\mathcal{M}_0(p_{ni}) = \emptyset$ for $n = 1, 2, 3$ and $i = 1, 3$ and $\mathcal{M}_0(p_{end}) = \emptyset$.

The system has to produce seven different types of lenses, i.e, the job type set is $J = \{J_1, \ldots J_i, \ldots, J_7\}$ with $\delta_i \in [1, 10]$ for $i = 1, ..., 7$.

In Table 6.1, the color of each job type and the corresponding processing times in the places $p_{n,3}$ for $n = 1, 2, 3$ are shown.

Table 6.1: Job types and timestamp function

Job	**Color**	**Processing Time**		
		p_{13}	p_{23}	p_{33}
Job 1	$J_1 =< r_{c1}, s_{c1} >$	30	10	10
Job 2	$J_2 =< r_{c2}, s_{c2} >$	10	20	14
Job 3	$J_3 =< r_{c1}, s_{c3} >$	30	30	21
Job 4	$J_4 =< r_{c2}, s_{c1} >$	10	10	12
Job 5	$J_5 =< r_{c1}, s_{c2} >$	30	20	18
Job 6	$J_6 =< r_{c3}, s_{c4} >$	25	15	17
Job 7	$J_7 =< r_{c3}, s_{c2} >$	19	32	11

We implement the TCPN simulation and the PSO optimization in a Matlab environment. The PSO is implemented by running 100 consecutive simulations of about 150 seconds each, and a new job enters the system at 5 second intervals. Figure 6.2 and Table 6.2 show five optimized job type sequences obtained by the PSO in the corresponding simulation and their throughput values (in jobs per hour, jph). We use these sequences to feed the system through t_{start} in cyclical fashion.

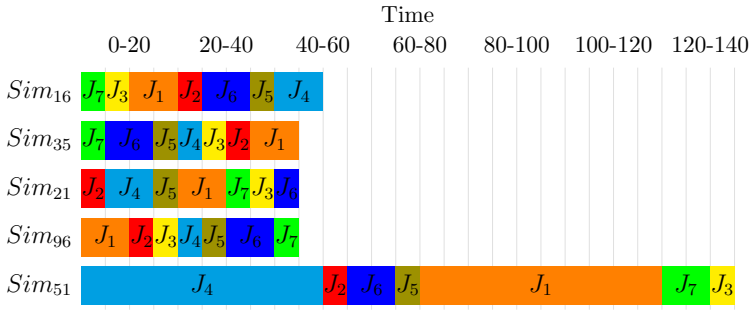


Figure 6.2: Optimized Jobs Sequences to cyclically feed the system through t_{start}

Table 6.2: Optimized job types sequences and their dimension

Sim.	ω	**Sequence**	**Job Units**							**Jph**
			J_1	J_2	J_3	J_4	J_5	J_6	J_7	
16	475	J_7,J_3,J_1,J_2,J_6,J_5,J_4	2	1	1	2	1	2	1	407
35	1	J_7,J_6,J_5,J_4,J_3,J_2,J_1	2	1	1	1	1	2	1	394
21	4028	J_2,J_4,J_5,J_1,J_7,J_3,J_6	2	1	1	2	1	1	1	393
96	5040	J_1,J_2,J_3,J_4,J_5,J_6,J_7	2	1	1	1	1	2	1	392
51	2675	J_4,J_2,J_6,J_5,J_1,J_7,J_3	10	1	1	10	1	2	1	356

In detail, we achieve the best result by simulation 16, in which the optimization process starts from 227 jph and converges to 407 jph when we feed the system cyclically with the computed sequence. Similar results are obtained by simulations 35, 21 and 96. However, the worst result is achieved in simulation 51 that converges to 356 jph. Figure 6.3 shows the throughput obtained at each PSO iteration in the five considered solutions. The best PSO algorithm execution (simulation 16) stops after 37 iterations and the 100 simulations are completed in about 4 hours on a standard *Intel Core vPro i7* processor.

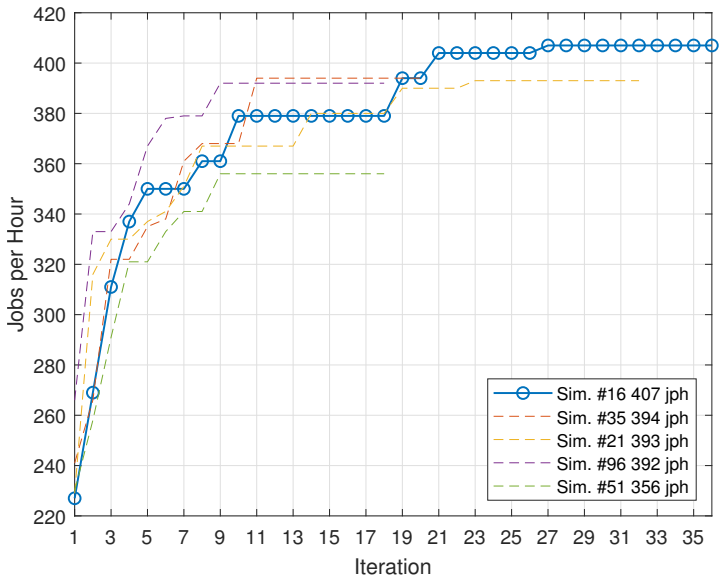


Figure 6.3: Throughput obtained at each PSO iteration optimization

In order to show the efficiency of the proposed method, we simulate the TCPN arranging the job types according to two well-known techniques: SPT (Shortest Processing Time) and LPT (Longest Processing Time) [107]. In SPT, the jobs having shorter time are processed first while the opposite occurs in LPT. For both techniques, a quantity of one job per type in the sequence is set. In detail, the throughput calculated according to LPT and SPT techniques is 329 and 330 jph, respectively. In addition, we calculate the throughput obtained using each of all the sequences in Ω and we get a mean value of 317 jph. On the other hand, we calculate the mean throughput obtained by the 100 simulations with the application of the PSO algorithm we obtain 389 jph.

Summing up, Figure 6.4 shows that by using the best solution of the proposed method the firm is able to increase the throughput of about 23% (i.e., 77 jobs) per hour with respect to the LPT and SPT techniques.

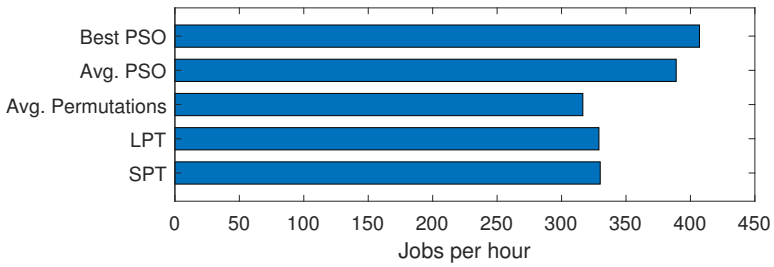


Figure 6.4: Throughput comparison between PSO and other methods

6.6 Conclusion

This work proposes a meta-heuristic approach based on Timed Coloured Petri Nets (TCPN) and Particle Swarm Optimization (PSO) to maximize the throughput of jobs in a production system by computing the optimal sequence of job types and the amount of units of each type. The computed sequence represents the order by which the jobs enter the system in a cyclical way.

A model in a TCPN framework describes the production system including CNC machines. Each machine works by a certain number of tools and is able to process a limited set of raw materials with different operation times. Given a set of job types to be produced in a determined sequence, we introduce a simulation algorithm that calculates the corresponding throughput. In addition, by the simulation algorithm the PSO is implemented in order to select the best sequence that maximizes the system throughput.

The case study of the mass production of ophthalmic lenses shows that the proposed approach enables the system to increase the throughput of about 23% in one hour. The results show the applicability of the approach with the complementary use of the TCPN model that allows simulating the system and the PSO technique that provides the optimal solutions.

Future developments concern an extended generalization of the modeling approach to include more complex job routing and larger number of CNC machines to optimize the job sequences.

Part III

Multi-Agent Systems for Autonomous Vehicles

7 A Cooperative DRL Approach for Autonomous Intersection Management

7.1 Introduction

In this chapter, the problem of autonomous intersection management at unsignalized intersections is addressed. The scenario includes not only connected and automated vehicles (CAV), but also connected vehicles driven by humans with high priority and regular vehicles. In order to have CAVs cross the intersection safely, preventing collisions and observing priorities, a Cooperative Deep Reinforcement Learning approach is suggested. Cooperative DRL has been chosen for its ability to consider the contributions of multiple agents in a distributed environment. This is particularly important in the presented scenario since the current state is only partially observable by each vehicle depending on its current position.

The constant increase of urban mobility over the past decades has improved the overall quality of life around the world. At the same time, when it comes to urban environments, various problems have been arising such as traffic congestion, collisions and pollution. As a consequence, continuous efforts are required to optimize traffic flow, reduce travel time, decrease fuel consumption and prevent collisions.

Among urban routes infrastructures, intersections represent one of the major bottlenecks of traffic flow [108]. Moreover, according to a report published by the U.S. Transportation Department, about 40% of all collisions occur at or nearby intersections [109].

A concrete chance to contribute to traffic flow optimization in critical areas, such as intersections, is being recently given by the advent of connected and automated vehicles (CAV). The capability of those vehicles

to exchange data by a Vehicle-to-Vehicle or Vehicle-to-Grid infrastructure makes it possible to implement different classes of optimization tasks and therefore schedule crossing times based on priorities, prevent collisions and ultimately improve traffic streams.

In this context, most of the literature has been focusing on optimal intersections management under full CAV environment, considering signalized and unsignalized intersections and using mostly rule-based and optimization methodologies with the support of simulators and numerical tests [110].

However, as the reality in the near future will be a mixture of CAV and regular vehicles (RV), mixed traffic scenarios have been recently receiving more attention. In this regard, in [111], the authors analyze the existing solutions for intersection management under mixed traffic environments at signalized intersections but point out that a full connected environment is often a requirement for the proposed methods to be effectively applied. Moreover, they conclude that Machine Learning (ML) approaches, thanks to their capability of continuous learning and adaption, can be an alternative reliable approach when it comes to hybrid environments in which not all vehicles are connected.

Among ML techniques, Deep Reinforcement Learning (DRL) has proved to be a powerful learning framework, capable of learning complex policies in high dimensional environments, and it has been recently employed in literature for different applications in the domain of Autonomous Driving [112, 113, 114].

More specifically, in [115], a novel approach for unsignalized intersection management is proposed that is based on a DRL-based centralized policy that optimizes traffic flow and arranges the right-of-way between vehicles. A method that combines 2D Lidar sensors observations and DRL in a game-theoretic decision-making context is proposed in [116]. The authors claim that their technique enables vehicles to make optimal decisions at unsignalized intersections without using any central coordinator or vehicle-to-vehicle infrastructure. A similar approach for complex intersection scenarios is proposed in [117], in which DRL is supported by long short-term memory (LSTM) that processes traffic images collected by a camera sensor mounted on the vehicle.

In addition, several recent works exist that employ Multi-agent DRL techniques for autonomous intersection management in a decentralized a cooperative fashion and show promising results in optimizing traffic flow, prevent collisions, decrease fuel consumption and reduce vehicles waiting time [118, 119].

However, among existing works, unsignalized intersection management in mixed traffic scenarios has received very little attention. Moreover, at the

best of our knowledge, none of the existing works consider that vehicles may have different priorities based on their class. For example, an ambulance or a police car must be given the right-of-way by all other autonomous or regular vehicles in order to safely cross the intersection.

Motivated by this gap, in this work, we introduce a novel Multi-agent Cooperative DRL approach for autonomous intersection management at unsignalized intersections under prioritized mixed traffic scenarios. In the proposed method, three different vehicle classes are considered: Connected Automated Vehicles, i.e. CAVs, that act as agents, Connected Priority Vehicles (CPVs), such as ambulances or police cars, that are connected but driven by humans and, lastly, Regular Vehicles (RVs), that are neither connected or automated.

In detail, the main contributions of this research are:

(i) A novel state representation of mixed traffic consisting of four overlapped grid layers representing the current route of each CAV or CPV connected vehicle, the speed, the distance to the intersection and, eventually, the distance from the vehicle in front, which could be even an unconnected vehicle. As a consequence, since RVs are not connected, the current traffic state is only partially observable by the agents.

(ii) A weighted global reward function that considers CAVs and CPVs current speeds at each step and applies strong penalties in case of collisions.

(iii) A Proximal Policy Optimization (PPO) [120] approach to find the optimal DRL-based policy that optimizes traffic flow by observing vehicles priorities, preventing collisions and reducing crossing times.

Compared to previous works, the main challenge addressed by the proposed DRL algorithm is to instruct the CAVs to take into account the class to which each other vehicle approaching the intersection belongs and to act accordingly.

The remainder of this chapter is organized as follows. In Section 7.2, the problem of autonomous intersection management problem at unsignalized intersections under prioritized mixed traffic scenarios is formulated and the resulting system is modeled. Section 7.3 introduces the Multi-agent Cooperative DRL approach to automate intersection management. In Section 7.4, a case study designed in SUMO [121] and trained by RLlib [122] shows the performance of the proposed method under simulated mixed traffic scenarios. Finally, Section 7.5 concludes the work and draws future research directions.

7.2 Problem Formulation

This section describes the Autonomous Intersection Management problem (AIM) addressed in this work and depicted in Fig. 7.1. In detail, three different classes of vehicles are approaching a 4-way double lane unsignalized intersection at the same time. The two red vehicles are CAVs, the blue vehicle is a CPV, for example an ambulance, whereas the yellow vehicle is a RV, which is not connected. The two CAVs wish to cross the intersection from right to left side and from left to right side respectively, while the ambulance is heading south and the RV wishes to turn left.

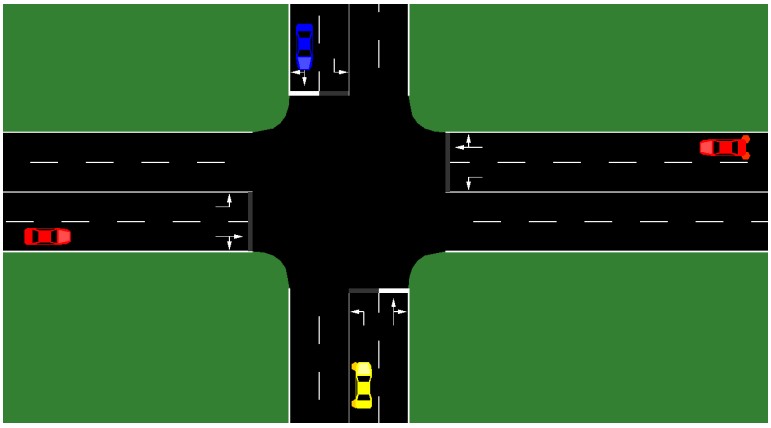
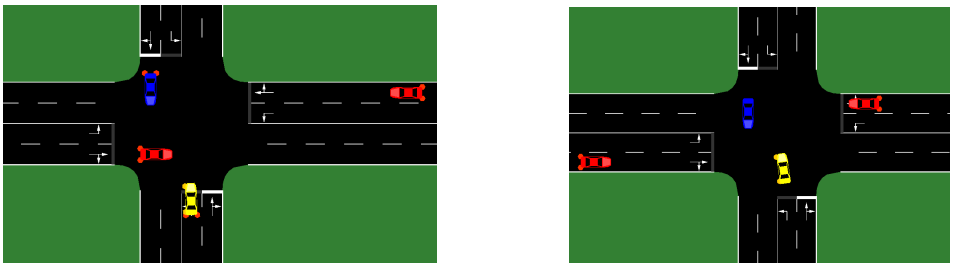


Figure 7.1: Standard mixed traffic scenario at unsignalized intersection including CAVs (red), CPVs (blue) and RVs (yellow) vehicles.

Since all vehicles are approaching the intersection at the same time, both a priority and a scheduling problem occur. In absence of traffic signals, one of the possible strategies to solve such problem is to follow the traffic rules and force all vehicles to give way to the right. However, in case a vehicle has higher priority over other vehicles on the basis of its class, that approach is not always feasible. Such case is depicted in Fig. 7.2a.

The blue priority vehicle slows down at intersection and gives right-of-way to one of the CAVs, that is not acceptable, as the blue vehicle has priority over the red one. Moreover, at unsignalized intersections, the risk of collisions is higher than at signalized intersections due to the high degree of discretionary power conferred to the drivers. In the proposed approach, CAVs and CPVs are both connected to a Vehicle-to-Infrastructure architecture and exchange data about their desired route, current lane, speed and distance to the intersection and eventually the distance from the vehicle in front, that could be even a regular vehicle.



(a) without priority (b) with priority

Figure 7.2: Mixed traffic scenario with and without CPVs priority.

The final objective is to design a control strategy for CAVs that regulates their speed at discrete time intervals in order to observe priorities, prevent collisions and reduce crossing times, even overcoming standard traffic rules in certain conditions. By following such ideal control strategy, we determine the scenario in Fig. 7.2b, in which the red vehicles, regardless of traffic rules, slow down as they know that a priority blue vehicle is about to cross the intersection.

In that respect, in this work, the control strategy is designed in terms of a policy trained by a DRL algorithm in a simulation framework. Accordingly, to model the interaction between CAVs and other vehicles, we model the AIM problem as a Markov Decision Process (MDP), where the agents follow a policy $\pi(a|s)$ in a predetermined environment. More specifically, at each time step t, given the current state $s(t)$, each agent chooses an action $a(t) \in \mathcal{A}$, according to the current policy, transits to the next state $s(t+1)$ and finally receives a reward $r(t) \in \mathbb{R}$. The agent purpose is to maximize the expectation of the return over time that is called the discounted cumulative reward and is defined as $G(s(t)) = \sum_{h=0}^{\infty} \gamma^h r(t+h+1)$, where $\gamma \in [0, 1]$ is the discount factor.

In the proposed problem formulation, the agents are the CAVs and the current state of each agent at each time step is designed on the basis of real-time traffic data shared by all connected vehicles. However, since priority vehicles are connected but driven by humans and regular vehicles are also unconnected, the future state of the environment not only depends on the agents, but also on certain unknown processes, such as the future intentions of CPVs drivers and the position of RVs. As a consequence, the process is only partially observable. In contrast, the global reward at each step is calculated on the basis of global traffic flow optimization objectives and considers both CAVs and CPVs current speeds and possible collisions. In this respect, in the proposed model, the agents work cooperatively to grant priorities, avoid accidents and reduce their crossing time.

7.3 Multi-agent DRL Optimization Approach

In this section, we introduce the details of the proposed Multi-agent DRL approach for AIM. For the sake of simplicity, we assume a 4-way unsignalized intersection, even if the method can be generalized to any kind of intersection.

Intersection scheme

The considered 4-way intersection is schematized in Fig. 7.3. Each way is made of two double lanes identified by a unique label. The first label identifies the incoming lane to the intersection, whereas the second one spots the outgoing lane. All possible routes are represented by the arrows and can be identified by joining the incoming and outgoing lanes labels. For example, considering E1 as incoming way, vehicles can designate either E1-N2, E1-W1 or E1-S1 as their planned routes.

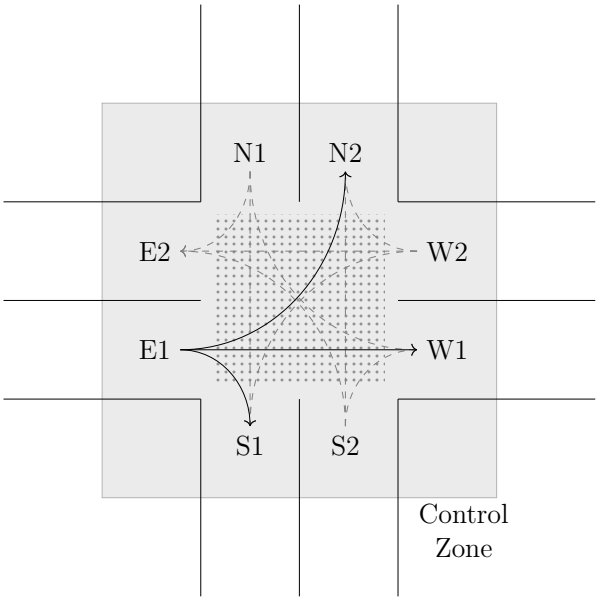


Figure 7.3: 4-way Intersection Scheme.

In addition, we define:

- $C = \{CAV_1, CAV_2, CAV_i, ..., CAV_n\}$ as the set of CAV agents,
- $P = \{CPV_1, CPV_2, CPV_j, ..., CPV_m\}$ as the set of CPV vehicles.

Similarly to the work [123], the intersection is divided in two different zones: (i) the control zone in which each CAV_i or CPV_j communicates with the centralized infrastructure and (ii) the merging zone where the collisions in the intersection may occur. The control zone is depicted in Fig. 7.3 as the gray rectangle surrounding the intersection. We assume that this zone extends up to 100 meters from the intersection center. On the contrary, the merging zone is the area shared by the vehicles when they cross the intersection and is depicted by the dotted rectangle in Fig. 7.3.

Within the control zone, at every time step t, each CAV_i sends the following data to the centralized infrastructure:

- the planned route, for example: E1-N2;
- the current speed $v_i(t)$ in m/s;
- the current distance $d_i(t)$ in meters to the intersection;
- the current distance $l_i(t)$ in meters from the in front vehicle.

In the same way, the planned route, $\hat{v_j}(t)$, $\hat{d_i}(t)$ and $\hat{l_i}(t)$ are defined for each CPV_j to denote the current speed and distances to the intersection and the in front vehicle, respectively.

State Representation

Having collected all data from CAVs and CPVs, the observation for each CAV_i agent at every time step t, is designed as a space in four dimensions. Every dimension in the space is a $k \times k$ grid, where k is the number of roads that make the intersection. In each $k \times k$ grid, rows and columns represent the incoming and outgoing ways respectively, whereas each cell identifies the corresponding route as defined in Section 7.3.

The four dimensions are used by CAVs and CPVs to mark in corresponding cells their planned routes, current speeds, distances to intersection and distances from the vehicle in front respectively.

In Table 7.1, an example observation of agent CAV_1, in the example 4-way intersection scenario, is shown.

In the proposed example, white cells of the tables represent all feasible routes on the intersection. On the contrary, gray boxes mark the unfeasible routes and cannot be filled. As shown in Table 7.1 (a), agents CAV_1 and CAV_2 (red), and priority vehicle CPV_1 (blue), have S2-W1, W2-S1 and S2-N2 as their planned routes respectively.

For each of the vehicles, the current speed in m/s is reported in Table 7.1 (b), while the distance to the intersection can be found in Table 7.1

Table 7.1: Observation Example for CAV_1

	N2	S1	W1	E2
N1				
S2	CPV_1		**CAV_1**	
W2		CAV_2		
E1				

7.1 (a)
Planned Route

	N2	S1	W1	E2
N1				
S2	7m/s		**9m/s**	
W2		6m/s		
E1				

7.1 (b)
Current Speed

	N2	S1	W1	E2
N1				
S2	30m		**25m**	
W2		20m		
E1				

7.1 (c)
Distance to Intersection

	N2	S1	W1	E2
N1				
S2	n/d		**6.5m**	
W2		n/d		
E1				

7.1 (d)
Distance from
Vehicle in Front

(c). In this regard, positive values are used as the vehicle approaches the intersection, whereas negative values indicate that the vehicle is leaving the junction. Finally, in Table 7.1 (d), the distance from vehicle in front, if any, is recorded. In the given example, only CAV_1 agent has vehicle in front that could be another CAV agent, a CPV or even a regular vehicle.

In general, the proposed approach can reproduce only partially the current state traffic as each agent has only visibility of the first connected vehicle in front, either CAV or CPV, for each route. Moreover, regular unconnected vehicles can be seen only in terms of their distance from a connected vehicle.

In addition, since more than one CAV may have the same planned route, agents observations generally differ each other. For example, suppose that two agents, i.e. CAV_1 and CAV_2, both planned the same route and that CAV_2 was the vehicle in front on the way. In that case, CAV_1 would see itself in the related cell in first grid and the distance to CAV_2 would be reported in fourth grid. At the same time, in CAV_2 observation, the value in the first grid would be CAV_2 itself, whereas no distance from vehicle in front would be found in fourth grid.

Also for those reasons, the state is only partially observable by each CAV agent and the cooperative approach is used to improve the performance of the trained policy.

Action Space

In this work, we assume that the kinematics of each autonomous CAV_i agent is driven by a separate low-level and autonomous control strategy and that only the speed $v(t)$ at each time step t needs to be set in order to fulfill the autonomous intersection management objectives. In this regard, we define the discrete set of three actions $\mathcal{A} = \{ACC,\ KEEP,\ DEC\}$ in which, given the predefined speed variation δ m/s, one of $ACC = \delta,\ KEEP = 0$ or $DEC = -\delta$ represents the action $a_i(t)$ chosen according to the policy at every time step t for each CAV_i. As a consequence, at each step, the new speed $v_i(t)$ of the agent CAV_i is calculated as $v_i(t) = v_i(t-1) + a_i(t)$.

Reward

As stated in Section 7.2, the common objectives of the proposed control strategy are: (i) force agents to observe priorities by giving right-of-way to connected priority vehicles, (ii) prevent collisions and (iii) reduce individual crossing times. In order to design a proper global reward function that takes into account all aforementioned requirements, including penalties in case of collisions, we first define the following binary variable $p_i(t)$ as follows:

$$p_i(t) = \begin{cases} 1, & \text{if } CAV_i \text{ detects collisions} \\ & \text{with other vehicles at time step } t \\ 0, & \text{otherwise.} \end{cases} \tag{7.1}$$

Then, the global reward function is defined by:

$$r(t) = w_1 \sum_{i=1}^{n} v_i(t) + w_2 \sum_{j=1}^{m} \hat{v_j}(t) - w_3 \sum_{i=1}^{n} v_i(t) p_i(t), \tag{7.2}$$

where $w_1, w_2, w_3 \in [0, 1]$ are arbitrary weights.

In detail, global reward at time step t is given by adding the weighted sum of CAVs current speeds and the weighted sum of current velocities of all priority vehicles. A more aggressive or conservative strategy against priorities is regulated by tuning w_1, which is the weight of CAVs speeds, and w_2, which is the weight of CPVs speeds. More specifically, greater the value of w_2 compared to w_1, higher is the probability of CPVs to be given right-of-way by CAVs.

In addition, a strong penalty, weighted by w_3 and proportional to the current speed v_i, is applied in case CAV_i collides with other vehicles at time step t. In this respect, setting a value for $w_3 \gg w_1, w_2$ drastically increases the probability to avoid accidents.

7.4 Case Study

In this section, we implement the proposed method on the example scenario schematized in Fig. 7.3 and we give the details about the simulation environment, the training setup and the performance evaluation.

Simulation Environment and Training Setup

The intersection is designed in SUMO environment. We set three different vehicle classes: CAVs, CPVs and RVs. Then, we connect SUMO by *TraCi* interface to a Python script in which we implement the Multi-agent DRL policy training strategy by RLlib and PPO. We run 500 training episodes with randomly generated traffic on the given simulation environment and we collect some metrics. More in detail, at the beginning of each episode, we place four vehicles at 30 meters from the intersection with randomly assigned routes. We choose to place two CAVs, i.e. the agents in the training strategy, one CPV and one RV. All vehicles are 5 meters long and have maximum speed set at 50 m/s.

As explained in Section 7.3, a shared fully connected network (FCN) is used as policy π and value function ϕ estimator. Hence, since $k = 4$, the network, that is shown in Fig. 7.4, has one input layer of size $4 \times 4 \times 4 = 64$, two hidden layers of size 256 and two output layers of size 3 for the policy and 1 for the value respectively.

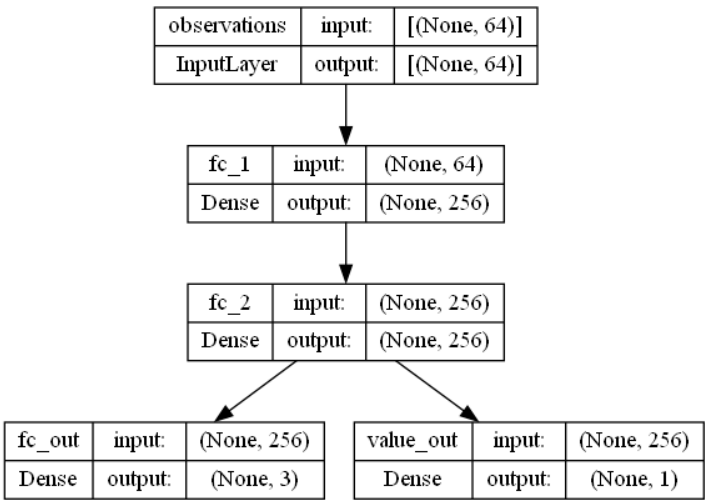


Figure 7.4: Fully Connected Network used as Policy π and Value Function ϕ estimator.

Although the control action on CAVs is effective as they cross the intersection, the speed should be regulated since when they approach the junction. Hence, rewards in the future are of great importance. In this regard, we set $\gamma = 0.99$ as discount factor. Moreover, we set $\alpha = 5e{-}5$ as starting learning rate and $ReLU$ as activation function.

Finally, since we wish to give more importance to collisions prevention and CPVs crossing than to CAVs speed, we set the weights in reward function (7.2) as: $w_1 = 1, w_2 = 2$ and $w_3 = 5$.

To decrease the number of steps, we terminate each episode when all vehicles are on the outgoing way and at least 20 meters from the intersection. In each step, a train batch of size 4000 is sampled and chunked down in mini-batches of size 128, each of those is used to update the agents for 30 consecutive iterations. We run the simulations for 10 to 15 hours on a personal computer with Intel(R) Core(TM) i7-1185G7 3.00 GHz CPU and 16 GB RAM.

Performance Evaluation

In Fig. 7.5, we show the performance of the training strategy. Firstly, we observe that the global reward converges after 50 episodes to a stable value. However, after around 320 episodes the number of accumulated collisions does not significantly increase anymore. Since one of the main objectives of the proposed approach is to ensure safety and avoid accidents, both the global reward and total collisions have to be considered to determine an effective training stop strategy.

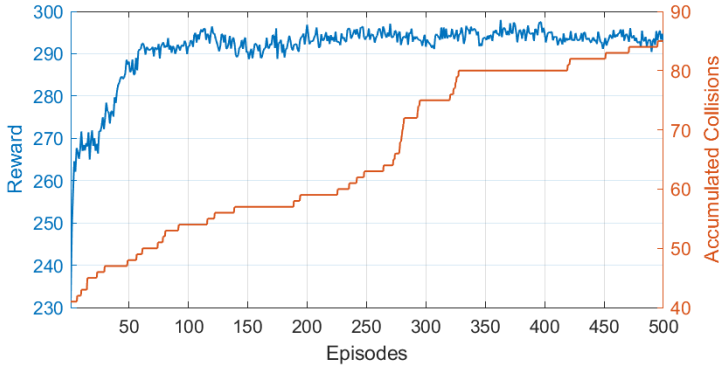


Figure 7.5: Global Reward obtained during training and accumulated collisions over all episodes.

In Fig. 7.6, the average speed over episodes for all connected vehicles is shown. Although the speed plays a central role in reward function (7.2), the global average value tends to slow down as collisions converge to a stable value around 320 episodes. This is due to the fact that CAV agents, as they approach the junction, are forced by the policy to give right-of-way to priority vehicles and to act, at the same time, in a conservative fashion to prevent collisions.

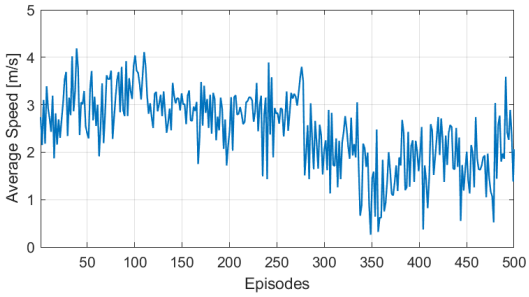


Figure 7.6: Average speed of connected vehicles.

The speed profiles of CAV_1 and CAV_2 agents, compared to the one of priority vehicle CPV_1, over a single episode after the training, are shown in Fig. 7.7. We observe that CAV_1, traveling on E1-W1 route, starts slowing down after 7 seconds entering the control zone and stops for around 3 seconds at time step 9. At the same time, CAV_2, traveling on W2-E2 route, starts decelerating after around 6 seconds and accelerates again at time step 9. In contrast, CPV_1, traveling on N1-S1 route, constantly increases its velocity over the whole time horizon and goes through the intersection without any interference of other vehicles. In conclusion, after around 320 episodes the trained policy is able to fulfill all prescribed objectives.

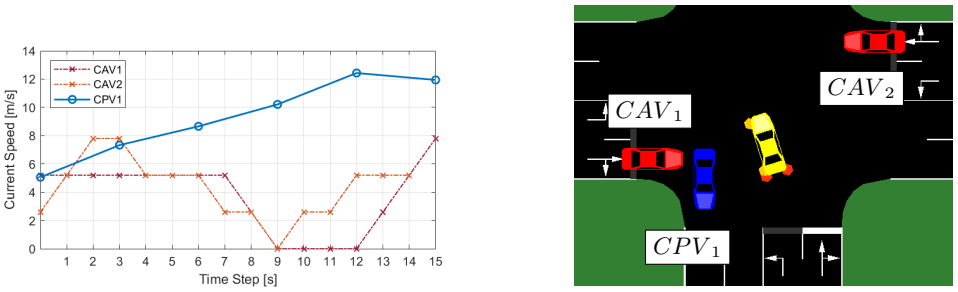


Figure 7.7: Speed profile of CAV_1 and CAV_2 agents vs. priority vehicle CPV_1 as they simultaneously cross the intersection.

7.5 Conclusion

In this work, we present a novel approach for autonomous unsignalized intersection management in mixed traffic scenarios with priority vehicles. The proposed method is based on cooperative Multi-agent Deep Reinforcement Learning and aims to optimize global traffic flow by giving right-of-way at intersection to specific classes of priority vehicles and ensuring safety, at the same time, by preventing collisions. Furthermore, since the scenario is of mixed traffic, we introduce a novel multidimensional intersection state representation that is only partially observable by each connected and autonomous vehicle since, in general, only some of other agents, priority connected vehicles and unconnected regular vehicles may appear in an observation.

We validate the proposed cooperative approach with Proximal Policy Optimization in a 4-way example scenario designed in SUMO, in which we place two connected and autonomous vehicles, one priority connected vehicle and one regular vehicle traveling on randomly assigned conflicting routes. Experiments results show that after around 320 training episodes the Fully Connected Network policy is able to ensure stable global reward, avoid accidents and give right-of-way to priority vehicles.

In future work, we plan to investigate the impact of partial observability in more complex intersections and traffic scenarios. Furthermore, we aim to assess the scalability of the presented method by extending simulations with more cooperative agents and test alternative approaches to accelerate the DRL training process.

8 Conclusion

The necessity to address multiple complex tasks in fast efficient and secure environments is one of the key requirements in the new Industry 4.0 paradigm. The contribution of *Cloud Computing* in this sense is crucial and ensures the availability of distributed and resilient infrastructures to provide a variety of services for many business domains.

Distributed systems have the ability to split the effort of multiple complex tasks across a set of nodes, often coordinated by a single leader, and complete the jobs more efficiently, compared to the centralized alternative. In addition, in decentralized systems, a subset of distributed systems, each node also constitutes a point of decision and contributes to the system behaviour.

A variety of well known optimization problem can be performed in such environments by using both classical and AI-based methodologies, to increase speed and improve resilience and efficiency.

In this thesis, the implementation of some of such problems in distributed environments has been presented. In Part I, the role of Blockchain, a special case of a decentralized system, has been specifically investigated. The main contributions of this part are reported hereafter:

- A collaborative decentralized *Ethereum*-based multi-agent platform to deliver complex software tasks in manufacturing environments by a dedicated *Smart Contract*, with the complementary use of Blockchain, *Docker* and Cloud Storage in a secure and tamper-proof environment;
- An approach for optimal task assignment, based on least agent runtime estimation. The runtime prediction is performed by a specifically designed Artificial Neural Network;
- A second platform designed in *HyperLedger Fabric*, in which the concept of task assignment procedure is driven by an auction and bidding scheme provided by a dedicated *Smart Contract*;

- A DRL-based runtime prediction algorithm that overcomes the limitations of the first proposed algorithm by providing incremental on-line learning and current load state estimation when multiple concurrent tasks are running on the agents;

- An *Ethereum*-based incentive platform for energy management that collects users consumption data in real-time and provides clustering and rating of the users in a district with a subsequent penalty and reward scheme based on an optimization problem.

Future research directions for this part regard the use of different container-based solutions such as *Kubernetes* as well as the implementation of different Cloud related optimization problems, such as optimal resources allocation, on-demand scaling and task re-allocation.

In Part II, the problem of Flexible Job Shop Scheduling (FJSSP) has been addressed with the implementation of a Swarm-based approach. In detail, the contributions of this part are:

- The design of a TCPN framework to simulate mass production systems;

- The implementation of PSO to find the optimal job sequence that maximizes throughput and minimizes machine waiting time.

Thanks to the intrinsic features of PSO, the proposed algorithm is likely to be implemented and solved in a distributed system context. In future developments, the investigated modeling approach should be more extensively generalized and the management of complex job routings should be included to allow a concrete industrial application.

Finally, in Part III, the problem of autonomous intersection management in the domain of Autonomous Driving has been explored. More specifically the contributions of this part are as follows:

- A novel Multi-agent Cooperative DRL approach for autonomous intersection management at unsignalized intersections;

- A partially-observable state representation scheme that includes not only pure autonomous vehicles, but also connected human driven vehicles and unconnected regular vehicles;

- A global reward scheme that considers vehicle priorities and prevent collisions.

The optimal control policy can be synthesized by training the proposed DRL scheme in a parallel fashion in the context of a distributed system. Future work for this last part is the investigation of the impact of partial observability in more complex intersections and traffic scenarios. Moreover, with the increasing number of the agents, the impact on performance should be studied.

References

[1] Gaetano Volpe, Agostino Marcello Mangini, and Maria Pia Fanti. An architecture combining blockchain, docker and cloud storage for improving digital processes in cloud manufacturing. *IEEE Access*, 10: 79141–79151, 2022. doi: 10.1109/ACCESS.2022.3194264.

[2] Gaetano Volpe, Agostino Marcello Mangini, and Maria Pia Fanti. A deep reinforcement learning approach for competitive task assignment in enterprise blockchain. *IEEE Access*, 11:48236–48247, 2023. doi: 10.1109/ACCESS.2023.3276859.

[3] Gaetano Volpe, Agostino Marcello Mangini, and Maria Pia Fanti. An architecture for digital processes in manufacturing with blockchain, docker and cloud storage. In *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pages 39–44, 2021. doi: 10.1109/CASE49439.2021.9551633.

[4] Giuseppe Olivieri, Gaetano Volpe, Agostino Marcello Mangini, and Maria Pia Fanti. A district energy management approach based on internet of things and blockchain. In *2022 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, pages 1–6, 2022. doi: 10.1109/DASC/PiCom/CBDCom/Cy55231.2022.9927910.

[5] Marco Fiore, Federico Carrozzino, Marina Mongiello, Gaetano Volpe, and Agostino Marcello Mangini. A blockchain-based modular architecture for managing multiple and quantum-safe encryption algorithms. In *2023 9th International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 598–601, 2023. doi: 10.1109/CoDIT58514.2023.10284090.

[6] Gaetano Volpe, Agostino Marcello Mangini, and Maria Pia Fanti. Job shop sequencing in manufacturing plants by timed coloured

petri nets and particle swarm optimization. *IFAC-PapersOnLine*, 55(28):350–355, 2022. ISSN 2405-8963. doi: https://doi.org/10.1016/j.ifacol.2022.10.365. URL https://www.sciencedirect.com/science/article/pii/S2405896322024028. 16th IFAC Workshop on Discrete Event Systems WODES 2022.

[7] Gaetano Volpe, Agostino Marcello Mangini, and Maria Pia Fanti. A cooperative drl approach for autonomous traffic prioritization in mixed vehicles scenarios. In *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*, pages 1–6, 2023. doi: 10.1109/CASE56687.2023.10260615.

[8] Francesco Paparella, Gaetano Volpe, Agostino Marcello Mangini, and Maria Pia Fanti. Collision avoidance strategy for autonomous intersection management by a central optimizer algorithm. In *2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2023 (to appear).

[9] Jiewu Leng, Guolei Ruan, Pingyu Jiang, Kailin Xu, Qiang Liu, Xueliang Zhou, and Chao Liu. Blockchain-empowered sustainable manufacturing and product lifecycle management in industry 4.0: A survey. *Renewable and Sustainable Energy Reviews*, 132:110112, 2020. ISSN 1364-0321. doi: https://doi.org/10.1016/j.rser.2020.110112. URL http://www.sciencedirect.com/science/article/pii/S1364032120304032.

[10] Xun Xu. From cloud computing to cloud manufacturing. *Robotics and Computer-Integrated Manufacturing*, 28(1):75–86, 2012. ISSN 0736-5845. doi: https://doi.org/10.1016/j.rcim.2011.07.002. URL https://www.sciencedirect.com/science/article/pii/S0736584511000949.

[11] JP Vergne. Decentralized vs. distributed organization: Blockchain, machine learning and the future of the digital platform. *Organization Theory*, 1(4):2631787720977052, 2020. doi: 10.1177/2631787720977052. URL https://doi.org/10.1177/2631787720977052.

[12] Fran Casino, Thomas K. Dasaklis, and Constantinos Patsakis. A systematic literature review of blockchain-based applications: Current status, classification and open issues. *Telematics and Informatics*, 36:55 – 81, 2019. ISSN 0736-5853. doi: https://doi.org/10.1016/j.

tele.2018.11.006. URL http://www.sciencedirect.com/science/article/pii/S0736585318306324.

[13] K. Christidis and M. Devetsikiotis. Blockchains and smart contracts for the internet of things. *IEEE Access*, 4:2292–2303, 2016. doi: 10.1109/ACCESS.2016.2566339.

[14] Giang-Truong Nguyen and Kyungbaek Kim. A survey about consensus algorithms used in blockchain. *J. Inf. Process. Syst.*, 14:101–128, 2018.

[15] Riikka Koulu. Blockchains and online dispute resolution: Smart contracts as an alternative to enforcement. *SCRIPTed*, 13:40–69, 05 2016. doi: 10.2966/script.130116.40.

[16] Solidity Documentation. https://docs.soliditylang.org/en/v0.8.1, Jan 2021. [Online; accessed 22. Feb. 2021].

[17] Vyper Documentation. https://vyper.readthedocs.io/en/stable, Feb 2021. [Online; accessed 22. Feb. 2021].

[18] Sishan Long, Soumya Basu, and Emin Gün Sirer. Measuring miner decentralization in proof-of-work blockchains, 2022. URL https://arxiv.org/abs/2203.16058.

[19] Tharaka Hewa, Mika Ylianttila, and Madhusanka Liyanage. Survey on blockchain based smart contracts: Applications, opportunities and challenges. *Journal of Network and Computer Applications*, 177:102857, 2021. ISSN 1084-8045. doi: https://doi.org/10.1016/j.jnca.2020.102857. URL http://www.sciencedirect.com/science/article/pii/S1084804520303234.

[20] Charles Anderson. Docker [software engineering]. *IEEE Software*, 32 (3):102–c3, 2015. doi: 10.1109/MS.2015.62.

[21] CouchDB: The Definitive Guide, Aug 2019. URL http://guide.couchdb.org/editions/1/en/index.html. [Online; accessed 22. Dec. 2021], http://guide.couchdb.org/editions/1/en/index.html.

[22] Diego Ongaro and John Ousterhout. In search of an understandable consensus algorithm. In *2014 USENIX Annual Technical Conference (USENIX ATC 14)*, pages 305–319, Philadelphia, PA, June 2014. USENIX Association. ISBN 978-1-931971-10-2. URL https://www.usenix.org/conference/atc14/technical-sessions/presentation/ongaro.

[23] Docker. https://docs.docker.com/get-started/overview, Oct 2023. [Online; accessed 09. Oct. 2023].

[24] Wenying Zeng, Yuelong Zhao, Kairi Ou, and Wei Song. Research on cloud storage architecture and key technologies. In *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human*, ICIS '09, page 1044–1048, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605587103. doi: 10.1145/1655925.1656114. URL https://doi.org/10.1145/1655925.1656114.

[25] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013. URL http://arxiv.org/abs/1312.5602.

[26] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Mach. Learn.*, 8(3):279–292, May 1992. ISSN 1573-0565. doi: 10.1007/BF00992698.

[27] Timothy Lillicrap, Jonathan Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, 09 2015.

[28] Global cloud and data center spending 2020 | Statista. https://www.statista.com/statistics/1114926/enterprise-spending-cloud-and-data-centers, May 2021. [Online; accessed 7. May 2021].

[29] Ming K. Lim, Weiqing Xiong, and Zhimei Lei. Theory, supporting technology and application analysis of cloud manufacturing: a systematic and comprehensive literature review. *Industrial Management & Data Systems*, 120(8):1585–1614, Jan 2020. ISSN 0263-5577. doi: 10.1108/IMDS-10-2019-0570. URL https://doi.org/10.1108/IMDS-10-2019-0570.

[30] C. Esposito, A. Castiglione, B. Martini, and K. Choo. Cloud manufacturing: Security, privacy, and forensic concerns. *IEEE Cloud Computing*, 3(04):16–22, jul 2016. ISSN 2372-2568. doi: 10.1109/MCC.2016.79.

[31] Shuai Wang, Liwei Ouyang, Yong Yuan, Xiaochun Ni, Xuan Han, and Fei-Yue Wang. Blockchain-enabled smart contracts: Architecture,

applications, and future trends. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(11):2266–2277, 2019. doi: 10.1109/TSMC.2019.2895123.

[32] U. Bodkhe, S. Tanwar, K. Parekh, P. Khanpara, S. Tyagi, N. Kumar, and M. Alazab. Blockchain for industry 4.0: A comprehensive review. *IEEE Access*, 8:79764–79800, 2020. doi: 10.1109/ACCESS.2020.2988579.

[33] Jiewu Leng, Shide Ye, Man Zhou, J. Leon Zhao, Qiang Liu, Wei Guo, Wei Cao, and Leijie Fu. Blockchain-secured smart manufacturing in industry 4.0: A survey. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 51(1):237–252, 2021. doi: 10.1109/TSMC.2020.3040789.

[34] J. E. Kasten. Engineering and manufacturing on the blockchain: A systematic review. *IEEE Engineering Management Review*, 48(1):31–47, 2020. doi: 10.1109/EMR.2020.2964224.

[35] Jiewu Leng, Douxi Yan, Qiang Liu, Kailin Xu, J. Leon Zhao, Rui Shi, Lijun Wei, Ding Zhang, and Xin Chen. Manuchain: Combining permissioned blockchain with a holistic optimization model as bi-level intelligence for smart manufacturing. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 50(1):182–192, 2020. doi: 10.1109/TSMC.2019.2930418.

[36] Tharaka Mawanane Hewa, An Braeken, Madhusanka Liyanage, and Mika Ylianttila. Fog computing and blockchain based security service architecture for 5g industrial iot enabled cloud manufacturing. *IEEE Transactions on Industrial Informatics*, pages 1–1, 2022. doi: 10.1109/TII.2022.3140792.

[37] J. A. Garcia-Garcia, N. Sánchez-Gómez, D. Lizcano, M. J. Escalona, and T. Wojdyński. Using blockchain to improve collaborative business process management: Systematic literature review. *IEEE Access*, 8:142312–142336, 2020. doi: 10.1109/ACCESS.2020.3013911.

[38] T. Allweyer. *BPMN 2.0: Introduction to the Standard for Business Process Modeling*. Books on Demand, 2010. ISBN 9783839149850. URL `https://books.google.it/books?id=fdlC7K_3dzEC`.

[39] Orlenys López-Pintado, L. García-Bañuelos, M. Dumas, and Ingo Weber. Caterpillar: A blockchain-based business process management system. In *BPM*, 2017.

[40] Turusha Ghimire, Atharva Joshi, Samgeeth Sen, Chinmay Kapruan, Utkarsh Chadha, and Senthil Kumaran Selvaraj. Blockchain in additive manufacturing processes: Recent trends & its future possibilities. *Materials Today: Proceedings*, 50:2170–2180, 2022. ISSN 2214-7853. doi: https://doi.org/10.1016/j.matpr.2021.09.444. URL https://www.sciencedirect.com/science/article/pii/S2214785321063604. 2nd International Conference on Functional Material, Manufacturing and Performances (ICFMMP-2021).

[41] Abhiram Haridas, Adil Abdul Samad, Vysakh D, Deepak Lawrence K, and Vinod Pathari. A blockchain-based platform for smart contracts and intellectual property protection for the additive manufacturing industry. In *2022 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES)*, volume 1, pages 223–230, 2022. doi: 10.1109/SPICES52834.2022.9774219.

[42] Ethereum Development documentation. https://ethereum.org/en/developers/docs, Jan 2021. [Online; accessed 21. Jan. 2021].

[43] Agustín Halty, Rodrigo Sánchez, Valentín Vázquez, Víctor Viana, Pedro Piñeyro, and Daniel Alejandro Rossit. Scheduling in cloud manufacturing systems: Recent systematic literature review. *Mathematical biosciences and engineering : MBE*, 17(6):7378—7397, October 2020. ISSN 1547-1063. doi: 10.3934/mbe.2020377. URL https://doi.org/10.3934/mbe.2020377.

[44] Marlon Dumas, Marcello La Rosa, Jan Mendling, and Hajo A. Reijers. *Fundamentals of Business Process Management*. Springer, Berlin, Germany, 2013. ISBN 978-3-642-33142-8. doi: 10.1007/978-3-642-33143-5.

[45] I. Barinov, V. Baranov, and P. Khahulin. POA Network Whitepaper. https://github.com/poanetwork/wiki/wiki/POA-Network-Whitepaper, Feb 2021. [Online; accessed 21. Feb. 2021].

[46] Grady Booch, James Rumbaugh, and Ivar Jacobson. *The Unified Modeling Language User Guide*. Addison Wesley Longman Publishing Co., Inc., USA, 1999. ISBN 0201571684.

[47] DCS - Data Communication Standard, v.3.12. https://www.thevisioncouncil.org/sites/default/files/DCS-v3_12_FINAL_v2.pdf, Aug 2019. [Online; accessed 10. May. 2021].

[48] David Harris and Sarah Harris. *Digital Design and Computer Architecture*. Morgan Kaufmann, Oxford, England, 2 edition, 2012.

[49] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL https://proceedings.mlr.press/v15/glorot11a.html.

[50] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pages 1–15, 2015.

[51] ERC-20 Token Standard | ethereum.org. https://ethereum.org/en/developers/docs/standards/tokens/erc-20, Feb 2021. [Online; accessed 11. Feb. 2021].

[52] Public cloud IaaS market worldwide 2023 | Statista, September 2022. URL https://www.statista.com/statistics/505251/worldwide-infrastructure-as-a-service-revenue. [Online; accessed 27. Sep. 2022].

[53] Jesus Flores-Contreras, Hector A. Duran-Limon, Arturo Chavoya, and Sergio H. Almanza-Ruiz. Performance prediction of parallel applications: a systematic literature review. *Journal of Supercomputing*, 77(4):4014–4055, 2021. ISSN 15730484. doi: 10.1007/s11227-020-03417-5. URL https://doi.org/10.1007/s11227-020-03417-5.

[54] Ali Mohammed, Ahmed Eleliemy, Florina M. Ciorba, Franziska Kasielke, and Ioana Banicescu. An approach for realistically simulating the performance of scientific applications on high performance computing systems. *Future Generation Computer Systems*, 111:617–633, 2020. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2019.10.007. URL https://www.sciencedirect.com/science/article/pii/S0167739X19308830.

[55] Younghyun Cho, Surim Oh, and Bernhard Egger. Performance modeling of parallel loops on multi-socket platforms using queueing systems. *IEEE Transactions on Parallel and Distributed Systems*, 31(2):318–331, 2020. doi: 10.1109/TPDS.2019.2938172.

[56] Peter Altenbernd, Jan Gustafsson, Björn Lisper, and Friedhelm Stappert. Early execution time-estimation through automatically generated timing models. *Real-Time Systems: The International Journal of Time-Critical Computing Systems*, 52(6):731–760, November 2016. URL `http://www.es.mdh.se/publications/4284-`.

[57] Tri Doan and Jugal Kalita. Predicting run time of classification algorithms using meta-learning. *International Journal of Machine Learning and Cybernetics*, 8, 12 2017. doi: 10.1007/s13042-016-0571-6.

[58] Xinnian Wang, Keyi Xing, Yanxiang Feng, and Yunchao Wu. Scheduling of Flexible Manufacturing Systems Subject to No-Wait Constraints via Petri Nets and Heuristic Search. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 51(10):6122–6133, 2021. ISSN 21682232.

[59] Hongzi Mao, Mohammad Alizadeh, Ishai Menache, and Srikanth Kandula. Resource management with deep reinforcement learning. *HotNets 2016 - Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, pages 50–56, 2016. doi: 10.1145/3005745.3005750.

[60] Zixuan Xie, Run Wu, Miao Hu, and Haibo Tian. Blockchain-Enabled Computing Resource Trading: A Deep Reinforcement Learning Approach. In *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, volume 2020-May, pages 1–8. IEEE, may 2020. ISBN 978-1-7281-3106-1. doi: 10.1109/WCNC45663.2020.9120521. URL `https://ieeexplore.ieee.org/document/9120521/`.

[61] Intro to Ethereum | ethereum.org, March 2023. URL `https://ethereum.org/en/developers/docs/intro-to-ethereum`. [Online; accessed 29. Mar. 2023].

[62] Andréa Matsunaga and José A.B. Fortes. On the use of machine learning to predict the time and resources consumed by applications. In *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pages 495–504, 2010. doi: 10.1109/CCGRID.2010.98.

[63] David A. Monge, Matěj Holec, Filip Železný, and Carlos García Garino. Ensemble learning of runtime prediction models for gene-expression analysis workflows. *Cluster Computing*, 18(4):1317–1329, 2015. ISSN 15737543. doi: 10.1007/s10586-015-0481-5.

[64] Muhammad Hafizhuddin Hilman, Maria A. Rodriguez, and Rajkumar Buyya. Task runtime prediction in scientific workflows using an online incremental learning approach. *Proceedings - 11th IEEE/ACM International Conference on Utility and Cloud Computing, UCC 2018*, pages 93–102, 2019. doi: 10.1109/UCC.2018.00018.

[65] Thanh Phuong Pham, Juan J. Durillo, and Thomas Fahringer. Predicting Workflow Task Execution Time in the Cloud Using A Two-Stage Machine Learning Approach. *IEEE Transactions on Cloud Computing*, 8(1):256–268, 2020. ISSN 21687161. doi: 10.1109/TCC.2017.2732344.

[66] Lan Jiang, Hongyun Huang, and Zuohua Ding. Path planning for intelligent robots based on deep q-learning with experience replay and heuristic knowledge. *IEEE/CAA Journal of Automatica Sinica*, 7(4):1179–1189, 2020. doi: 10.1109/JAS.2019.1911732.

[67] Rongbo Zhu, Hao Liu, Lu Liu, Xiaozhu Liu, Wenjie Hu, and Bo Yuan. A blockchain-based two-stage secure spectrum intelligent sensing and sharing auction mechanism. *IEEE Transactions on Industrial Informatics*, 18(4):2773–2783, 2022. doi: 10.1109/TII.2021.3104325.

[68] Haojun Huang, Wang Miao, Zhaoxi Li, Jialin Tian, Chen Wang, and Geyong Min. Enabling energy trading in cooperative microgrids: A scalable blockchain-based approach with redundant data exchange. *IEEE Transactions on Industrial Informatics*, 18(10):7077–7085, 2022. doi: 10.1109/TII.2021.3115576.

[69] Chi Harold Liu, Qiuxia Lin, and Shilin Wen. Blockchain-enabled data collection and sharing for industrial iot with deep reinforcement learning. *IEEE Transactions on Industrial Informatics*, 15(6):3516–3526, 2019. ISSN 19410050. doi: 10.1109/TII.2018.2890203.

[70] Weijun Zheng, Wenhua Wang, Guoqing Wu, Chenzi Xue, and Yifei Wei. Fog computing enabled smart grid blockchain architecture and performance optimization with drl approach. In *2020 IEEE 8th International Conference on Computer Science and Network Technology (ICCSNT)*, pages 41–45, 2020. doi: 10.1109/ICCSNT50940.2020.9305000.

[71] Le Yang, Meng Li, Pengbo Si, Ruizhe Yang, Enchang Sun, and Yanhua Zhang. Energy-Efficient Resource Allocation for Blockchain-Enabled Industrial Internet of Things With Deep Reinforcement

Learning. *IEEE Internet of Things Journal*, 8(4):2318–2329, 2020. ISSN 2327-4662. doi: 10.1109/jiot.2020.3030646.

[72] Ye Xu, Liang Yu, Gang Bi, Meng Zhang, and Chao Shen. Deep Reinforcement Learning and Blockchain for Peer-to-Peer Energy Trading among Microgrids. In *2020 International Conferences on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*, pages 360–365. IEEE, nov 2020. ISBN 978-1-7281-7647-5. doi: 10.1109/iThings-GreenCom-CPSCom-SmartData-Cybermatics50389.2020.00071. URL `https://ieeexplore.ieee.org/document/9291556/`.

[73] Chao Qiu, Xiaoxu Ren, Yifan Cao, and Tianle Mai. Deep reinforcement learning empowered adaptivity for future blockchain networks. *IEEE Open Journal of the Computer Society*, 2:99–105, 2021. doi: 10.1109/OJCS.2020.3010987.

[74] Alan A.A. Donovan and Brian W. Kernighan. *The Go Programming Language*. Addison-Wesley Professional, 1st edition, 2015. ISBN 0134190440.

[75] Nicolas Auger, Vincent Jugé, Cyril Nicaud, and Carine Pivoteau. On the Worst-Case Complexity of TimSort. In Yossi Azar, Hannah Bast, and Grzegorz Herman, editors, *26th Annual European Symposium on Algorithms (ESA 2018)*, volume 112 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 4:1–4:13, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-95977-081-1. doi: 10.4230/LIPIcs.ESA.2018.4. URL `http://drops.dagstuhl.de/opus/volltexte/2018/9467`.

[76] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL `https://doi.org/10.1038/s41586-020-2649-2`.

[77] Charles AR Hoare. Quicksort. *The Computer Journal*, 5(1):10–16, 1962.

[78] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[79] Robert Sedgewick and Kevin Wayne. *Algorithms, 4th Edition.* Addison-Wesley, 2011. ISBN 978-0-321-57351-3.

[80] Dan-Dan Zhu and Jun-Qing Sun. A new algorithm based on dijkstra for vehicle path planning considering intersection attribute. *IEEE Access*, 9:19761–19775, 2021. doi: 10.1109/ACCESS.2021.3053169.

[81] Alina Buzachis, Antonio Celesti, Antonino Galletta, Jiafu Wan, and Maria Fazio. Evaluating an application aware distributed dijkstra shortest path algorithm in hybrid cloud/edge environments. *IEEE Transactions on Sustainable Computing*, 7(2):289–298, 2022. doi: 10.1109/TSUSC.2021.3071476.

[82] Filipo Studzinski Perotto and Laurent Vercouter. Tuning the discount factor in order to reach average optimality on deterministic mdps. In Max Bramer and Miltos Petridis, editors, *Artificial Intelligence XXXV*, pages 92–105, Cham, 2018. Springer International Publishing. ISBN 978-3-030-04191-5.

[83] Michele Roccotelli, Agostino Marcello Mangini, and Maria Pia Fanti. Smart district energy management with cooperative microgrids. *IEEE Access*, 10:36311–36326, 2022. doi: 10.1109/ACCESS.2022.3163724.

[84] Maria Pia Fanti, Agostino Marcello Mangini, Michele Roccotelli, and Walter Ukovich. A district energy management based on thermal comfort satisfaction and real-time power balancing. *IEEE Transactions on Automation Science and Engineering*, 12(4):1271–1284, Oct 2015. ISSN 1558-3783. doi: 10.1109/TASE.2015.2472956.

[85] D. Muralidar, Meikandasivam, and Vijayakumar. A review about energy management techniques in industrial buildings. In *2017 Innovations in Power and Advanced Computing Technologies (i-PACT)*, pages 1–9, April 2017. doi: 10.1109/IPACT.2017.8245086.

[86] Ozan Erdinç, Akın Taşcikaraoğlu, Nikolaos G. Paterakis, and João P. S. Catalão. Novel incentive mechanism for end-users enrolled in dlc-based demand response programs within stochastic planning context. *IEEE Transactions on Industrial Electronics*, 66(2):1476–1487, Feb 2019. ISSN 1557-9948. doi: 10.1109/TIE.2018.2811403.

[87] Hao Wang and Jianwei Huang. Incentivizing energy trading for interconnected microgrids. *IEEE Transactions on Smart Grid*, 9(4): 2647–2657, 2018. doi: 10.1109/TSG.2016.2614988.

[88] Miroslav Markovic, Marko Maljkovic, and Rini Nur Hasanah. Smart home heating control using raspberry pi and blynk iot platform. In *2020 10th Electrical Power, Electronics, Communications, Controls and Informatics Seminar (EECCIS)*, pages 188–192, Aug 2020. doi: 10.1109/EECCIS49483.2020.9263441.

[89] Ivan Ganchev, Zhanlin Ji, and Mairtin O'Droma. An iot-based smart electric heating control system: Design and implementation. In *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*, pages 760–762, July 2017. doi: 10.1109/ICUFN.2017.7993895.

[90] Research Department Statista. Global smart home devices shipment share 2018-2025, by category. Technical report, Statista, mar 2022. URL `https://www.statista.com/statistics/920694/smart-home-device-shipment-share-worldwide-by-category/`.

[91] Marah R. Bataineh, Wail Mardini, Yaser M. Khamayseh, and Muneer Masadeh Bani Yassein. Novel and secure blockchain framework for health applications in iot. *IEEE Access*, 10:14914–14926, 2022. doi: 10.1109/ACCESS.2022.3147795.

[92] Shajulin Benedict, P. Rumaise, and Jaspreet Kaur. Iot blockchain solution for air quality monitoring in smartcities. In *2019 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, pages 1–6, Dec 2019. doi: 10.1109/ANTS47819.2019.9118148.

[93] Rahma A Alzahrani, Simon J Herko, and John M Easton. Blockchain application in remote condition monitoring. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 2385–2394, Dec 2020. doi: 10.1109/BigData50022.2020.9377895.

[94] Anjali Rajith, Sakurai Soki, and Mine Hiroshi. Real-time optimized hvac control system on top of an iot framework. In *2018 Third International Conference on Fog and Mobile Edge Computing (FMEC)*, pages 181–186, April 2018. doi: 10.1109/FMEC.2018.8364062.

[95] Maker - Dai stable coin, June 2022. URL `https://developer.makerdao.com/dai/1`. https://developer.makerdao.com/dai/1, [Online; accessed 15. Jun. 2022].

[96] Douglas Steinley and Michael J Brusco. Initializing k-means batch clustering: A critical evaluation of several techniques. *Journal of Classification*, 24(1):99–121, 2007.

[97] I.T. Jolliffe and Springer-Verlag. *Principal Component Analysis*. Springer Series in Statistics. Springer, 2002. ISBN 9780387954424. URL `https://books.google.it/books?id=_olByCrhjwIC`.

[98] MATLAB. *version 9.11.0 (R2021b Update 1)*. The MathWorks Inc., Natick, Massachusetts, 2021.

[99] Fuqing Zhao, Jianlin Zhang, Chuck Zhang, and Junbiao Wang. An improved shuffled complex evolution algorithm with sequence mapping mechanism for job shop scheduling problems. *Expert Systems with Applications*, 42(8):3953–3966, 2015. ISSN 0957-4174.

[100] Michael R Garey, David S Johnson, and Ravi Sethi. The complexity of flowshop and jobshop scheduling. *Mathematics of operations research*, 1(2):117–129, 1976.

[101] K. Jensen and L. Kristensen. *Coloured Petri Nets*. Modelling and Validation of Concurrent Systems. Springer, Berlin, 2009.

[102] Abel Gómez, Ricardo J. Rodríguez, María-Emilia Cambronero, and Valentín Valero. Profiling the publish/subscribe paradigm for automated analysis using colored petri nets. *Software & Systems Modeling*, 18(5):2973–3003, Oct 2019. ISSN 1619-1374.

[103] R. E. Perez and K. Behdinan. Particle swarm approach for structural design optimization. *Comput. Struct.*, 85(19–20):1579–1588, oct 2007. ISSN 0045-7949.

[104] Monica Clemente, Maria Pia Fanti, Giorgio Iacobellis, Massimiliano Nolich, and Walter Ukovich. A decision support system for user-based vehicle relocation in car sharing systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 48(8):1283–1296, 2018.

[105] Riccardo Poli, James Kennedy, and Tim Blackwell. Particle swarm optimization. *Swarm Intelligence*, 1(1):33–57, Jun 2007. ISSN 1935-3820.

[106] Efrén Mezura-Montes and Carlos A. Coello Coello. Constraint-handling in nature-inspired numerical optimization: Past, present and future. *Swarm and Evolutionary Computation*, 1(4):173–194, 2011. ISSN 2210-6502.

[107] J. H. Blackstone, Don T. Phillips, and G. L. Hogg. A state-of-the-art survey of dispatching rules for manufacturing job shop operations. *International Journal of Production Research*, 20(1):27–45, 1982.

[108] Lina Wu, Yusheng Ci, Jiangwei Chu, and Hongsheng Zhang. The influence of intersections on fuel consumption in urban arterial road traffic: A single vehicle test in harbin, china. *PLOS ONE*, 10(9):1–10, 09 2015. doi: 10.1371/journal.pone.0137477. URL `https://doi.org/10.1371/journal.pone.0137477`.

[109] Eun-Ha Choi. Crash Factors in Intersection-Related Crashes: An On-Scene Perspective, September 2010. URL `https://trid.trb.org/view/1083638`. [Online; accessed 15. Feb. 2023].

[110] Elnaz Namazi, Jingyue Li, and Chaoru Lu. Intelligent intersection management systems considering autonomous vehicles: A systematic literature review. *IEEE Access*, 7:91946–91965, 2019. doi: 10.1109/ACCESS.2019.2927412.

[111] Mohammed Al-Turki, Nedal T. Ratrout, Syed Masiur Rahman, and Khaled J. Assi. Signalized intersection control in mixed autonomous and regular vehicles traffic environment—a critical review focusing on future control. *IEEE Access*, 10:16942–16951, 2022. doi: 10.1109/ACCESS.2022.3148706.

[112] B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A. Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(6):4909–4926, 2022. doi: 10.1109/TITS.2021.3054625.

[113] Giuseppe Benedetti, Maria Pia Fanti, Agostino Marcello Mangini, and Fabio Parisi. Application of deep reinforcement learning for traffic control of road intersection with emergency vehicles. In *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 182–187, 2021. doi: 10.1109/SMC52423.2021.9658968.

[114] Maria Pia Fanti, Agostino Marcello Mangini, Daniele Martino, Ignazio Olivieri, Fabio Parisi, and Francesco Popolizio. Safety and comfort in autonomous braking system with deep reinforcement learning. In *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1786–1791, 2022. doi: 10.1109/SMC53654.2022.9945383.

[115] Shengchao Yan, Tim Welschehold, Daniel Büscher, and Wolfram Burgard. Courteous behavior of automated vehicles at unsignalized intersections via reinforcement learning. *IEEE Robotics and Automation Letters*, 7(1):191–198, 2022. doi: 10.1109/LRA.2021.3121807.

[116] Mingfeng Yuan, Jinjun Shan, and Kevin Mi. Deep reinforcement learning based game-theoretic decision-making for autonomous vehicles. *IEEE Robotics and Automation Letters*, 7(2):818–825, 2022. doi: 10.1109/LRA.2021.3134249.

[117] Guofa Li, Siyan Lin, Shen Li, and Xingda Qu. Learning automated driving in complex intersection scenarios based on camera sensors: A deep reinforcement learning approach. *IEEE Sensors Journal*, 22(5):4687–4696, 2022. doi: 10.1109/JSEN.2022.3146307.

[118] Guillen-Perez Antonio and Cano Maria-Dolores. Multi-agent deep reinforcement learning to manage connected autonomous vehicles at tomorrow's intersections. *IEEE Transactions on Vehicular Technology*, 71(7):7033–7043, 2022. doi: 10.1109/TVT.2022.3169907.

[119] Antonio Guillen-Perez and Maria-Dolores Cano. Learning from oracle demonstrations—a new approach to develop autonomous intersection management control algorithms based on multiagent deep reinforcement learning. *IEEE Access*, 10:53601–53613, 2022. doi: 10.1109/ACCESS.2022.3175493.

[120] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL `https://arxiv.org/abs/1707.06347`.

[121] Arpan Kusari, Pei Li, Hanzhi Yang, Nikhil Punshi, Mich Rasulis, Scott Bogard, and David J. LeBlanc. Enhancing sumo simulator for simulation based testing and validation of autonomous vehicles. In *2022 IEEE Intelligent Vehicles Symposium (IV)*, pages 829–835, 2022. doi: 10.1109/IV51971.2022.9827241.

[122] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. RLlib: Abstractions for distributed reinforcement learning. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3053–3062. PMLR, 10–15 Jul 2018. URL `https://proceedings.mlr.press/v80/liang18b.html`.

[123] Gianvito Difilippo, Maria Pia Fanti, and Agostino Marcello Mangini. A consensus protocol for connecting automated vehicles at signal-free intersection. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pages 6568–6573, 2022. doi: 10.1109/CDC51059.2022.9993242.