UNIVERSITÉ DE SHERBROOKE
Faculté de génie
Département de génie électrique et de génie informatique

Allocation des ressources dans les environnements informatiques en périphérie des réseaux mobiles

Resource Allocation in Mobile Edge Computing Network Environments

Thèse de doctorat
Specialité: génie électrique

Amine ABOUAOMAR

Sherbrooke (Québec) Canada

Novembre 2021

JURY MEMBERS

Pr. Soumaya CHERKAOUI
Supervisor

Pr. Abdellatif KOBBANE
Co-supervisor

Pr. João Pedro TROVÃO
Examiner

Pr. Abdelhakim SENHAJI HAFID
Examiner

Pr. Abdelkrim HAQIQ
Examiner

RÉSUMÉ

L'évolution des technologies de l'information entraîne la prolifération des dispositifs connectés qui mène à l'exploration de nouveaux champs d'application. Ces applications demandent une latence ultra-faible, qui ne peut être atteinte par les infrastructures en nuage traditionnelles étant donné la distance qui les sépare des utilisateurs. En rapprochant les ressources aux utilisateurs, le paradigme de l'informatique en périphérie, récemment apparu, vise à répondre aux besoins de ces applications. L'informatique en périphérie s'inspire de l'informatique en nuage, en l'étendant à la périphérie du réseau, à proximité de l'endroit où les données sont générées. Ce paradigme tire parti de la proximité entre l'infrastructure de traitement et les utilisateurs pour garantir une latence ultra-faible et un débit élevé des données.

L'objectif de cette thèse est l'amélioration de l'allocation des ressources à la périphérie du réseau pour offrir une meilleure qualité de service et expérience pour les applications à faible latence. Pour une meilleure allocation des ressources, il est nécessaire d'avoir une bonne connaissance sur les ressources disponibles à tout moment.

La première contribution de cette thèse consiste en la proposition d'une représentation des ressources pour permettre à l'entité de supervision d'acquérir des informations sur les ressources disponibles à chaque dispositif. Ces informations sont ensuite exploitées par le schéma d'allocation des ressources afin d'allouer les ressources de manière appropriée pour les différents services. Le schéma d'allocation des ressources est basé sur l'optimisation de Lyapunov, et il n'est exécuté que lorsque l'allocation des ressources est requise, ce qui réduit la latence et la consommation en ressources sur chaque équipement de périphérie.

La deuxième contribution de cette thèse porte sur l'allocation des ressources pour les services en périphérie. Les services sont composés par le chaînage d'un ensemble de fonctions réseau virtuelles. L'allocation des ressources pour les services consiste en la recherche d'un placement, d'un routage et d'un ordonnancement adéquat de ces fonctions réseau virtuelles. Nous proposons une solution basée sur la théorie des jeux et sur l'apprentissage automatique pour trouver un emplacement et routage convenable ainsi qu'un ordonnancement approprié de ces fonctions en périphérie du réseau.

La troisième contribution de cette thèse consiste en l'allocation des ressources pour les services véhiculaires en périphérie du réseau. Dans cette contribution, les services sont migrés et déplacés sur les différentes infrastructures en périphérie pour assurer la continuité des services. Les services véhiculaires sont en particulier sensibles à la latence et liés principalement à la sûreté et à la sécurité routière. En conséquence, la migration des services véhiculaires constitue une opération complexe. Nous proposons une approche basée sur l'apprentissage par renforcement profond pour migrer de manière proactive les différents services tout en assurant leur continuité sous les contraintes de mobilité élevée.

Mots-clés : Allocation des ressources en périphérie du réseau, fonctions chainées des services, migrations des services.

ABSTRACT

The evolution of information technology is increasing the diversity of connected devices and leading to the expansion of new application areas. These applications require ultra-low latency, which cannot be achieved by legacy cloud infrastructures given their distance from users. By placing resources closer to users, the recently developed edge computing paradigm aims to meet the needs of these applications. Edge computing is inspired by cloud computing and extends it to the edge of the network, in proximity to where the data is generated. This paradigm leverages the proximity between the processing infrastructure and the users to ensure ultra-low latency and high data throughput.

The aim of this thesis is to improve resource allocation at the network edge to provide an improved quality of service and experience for low-latency applications. For better resource allocation, it is necessary to have reliable knowledge about the resources available at any moment.

The first contribution of this thesis is to propose a resource representation to allow the supervisory xentity to acquire information about the resources available to each device. This information is then used by the resource allocation scheme to allocate resources appropriately for the different services. The resource allocation scheme is based on Lyapunov optimization, and it is executed only when resource allocation is required, which reduces the latency and resource consumption on each edge device.

The second contribution of this thesis focuses on resource allocation for edge services. The services are created by chaining a set of virtual network functions. Resource allocation for services consists of finding an adequate placement for, routing, and scheduling these virtual network functions. We propose a solution based on game theory and machine learning to find a suitable location and routing for as well as an appropriate scheduling of these functions at the network edge. Finding the location and routing of network functions is formulated as a mean field game solved by iterative Ishikawa-Mann learning. In addition, the scheduling of the network functions on the different edge nodes is formulated as a matching set, which is solved using an improved version of the deferred acceleration algorithm we propose.

The third contribution of this thesis is the resource allocation for vehicular services at the edge of the network. In this contribution, the services are migrated and moved to the different infrastructures at the edge to ensure service continuity. Vehicular services are particularly delay sensitive and related mainly to road safety and security. Therefore, the migration of vehicular services is a complex operation. We propose an approach based on deep reinforcement learning to proactively migrate the different services while ensuring their continuity under high mobility constraints.

Keywords: Resource allocation at the network edge, service function chaining, service migration.

À mes parents, ma soeur et mes frères.

ACKNOWLEDGEMENTS

I would like to thank my dear mother Amina and my dear father Mustapha who continually supported and encouraged me. I also thank my sister Safaâ, and my two brothers Anas and Mehdi for their love and encouragement. A special thanks to you all!

I would like to thank my advisor Professor Soumaya Cherkaoui and my co-advisor Professor Abdellatif Kobbane for their insightful guidance, advice, and motivation during the development of my thesis.

I would like to thank my lab mates Zoubeir, Abderrahime, Afaf, Boubakr, Hajar, and Oussama for being wonderful colleagues and more than just friends. I would also like to thank all my friends, and especially Soukaina, Manal and Okba for their encouragement and support.

A special acknowledgement for the examination committee for the effort they made to evaluate my thesis. I would like to thank Professor Wael Suleiman, Professor João Pedro Trovão, Professor Abdelhakim Senhaji Hafid, and Professor Abdelkrim Haqiq, for their valuable comments and suggestions.

Thank you all!

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

Acronym	Definition
CC	Cloud computing
DAA	Deferred Acceptance Algorithm
DDQN	Double Deep Q-Network
DQL	Deep Q-Learning
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
EC	Edge computing
eMBB	enhanced Mobile Broadband
EN	Edge node
eNB	eNodeB
ETSI	European Telecommunications Standards Institute
FC	Fog computing
GA	Genetic Algorithm
gNB	gNodeB
IID	Independent and Identical Distributed
IMLA	Ishikawa-Mann Learning Algorithm
IoE	internet of everything
IoT	Internet of Things
ITS	Intelligent Transportation Systems
LTE	Long-Term Evolution
MDP	Markov Decision Process
MEC	Multi-access edge computing
MFG	Mean-Field Game
MG	Matching Game
MILP	Multi-Integer Liner Program
MMDP	Multi-Agent Decision Process
mMTC	massive Machine Type Communication
MSDA	Multi-Stage Deferred Acceptance Algorithm
NFV	Network function virtualization
QoE	Quality of experience
QoS	Quality of service
RA	Resource allocation
RB	Resource block
RL	Reinforcement Learning
RSU	Roadside unit
SDN	Software defined networks
SFC	Service function chain
uRLLC	ultra-Reliable Low-Latency
V2I	Vehicle to infrastructure
V2V	Vehicle to Vehicle

VANET	Vehicular Adhoc networks
VM	Virtual machine
VNF	Virtual network function

CHAPTER 1

Introduction

Throughout the last decade, we have witnessed a fast evolution of information system technologies, and the word 'smart' is part of the names of most of the devices we use in daily life. Smart devices are often delivered with applications to make our daily life more pleasant through the gathering and analysis of data. Nevertheless, processing the increasing amount of data generated by smart devices requires a significant amount of resources. The fifth generation of mobile network technologies (5G) was conceived to address the increasingly high amounts of data generated from different smart devices while offering a high quality of experience (QoE) and quality of service (QoS). Additionally, the envisioned services require low to ultra-low latency and good resource provisioning. A key feature of 5G is the edge computing (EC) paradigm. EC has emerged from the concept of cloud computing to offer resources at the edge of the network near the data source.

EC is shaped to meet the crucial performance needs of data-based ser- vices by moving computational, storage, and communication resources to the end user's vicinity [3, 4]. Unlike cloud computing, which is a homogeneous, scalable, and highly maintained environment, the network edge is heterogeneous and resource-constrained, and achieving high performance at the network edge requires the consideration of constraints such as latency, bandwidth utilization, and the limited capabilities of edge devices. Indeed, promising resource allocation schemes need to be proposed to provide the expected edge efficiency. Additionally, a software-defined network (SDN) is an innovative approach to network management that provides a dynamic network configuration for efficient network performance and monitoring, and it is considered to be a key feature of 5G. An SDN has the role of separating the data plane (i.e., the traffic of the network) from the control plane (i.e., the intelligent part of the network), which has a global view over the network [5]. Another key feature of 5G is network functions virtualization (NFV), which allows the virtualization of entire classes of network functions into pieces of software to enable one to flexibly add and remove network functions [6].

To achieve a promising ecosystem, it is crucial to propose effective and efficient resource provisioning schemes. From an architectural perspective, EC is a heterogeneous ecosystem, in which different EC devices with different purposes and capabilities need to communicate to achieve collaborative data processing. Resource-wise, EC devices that are deployed

among different service providers (and even users in some cases) are resource-constrained in comparison to cloud computing facilities. For such reasons, many research challenges arise regarding resource provisioning at the edge.

From a service deployment perspective and given that services can be formed in many ways, this research area is challenging. SDNs and NFV can be leveraged to create complex array services by linking/chaining multiple network functions across multiple EC facilities. Such a paradigm is referred to as service function chaining (SFC), and as mentioned previously, many challenges arise in this context. These challenges include choosing the appropriate location for network functions, determining proper schemes to chain them, and determining how the network functions should be executed (scheduled). Moreover, a specific use case of SFC resource provisioning is the vehicle edge network, where services must not only be placed and scheduled appropriately but should be migrated due to the high mobility of the vehicles. Hence, it is of great importance to investigate the migration of services at the network edge.

1.1 Objectives

In this thesis, we will investigate resource provisioning at the edge from a range of viewpoints and levels of EC architecture. One part of resource provisioning is overcoming the heterogeneity of re- sources forming the EC. To fulfil this objective, it is highly important to propose a unified representation of EC resources, in such a way that various heterogeneous devices can expose and share information about their resources with other devices. Moreover, there is also a need to provide efficient schemes to wisely and properly exploit the exposed resources to reach the required network performance. Finally, the resource provisioning process itself is resource-demanding and only needs to be performed when required.

On the service side, the entire process of allocating EC resources consists of placing the virtual network functions (VNFs) appropriately in the edge facilities with sufficient resources. Next, the proper route must be found to provide the communication links for VNF data streams. At last, the VNFs must be scheduled, which refers to the various VNFs being executed in the different facilities. This problem is NP-hard [7, 6, 5], so advanced mathematical tools are required to solve it. As an example, game theory is an effective tool to use to formulate the problem as a game or multiple games, where the VNFs are players competing for EC resources. However, solving these games in some formulations requires machine learning due to the complex structure of the games.

In some cases, SFC resource provisioning requires the migration of services due to the users' mobility. Specifically, there is the case of vehicular services, for which the mobility is high, and the services are related to the safety and security of the users. Therefore, the mobility constraints require some services to be moved (i.e., migrated) from one infrastructure to another to guarantee service continuity. It is therefore an attractive use case to study because it requires the consideration of the mobility constraints and low latency requirements for different services.

To accomplish efficiently the main objective of this thesis, we divided it into a set of intermediate objectives. The first intermediate objective is dedicated to resource provisioning at the edge, which requires complete knowledge of over-the-edge resources. This knowledge represents valuable information on how resources are to be provisioned for the required performance. In addition, resource allocation rates must be reasonable so as not to deprive users who benefitted from several resources during a previous allocation cycle. The corresponding research questions for this objective are as follows,

- How can the resource exposure of the heterogeneous edge devices be enabled to formulate an efficient resource provisioning scheme?
- What is the scheme that is most suited to exploiting the full potential of the exposed resources?
- How many times should a resource provisioning operation be performed, and at what cost?

The second intermediate objective is dedicated to resource provisioning for the SFC . SFCs at the edge are delivered by multiple parties sharing the same infrastructure. Indeed, it is crucial to have a resource provisioning scheme for the requirements of the different SFCs, and more importantly, for the VNFs that constitute these SFCs. It is therefore important to investigate this particular problem at all of its various steps. We formulate the following research questions for this intermediate objective,

- How can VNFs be adequately placed over different EC infrastructures to meet the low latency requirements, and at the same time guarantee the proper links allocation?
- How can the placement policy be efficiently exploited to achieve stable resource scheduling for different VNFs over different EC facilities?

Finally, the migration of services at the vehicular edge requires not only a good knowledge of the various resources but also predictive models to initiate the migration process at a convenient time. Furthermore, vehicles' mobility patterns are often not known a priori,

making the system model a bit more complex than that of a static service placement. Consequently, we propose the following research questions regarding this intermediate objective,

- What are the mechanisms that can be used to enable fast service migration at the vehicular edge?
- What is the impact of the mobility patterns on the migration cost?

1.2 Contributions and Originality

The main contributions and originality of this thesis are summarized as follows.

The first contribution offered by this thesis is efficient resource provisioning in edge computing. We propose the following: (1) a resource representation model that enables different edge equipment devices to expose their resources information to solve the problem of heterogeneity. Such a problem is crucial since some devices need to perform tasks that they are not supposed to perform or that they are not able to perform. (2) We also propose a resource provisioning model that leverages the resource information gathered from the previous stage and adequately distributes the tasks among the different edge equipment devices. Finally, (3) we propose a workload-based resource reallocation to reduce the frequency of the resource provisioning process itself. To the best of our knowledge, our contributions represent one of the first attempts, if not the first attempt , to investigate the resource representation and the resource reallocation frequency at the edge of the network. Additionally, we performed simulations on a testbed by running a face recognition application that we deployed.

The second part of this thesis concerns service function chaining resources provisioning. In this problem, the main contributions are clear in the methodology that we adopted to solve such a problem. Resource provisioning at the edge for the SFCs is divided into three parts, namely, (i) VNF placement, (ii) VNF chaining, and (iii) VNF scheduling. The first two parts are often tackled together, while the third is tackled alone under the assumption that (i) and (ii) have already been solved. In our contributions, we propose a mean-field game-based (MFG) formulation of the problem of VNF placement, and due to the complexity of the problem, we adopt a reinforcement learning (RL)-based solution for the MFG. The RL-based approach leverages the iterative learning algorithm of Ishikawa-Mann, and the output of this solution is then used by the matching-based formulation of VNF scheduling. To solve the matching-based game, we used the classic deferred acceptance (DA) algorithm to build our enhanced multi-stage DA (eMSDA). The main particularity of the eMSDA is

that it guarantees stability (i.e., all VNFs will get a share of resources). Both algorithms offer better performance than benchmarks from the literature.

The last contribution is dedicated to a special case of service resource allocation: vehicular services. Additionally, due to the previously discussed three stages for resource provisioning services, in some cases, the services are migrated from one edge infrastructure to another due to mobility constraints. We formulated the problem of service migration as a non-linear integer program and then we linearized it; next, we leveraged CPLEX tools to solve it. The solution to the problem is provided through deep reinforcement learning, specifically, the deep Q-networks (DQN) framework.

1.3 Thesis Plan

This thesis is divided into six chapters and is organized as follows:

- The first chapter presents a brief contextualization of this thesis and its objective, originality, and contributions to resource allocation in edge computing. The first chapter also includes a general overview of the research project and the challenges to be faced.
- The second chapter provides a detailed literature review related to this thesis. Namely, it concerns edge computing, resource allocation in edge computing, service function chaining, and the migration of services at the edge.
- The third chapter is dedicated to proposing a resource representation scheme to cope with the heterogeneity of resources at the network edge. It also provides a resource allocation scheme that leverages the resource representation to adequately distribute the resources among the requests. Additionally, a testbed was deployed to test the performance of the model in the real-world scenario of edge face recognition.
- The fourth chapter addresses SFC resource provisioning at the edge. This chapter tackles the problem from the perspective of VNF placement, chaining, and then scheduling. We propose a game theory-based approach in addition to a machine learning-based solution.
- The fifth chapter deals with the special case of resource provisioning at the vehicular edge. Specifically, in this scenario, the services must be migrated to cope with the high-mobility aspect of the problem.
- The final chapter concludes this research project and provides a set of research directions that can be investigated to enhance this study in future works.

CHAPTER 2

State of the Art

This literature review will be divided into three sub-parts. First, (i) a general overview of EC will be presented, including different architectures and standardization efforts in this field. (ii) Then, there will be a review of the most relevant resource allocation methods at the edge concerning low latency and resource consumption constraints. (iii) Next, there will be a review of the most significant resource allocation methods for SFCs in EC. Finally, (iv) a special use case of SFC, discussing the migration of services at the edge under mobility constraints, will be presented.

2.1 Edge Computing / Multi-Access Edge Computing

2.1.1 Background

Edge computing (EC) is an extension of cloud computing to the edge of the network that consists of placing computational, storage, and networking resources in the vicinity of the data source [8, 4]. EC emerged from cloudlet concepts [3] and has also been referred to as micro-datacentres (mDCs) [9], fog computing (FC) [10], and multi-access edge computing (MEC) [4]. The origins of EC are back in the early 1990s, when content delivery networks (CDNs) emerged to improve the performance of the web [11] by placing copies of content in different locations to reduce the latency. However, the first real EC deployments efforts occurred in early 2013, when IBM, in a joint project with Nokia, deployed an EC platform for 4G/LTE networks. Another project was proposed in mid-2015 by three giants of technology and communication: Vodafone, Intel, and Huawei participated in the Open Edge Computing (OEC) Initiative. These efforts, in addition to many others, gave birth to many standardization efforts, which were mainly incubated by the European Telecommunications Standards Institute (ETSI) [12, 13]. The authors of [4] wrote a holistic review of the proposed architectures and discussed how to integrate them with different communication network generations.

In the literature, EC takes many shapes and forms depending on the use case studied. However, it is possible to group these architectures into three main architectures, which are summarized in Fig. 2.4, Fig. 2.2, and Fig. 2.3. Table 2.1 summarizes the various definitions of EC from different perspectives using the corresponding architectures proposed in Fig. 2.4, Fig. 2.2, and Fig. 2.3. The common point between these architectures is

the cloud infrastructure, which is defined as the last level and as the permanent host for different applications.

Fig. 2.4 illustrates the most abstract shape for the edge computing architecture, a three-layered architecture, where the edge represents the level directly beneath the users. The edge in this case can take the form of any kind of computing and storage facility.

The definition of edge computing in [14, 15] assumes that the edge is the ideal place to perform data processing because there is no need to go through the Internet to access distant cloud infrastructures. These architectures decompose the edge layer into three categories of components, namely, the edge controllers compose the far-edge layer, edge gateways compose the mid-edge layer, and edge servers compose the near-edge layer. The edge controller (the far-edge layer) is considered to be the first place where the data (sensing data, for instance) can be processed. It includes the controlling components, development components, such as algorithms and libraries, and finally, the networking components, which ensure protocol conversion and device access. The mid-edge layer is considered the intelligent part of this edge. This layer is a place for the edge gateways and includes management modules. Equipment registration, access authorization, and communication management modules are examples of management modules. The mid-edge layer also includes storage components where the data aggregation, edge caching, and data pre-processing are performed. Finally, it also includes computing components such as data analysis tools and virtualization hypervisors.

Like the other definition considering the edge to be a place closer to where the data is generated, the definition in [17, 18], as illustrated in Fig. 2.3, not only considers the edge an independent layer but considers it a layer in which users can participate in the formation of the edge layer. As illustrated in Fig. 2.3, users can be data producers or consumers and can offer to participate in the task processing operation by allowing some amount of their resources to be managed by the edge supervisor, if it exists, or just by acting in a distributed manner, as in mobile ad hoc networks (MANETs), for instance.

The cloud computing (CC) paradigm offers almost infinite resources, including elastic and expendable resources, and most of the infrastructure is homogeneous (servers and equipment are from the same supplier with the same performance indices) and within the premises of the same service provider. EC, by contrast, is resource-limited, and heterogeneous equipment, such as routers, switches, edge servers, and in some scenarios, users' equipment (i.e., the equipment is outside the service provider's (SP) premises), is used, due to the placement of the edge within the network architecture. Despite all these

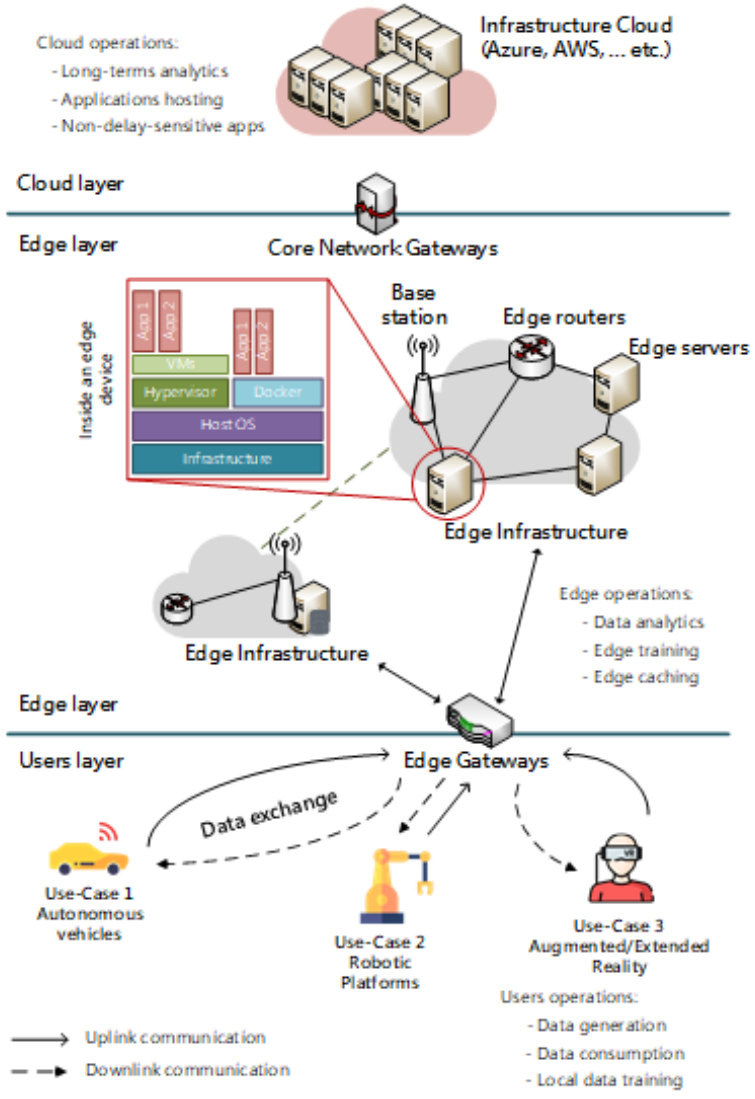

Figure 2.1 Abstract three layered EC architecture.

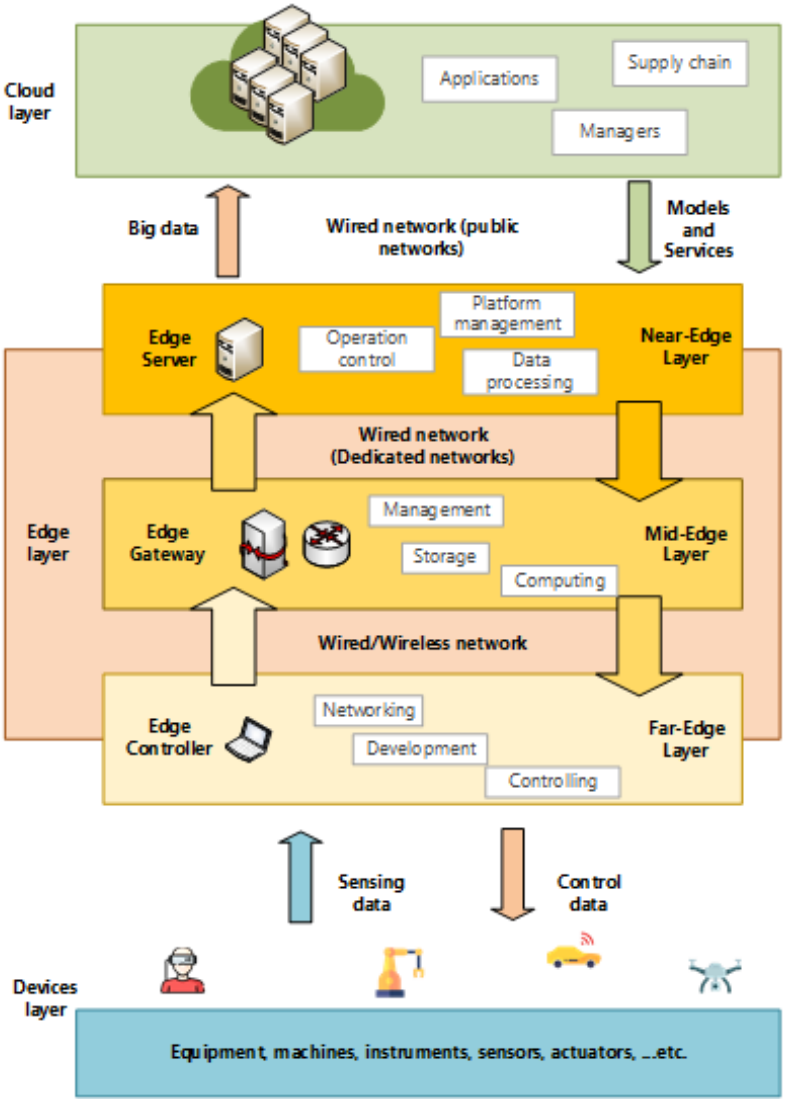

Figure 2.2 Specific three layered EC architecture, where the edge consists of edge controllers, gateways and servers.

Table 2.1 Definitions of EC

Authors	Definition
Satyanarayanan [8] - Zhang et al. [16]	EC represents the paradigm under which computing and storage resources are placed at the network's edge in the vicinity of users. (Fig 2.4)
Tie et al. [14] and CISCO [15]	Edge Computing or simply Edge, brings processing capabilities closer to the data source, without the need to send data to remote Cloud infrastructures or other centralized infrastructures. EC aims to reduce the additional time to access distant cloud infrastructures by eliminating the distance that data needs to travel to centralized sources. (Fig 2.2)
Weisong et al. [17] and Lopez et al. [18]	Edge Computing refers to the paradigm where diverse private or public platforms expose the core capabilities of networks, computing, storage, and applications to provide intelligent services at the network edge closer to where the data is being generated to meet the crucial requirements of applications such as real-time services, data optimization, the intelligence of application, security and privacy. (Fig 2.3)

limitations, EC represents a promising solution for processing data, enabling real-time applications. However, under such constraints, many challenges related to resource provisioning, management, and fault tolerance need to be investigated to enable the real-world use of EC.

In the remainder of this thesis, we will adopt a three-layered EC architecture, and we will adopt the definition described in [8, 4]: we will consider the terms EC and MEC to be the same. In the next section, we will discuss the literature regarding resource allocation at the edge; specifically, we will discuss different types of resources and the objectives that are sought after resource management.

2.2 Resource Allocation at the Edge

Resource allocation is the process of providing available resources from a set of physical devices to different users subject to specific constraints related to the use case. From the point of view of EC, it is the allocation of the EC node's resources to given services requested by the users. A proper resource allocation, depending on the studied resources,

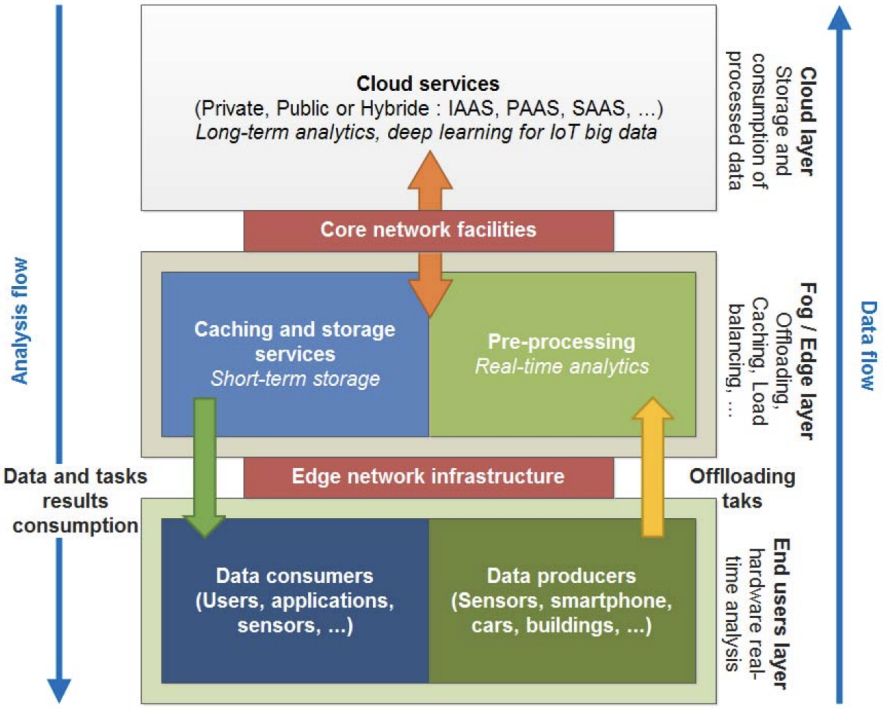

Figure 2.3 Three layered EC architecture.

must guarantee a high QoE and QoS, improved load balancing, and a very low / ultra-low latency.

Before going further into the resource allocation literature, it is very important to identify what types of resources can be allocated within EC use cases. The widely studied types of resources are computational and communication resources [4]. These resources are well investigated in EC, specifically from the perspective of minimizing the latency while maintaining the appropriate utilization of the available resources. Storage resources are another type of resource: they are widely studied from the perspective of edge caching [19]. Since EC is resource-limited compared to CC, it makes sense to talk about energy consumption within the EC context. Resource allocation, from the conducted literature review, can be studied from the perspective of placement and the perspective of scheduling. The placement perspective is about where to process data within the EC facilities and is mostly related to the offloading of tasks [20, 21, 22]. The scheduling perspective, on the other hand, is about how to process data at the EC facilities [23, 24, 25].

In this thesis, we focus on computational, communication, and storage resources, and we can divide the literature into two categories of works. The first category is comprised of a single resource focus: the studies in this category focus only on one type of resource without

considering the others. The second category of literature focuses on multiple resource allocation, in which multiple resources are considered even under complex constraints. For both categories, we will also specify the metrics used to study the performance of the proposed schemes and solutions.

2.2.1 Single-resource focus

Resource allocation in this context is either tackled through specifying a given type of resources (i.e., computation or communication) or tackled as a generic type of resources (i.e., an abstraction of resource type).

The authors of [26] defined a generic measurement unit referred to as the virtual resource value. The work proposed a scheme for resource provisioning that leverages fog computing by measuring the fluctuating relinquish probability of the user, the type of service, the pricing of services, and the variance of the relinquish probability. In addition, the work in [27] considered a generic resource type and proposed a scheme that covers the challenges of resource prediction, customer resource demand estimation, and the pricing for different users based on their application types and requirements. In [28], the authors proposed a platform for collaborative computing that enables proximate devices to act in an ad hoc network fashion to provide diverse capabilities such as cloud services. The proposed scheme is context-aware and exploits the collective capabilities of the participating devices. Additionally, the authors presented a revised version of the Hungarian algorithm to assign tasks among the devices to reduce the load balancing and the latency. The work of [29] focused on the computational resources under the constraints of energy consumption and response time in a general-purpose fog computing architecture. The authors proposed an algorithm to solve the energy and response time minimization problem that improves the energy consumption while keeping a low completion time for the requests. In [30], the authors investigate the resource allocation of computational resources for users that request computational resources from the MEC infrastructure at the base stations, with the aim of reducing the energy consumption. The work proposed a time-slotted MEC system with queuing dynamics for tasks and energy, and then proposed a dynamic throughput maximization algorithm using Lyapunov optimization. It is important to mention that in this work, energy is used as a performance metric.

Other works considered data as resources, such as [31] and [20]. The work in [20], which was based on the follow-me cloud concept proposed in [31], included a migration scheme at the edge with mobility prediction as an enabler.

2.2.2 Multiple-resource focus

The multiple-resource allocation works study two or more types of resources. Computation and communication are investigated jointly to reduce the overall response time. Other researchers studied also computation and communication resource types in addition to storage or energy.

The authors in [32] investigated network resource optimization, specifically the channel selection, which is critical to ensuring reliable resource allocation. The authors also proposed a learning-based channel selection with a focus on the service reliability, energy, backlog, and contention; they used the Lyapunov optimization framework to optimize the channel provisioning strategy, and they used a matching game to solve the channel selection subproblem. The authors of [33] considered the wireless bandwidth and computing resources to decide whether to handle a request in a cloudlet or in the cloud. Another example is the work by Bittencourt et al. [23], who considered the bandwidth between the cloud and a cloudlet, as well as cloudlet processing capabilities, when evaluating different scheduling strategies.

Computational resources can be addressed at a physical level, for example, when discussing CPU cycles, or at a conceptual level, such as when virtual machines (VMs) are used as resource elements. In the surveyed articles, Wang et al. [34] considered CPU cycles, Singh et al. [25] considered millions of instructions per second (MIPS), and Rodrigues et al. [35] considered the number of processors per cloudlet. At a conceptual level, Zamani et al. [36] considered different computing resources based on the average number of tasks completed per unit of time, and Plachy et al. [22] allocated computational resources in the form of VMs. Sometimes the VMs are used to ensure that a task can run, given enough underlying resources, in the device hosting the VM [37].

Instead of using VMs, Yi et al. [38] adopted lightweight OS-level virtualization and a container technique, arguing that resource isolation can be provided at a much lower cost using OS-level virtualization. They also point out that the creation and destruction of container instances are much faster and thus enable the deployment of an edge computing platform with minimal effort. In [39], the authors formulated the resource allocation as a mixed-integer nonlinear programming problem to optimize computational task offloading and communication resources. The problem was relaxed through the Lyapunov optimization framework, and then solved using a convex decomposition approach and a matching game. In their proposed algorithm for optimal task offloading, the decision concerning resource allocation is made at each time slot, which may lead, in some scenarios, to the modification of the overall decision to offload a small portion of the tasks.

2.2.3 Conclusion

By far, the majority of the works surveyed on resource provisioning in EC investigate computational and communication resources. As summarized in Table 2.2, few papers consider storage resources, while energy is considered a performance metric. Moreover, in the works where storage and/or energy are addressed, these resources are studied either independently or considered to be a metric (as in energy); the optimizations of these types of resources are carried out independently of each other. Additionally, resource provisioning at the edge is always tackled for a given type of resources; for a specific use case, it is meant to reduce the response time, maximize the throughput, or reduce the energy consumption, without considering the composition of the resources at the edge and the nature of the infrastructure that is used. Moreover, it is worth considering the resource allocation frequency, which represents the number of times a resource allocation operation takes place. In the literature, nearly all of the proposed works assume that this operation takes time at each time slot. However, it is possible to reduce this time through the investigation of the composition of the EC infrastructure/nodes. Even though the network edge is essentially formed by heterogeneous devices and these devices are situated away from the service provider's facilities, no work in particular, to the best of our knowledge, has addressed the representation of edge resources.

Resource focus	Ref	Tools	Resources	Contributions & Assumptions	Parameters
Single resource	[34]	Iterative algorithm	Networking	– Minimizing operators' cost while meeting the task's latency constraints – Mobile cloud completes tasks for the mobile user then transmits the outcome back to the users using the C-RAN – Cost-effective resource allocation between MCC and C-RAN	Cost
	[25]	Custom algorithm	Computational	– Providing balance performance and data privacy/security constraints for different applications – Real-time scheduling solution to support the integration of a micro-datacenter and cloud datacenter	– Throughput – Success ration
	[26, 27]	Custom algorithm	Generic type of resources	– Resource prediction based on the resource demand for different applications and type of requirements – Measuring the fluctuating relinquish probability of users and services	– Latency – Generic measurement unit (virtual resource value) – Pricing of resources
	[28]	Hungarian algorithm for task assignment	Generic type of resources	– Context-aware scheme for tasks offloading – Collaboration between the equipment – Tasks assignment is performed using the Hungarian algorithm	– Latency – Load balancing

Single resource	[29]	– Integer Linear Programming – Custom algorithm	Computational resources	– General purpose fog architecture – Energy and time efficient task offloading and resource allocation within IoT-fog-cloud architecture – Leveraging advantages of both fog and cloud and studied the resource allocation as an energy and time cost minimization problem – Resources allocation is performed in two stages: (i) computation offloading selection and (ii) transmission power allocation	– Energy consumption – Latency
	[30]	Lyapunov optimization tool	Computational resources	– Lyapunov optimization for queuing dynamic – Time-slotted queuing dynamic	– Latency – Energy consumption – Throughput
	[31]	Custom algorithm	Data as resources	– MEC services follow their users during their movement by migrating all of services components – Migration decision is based on mobility constraints and network policies of the operator (e.g., P-GW relocation operation)	– QoE – Latency – Migration cost
	[20]	Multiple Attribute Decision Making	Storage	– Proposition of a cohesive end-to-end architecture based on information-centric networking together with MEC – Enhancement of the migration cost and content caching in MEC – The proposed algorithm reaches up to 500% in content availability when requested by a user	– Latency – Users' satisfaction – Cache hit ratio

Multiple resources	[32]	– Machine learning – Lyapunov optimization tools – Matching game	– Networking – Computational – Energy	– Optimization of channel selection for efficient and reliable distribution of tasks	– Throughput – Energy pricing – Latency
	[40]	– Semi – Markov decision process – Integer Linear Programming	– Computational – Networking	– A multi – resource allocation scheme for the cloudlet environment to overcome bottlenecks of cloudlet's computational resources, and the networking resource between users and cloudlets – Admission control optimization with a focus on services reliability	– Latency – Queue's length – Energy backlog
	[23]	– Custom algorithm	– Networking – Computational	– An introduction to the scheduling problem in the hierarchical composition of fog and cloud computing – Scheduling policies are designed to deal with various applications based on users' demand, benefiting from fog proximity to the users and cloud computing properties	– Number of application modules migrated
	[36]	– Custom algorithm – Experimental testbed	– Storage – Networking – Computational	– Leveraging SDN capabilities to control data transport services aiming to dynamically establish data routes to exploit computational capabilities located along the network path	– Acceptance ratio – Admitted jobs – Latency – Completion time

Multiple resources

[37]	– Custom algorithm	– Networking – Computational – Storage	– Proposing a model that captures the heterogeneity of cost and capacity of a MEC network – The model bridges MEC and DCC paradigms by modeling multiple types of resources among the network and serves both mobile devices but client within and beyond the network perimeter – The algorithm is seen as an application placement scheme that considers features such as network links capacities, user latency threshold, and user mobility	– Execution cost – Round trip time – Resource consumption
[38]	– Mixed integer non – linear programming problem (MINLP) – Sequential Quadratic Programming – Branch and bound	– Networking – Computational – Storage	– Leveraging containers to significantly reduce the latency due to the fast startup and destruction of different services – The proposed solution ensures low latency and flexibility in using hierarchical resources from client nodes – Computation offloading default choice is set to be the edge infrastructure	– Execution time – Latency – Throughput – Tasks placement

Multiple resources	[39]	– Mixed – integer nonlinear programming – Lyapunov optimization	– Energy – Computational – Networking	– Decision and resource allocation is made at each time slot, which may lead in some scenarios to modify the overall decision to offload a small portion of the tasks – Offloading decision and computation resource scheduling is reached through decomposition methods, while radio resource allocation is addressed by matching game and geometric programming – Proposing an implementation in a semi – distributed way with low complexity	– Energy efficiency – Queue length – Latency
	[41]	Integer Linear Programming	– Computation – Networking	– Independent from time migration cost of VNFs – The VNFs tolerate a given threshold for the latency violation – The placement of VNFs takes place at any host of the network – The model adapts to changing network dynamics, demands and mobility of users	– Latency – Num. of violations

Table 2.2 Summary of contributions for resource allocation in edge computing

2.3 Service Function Chaining Resource Provisioning at the Edge

2.3.1 Background

In this section, we will discuss various SFC approaches and introduce the different topologies and the definition adopted for this thesis. Table 2.3 summarizes the most relevant literature for service function chain resource allocation in EC.

With the emergence of SDNs and NFV, network functions (e.g., firewalls, network monitoring, and compression network functions) are being deployed across multiple geographically dispersed computing infrastructures, namely, cloud and edge facilities. Chaining these network functions to form a full end-to-end network service is a complex, time-consuming, and cost-intensive operation. Service function chaining (SFC) is a technique that allows different service functions to be interconnected to form a single service or multiple services that allow operators to take advantage of a fully virtualized software environment [42, 5]. SFC provides a flexible and cost-effective replacement for the current static environment, which consists of hardware-based service providers.

As mentioned before, NFV is a fully virtualized concept in which abstract network functions are deployed over an infrastructure. Therefore, resource allocation for these functions is a necessity.

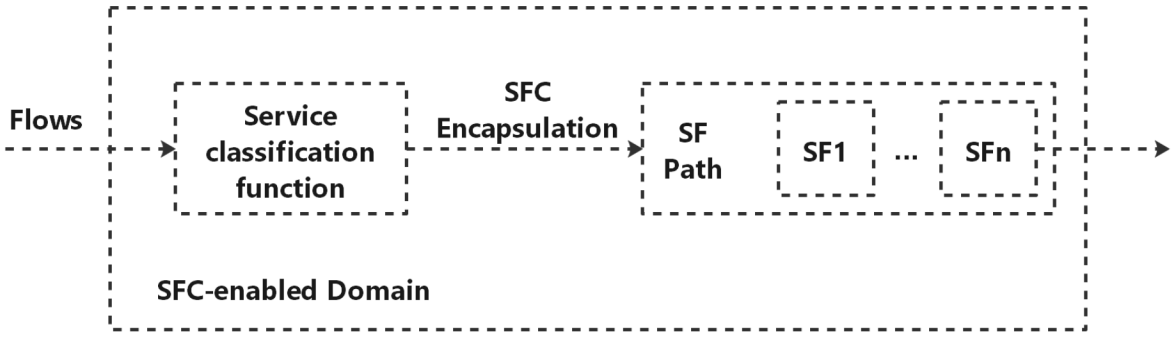

Figure 2.4 Standard SFC architecture [1].

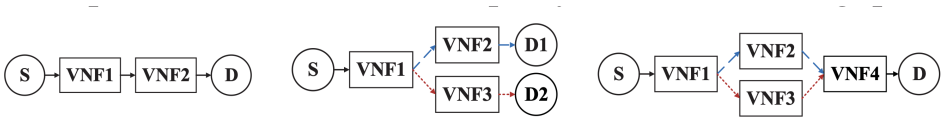

Figure 2.5 Different SFCs topologies [2].

By way of illustration, Fig. 2.5 illustrates different SFC topology types. The right side of the figure shows the linear topology, which is the most commonly adopted topology for

SFC. The middle of the figure shows the split topology, where the egress of a VNF is split onto two different VNFs. Finally, the right side of the figure shows the split and merge topology, where the ingress of a VNF takes two or more egress results of other VNFs.

From the perspective of EC, the SFC resource provisioning process consists of three stages. The first stage is related to the placement of the VNFs within the edge network and the resource allocation at the hosting infrastructure. The second stage deals with the routing of the data flow among the VNFs and links the allocation between different EC facilities. The third and final stage is the scheduling, which represents the order of the execution of the VNFs within the EC infrastructure. In the literature, the first two stages are often tackled together and are referred to as SFC composition. It is justified to tackle both placements and routing at the same time due to the relationship between them. Indeed, sometimes, it is important to check for the capacity of the links before placing the VNFs. Meanwhile, the scheduling is tackled independently. In this literature review, we will divide the literature into two categories: (i) SFC composition and (ii) SFC scheduling.

The first category, which is SFC composition, consists of first placing the VNFs in adequate EC facilities, and then linking these VNFs through virtual links allocation. Let us mention here that an SFC can be deployed over a set of EC facilities that are distributed over a geographical area and communicate over physical links. The VNF placement problem seeks to find proper locations for VNFs (either VMs or containers) within the EC facility, where each EC facility has various physical devices serving as VM/container hosting facilities depending on the demand. The benefit of deploying VNFs at the edge is the proximity to the users, which enables the low latency feature.

2.3.2 SFC Composition

The proper placement of VNFs is driven by the optimization of resource utilization and the overall network performance while maintaining a minimum cost (in terms of energy consumption, latency, and financial costs) or reducing service level agreement (SLA) violation penalties. Although these parameters are quite suitable for NFVs, most of the work that has been proposed in this context concentrates exclusively on reducing the cost of deployment and improving QoS and availability, overlooking the constraints associated with the EC facility itself.

As demonstrated in Fig. 2.6, the VNFs are deployed over different EC nodes and linked virtually over the physical links between the EC nodes. Note here that the orchestration of the VNFs can take place on the SDN controller, since it is a central entity with a global view of the network, as detailed in [4]. The orchestration can also take place on

the EC nodes independently; however, in this case, it is required that the EC nodes exchange information about their resources to allow dynamic VNF placement, routing, and scheduling, which is also exhaustively discussed in [4]. This exchange can be made possible through the resource representation proposed in [43, 44].

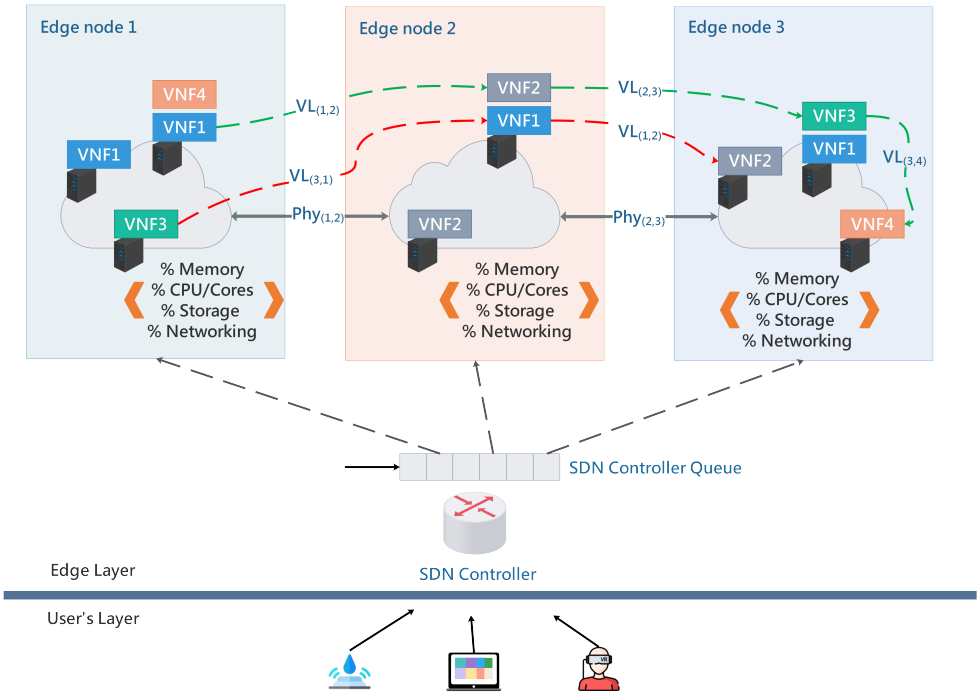

Figure 2.6 Standard SFC architecture [1].

In [45], the authors studied the online provisioning for NFVs by defining the optimal SFC composition policy. The work divided the instantiated VNFs into mandatory and best-effort network functions. They proposed a primal-dual solution to approximate the optimal solution. The authors of [46] proposed a uRLLC-aware VNF placement scheme as an optimization problem to minimize the access latency and maximize the service availability. The authors proposed a genetic algorithm to solve the optimization problem and compared the results to the optimal solution. In [47], the authors proposed joint optimal decision-making for VNF and CPU resource allocation at the hosting facility using a queuing-based system model, taking into account all the entities of the 5G networks, such as arbitrary VNF graphs and flexible CPU usage; it has the ability to instantiate the same VNF multiple times. The solution is given based on the placement heuristic to make joint VNF placement and CPU allocation decisions. The work of [48] investigated the routing problem for requests through the consideration of dynamic VNF placement and multiple resource constraints within an NFV-enabled SDN network. The authors divided the problem into two subproblems formulated as integer linear programming problems. The first subproblem is the dynamic placement of VNFs on the networks and the second

subproblem is the request routing subproblem. As in the majority of the literature, the metrics investigated in this work are the delay, packet loss, and jitter. To solve the subproblems, the authors proposed an auxiliary edge-weight graph-based algorithm. The authors of [49] proposed a centralized scheme allowing SFC partitioning and embedding over multiple domains under the constraints of the global infrastructure visibility. The SFC placement was formulated as a multi-objective optimization problem through the physical programming method. Such a method has the ability to describe the decision maker's preferences using meaningful parameters and then propose an adequate method in addition to a scalable heuristic solution.

2.3.3 SFC Scheduling

The scheduling stage can be defined as the sequence in which the VNFs that are co-hosted on the EC facilities will be executed to meet the minimum overall execution time requirements (i.e., the delay experienced between executing the ingress of the VNF and the completion at the egress of the VNF). This particular problem is NP-complete and thus cannot be solved in polynomial time [7, 50, 51].

Fig. 2.7 illustrates the scheduling of two SFCs with their VNFs hosted on three different nodes. In the literature, different performance parameters were used, but the most investigated parameter is the scheduling delay. The scheduling delay is often considered to be, as illustrated in 2.7, the sum of the processing time at the EC facility, the queuing time, and the transmission time, which depends on the quality of the links and their capacities.

The scheduling problem in the literature is frequently formulated as a mixed-integer linear program and then reduced to a less complex formulation. For instance, in [52, 53], the authors reduced the problem to a Markov decision process, which was solved using reinforcement learning. The learning results provide the best scheduling policy through continuous interaction with the networks. Meanwhile, in [54], the authors jointly investigate the scheduling and mapping of VNFs to enhance the performance of service provisioning. The problem was relaxed using a two-stage algorithm. The VNFs were mapped and scheduled on an SFC through the greedy minimization of the waiting time of VNFs. When the requirements in terms of the delay are difficult to satisfy, the algorithm reschedules the existing VNFs. Such an approach guarantees the flexibility of VNF placement and increases the service acceptance ratio. Both advantages, which are studied in this work, are rarely studied in the literature. The authors of [7]proposed a matching game-based approach to properly schedule the execution of the VNFs within different hosting infrastructures. Although the problem is NP-complete and the resolution takes a non-polynomial time, the authors proposed a solution that guarantees stability but not optimality. The work

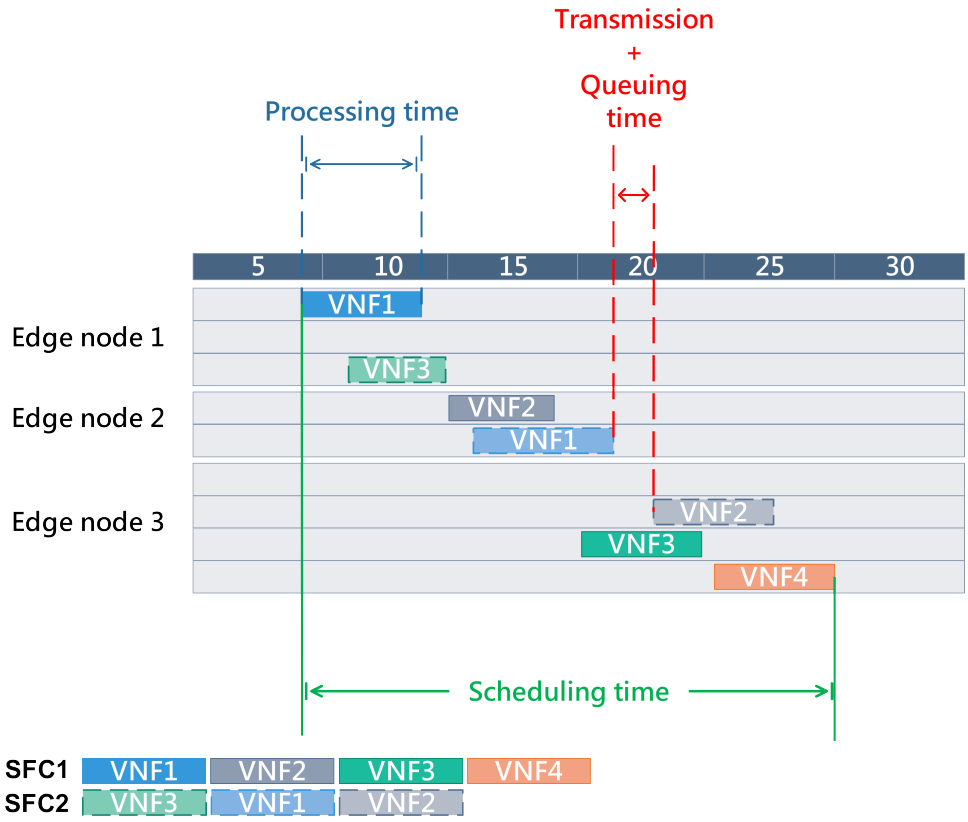

Figure 2.7 Standard SFC architecture [1].

proposed a one-to-one matching game where the VNF resources are allocated on different nodes forming the networks. Within the same context, [55] enhanced the intelligent part of the NFV infrastructure (NFVI), which is the NFV orchestrator (NFVO), by providing a latency-aware placing and routing scheme. In the proposed scheme, VNFs are initially allocated to the proper entity, and then they are scheduled on their instance locations, scaled, or migrated, and destroyed based on the network status. The problem of scheduling in [56] was tackled as a flexible job-shop problem to minimize the overall scheduling latency. The problem was solved through an RL-based algorithm. Additionally, this approach can detect the fluctuations of the MEC infrastructure and adequately adapt the SFC scheduling.

2.3.4 Conclusion

From the conducted literature review, the SFC resource allocation problem at the edge is divided into three subproblems, namely, the placement, routing, and scheduling of the VFNs, which represent the vital component of the SFC. However, in the literature, these subproblems are tackled independently, and the main contributions investigate the placement and the routing subproblems, with very few contributions to VNF scheduling.

This is due to the aforementioned issue that the majority of the contributions focus on the problem of resource allocation without considering the composition of the resources; they assume that a given resource is a homogeneous entity. However, at the edge, the resources can be distributed according to various forms and shapes, as discussed previously in the description of the EC architecture. Another important aspect within this context is the request arrival model, which directly impacts the way that resources are allocated in different EC facilities. In the literature, request arrival models are not well investigated, and most of the contributions adopt existing queue systems, without considering the SFC constraints related to the routing and composition of chains [6]. However, the various SFC architectures can take many forms, as previously discussed, and thus, they require adequate request arrival models. The problem of SFC resource allocation in the EC context should be studied as an integral problem, since the scheduling subproblem requires knowledge of where the VNFs are placed and how they are routed.

Ref	Tools	Resources of focus	Contributions & Assumptions	Parameters
[45]	Primal-method	– Networking – Computational	– Two kind of VNFs, the mandatory and the best effost VNFs – Requests arrives in batches having the same requirements of resources – No precision over the request arrival model	– Throughput – Profit value (generic; adaptable to different parameters)
[46]	Genetic algorithm	– Computational – Networking	– Leveraging genetic algorithm and comparing the results to the optimal – URLLC-aware scheme under latency constraints	– Service availability – Latency
[47]	Heuristics	Computational	– Joint optimal decision making for the VNFs and CPU resource allocation – Queuing-based model taking into consideration all the 5G entities – Stochastic requests arrival – Account for the flexible CPU allocation to VNFs running on the same host	Latency
[48]	Integer Linear Programing	– Networking – Computational	– Auxiliary edge-weight graph-based algorithm to solve the placement and chaining problem – Joint dynamic VNF placement under multiple resources and QoS constraints – Leveraging Lagrange relaxation to cope with the delay and the packet loss to dynamically place and chain VNFs	– Latency – Packet loss – Jitter

[49]	– Multiple-Integer Linear Programing – Physical Programming – Meta-heuristics	Networking	– Proposition of a centralized scheme to partition and embed SFCs over multiple domains while ensuring a global infrastructure visibility – Flexible description of the decision makers' preferences using meaningful parameters	– Latency – Generic cost – Resources consumption
[52, 53]	– Multiple-Integer Linear Programming – Markov Decision Process – Reinforcement Learning	Generic resource	– Minimizing the overall completion time of services and satisfying differentiated E2E delay requirements – MDP formulation for the problem of placement and scheduling – RL-based solution to find the optimal placement and schedule for VNFs on different infrastructures	– Latency – Makespan – Average reward – Scalability
[54]	– Greedy algorithms – MILP	Generic resource	– Investigation of VNFs mapping and scheduling – two-stage online algorithm to address the NP-hardness of the MILP – Proposition of a delay-aware rescheduling scheme in which selected existing VNFs are remapped and rescheduled – Minimizing the waiting time greedily	– Acceptance ratio – Latency – Generic cost (revenue)
[7]	Game theory	– Computational – Networking – Memory	– Focusing on the stability rather considering the optimality – Scheduling different VNFs over different infrastructures	– Execution time – Completed VNFs – Resource consumption

[55]	Testbed implementation	Generic resource	– Enhancing the orchestrator (NFVO) through providing a latency-aware placing and routing scheme – VNFs allocate resources on the proposer infrastructure entity then scheduled on their instance locations, scaled, or migrated, and destroyed based on the network status	– Resources consumption – Latency
[56]	RL-based solution	Generic resource	– Ability to detect fluctuations of MEC infrastructure to adequately adapt the SFC scheduling – A job-shop problem. formulation to minimize the overall scheduling latency	– Latency – Resources consumption – Completion of SFC
[57]	Linear Programming	– Computational – Networking	– Considering reliability, delay, and resource allocation for SFC – Proposition of an SFC sub chaining method to deal with reliability – ILP-based formulation for SFC placement using matching algorithm – Providing new-optimal polynomial solution for the SFC resource allocation	– Reliability – Latency – Cost – Resource consumption
[58]	– Heuristics – Integer Non-Linear Programming	– Computational – Networking	– Proposition of a configurable service allocation scheme for VNFs embedding and routing – Greedy algorithm-based solution	– Resource consumption – Latency

[59]	Integer non-linear programming	– Networking – Computation	– Enhancing network resources usage as well as the minimization of end-to-end delay for network services – Proposing a genetic algorithm with improved crossover and mutation operations – High computational cost	– Completed services – Resources cost – Resource consumption

Table 2.3 Summary of important works investigating resource provisioning for SFC.

2.4 Service Migration at the Edge: Vehicular Network Use Case

As discussed before in the section on service function chaining, functions can be placed, chained, and scheduled. However, services under mobility constraints need to be moved and migrated to follow the mobility of the users.

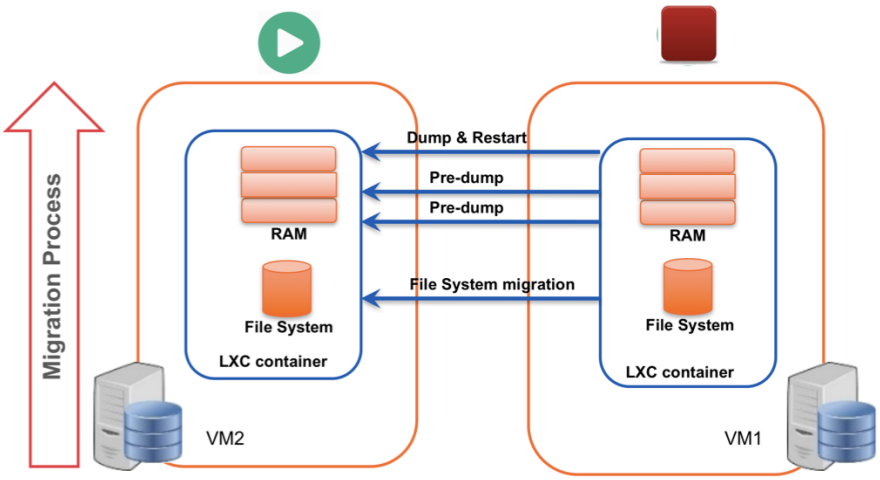

Figure 2.8 Migration process

The authors of [60] provide a QoE-aware system to ensure service continuity in a mobile cloud computing environment. This scheme is based on a buffer usage threshold scheme that classifies new requests from mobile users. The suggested scheme protects the migrated service from the traffic fluctuation of the new demands. Moreover, the cloud server can change the buffer threshold dynamically for different categories of requests. In [61] and [62], the authors proposed the Follow-Me Chain algorithm to address the problem of SFC placement and migration in EC networks. In particular, the work studied the problem of cross-MEC handovers to achieve higher user satisfaction in high-mobility situations. This problem is NP-hard, and the authors proposed an integer programming formulation of the problem, which is solved using the Follow-Me Chain algorithm. The authors of [63] studied the problem of VNF relocation in a cloud infrastructure under mobility and resource heterogeneity constraints. Specifically, the authors investigated the effect of the relocation process on the service latency and the frequency of VNF relocations (i.e., the number of times the same VNF is moved from one cloud to another). The relocation problem was formulated as a MILP problem and solved using an ant colony optimization meta-heuristic approach. In a similar context, the works of [64] and [65] proposed an evaluation of three container-based schemes for VNF migration as a way to achieve service continuity. In

particular, the schemes consider two cases of mobility patterns, respectively, the a priori known and unknown mobility patterns. For the a priori known pattern, temporary file systems and driverless migration are discussed, but the work emphasizes the unknown mobility pattern: the authors proposed a solution that involves storing the container file system in the system images in a shared pool. Additionally, the authors of [66] presented a holistic overview of different service migration methods, namely, full slice migration, and partial slice migration through slice breathing, splitting, and merging.

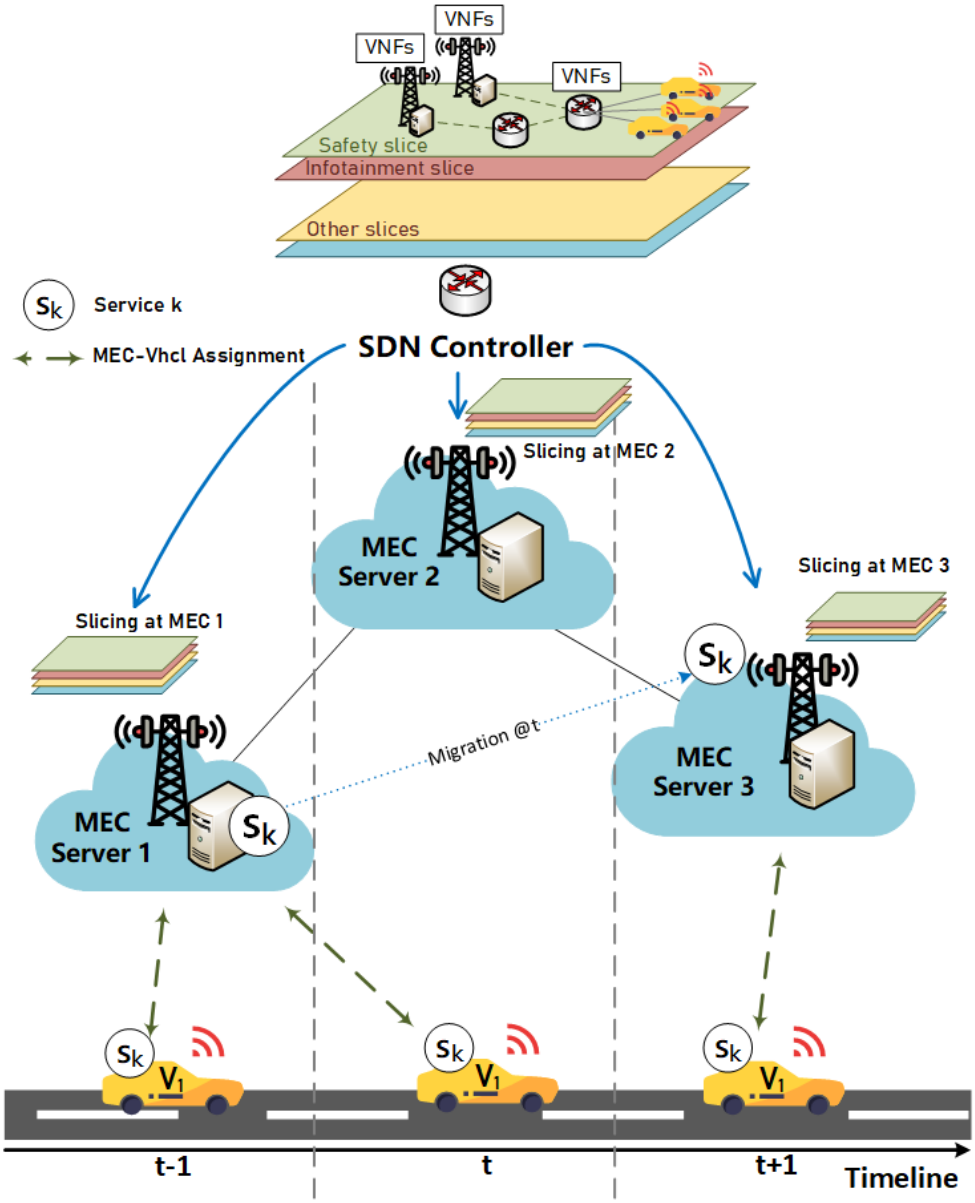

Figure 2.9 Migration process at the vehicular edge

From the resource perspective, previous works focus only on a single type of resource, either communication or computational resources, and occasionally they are studied in a

joint formulation. Storage represents an important resource, especially in caching-related scenarios, which makes it a resource worth investigating due to the importance of edge caching in 5G and beyond network architectures. From the structure perspective, SFCs in the literature are often considered to be linear, which means that the data flow traverses a set of VNFs in a pre-set order. However, in some cases, when a VNF is migrated from one MEC node to another, it is necessary to consider that a VNF can take place in two different SFCs. For instance, the splitting and merging of the data flow may occur during the execution of the SFC. Consequently, the migration of one VNF from one MEC node to another may impact the performance of other SFCs. From the diversity perspective of SFCs, VNFs from different service providers may be deployed in the same EC facility, which requires schemes to deal with the heterogeneity of the SFCs. Therefore, it is necessary to consider the diversity of the SFCs and VNFs as a key pillar parameter in shaping solutions for VNF migration. Finally, there are few works that investigated cases in which the mobility pattern is unknown, and most of the contributions assume that the mobility pattern is a priori known. However, the mobility pattern should be studied from different angles and through extensive complex scenarios to ensure the efficient continuity of the service under extremely unpredictable use cases.

CHAPTER 3

Resource Provisioning in Edge Computing for Latency Sensitive Applications

Date de parution: 18 January 2021
Revue: IEEE Internet of Things Journal
Titre français: Approvisionnement des ressources dans l'informatique de périphérie pour les applications sensibles à la latence.

Resumé français
Les applications IoT à faible latence, telles que les véhicules autonomes, les dispositifs de réalité augmentée/virtuelle et les applications de sécurité, nécessitent des ressources de calcul élevées pour prendre des décisions instantanément. Cependant, ces types d'applications ne peuvent pas tolérer de confier leurs opérations à une infrastructure en nuage en raison de la latence qu'elles subissent. C'est pourquoi l'informatique de périphérie est introduite pour permettre une faible latence en rapprochant le traitement des tâches des utilisateurs à la périphérie du réseau. La périphérie du réseau est caractérisée par l'hétérogénéité des dispositifs de périphérie qui la composent. Par conséquent, il est donc crucial de concevoir de nouvelles solutions qui tiennent compte des différentes ressources physiques de chaque équipement. Dans cet article, nous proposons un schéma de représentation des ressources, permettant à chaque dispositif de périphérie d'exposer les informations relatives à ses ressources au superviseur du nœud de périphérie par l'intermédiaire des interfaces de programmation d'applications proposées par ETSI. L'information sur la ressource de chaque équipement est exposée au superviseur du nœud périphérique chaque fois qu'une allocation de ressource est requise. À cette fin, nous utilisons l'optimisation de Lyapunov pour allouer dynamiquement les ressources aux équipements. Afin d'étudier la performance du modèle proposé, nous avons effectué des simulations théoriques et expérimentales intensives sur un déploiement d'essais pour valider le schéma proposé et son impact sur les différents paramètres du système. Les simulations ont montré que l'approche proposée surpasse les autres approches de référence et fournit une faible latence et une consommation optimale des ressources.

3.1 Abstract

Low-Latency IoT applications such as autonomous vehicles, augmented/virtual reality devices, and security applications require high computation resources to make decisions on the fly. However, these kinds of applications cannot tolerate offloading their tasks to be processed on a cloud infrastructure due to the experienced latency. Therefore, edge computing is introduced to enable low latency by moving the processing of the task closer to the users at the edge of the network. The edge of the network is characterized by the heterogeneity of edge devices forming it; thus, it is crucial to devise novel solutions that take into account the different physical resources of each edge device. In this paper, we propose a resource representation scheme, allowing each edge device to expose its resource information to the supervisor of the edge node through the mobile edge computing application programming interfaces proposed by European Telecommunications Standards Institute. The information about the edge device resource is exposed to the supervisor of the EN each time a resource allocation is required. To this end, we leverage a Lyapunov optimization framework to dynamically allocate resources at the edge devices. To test our proposed model, we performed intensive theoretical and experimental simulations on a testbed to validate the proposed scheme and its impact on different system parameters. The simulations have shown that our proposed approach outperforms other benchmark approaches and provides low latency and optimal resource consumption.

3.2 Introduction

The evolution of technologies and communication systems in the last decade gave birth to new Internet-based applications and services that require low latency. Cloud computing was proposed as a powerful technology to enable low latency requirements and optimized resource consumption by offering many advantages such as high availability, scalability, and reduced costs [67]. However, to meet the latency requirement in tasks processing, a remote cloud computing infrastructure may not be suitable for latency-sensitive applications such as industrial process monitoring, automated vehicles, virtual/augmented reality, surveillance and security, and human emotions detection. Therefore, the proximity to the processing infrastructure is the key when it comes to reducing the experienced latency [8]. Such infrastructure is located at the edge of the network, enabling to address of the limitations of cloud computing infrastructure through distributed and low latency computation.

EC has emerged as a promising solution to lower the experienced latency by distributing the processing, communication, and control closer to where the data is generated [4, 8, 68]. EC, therefore, extends cloud computing by providing applications with the computational, storage, and communication resources at the edge of the network. The main standard-

ization efforts in EC were initially proposed by the Industry Specification Group (ISG) within the ETSI [13, 69, 70, 71, 12]. Unlike cloud computing, which is scalable and highly maintained, EC is more heterogeneous and resource-constrained. The edge of the network is built-up on edge nodes (ENs). ENs are the aggregation of heterogeneous edge devices (EDs) such as edge routers and switches, edge servers, sensors, and even some end users (EUs) equipment (e.g. laptops and smartphones). From the location point of view, the EDs are located outside the premises of the operator which makes the maintenance and management difficult. From the architectural point of view, EDs could have different hardware architectures and have different capabilities. For instance, an EN could have several edge routers and edge servers. Some edge servers may have high processing resources or specialized ones such as graphical processing units (GPUs), which make them suitable to perform intense computation tasks. Some other edge servers could have big storage space to serve as caching and storing infrastructure at the edge of the network. Finally, edge routers may have moderate processing resources but powerful networking resources, allowing them to handle more traffic than others.

The disparity of available resources at the edge of the network requires efficient and optimal resource provisioning, especially because of the limited available resources compared to the cloud. However, adequate and optimal resource provisioning requires a good knowledge of available resources. The unavailability of information about EDs capabilities at the ENs can increase the latency and cause additional delay because some EDs could be asked to perform tasks they are noted best suited for. Therefore, providing the information about ED's resources at an EN will make the process of tasks distribution more optimal. Thus, any supervising entity that oversees tasks distribution can build suitable resource provisioning schemes based on this information.

In this paper, we propose a resource representation model to characterize the different physical resources of the EDs. There are four resources of interest in this paper: processing, storage, memory, and networking. Since each ED is aware of its different available resources, the ENs will be capable of discovering the resource information of the EDs independently of the architecture or manufacturer. In the literature, most of the works investigate the problem of edge resource allocation considering only communication and computation resources [72, 40, 73, 74, 75, 32, 76, 77]. A few works also consider other types of resources but without specifically targeting the delay minimization problem [78, 79, 80]. In this work, we formulate the resource allocation optimization problem as a Lyapunov optimization with the aim of minimizing the overall applications experienced latency by

jointly considering all the above-mentioned resources. The main contributions of this paper are summarized as follows:

- To characterize the physical resource in an EC environment, we propose a resource representation model which adheres to the ETSI standard [13, 69, 70, 71, 12]. The model represents different kinds of resources (processing, storage, memory, and networking) of EDs and their capabilities.
- We propose a resource allocation scheme based on Lyapunov optimization to minimize the experienced delay through the study of the service queues dynamics at the ENs.
- We propose an algorithm to optimize the frequency of resource allocation and information exchange. The proposed algorithm is based on the workload at each EN.
- We simulate the proposed scheme under different configurations of parameters. We also use a testbed inspired by the work in [81] to experiment the proposed scheme. We evaluate the latency, the consumption of different resources at the EDs, and the queue evolution over time. The simulations show that our proposed approach outperforms other benchmark approaches and provides low latency and optimal resource consumption.

The remainder of the paper is organized as follows. The system model is first detailed in Section II. Subsequently, we formulate the problem of latency minimization, the dynamic of the queues, and the different requirements in Section III. In Section IV, we detail the proposed resource representation scheme, the resource allocation scheme, and the task distribution model. The performance of the system is simulated in Section V. We discuss the related works in Section VI. And we conclude the paper in Section VII.

3.3 System Model

3.3.1 Network architecture

Let us consider an edge computing network architecture as the one depicted in Fig. 4.1 that adheres to the requirement of ETSI standard based architecture [13, 8, 70, 71, 12]. In this architecture, the end users are depicted in the lower layer which is called EUs' layer. EUs can be drones, surveillance cameras or virtual reality equipment. For simplicity, EU are to be referred to as users. Let $\mathcal{U}$ be the set of N users $\mathcal{U} = \{u_1, ..., u_N\}$. Users communicate with the edge layer through gateways that connect to one or several edge nodes (ENs). The gateways are used as intermediary devices to connect the user equipment to the edge nodes. For example, these intermediary devices can be routers or switches within the same

Table 3.1 Summary of System Variables

Symbol	Description
$\mathcal{U}$	Set of users
$\mathcal{E}$	Set of EN
x_i^u	Association index
D_i	Devices of the EN i
$R_{(i,j)}$	Set of resources of device j
$\mathcal{C}_{(i,j)}$	Set of containers of device j
$S_{(i,j)}$	Set of services
$V_r^{s_{(i,j)}}$	Requirement resource vector a service
$\mathbb{1}_r$	Identification function
$A_{s_{(i,j)}}$	Matrix of required resources per service
T_{out}	Timeout of a request
$\varrho_{(i,j)}$	Request
$\Xi_{(i,j)}^{\sim,p}$	Amount of resources consumed by a container
λ_i	Arrival rate of requests
Rq_i	Number of request at a time slot
$\delta_L^{\sim}$	Delay for a given resource
$D_{(i,j,l)}$	Data size for local processing
$f_{\mathcal{P}}$	Processing capacity
$idx_{r/w}$	Read/Write index
ξ_i^u	Link data rate
$h_.$	Channel gain
$P_.$	Power of communication
$\rho_{\sim}^i$	Data size for edge processing
Q_i	Queue vector of an EN i
$\mathcal{H}$	History vector
b_i	Queue's request arrival process
a_i	Queue's service arrival process
Z_i	Queue dynamic of EN i
y_k	Penalty process variable
$L_{(i,j,k)}^{r_{(i,j)}^{\sim}}$	Load for a given resource
V	Trade off parameter
p_m^{avg}	average resource loss
$\alpha_i(t)$	Resource allocation scheme
$\beta_i(t)$	Resource state vector
$y_i(t)$	Vector of penalties
$L_{(i,j,k)}^{norm}$	Normal load
$rate_{norm}$	Rate of request arrival

level of the hosting organization. ENs belong to the set $\mathcal{E} = \{e_1, e_2, ..., e_M\}$ of M EN. We assume that users are already associated to ENs following the strategy proposed in [82].

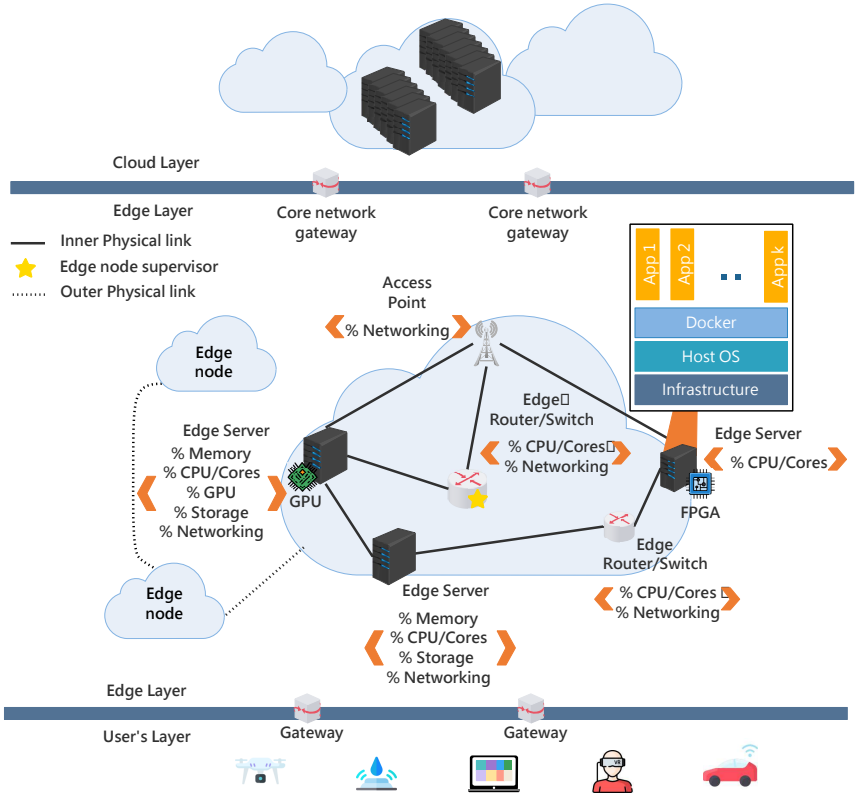

Figure 3.1 Overview of the different entities of the edge computing architecture.

Let $x_i^u(t)$ be a binary variable that ensures the user-EN association, described as follows:

$$\begin{cases} x_i^u(t) \in \{0,1\}, \forall i \in \mathcal{E}, \forall u \in \mathcal{U} \\ \sum_{i \in \mathcal{E}} x_i^u(t) = 1, \forall u \in \mathcal{E} \end{cases} \tag{3.1}$$

Each EN is considered to be an aggregation of heterogeneous EDs denoted by $D_i = \{d_{(i,1)}, d_{(i,2)}, ..., d_{(i,K)}\}$. In addition, users-EN matching is distributed which makes the system standalone, but in case we have a huge number of EDs, the discovery phase could take much more time. Therefore, each EN e_i is supervised by an edge node supervisor (ENS), denoted e_i^* . Each ED has resources in terms of processing ($\mathcal{P}$), storage ($\mathcal{S}$), memory ($\mathcal{M}$) and networking ($\mathcal{N}$), denoted by the set $R_{(i,j)} = \{r^{\sim}_{(i,j)}, \sim \in \mathcal{P}, \mathcal{S}, \mathcal{N}, \mathcal{M}\}$, with i representing the EN and j the ED. Each ED j hosts a set of containers [83] $C_{(i,j)} = \{c^1_{(i,j)}, ..., c^P_{(i,j)}\}$ representing the different services deployed on the EN i. We denote by $\Xi^{\sim,k}_{(i,j)}$ the amount of resource $\sim$ consumed by the container k executed on the ED j at the EN i. Let $S_{(i,j)} = \{s_{(i,j)} \mid i \in \mathcal{E} \text{ and } j \in D_i\}$ be the set of services hosted on the j^{th} ED at the i^{th} EN. Each service requires resources in terms of processing, storage,

memory and networking, which are represented by a vector $V_r^{s_{(i,j)}} = \langle \mathbb{1}_r, A_{s_{(i,j)}} \rangle$ where $\mathbb{1}_r(\sim) = 1$ if the resource $\sim$ is required by $s_{(i,j)}$ and $\mathbb{1}_r(\sim) = 0$ otherwise. $A_{s_{(i,j)}}$ represents the resource amounts required by service $s_{(i,j)}$. The resource capacity at each device is a crucial parameter that should be considered for a good resource allocation. The following equation describes the constraint on the resource capacities and is given as:

$$\sum_{k=1}^{|C_{(i,j)}|} \Xi_{(i,j)}^{\sim,k} \leq r_{(i,j)}^{\sim} \mid \sim \in \{\mathcal{P}, \mathcal{M}, \mathcal{N}, \mathcal{S}\} \tag{3.2}$$

$C_{(i,j)}$ represents the set of containers, $\Xi_{(i,j)}^{(\sim,k)}$ represent the amount of resources required by the container k, $r_{(i,j)}$ represents the maximum capacity of the resource $\sim$. Users perform service requests on the EN. We denote a single request by $\rho_{(i,j)} = \langle D_{(i,j,l)}, s_{(i,j)}, T_{out} \rangle$, where $D_{(i,j,l)}$ is the data size to be processed by service $s_{(i,j)}$ and T_{out} is the time duration upon which the user gives up service $s_{(i,j)}$. Without loss of generality, the time is considered to be discreet and indexed by slots $t \in \mathbb{N}$ and the requests are identically independent distributed (i.i.d.). The requests arrival rate is then proportional to the number of request, and for a given EN i we have the arrival rate [40]:

$$\lambda_i = \frac{\mathbb{E}\left[V_r^{s_{(i,j)}}\right]}{t} \tag{3.3}$$

where $V_r^{s_{(i,j)}}$ represents the vector of the required resources by the service $s_{(i,j)}$ at the time slot t. In this paper, we consider that requests are either processed locally, using the residual resources of the user, processed at the edge, or both in a proportional manner; locally and at the EN.

3.3.2 Local processing

Local processing delay consist in both computational and storage delay. The computational delay is given as follows:

$$\delta_L^{\mathcal{P}} = \frac{D_{u_k}}{f_{\mathcal{P}}} \tag{3.4}$$

where, $f_{\mathcal{P}}$ represents the computing capacity of the ED which is the amount of data that the processing unit can process per time unit. D_{uk} represents the data size. At this stage there is no communication. Therefore, no queuing delay which represents the time that the request spends at the queue of a shared resources (the EN in this case). For the local processing case, the user equipment uses its own capabilities to process the task. Thus, the communication delay is, $\delta_L^C = 0$. We considered the storage delay which is the delay from storing/reading to/from the storage support of ED. Since the ED are heterogeneous,

some of the EDs can be equipped with HDD storage supports while others might have SSD, and it is well known that writing/reading to/from SSD is faster than HDD [84, 85]. In addition the caching at the edge represents a use case in which the storage performance have an important impact on the latency [86]. For these considerations we considered the storage delay, and we put:

$$\delta_L^{\mathcal{S}} = D_{u_k}\left(\frac{idx_w}{s_w} + \frac{idx_r}{s_r}\right) \tag{3.5}$$

where, s_w and s_r are respectively the storage support writing and reading speed of the ED, idx_w and idx_r are binary variables denoting either writing/reading is active. Reading index is defined as follows:

$$idx_r = \begin{cases} 0 \text{ if request requires no reading} \\ 1 \text{ otherwise} \end{cases} \tag{3.6}$$

and writing index as:

$$idx_r = \begin{cases} 0 \text{ if request requires no writing} \\ 1 \text{ otherwise} \end{cases} \tag{3.7}$$

In this case of local processing, the total delay experienced by a given request is given as:

$$\delta_L^{total} = \delta_L^{\mathcal{P}} + \delta_L^{\mathcal{S}} \tag{3.8}$$

In this paper, the queuing delay represents the time that the request spends at the queue of a shared resources in the edge network. For this reason, we considered the queuing delay in the edge processing scenario and we adopted the Lyapunov framework to study the queues dynamics. For the local processing case, the queuing delay is not considered, because the user equipment uses its own capabilities to process the task and thus it has no queue of a shared resources.

3.3.3 Edge processing

We consider that the links between the users and EN are reliable, also, the communication setup is already done. Processing at the EN entails a queuing delay and a propagation delay. The propagation delay is the sum of, (i) the delay that a service request experiences to get to the EN and (ii) the delay t that the corresponding response experiences to get

back to the user from the EN. This propagation delay depends on the communication link's data rate, which is defined as follows [40]:

$$\xi_i^u = B_w . \log_2 \left(1 + \frac{P_u \left| h_{(u,i)} \right|^2}{\sigma^2 + \sum\limits_{\substack{v \in \mathcal{U} \\ v \neq u}} P_v \left| h_{(v,i)} \right|^2} \right), \tag{3.9}$$

where B_w represents the bandwidth of the communication link, P_u is the transmission power or user u, $h_{(u,i)}$ is the channel gain between user u and EN i and σ^2 represents the noise variance. The transmission is the ratio of the packet size to be transmitted to the communication data rate and is given as follows:

$$\delta_{(i,u)}^{\mathcal{C}} = \frac{\widetilde{\rho_i}}{\xi_i^u} \tag{3.10}$$

The total delay for task processing is given as:

$$\delta_{(i,u)}^{total} = \delta_{(i,u)}^{\mathcal{C}} + \delta_{(i,u)}^{\mathcal{S}} + \delta_{(i,u)}^{\mathcal{P}} + \delta_{(i,u)}^{wait} \tag{3.11}$$

where $\delta_{(i,u)}^{\mathcal{C}}$, $\delta_{(i,u)}^{\mathcal{S}}$, $\delta_{(i,u)}^{\mathcal{P}}$ and $\delta_{(i,u)}^{wait}$ represent respectively, the transmission delay, storage delay, processing delay and the waiting time at the queue of the EN.

3.3.4 Problem Formulation

In order to optimize the total delay, we start by analyzing the queuing delay which is an essential component of the overall delay. To achieve that, we could use tools such as Little's Law that affirms that the queuing time is proportional to queue length. However, this would allow to consider the queue length and not the queue tail, which would not help guaranteeing the constraints regarding the low latency. We leverage the Lyapunov optimization framework due to its capabilities to provide optimized and stable queuing dynamic [87].

We assume that all the requests are stored in the same queue and i.i.d. Let $Q_i(t)$ be the queue vector of EN i that evolves in time. The evolution of the queue $Q_i(t)$ is based on the event of requests arrival. Let us assume that the value of $Q_i(t)$ have the structure of $\{\mu(-1), \mu(0), \mu(1), ...\}$ where $Qi(0) = \mu(-1)$ is the initial state of the vector Q_i, where $\mu(t)$ represents the requests arrived in time t. The values of $Q_i(t)$ are based on the values of $\{Q(0), \mu(0), ..., \mu(t-1)\}$ and let $\mathcal{H}(t)$ be the history vector up to the time $t-1$. The

queue $Q_i(t)$ evolves in time as follows:

$$Q_i(t+1) = \max\left[\left(Q_i(t) - b_i(t) + a_i(t)\right), 0\right] \tag{3.12}$$

Where $a_i(t)$ and $b_i(t)$ represents respectively the arrival process and the service process [87]. Values of $a_i(t)$ and $b_i(t)$ are defined by general functions $\hat{a_i}(t)$ and $\hat{b_i}(t)$ respectively ads defined in Eq. (3.14). In addition, we define the queue dynamic at the EN i as follows:

$$Z_i(t+1) = \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=1}^{t} P(\max\left[\left(Q_i(t) + b_i(t)\right) - a_i(t), 0\right] > \Delta_{T_{out}}(t)) \tag{3.13}$$

Where, $Q_i(t)$ is the queuing dynamic at the EN i, and $\Delta_{Tout}(t)$ is the queue length bounds when the tolerable bound relative to the timeout of the tasks. For a given resource allocation scheme $\alpha_i(t)$ and a resource state $\beta_i(t)$ of the EDs in the EN i. $\beta_i(t)$ is used as intermediary variable to find the optimal value of $\alpha_i(t)$ that will be used as input to our resource allocation algorithm. For each $\beta_i(t)$, which is the resource state represented as a vector with values of different values of available resources there is an associated $\alpha_i(t)$, which is the resource allocation scheme.

The objective is to find an optimal value of $\alpha_i(t)$. Each allocation scheme $\alpha_i(t)$ incurs a vector of penalties $y_i(t) = \{y_0(t), y_1(t), ..., y_K(t)\}$. The arrival process $a_i(t)$, the service process $b_i(t)$, and the penalty vector $y_i(t)$ can be expressed as a function of $\alpha_i(t)$ and $\beta_i(t)$ as follows [87]:

$$\begin{cases} a_i(t) = \hat{a_i}(\alpha_i(t), \beta_i(t)) \\ b_i(t) = \hat{b_i}(\alpha_i(t), \beta_i(t)) \\ y_k(t) = \hat{y_k}(\alpha_i(t), \beta_i(t)) \end{cases} \tag{3.14}$$

For $t > 0$, we define $\bar{a}_i(t)$, $\bar{b}_i(t)$, $\bar{y}_k(t)$ and $\bar{Q}_i(t)$ the average sizes of $a_i(t)$, $b_i(t)$, $y_k(t)$ and $Q_i(t)$ defined as :

$$\begin{cases} \bar{a}_i(t) = \frac{1}{t}\sum_{\tau=0}^{t-1} a_i(\tau) \\ \bar{b}_i(t) = \frac{1}{t}\sum_{\tau=0}^{t-1} b_i(\tau) \\ \bar{y}_k(t) = \frac{1}{t}\sum_{\tau=0}^{t-1} y_k(\tau) \\ \bar{Q}_i(t) = \frac{1}{t}\sum_{\tau=0}^{t-1} Q_i(\tau) \end{cases} \tag{3.15}$$

The ENS compute the optimal value of $\alpha_i(t)$ through solving the following optimization problem:

$$\limsup_{t\to\infty} \bar{y}_0(t) \tag{3.16}$$

Subject to

- (C1') : $\limsup_{t\to\infty} \bar{y}_k(t) \leq 0$
- (C2') : Stability of $Q_i(t)$ $\forall$ t $\in \{0, 1, ...\}$

The delay optimization problem is formulated as follows:

$$minimize\ \{\delta^{total}_{(i,u)}\} \tag{3.17}$$

subject to:

- (C1): Eq. (3.2) to ensure resource consumption and maximum capacities.
- (C2): Eq. (3.1) for user-EN association.
- (C3): Eq. (3.12), (3.13) to ensure the queue dynamic.
- (C4): Eq. (3.14), (3.15) for resource allocation.
- (C1') and (C2') for penalties conditions.

To solve the problem in (3.17), problem (3.16) should be solved and thus constraints (C1') and (C2') should be respected. Further, constraints (C1)-(C4) should be respected to obtain an optimal solution to (3.17). It is known that problem (3.16) can be solved in an optimal way but tuning a parameter denoted as V [87]. Thus, after verifying the constraints C1-C4, we can solve (3.17) optimally as well. Our detailed solution is described in the sequel.

3.4 Proposed Solution

3.4.1 Solution Overview

Before going deep into the details of the solution, we provide the basic idea of the proposed resource provisioning approach. At the beginning of each resource provisioning cycle, we verify the constraints (C1)-(C4) related to the association, maximum resource capacities, type and requirement of each task. Then, we provide the resource provisioning through solving the optimization problem in (3.16) by leveraging the Lyapunov optimization whenever a resource provisioning is needed under the constraints of (C1') and (C2'). Note here that we focus on the tasks that are to be processed at the edge of the network. At the EN side, the ENS checks the requirements for each task in terms of type and required amounts of resources. More details about our proposed approach are provided in the remainder of this section.

3.4.2 Proposed Solution

In order to solve problem in (3.16), we consider y_0^{min} as the minimum value of the penalty; in our case is the resource consumed over time slot t, and we put:

$$y_k(t) = p_k(t) - p_k^{avg} \tag{3.18}$$

$p_k(t)$ is the resource loss in the ED k in the EN i at the time slot t, and p_k^{avg} is the average resource loss, and the constraint (C1') in (3.16) holds if:

$$\limsup_{t\to\infty} \bar{y}_k(t) \leq p_k^{avg} \tag{3.19}$$

Therefore, considering the Eq. (3.13), we can rewrite $Z_k(t+1) = \max\left[Z_k(t) + y_k(t), 0\right]$ and we have $Z_k(\tau+1) \geq Z_k(\tau) + y_k(\tau)$ for $\tau \in \{0, 1, ..., t-1\}$, due to requests arrival process which is i.i.d. Thus we can write [87]:

$$Z_k(t) - Z_k(0) \geq \sum_{\tau=0}^{t-1} y_k(\tau) \tag{3.20}$$

we divide by t in (3.20), and we obtain:

$$\frac{Z_k(t)}{t} - \frac{Z_k(0)}{t} \geq \frac{1}{t}\sum_{\tau=0}^{t-1} y_k(\tau) \tag{3.21}$$

Clearly, $\frac{Z_k(t)}{t} \to 0$ when $Z_k(t)$ is stable, thus, $Q_i(t)$ is stable and (C2') is met.

Assuming that the queues are empty initially, we define $\theta(t) = [Qi(t), Z_k(t)]$ as the combination of the queue vectors, the Lyapunov function is expressed as follows:

$$L(\theta(t)) = \frac{1}{2}\left[\sum_{i\in\mathcal{E}} Q_i(t)^2 + \sum_{k\in\mathbb{D}_i} Z_k(t)^2\right] \tag{3.22}$$

The drift plus penalty process requires the minimization of $\mathbb{E}[\Delta(t) + V.y_k(t) \mid \mathcal{H}(t)]$, where $\Delta(t) = L(Q(t+1)) - L(Q(t))$ and V is a performance tradeoff parameter.
Let us assume that the functions of (3.14) satisfy the following conditions for all values of $\alpha_i(t)$ and $\beta_i(t)$:

$$\begin{cases} a_i(t) \geq 0 \\ b_i(t) \geq 0 \\ y_0^{min}(t) \leq y_0(t) \leq y_0^{max} \end{cases} \tag{3.23}$$

Where y_0^{min} and y_0^{max} are the maximum and minimum values of $y_0(t)$. Let $D \geq 0$ be a constant that for every resource allocation scheme $\alpha_i(t)$ based on values of $\beta_i(t)$ we have [87]:

$$\begin{cases} \mathbb{E}\left[\hat{a_i}(\alpha_i(t), \beta_i(t))^4\right] \leq D \\ \mathbb{E}\left[\hat{b_i}(\alpha_i(t), \beta_i(t))^4\right] \leq D \\ \mathbb{E}\left[\hat{y_k}(\alpha_i(t), \beta_i(t))^4\right] \leq D \end{cases} \tag{3.24}$$

Where $\mathbb{E}[.]$ represents the expectations taken, considering $\beta_i(t)$ and the decisions $\alpha_i(t)$. The drift-plus-penalty expression for a finite constant $B > 0$, satisfies the following [87]:

$$\mathbb{E}\left[\Delta(t) + V.y_0(t)|\mathcal{H}(t)\right] \leq B + V.\mathbb{E}\left[y_0(t)|\mathcal{H}(t)\right] + \sum_{i\in\mathcal{E}} \left(Q_i(t)\right)\mathbb{E}\left[a_i(t) - b_i(t)|\mathcal{H}(t)\right] + \sum_{k\in\mathcal{D}_i} \left(Z_k(t)\right)\mathbb{E}\left[y_k(t)|\mathcal{H}(t)\right] \tag{3.25}$$

The right-hand-side of Eq. (3.25) is minimized when choosing a minimal value of $\alpha_i(t)$ corresponding to the states of the $Q_i(t)$ and $Z_k(t)$ and the state of resource at EN i, $\beta_i(t)$:

$$V.y_0(t) + \sum_{i\in\mathcal{E}} Q_i(t)\left[a_i(t) - b_i(t)\right] + \sum_{k\in\mathcal{D}_i}\left[Z_k(t)y_k(t)\right] \leq C \tag{3.26}$$

$Q_i(t)$ and $Z_k(t)$ are updated according to (3.12) and (3.13), with the existence of a given constant $C \geq 0$, optimal value of $\alpha_i(t)$ is chosen when C is small as possible, and:

$$\begin{aligned} \mathrm{V}.y_0(t) + \sum_{i\in\mathcal{E}} Q_i(t)\left[a_i(t) - b_i(t)\right] + \sum_{k\in\mathcal{D}_i}\left[Z_k(t)y_k(t)\right] \\ \leq C + \\ \inf_{\alpha_i(t)}\left[V.y_0(t) + \sum_{i\in\mathcal{E}} Q_i(t)\left[a_i(t) - b_i(t)\right] + \right. \\ \left. \sum_{k\in\mathcal{D}_i}\left[Z_k(t)y_k(t)\right]\right] \end{aligned} \tag{3.27}$$

In case $C = 0$, the exactly minimum value of $\alpha_i(t)$ is reached, thus the optimal resource allocation scheme is obtained.

The resource representation allows characterizing the exact capabilities of the EDs precisely and uniformly. The resource representation allows the ENS to get information about the EDs resource status, which correspond to the values of $\beta_i(t)$ in our proposed approach in the previous section. Each ED is aware of its purpose (the types of operations that could

be handled), its capacities and available resources. For this purpose, each ED exposes its available resource to the ENS through the MEC API standard of ETSI [13, 69, 70, 71, 12].

```
<device id="47df0be20283"><cpuinfo_version>(4, 0, 0)</cpuinfo_version>
<vendor_id>GenuineIntel</vendor_id>
<hardware_raw></hardware_raw>
<brand>Intel(R) Core(TM) i5-4310U CPU @ 2.00GHz</brand>
<hz_advertised>2.0000 GHz</hz_advertised>
<hz_actual>2.6010 GHz</hz_actual>
<hz_advertised_raw>(2000000000, 0)</hz_advertised_raw>
<hz_actual_raw>(2601000000, 0)</hz_actual_raw>
<arch>X86_64</arch>
<bits>64</bits>
<count>4</count>
<raw_arch_string>x86_64</raw_arch_string>
<l1_data_cache_size>32 KB</l1_data_cache_size>
<l1_instruction_cache_size>32 KB</l1_instruction_cache_size>
<l2_cache_size>256 KB</l2_cache_size>
<l2_cache_line_size>6</l2_cache_line_size>
<l2_cache_associativity>0x100L</l2_cache_associativity>
<l3_cache_size>3072 KB</l3_cache_size>
<stepping>1</stepping>
<model>69</model>
<family>6</family>
<processor_type></processor_type>
<extended_model>4</extended_model>
```

Figure 3.2 Example of output for an ED exposing information about its CPU

The resources are exposed in XML format to the supervising entity. In order to get information about the physical resource, each ED uses low-level operations and commands including *CPUID* to get the CPU properties such as the architecture, number of cores and the frequency. The ED, depending on its OS, uses commands such as *cpuinfo* and *meminfo* for Linux based machines to obtain the current state of the CPU usage and memory respectively, *wmic* for Windows-based EDs. We proceeded with the same to cover the majority of the used OS in the EDs. Collecting all these commands within the same program that, depending on the ED used OS, collects the information about the EDs and their available resources. Figure 3.2 shows an example of an ED's response when the supervisor requested information about the CPU resource. The output is an XML response with all the different information about the CPU such as the family, the frequency, number of cores and the architecture. Also, XML can be parsed by almost all EDs, which makes the information about the resource easier and more understandable to the ENS.

Algorithm 1 is performed in order to define the optimal allocation scheme for determining the adequate EDs that will participate in processing the incoming request. The time complexity of the proposed Lyapunov-based optimization algorithm is around $O(N^2)$ [87], which ensures an low complexity of our proposed algorithm in Algorithm 1, since it is the only heavy task to perform by the ENS.

The frequency of checking ED resources (through resource representation model) depends essentially on the frequency of reallocating the resources. More precisely, the resource's status is checked every time the resource allocation is needed instead of being checked at each given time slot as in [72, 40, 73, 74, 75, 32, 76, 77]. We could solve this problem either statically or dynamically. In the static scheme, the problem is solved by fixing a time interval on which we perform the resource's checking, which is not optimal since the resource reallocation interval on a given EN is longer under a low load and shorter under a high load. We can solve the problem of reallocation resources dynamically by monitoring the load on devices [88]. Below, we consider that the moment of checking resource's status is the moment of reallocating resources. In fact, reallocating virtual resources for edge computing applications costs additional computational resource. Thus, overly frequent reallocation of resources might decrease the efficiency of the EN resources usage. Inspired from the the work in [88], we define the workload of an ED for a given resource $\sim$ (where $\sim \in \{\mathcal{P}, \mathcal{S}, \mathcal{N}, \mathcal{M}\}$) as follows:

$$L_{(i,j,k)}^{r_{(i,j)}^{\sim}} = r_{(i,j)}^{\sim} - \sum_{k=1}^{|C_{(i,j)}|} \Xi_{(i,j)}^{\sim,k} \tag{3.28}$$

Algorithm 1: Resource allocation based on resource representation and Lyapunov optimization (LRR)

Input: incoming requests $\varrho_{(i,j)}$
Initialization: place $\varrho_{(i,j)}$ at the queue
get D_{u_k} and $s_{(i,j)}$;
if *free*$(r_{(i,j)}^{\sim})$ **then**
    Allocate resource at ED $d_{(i,j)}$;
    Instantiate container $c_{(i,j)}^{k}$ for the requested service with respect to constraint in equation (2);
    Destroy the $c_{(i,j)}^{k}$ after task completion;
end
Find α_i from the problem in (3.16) determine the candidate ED with respect to T_{out};
Allocate the adequate amount of resource on each selected ED from (3.16);
Distribute the subtasks to the ED;
Create containers on each ED;
Destroy containers after all subtasks are finished;
Wait for the next request;

If we assume that the maximum workload of an ED is given by $L^{max}_{(i,j,k)}$, we define the normal workload $L^{norm}_{(i,j,k)}$ as follow:

$$L^{norm}_{(i,j,k)} = L^{max}_{(i,j,k)} \times rate_{norm} \text{ with } rate_{norm} \in [0,1] \tag{3.29}$$

Finally, we can define $L^{th}_{(i,j,k)}$ as the threshold that triggers the resource reallocation process as :

$$L^{th}_{(i,j,k)} = rate_{th} \times (1 - rate_{norm}) \times K \times L^{max}_{(i,j,k)} \tag{3.30}$$

Where K represents the number of container on execution on the ED j. A resource reallocation takes place when the value of $L^{th}_{(i,j,k)}$ changes.

The time complexity of algorithm (2) depends on the number of ED in the EN. Since the ENS has the information about each ED at the beginning of the network operations, information about the loads are available, since the ENS is aware of the tasks' sizes and the capabilities of each ED. Moreover, the number of the resource allocation scheme changing in some cases remains unchanged due to the availability of resource on a given ED, which ensures the optimality of the resources used to compute the optimal allocation scheme $\alpha_i(t)$ [88]. In other words, the frequency of performing resource allocation is lower when compared to other schemes that perform the same operation not only at each request arrival but at each time slot [72, 40, 73, 74, 75, 32, 76, 77].

3.5 Simulation Results

We evaluated our proposed resource provisioning approach through extensive simulations. The experimental testbed was inspired from the work of [81]. For simplicity, we used the same OS (Linux) and we deployed Docker on these VMs, then we used Kubernetes as container management engine. We then configured the VMs to act as a swarm from the point of view of Docker. We deployed our resource representation component over these VMs, and we deployed our resource allocation algorithm on the master device (the

Algorithm 2: Resource Reallocation Calculation

```
1 Use (29) and (30) to calculate L^norm_(i,j,k) , L^th_(i,j,k) and L^max_(i,j,k);
2 for each d(i,j) in D_i do
3 |   if L^curr_(i,j,k) > L^norm_(i,j,k) + L^th_(i,j,k) then
4 |   |   Send resource information request to d(i,j);
5 |   end
6 end
```

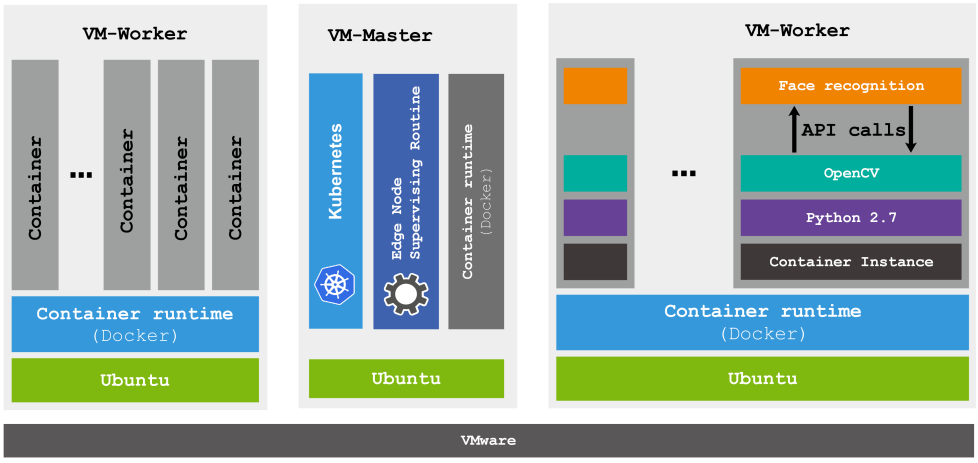

Figure 3.3 Testbed components; The VM-Workers (EDs) on left and right and the VM-Master (ENS) on the center.

ENS). Moreover, we tested the resource representation component on different versions of OSs and different hardware architectures (x86, x64 and ARM). However, the testbed with heterogeneous configuration is yet to be investigated in a future work. The experimental simulations consist of an EN with 3 EDs. As illustrated of Fig. 3.3, with the ENS (master) is at the middle having the role of supervising the EN, receiving the request and executing the resource provisioning schemes. The central ED in the Fig. 3.3 is considered to host the master of the EN, thus, the ENS. The other EDs are the workers. The ENS is responsible for checking the EDs resource information, performing the resource provisioning (computing the optimal scheme and values of $\alpha_i(t)$) and forwarding the requests to the adequate nodes based on the $\alpha_i(t)$ values). We tested the proposed approach by implementing a face recognition based on OpenCV framework [89]. We simulated the proposed approach using a setup that is closely equal to the experimental one, having a power of processing of $2.5Ghz$, a memory variation between $\{2-4\}$ Gb and a storage between $\{20-30\}$ Gb. We also considered the V parameter ranging in $0-100$. To put the testbed to work, we considered the following scenario, in which we used a face recognition application inspired from [90]. The user's device (smartphone for instance) sends the images to the EN for recognition. The face recognition classifier is already trained in the EN. After receiving the image, the EN solves the optimization problem and select the EDs in which the face detection should take place. The EDs have different capabilities and configurations as described in the table follows:

We compare our approach to the MGRA benchmark approach [74] both theoretically and in simulations. We also compare our approach when using both resource representation and Lyapunov optimization (LRR in Fig. 4-11) with using the proposed Lyapunov frame-

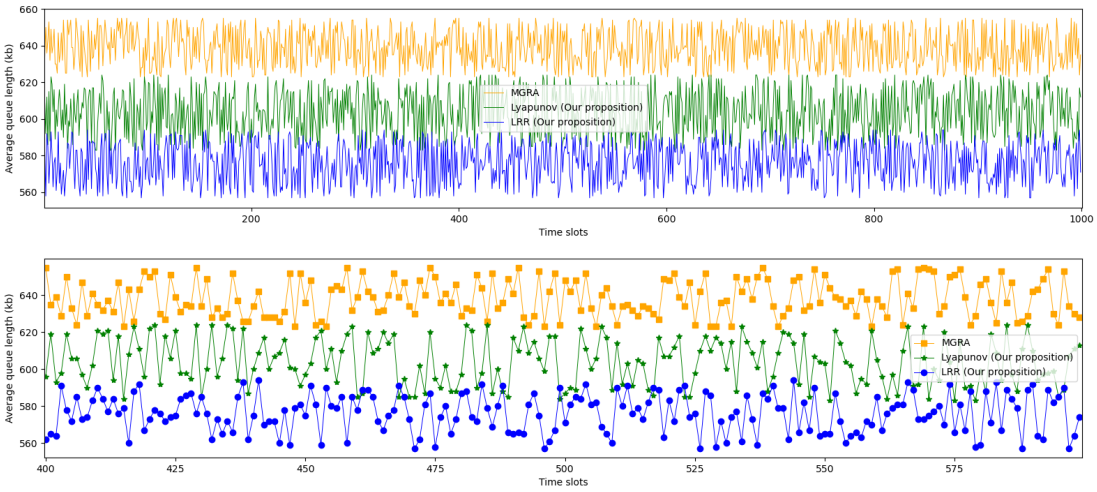

Figure 3.4 Average queues sizes over time

Table 3.2 Experimental Setup

Nodes/Resource capabilities		Memory	Processing	Storage
EN	ED1	2Gb	2 Cores	20Gb
	ED2	2 Gb	4 Cores	30Gb
	ED3	4Gb	2 Cores	20Gb

work alone. Fig. 3.5 illustrates the average queue size evolution in time. The queue size is minimized in LRR compared to MGRA. The figures also show that using resource representation in our approach has a positive effect in minimizing the queue size compared to using our proposed Lyapunov framework alone. In addition, the mechanism we used is lightweight and could be executed within a tiny shred of resources.

In order to test the impact of the parameter V of Eq. (3.26) on the overall system performance, we evaluated the average queue size under the different values of V to find the optimal configuration of the performance-delay tradeoff. As a reminder the V parameter represents the tradeoff between the performance and the allowable latency.

We illustrate the evolution of the queue under the variation of the V parameter. Fig. 3.5 which shows that a lower value of V means that the user is interested in being served within a low delay. In the MGRA scheme, there is no consideration to such parameter, which causes the constant behavior of the average queue size. Fig. 3.5 shows that the queue size of is less congested when using our Lyapunov framework and even less congested

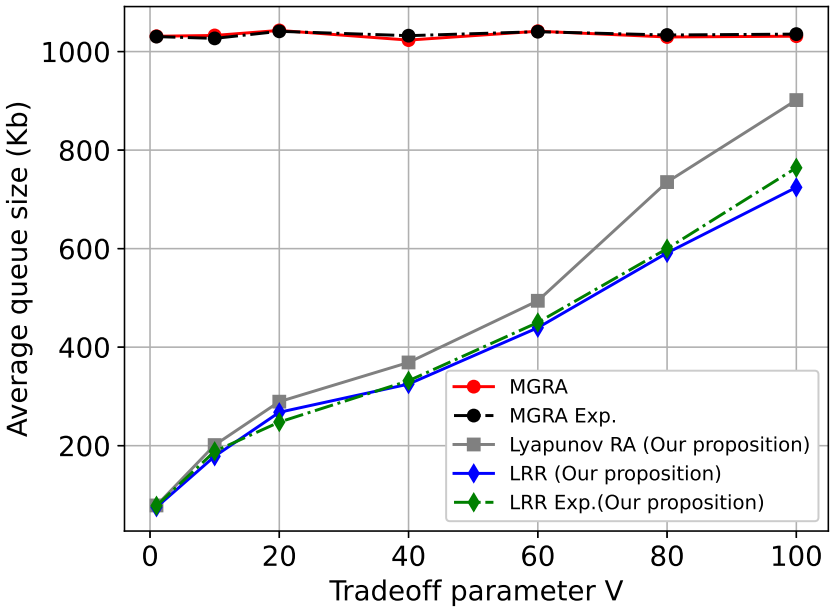

Figure 3.5 Average queue size evolution in function of the parameter V

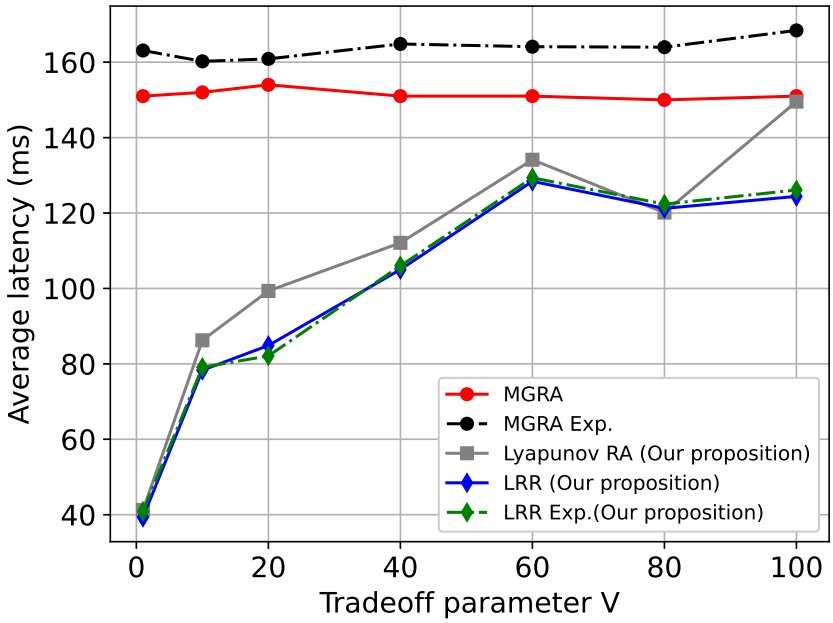

Figure 3.6 Average latency evolution in function of the parameter V

when using LRR. Even for high values of V , the queue is far from reaching the queue size of the benchmark MGRA scheme.

We evaluated the impact of the parameter V on the average latency and we conclude that a higher value of V implies a higher latency, which is also very clear from the figure 3.4. Our proposed approach shows an improvement in terms of latency. Fig. 3.9 illustrates

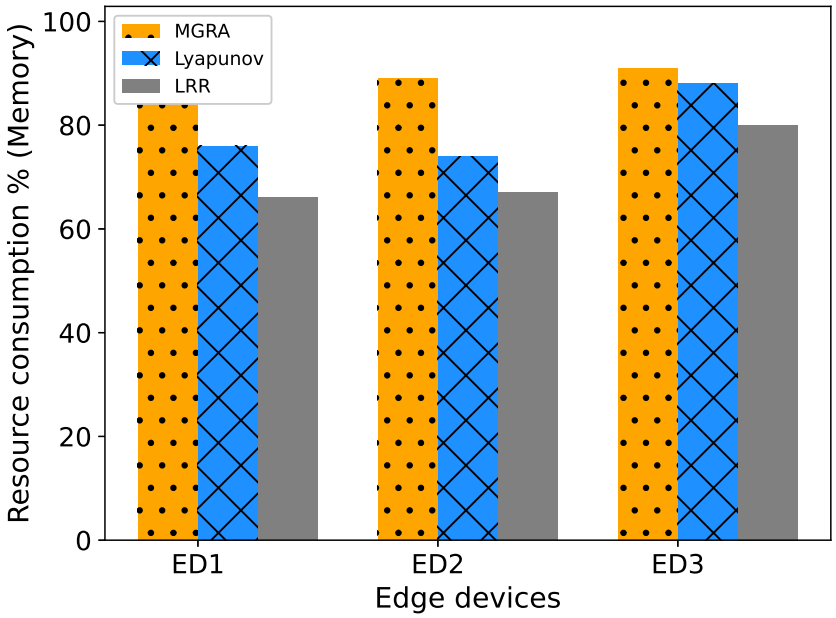

Figure 3.7 The average network interface utilization on each ED of the simulation setup

the average latency evolution with the queue size under the different schemes of resource provisioning. The average latency represents the overall delay spent from the moment of requesting a task processing, to selecting the adequate ED to process the task and then receiving the results. Our proposed LRR approach outperforms the MGRA approach.

Our approach achieves the lowest latency with the evolution of the queue size, and with further examination, Fig. 3.9 shows that LRR improves the latency up to 40%. This can be explained by the fact that the ENS is all the time aware of the available resources at the ED and their capabilities. Therefore, based on the resource information, the ENS allocates the resources on the adequate EDs, which distribute the tasks in an adequate way that guarantees a lower latency for the users and resource consumption at the ED following the Lyapunov optimization framework.

The results for our study of the testbed ED resources behavior under the different resource provisioning schemes are illustrated in Fig. (3.4-3.11). Our proposed LRR scheme shows a significant improvement in terms of CPU consumption as shown in Fig. 3.10. This enhancement is due to the ENS awareness of the resource status of each ED. When adopting the benchmark MGRA approach, a portion of the resources from the devices are used to compute the optimal matching between the users and the EN which takes almost $O(N^2)$ compared to Lyapunov which takes only $O(\frac{1}{N2})$. In some cases, the MGRA benchmark approach which is based on a matching game, takes long delay to reach a stable distribution,

which may cause higher consumption of resources. In LRR, the resource representation guarantees the information about the ENs resources. Using Lyapunov optimization for addressing the queue's congestion offers better resources utilization on one hand. On the other hand, it is mathematically proved that an optimal resource allocation scheme is always guaranteed within a reasonable delay. Fig. 3.10 shows that LRR guarantees a lower resource consumption in terms of CPU compared to the benchmark approach; specifically approximately a 11% lower consumption on ED1 and ED2, and approximately a 18% lower on ED3.

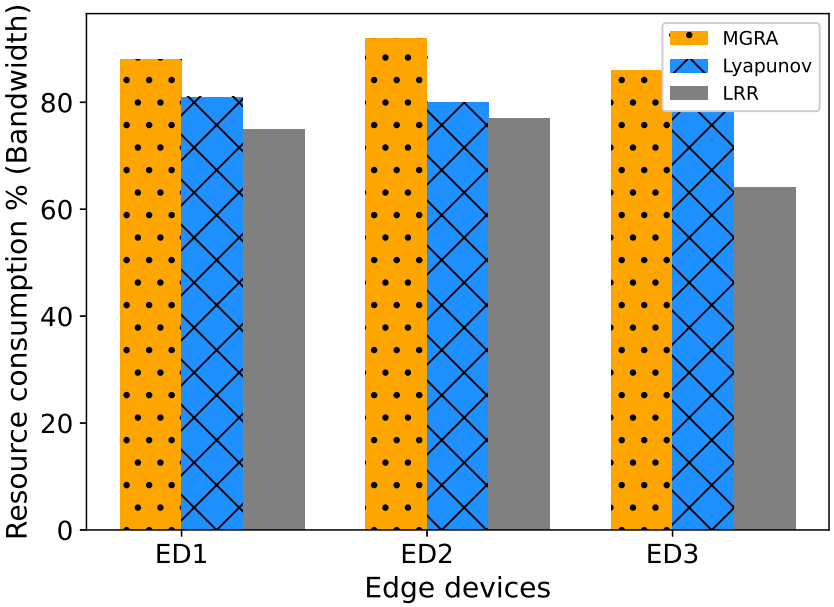

Figure 3.8 The average storage utilization of each ED from the setup

Fig. 3.11 illustrates the storage consumption under the MGRA, and our approach using Lyapunov with and without considering the resource representation. Fig. 3.11 shows that the different schemes have almost the same performance for ED1 and ED2. However, in ED3, LRR significantly improves the performance of storage use by almost 25% compared to MGRA, due to the fact that in our proposed scheme we used containers as a virtualization technology instead of VMs used in the MGRA approach.

The network performance is illustrated in Fig. 3.8, in which the Lyapunov-based approach shows an enhancement of the bandwidth consumption as it was already been discussed in previous papers such as the works in [39, 40]. In addition, the resource representation gives better results, 15% lower than MGRA and 6% lower than the Lyapunov-only based approach.

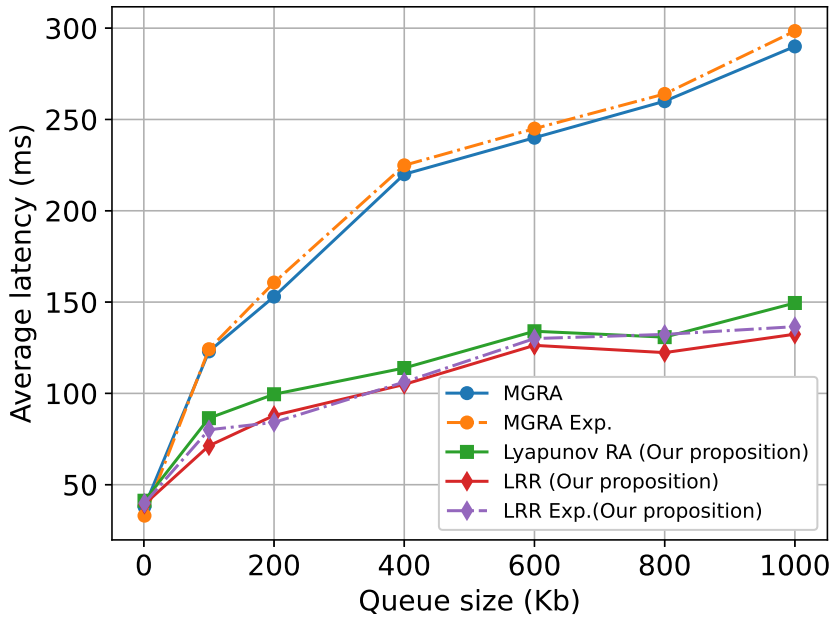

Figure 3.9 Latency evolution over the queue size

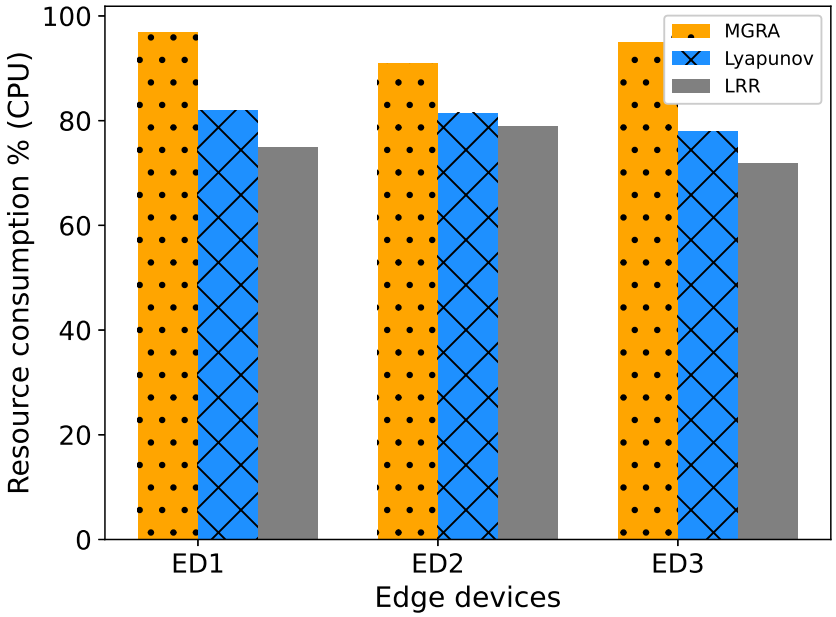

Figure 3.10 Average CPU utilization at the EDs.

Fig. 3.7 illustrates the memory consumption on the different EDs of our testbed. The results show that LRR uses 14% less resources compared to the benchmark scheme. This result could be explained by the fact that in a matching game approach, EDs are not aware of what the other devices have in terms of capabilities. Also, the matching game is distributed and the discovery phase could take much more time. Our LRR approach

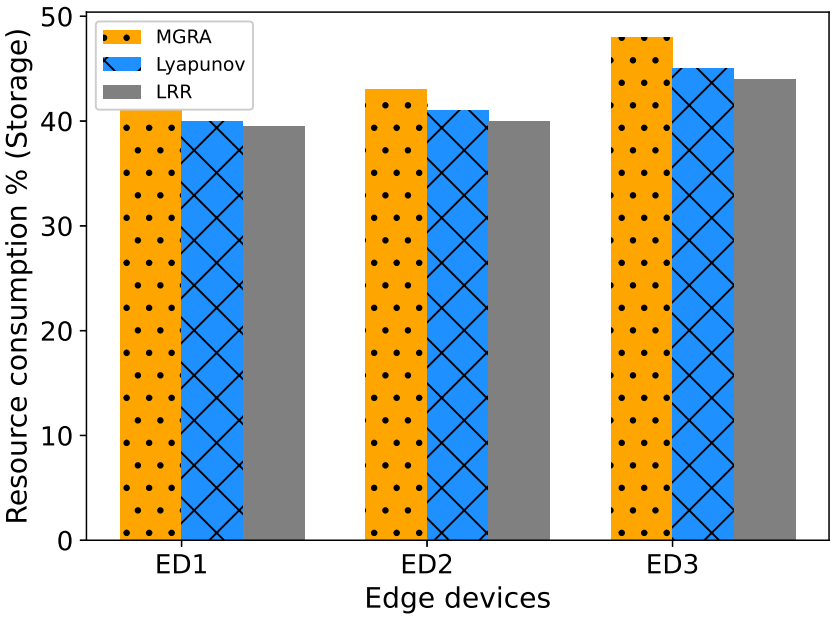

Figure 3.11 The average storage utilization of each ED from the setup

which makes the information about available resources accessible to the ENS while using a Lyapunov approach makes the resource allocation more efficient.

Fig. 3.12 illustrates the evolution of the reallocation frequency in LRR compared to a baseline scheme. The baseline scheme referred to in Fig. 3.12, is used in several previous works [39, 40, 73, 74, 75, 32, 76, 77, 88] and consists in allocating resources at each time slot where a request is received. However, adopting such policy may make the EN reallocate resources continuously at each node to satisfy the requests, which may have a direct impact on the latency. In LRR, the ENs have sufficient information about each ED's capabilities and up-to-date resource states, making it easier to allocate resources on nodes with available resource without the need of recomputing the allocation scheme.

3.6 Related Works

Several previous works investigated the resource allocation and management from the edge computing perspective. Most of these works, such as in [39, 40, 73, 74, 75, 32, 76, 77] investigate the resource allocation problem for only one or two types of resources, and very few investigated the problem of resource allocation with more than two types of resources [68]; mainly the computational and networking resources. In this paper, we propose a resource allocation scheme for four kind of resources including processing, storage, memory and networking through resource representation. Resource representation allows the ENS to acquire information about all the available resources at each device in the EN. Some

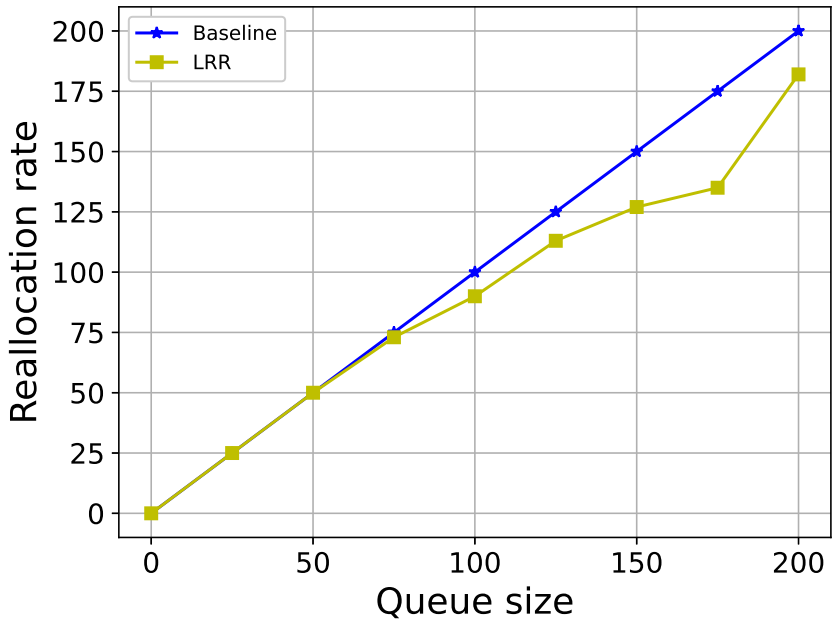

Figure 3.12 The reallocation frequency in function of the queue size

of the previous works did address the resource allocation problem using the Lyapunov optimization framework for minimizing the queuing time [39, 40, 73, 32, 91]. However, these works focus on few types of resource and the allocation cycle takes place at each time slot, which may lead in some cases to compute the optimal allocation scheme even when the current scheme can deal with incoming requests. Therefore, in our proposed approach the resource allocation scheme is only performed when the incoming request requires a re-computation of the resource allocation scheme. Other studies such as in [74, 75, 32, 76, 77] used theoretical game-based approaches such as matching games and coalitional games.

In some papers, authors propose a combination of different techniques such as in [39, 40, 73], combining Lyapunov optimization and matching game frameworks or combining multiple optimization framework such as in [32]. In [39] the authors proposed a mixed-integer nonlinear programming problem to optimize the task offloading, computation scheduling and the radio resources. The problem was relaxed to subproblems by leveraging the Lyapunov optimization framework, then proposing a convex decomposition approach and matching game to solve the subproblems. In their proposed algorithm for the optimal task offloading, decision and resource allocation is taken at each time slot, which may lead in some scenarios to changing the overall decision to offload a small portion of tasks. In the study in [40] the authors proposed a user-server association scheme that takes the channel quality in account in addition to the computational capabilities and workloads of the servers. The authors also used a Lyapunov optimization and a matching game

to propose a dynamic task offloading and resource allocation policy. The authors of [73] proposed a resource allocation scheme for computational and communication resources jointly with users' access point association. The proposed solution in [73] is based also on a Lyapunov framework to stabilize the queue and a matching game theory to associate the users to the access points. From the game theoretical perspective, the resource allocation scheme proposed in [74] is based on a joint Stackelberg game to efficiently allocate computing resources to devices, and a one-to-many matching game to match the users to access points. The approach is similar to the one in [75], where instead of using different game theory frameworks, the authors proposed two-tiers matching game to optimize computing resource allocation and pricing, under limited computing and communication resources constraints. The first tier aims at associating users and small base stations with the goal of maximizing the social welfare. The second tier aims at achieving the collaboration between the different small base stations in order to ensure an efficient computing resources consumption. In some papers such as in [32], the authors investigated the optimization of the network resources, more specifically the channel selection which is critical to ensure a reliable task allocation. The authors proposed the combination of three optimization tools to optimize the long-term throughput under energy cost and service reliability constraints. The authors also proposed a learning-based channel selection with service reliability, energy, backlog, and conflict awareness through with a Lyapunov optimization framework to optimize the strategy to allocate the channel, and a matching game approach for the channel selection. However, in these works, the aspect of resource is used in an abstract manner, and the proposed approaches does not put assumptions about either the heterogeneity of the equipment used, neither the interaction between the equipment.

3.7 Conclusion

In this paper, we investigated resource provisioning at the edge of the network under latency and resource consumption constraints. In order to reduce the latency, we studied the experienced delays at the different levels of the considered architecture. Precisely, we studied the queue dynamic at the EN by leveraging a Lyapunov optimization framework. We also proposed a resource representation for EDs which allows the exposition of EDs resource information (processing, storage, memory, and networking) at any time through the ETSI standard for edge computing applications. The ENS uses the gathered information on available resources to define the optimal resource provisioning scheme based on the drift-plus penalty of the Lyapunov optimization framework. We also studied the frequency of resource reallocation and we proposed an algorithm based on the workload at

each ED of the EN to reduce the number of times at which the ENS performs a resource reallocation operation. Moreover, we performed extensive theoretical and experimental simulations to prove the effectiveness of our proposed approaches. The numerical results have shown that our proposed approach outperforms the benchmark approach in terms of latency which drops up to 25% in some cases, and up to 40% in terms of lower resource consumption.

CHAPTER 4

Service Function Chaining in MEC: A Mean-Field Game and Reinforcement Learning Approach

Date de parution: -
Status: Soumis
Revue: IEEE Systems Journal
Titre français: Chaînage des fonctions de service dans les MEC : un jeu de champs moyens et une approche d'apprentissage par renforcement.

Resumé français: Les technologies de l'informatique en MEC et la virtualisation des réseaux sont des outils importants pour que les réseaux de cinquième génération (5G) puissent fournir diverses applications et services. Les services sont souvent fournis sous forme de VNFs entièrement connectées, par le biais du SFC. Cependant, le problème de l'allocation des ressources SFC à la périphérie du réseau reste confronté à de nombreux défis liés à la manière dont les VNF sont placées, enchaînées et exécutées. Dans cet article, nous proposons une approche basée sur la théorie des jeux dont l'objectif est de réduire la latence des services dans le contexte du SFC à la périphérie du réseau. Le problème de l'allocation des ressources SFC peut être divisé en deux sous-problèmes. 1) le sous-problème du placement et du routage des VNF, et 2) le sous-problème de l'exécution des VNFs. Pour le premier sous-problème, nous le formulons comme un jeu de champ moyen (MFG) dans lequel les VNFs sont modélisés comme des entités se disputant les ressources de la périphérie dans le but de réduire la consommation de ressources des nœuds MEC et de réduire la latence pour les utilisateurs. Nous proposons une technique basée sur l'apprentissage par renforcement, où l'algorithme d'apprentissage Ishikawa-Mann (IMLA) est utilisé. Pour le dernier sous-problème, nous le formulons comme un jeu de correspondance entre les VFNs et une ressource du MEC afin de trouver l'ordre d'exécution des VNFs tout en réduisant la latence. Pour le résoudre efficacement, nous proposons une version modifiée de l'algorithme d'acceptation différée classique plusieurs-à-un, appelée algorithme d'acceptation différée à étapes multiples. Pour illustrer les performances des

approches proposées, nous des simulations approfondies. Les résultats obtenus montrent que les approches proposées sont plus performantes que les autres méthodes de l'état de l'art attendant jusqu'à 37,5% de performance.

4.1 Abstract

MEC and network virtualization technologies are important enablers for fifth generation (5G) networks to deliver diverse applications and services. Services are often provided as fully connected VNFs, through SFC. However, the problem of allocating SFC resources at the network edge still faces many challenges related to the way VNFs are placed, chained, and scheduled. In this chapter, to solve these problems, we propose a game theory-based approach with the objective to reduce service latency in the context of SFC at the network edge. The problem of allocating SFC resources can be divided into two subproblems. 1) The VNF placement and routing subproblem, and 2) the VNF scheduling subproblem. For the former subproblem, we formulate it as a mean-field game (MFG) in which VNFs are modeled as entities contending over edge resources with the goal of reducing the resource consumption of MEC nodes and reducing latency for users. We propose a reinforcement learning-based technique, where the Ishikawa-Mann learning algorithm (IMLA) is used. For the later subproblem, we formulate it as a matching game between VFNs and edge resources to find the execution order of the VNFs while reducing the latency. To efficiently solve it, we propose a modified version of the many-to-one deferred acceptance algorithm (DAA), called the enhanced multi-step deferred acceptance algorithm (eMSDA). To illustrate the performance of the proposed approaches, we performed extensive simulations. The results show that the approaches achieve up to 40% less resource consumption, and up to 38% less latency than the benchmarked state-of-the-art methods.

4.2 Introduction

The fifth-generation (5G) of mobile systems is being promoted to accelerate the development of smart cities, not only through improved data throughput but also through support for the expected large amount of connected devices [92]. Many of the use cases involving latency-sensitive applications and highly responsive services are now achievable with less effort and cost [93]. To enable such services, 5G relies on emerging technologies such as cloud computing, multi-access edge computing (MEC), and virtualization technologies that can meet the demands of network flexibility and elasticity. MEC on one hand [4] emerged to address the limitations of cloud computing primarily in terms of latency. MEC enables data processing closer to where it is generated and acts as an extension to the cloud computing paradigm at the network edge [4]. In addition, MEC can assist 5G in

enabling ultra-reliable and low-latency communication (URLLC), enhanced mobile broadband (eMBB), and support massive machine-like communication (mMTC) [92]. On the other hand, virtualization technologies have appeared as a remarkable concept to provide efficient provisioning for 5G and beyond networks through software-defined networking (SDN) and network functions virtualization (NFV) [94, 95].

With the help of virtualization technologies, physical network components and hardware devices can be abstracted into software called virtual network functions (VNFs). VNFs can be instantiated and run in the data plane as virtual machines or containers hosted in devoted infrastructures such as cloud or MEC platforms. Services are often delivered as a bundle of multiple VNFs to compose a service function chain (SFC). For an SFC to meet the required performance, VNFs that compose it need to acquire resources from the MEC infrastructure in an efficient manner. Such a process is often referred to as SFC resource provisioning. Resource provisioning in the context of SFCs requires solving multiple subproblems, including (i) VNF placement, (ii) VNF chaining, and (iii) VNF scheduling [96, 4]. To achieve all the required performance, it is necessary to treat these three sub-problems as an indivisible part of the SFC resource provisioning process.

The SFC resource provisioning problem should be solved regarding different types of resources, namely computation, storage, and transmission. In fact, many works have studied the problem of SFC resource provisioning but consider only computation or transmission, ignoring the storage resources. However, on many occasions, services may require content storage for caching purposes [97] or for storing results of the processing as in some machine learning paradigms [98]. In addition, most existing solutions [7, 99, 57, 54, 52, 56, 100, 58] assume that the VNFs have equal requirements and are homogeneous within a single MEC node. However, in real use cases, VNFs often require different amounts and types of resources and have different purposes.

The VNFs placement and chaining problem aims to find a suitable location for the VNFs (VMs/containers) in MEC nodes while finding an optimal path to transmit the processed packets when the number of transmission links or their capacities are limited. MEC has physical devices on which VMs/containers can be instantiated according to demand [96]. Proper placement of VNFs is motivated by optimizing resource consumption or overall network performance, while minimizing the cost (in terms of energy consumption, latency, and financial) or penalties for service level agreement violations. Although these parameters are quite appropriate for NFVs, most of the work proposed in this context focuses only on reducing the cost of deployment, improving QoS and availability, while neglecting the constraints related to MEC itself. The MEC constraints are related to the

fact that the MEC nodes are not entirely in the premises of the service provider, but many service providers can share the same MEC nodes. In addition, the equipment used in the MEC nodes is not always homogeneous, which in consequence makes the operation of resource allocation for different services and service providers hard. Moreover, VNFs can be deployed/duplicated on several network sites.

Finally, the SFC scheduling subproblem should be solved considering different MEC nodes, since SFCs are often running within a shared physical network and thus, several VNFs can be hosted in the same MEC node [96]. The VNFs scheduling subproblem can be defined as finding the sequence in which the collocated VNFs on each MEC node will be executed in order to achieve the minimum SFC execution time (i.e., the experienced delay between the execution of the source VNF and the completion of the destination VNF). This problem is in the NP-complete complexity class and thus cannot be solved in polynomial time unless $P = NP$ [7].

To summarize, the main contributions of this paper are synthesized as follows:

- We study the SFC resource provisioning problem as an integral problem in which we consider the subproblems of VNFs placement, VNFs chaining, and VNFs scheduling with the aim to reduce the overall latency and resource consumption while considering different types of resources, namely computational, storage and transmission resources.
- We model the VNF placement and chaining subproblems on different MEC nodes as a mean-field game (MFG) to provide the adequate placement and chaining of VNFs on different MEC nodes. The game involves heterogeneous VNFs that require different amounts and types of resources. We also consider a stochastic arrival of requests.
- We model VNF scheduling as a matching game to address the problem of multiple heterogeneous VNFs running within the same MEC node. We extend the classical deferred acceptance algorithm, and we propose the enhanced multistage deferred acceptance algorithm (eMSDA) to support constraints on processing completion, routing time, and resource consumption.
- We provide a theoretical study for both games by proving their stability and equilibrium.

The remainder of this paper is structured as follows. Section II presents the system model. Subsequently, we formulate the problem of resource provisioning and we present the different system requirements in Section III. In Section IV, we provide the details

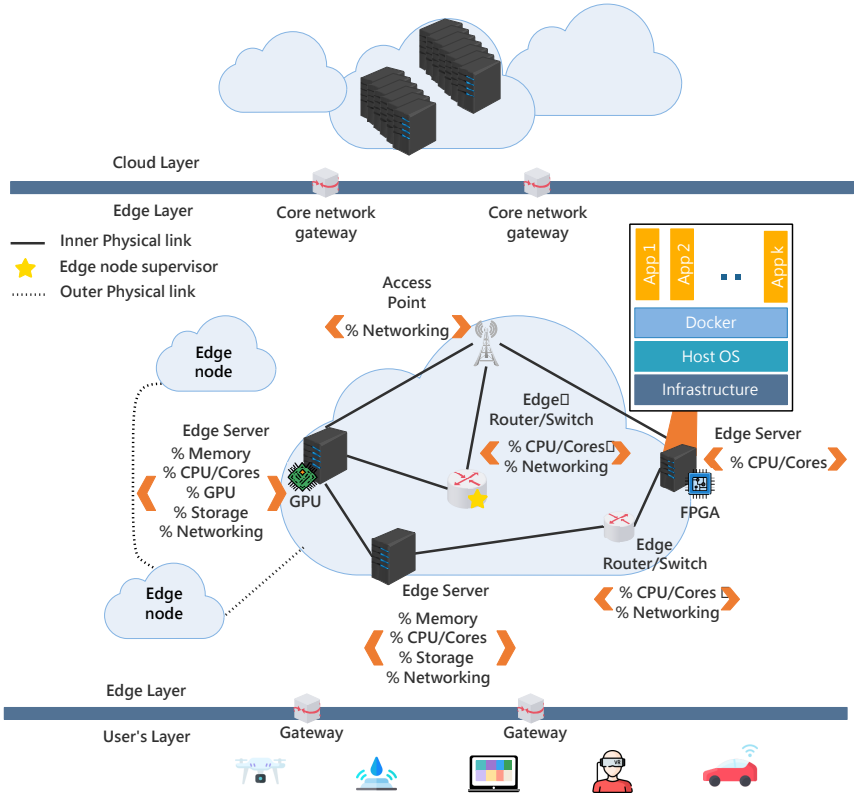

Figure 4.1 An illustration of the considered MEC architecture and the different system's entities.

about the proposed solutions for the VNFs placement, chaining, and scheduling. The performance of the system is simulated in Section V. Last but not least, the related works are discussed in Section VI. Finally, we conclude the paper in Section VII.

4.3 System Model

In this section, we present the considered entities of the system model, followed by the system description, namely, service requests and the physical resources of edge nodes. Table 4.1 summarizes the important notations used in the system model.

For sake of clearness, in what follows, we denote by the index i represents the MEC nodes, j represents the users, k represents the services, and v represents the VNFs.

4.3.1 Physical network substrate

In this work, we consider a slotted system with $t \in \mathbb{N}$ that represents the time slots, where $\mathbb{N}$ represents the set of natural numbers. We consider a MEC network consisting of N edge nodes (ENs) $\mathcal{E} = \{e_1, \ldots, e_N\}$ distributed over a geographical area and interconnected through physical links (PhyL) having transmission capacity of $L_{(i,i')}$ with $(i, i' \in \mathcal{E})$ in $packet/t$ for horizontal data exchanges. Fig. 4.1 illustrates the considered MEC ar-

Table 4.1 Summary of important notations

Symbols	Definition
$\mathcal{E}$	Set of edge nodes
$\mathcal{U}_i$	Set of users
$\mathcal{F}_i$	Set of VNFs hosted in EN i
$\mathcal{F}_k$	Set of VNFs composing the SFC k
$\mathcal{S}$	Set of SFCs
$\rho_{(j,k)}$	Request for service k by yser j
$\bar{\alpha}_v$	Requirement of VNF v
c_i	Computational capacity of EN i
s_i	Storage capacity of EN i
ω_i	Transmission of EN i
f_{src}	First VNF of SFC k
f_{dest}	Destination VNF of SFC k
L_k	Virtual links of SFC k
x_j^i	User-EN assignment
$x_{(i,v)}^f$	VNF-EN assignment
y_z^l	VNF-SFC assignment
$z_{z,l}^i$	VNF-SFC-EN assignment
m_v	Mean field term
r_v	Payoff function
a_j	Strategy taken by VNF j
$\mathcal{A}$	Set of strategies
λ	Learning rate
$\succ_{f_{(i,j)}}$	Preferences of VNFs overs EN
$\succ_i$	Preferences of EN over VNFs
$\mu(.)$	Many-to-One mapping
q_i^{min}	Quota min
q_i^{max}	Quota max

chitecture where we consider three layers. The bottom layer represents the users who communicate with the edge layer through edge gateways. Gateways forward the requests to the associated edge node. In this paper, we consider that the network is supervised by an SDN controller. Both user's layer and edge layer belong to the edge of the network and communicate with the top layer that consists of the core network layer. An example of users could be an extended reality equipment [101], unmanned aerial vehicles [58], or connected vehicles [102]. We denote the set of users associated to a given EN by $\mathcal{U}_i = \{u_{(i,1)}, \ldots, u_{(i,J)}\}$. Each $i \in \mathcal{E}$ is an aggregation of heterogeneous edge devices such as edge servers, routers, access points, and even eNodeBs/gNodeBs. These devices host a set of VNFs $\mathcal{F}_i = \{f_{(i,1)}, \ldots, f_{(i,v)}\}$. Without loss of generality, we consider the placement of the VNFs within a given EN regardless of the specific assignment between the VNFs and the devices of the corresponding EN. Moreover, in this paper, we leverage our previous works [44, 43] on resource representation to allow the EN to exchange their resource availability status and we consider the overall available resources of the EN. Each VNF requires an amount of resources to perform its tasks in terms of computing, storage, and transmission. The available resource at the EN i at time-slot t is given by,

$$\alpha_i(t) = \langle c_i(t), s_i(t), \omega_i(t) \rangle \tag{4.1}$$

where $c_i(t)$ is the available computational resource, $s_i(t)$ the available storage resource, and $\omega_i(t)$ the available transmission resource. Services are represented by the set $\mathcal{S} = \{s_1, ..., s_O\}$ in which a single service s_k is given by the following tuple:

$$s_k = \langle F_k, f^k_{src}, f^k_{dest}, L_k \rangle, \tag{4.2}$$

where F_k denotes the set of VNFs composing the SFC s_k in the ascending order. The source and destination VNFs of SFC s_k are given by f_{src} and f_{dest}, respectively. The virtual links (VLs) between the VNFs of the SFC s_k is denoted as L_k and is represented with $L_k = \{l_{(v,v')} | (v, v') \in \mathcal{F}_k \text{ with } v \neq v'\}$. We also consider that VNFs have different requirements,

$$\bar{\alpha}_v = \left\langle c^v_{(i,j)}, s^v_{(i,j)}, \omega^v_{(i,j)} \right\rangle, \tag{4.3}$$

where, $c^v_{(i,j)}$ represents the computational requirement, $s^v_{(i,j)}$ the storage requirement, and $\omega^v_{(i,j)}$ transmission. We assume that users are already associated to the ENs as proposed in [82]. We consider a binary parameter to represent the user-EN assignment x^i_j, defined

as:

$$x_j^i = \begin{cases} 1, & \text{if user } j \text{ is associated to EN } i, \\ 0, & \text{otherwise.} \end{cases} \tag{4.4a}$$

where

$$\sum_{e_i \in \mathcal{E}} x_j^i = 1, \forall j \in \mathcal{U}_i. \tag{4.4b}$$

We define the VNF-EN association with the variable $x^f_{(i,v)}$ as,

$$x^f_{(i,v)} = \begin{cases} 1, & \text{if the replica of VNF } v \text{ is selected in EN } i, \\ 0, & \text{otherwise.} \end{cases} \tag{4.5}$$

4.3.2 Service Requests

In the literature, many request arrival models have been studied and applied to queuing based systems such as the Poison processes [103], statistic arrivals based on observing the state of the system to build an arrival model [104, 105] and stochastic arrival models [106, 107, 108]. The impact of the request arrival model is crucial, especially when investigating an online-related problem such as in [45]. In [45], the authors considered that the requests arrive in a sequence but did not consider a precise model. However, the requests arrival model has a crucial role in studying resource allocation in a MEC architecture. The work in [106] investigates the problem of stochastic online matching and considered an independent and identical distributed (IID) request arrival model. This model's requests arrival is stochastic-based that has been studied in a variety of online problems such as the k-server problem [107] and matching problem [108]. Adopting an IID model for request arrival with a stochastic framework can improve the quality of the results regarding the resource allocation [106].

In this paper, we adopt IID request arrival model as described in [106]. We model the request arrival to denote the means of requests received by an EN using the variable σ described as follows:

$$\sigma = \{\sigma_i, i \in \mathcal{E}\}. \tag{4.6}$$

Let $\rho_{(j,k)}$ represents a request for the service k by the user j, and is defined as,

$$\rho_{(j,k)} = \langle s_k, d_j, T_{out} \rangle, \tag{4.7}$$

where, s_k is the requested service, d_j is the data packet and T_{out} is the timeout; the time at which the users must complete the service.

4.3.3 EN Physical Resources

As mentioned before, the ENs are formed by aggregating different heterogeneous edge devices such as servers, edge routers, small base stations, or even eNodeBs and gNodeBs. In [44, 43], we proposed a resource representation model that allows these heterogeneous devices to exchange information about their resource capabilities in terms of the type of operations that can handle. For instance, the edge routers are more into network operations, unlike edge servers that are more into data processing and edge learning. In this paper, we will leverage our proposed resource representation model to get the overall capacities of each EN. In other words, we will focus on the position of the VNFs at the EN in general. The ENs communicate horizontally through PhyL that can generally be wired or wireless.

Virtual Links Allocation

The virtual links must allocate a portion of the physical links resources to transmit the packets. We introduce $r^i_{(v,v')}$ as binary parameter to denote whether VNFs are located into different EN. In fact, $r^i_{(v,v')}$ defines whether a VL allocation on PhyL is necessary or not. We also define the binary parameter y^z_l to denote VL-PhyL assignment. $r^i_{(v,v')}$ and y^z_l are defined as follows:

$$y^z_l = \begin{cases} 1, & \text{if VL } z \text{ is allocated on PhyL } (i, i'), \\ 0, & \text{otherwise,} \end{cases} \tag{4.8a}$$

$$r^i_{(v,v')} = \begin{cases} 0, & \text{if VNFs are within the same EN,} \\ 1, & \text{otherwise,} \end{cases} \tag{4.8b}$$

$$\sum_{z \in L_k} \sum_{l \in L_{(i,i')}} r^i_{(v,v')} \times y^z_l = 1 |\ i \neq i' \text{ and } v \neq v', \tag{4.8c}$$

In case VNFs of the same SFC are located within the same EN, Eq. (4.8c) ensures that the PhyL-VL allocation operation is not performed.

Processing Model

In the considered system model, each VNF process the forwarded packets using the allocated resource from the EN. Given a request $\rho_{(j,k)}$ with a packet size d_j, the processing

delay of a packet j by the VNF v is given as follows:

$$\gamma^{v}_{(i,j)}(t) = \frac{d_j}{\alpha^{v}_{i}(t)} \tag{4.9}$$

where $\alpha^{v}_{i}(t)$ is the portion of resource allocated to the VNF v. In fact, the division by $\alpha^{v}_{i}(t)$ can be written explicitly as follows,

$$\gamma^{v}_{(i,j)}(t) = x^{c}_{i,j}\frac{d^{c}_{j}}{c_i} + x^{s}_{i,j}\frac{d^{s}_{j}}{s_i} + x^{\omega}_{i,j}\frac{d^{\omega}_{j}}{\omega_i}$$

where $x^{c}_{i,j}$ is a binary variable to denote if the request requires the CPU capabilities, $x^{s}_{i,j}torage$ for storage capabilities, and $x^{\omega}_{i,j}$ for transmission capabilities. ($x^{\sim}_{i,j} = 1$) if the request requires the resource . Also, d^{c}_{j} represents the data size in terms of CPU, d^{s}_{j} for the storage, and d^{ω}_{j} for the packet size for transmission. For simplicity, we consider the notation in eq. (4.9). The transmission delay between two consecutive VNFs in EN i and i' is given as follows,

$$\gamma^{z}_{j}(t) = r^{i}_{(v,v')}\frac{d_j}{l^{z}_{(i,i')}(t)} \tag{4.10}$$

where $l^{z}_{(i,i')}$ represents the virtual link z capacity, given as follows,

$$l^{z}_{(i,i')} = B.\log_2\left(1 + \frac{P_i\left|g_{(i,i')}\right|^2}{\zeta^2 + \sum_{\substack{i''\in\mathcal{E}\\ i''\neq i}} P_{i''}\left|g_{(i'',i')}\right|^2}\right), \tag{4.11}$$

where B is the bandwidth, P_i the power of transmission of i, $g_{(i,i')}$ the channel gain between the ENs, and ζ^2 the noise variance. The overall delay experienced from requesting service k is given as follows,

$$\gamma_j(t) = \sigma_i\sum_{e_i\in\mathcal{E}}\gamma^{v}_{(i,j)}(t) + \sum_{l\in L_k}\gamma^{l}_{j}(t) + \gamma_{queue} \tag{4.12}$$

where γ_{queue} is the queuing time at the full controller queue. Without loss of generalities, we consider an $M/M/1$ queue, and we adopt the Little's law to cope with queuing delay.

4.4 Problem Formulation

4.4.1 The VFN placement and chaining subproblem

The problem of VNFs placement and chaining aims at finding the adequate placement of VNFs in their different virtual forms (e.g., VMs or containers) within the MEC nodes,

where each MEC has an aggregation of physical edge devices on which VMs/containers can be lunched following the demand. We formulate the problem as a mean-field resource allocation game [109, 110] in which many VNFs in the ENs are trying to operate within the available resources. We denote the game in MEC node i with $\mathcal{G}_i$. We consider a set $\mathcal{A}$ to represent the set of actions (strategies) to be chosen by players, which are given by the VNFs. An action $a_j \in \mathcal{A}$ could be a demand for a resource such as CPU, storage, or transmission. The game $\mathcal{G}_i$ is given as follows:

$$\mathcal{G}_i = \langle \mathcal{F}_i, \mathcal{A}, (r_j)_{j \in \mathcal{F}_i} \rangle, \tag{4.13}$$

where r_j denotes the payoff function of player j. It represents the amount of the allocated resources, given as follows [110]:

$$r_j(\mathcal{A}) = \alpha_i \times \frac{h(a_j)}{\sum\limits_{\substack{j' \in \mathcal{F}_i \\ j \neq j'}} h(a_{j'})} - a_j \times \gamma_{(i,j)}, \tag{4.14}$$

where $\gamma_{(i,j)}$ is the expected delay of using the resources at the EN i (i.e. $a_j \times \gamma_{(i,j)}$ represents the cost of the service) and $h(\cdot)$ is a mapping that should meet the following conditions:

$$\begin{cases} \sum\limits_{\substack{v' \in \mathcal{F}_i \\ v \neq v'}} h(a_{j'}) > 0, \\ h(0) = 0, \text{ and } h(\cdot) \text{ is a positive function.} \end{cases} \tag{4.15}$$

We define the mean-field term m_v, which represents an important parameter in the study of the mean-field game,

$$m_v = \left(\frac{1}{F} \sum_{j' \in \mathcal{F}_k} a_{j'}^{\beta} \right)^{\frac{1}{\beta}} \tag{4.16}$$

where $a_{j'}^{\beta}$ represents the action taken by the player j', *beta* represents the weight of the action (should be always $\beta > 0$), and $\mathcal{F}_k$ is the set of VNFs in the requested SFC. Hence,

we can express the payoff function and the mean-field terms differently as follows:

$$\begin{cases} r_v^* \left(a_j, m_{(v,-j)}\right) = \frac{\alpha_i}{v} \left(\frac{a_j}{m_v}\right)^\beta - a_j \times \gamma_{(i,j)} \\ m_{(v,-j)} = \frac{1}{v-1} \sum\limits_{\substack{j,j' \in \mathcal{F}_i \\ j \neq j'}} a_{j'}^\beta \end{cases} \tag{4.17}$$

We could then simplify the term of $m_{(v,-j)}$ and rewrite it as:

$$m_{(v,-j)} = \frac{1}{v-1} \left(m_v^\beta - \frac{a_j^\beta}{v}\right) \tag{4.18}$$

where $m_{(v,-j)}$ is the weighted mean when the action of user j is not chosen, and α_i is the amount of resource available of the requested resources. We note here that the payoff depends only on the chosen action and the mean m_v. Finally, we rewrite the term of the payoff as:

$$r_v^* \left(a_j, m_{(v,-j)}\right) = \frac{a_j^\beta \times \alpha_i}{m_{(v,-j)}(v-1) + a_j^\beta} - a_j \times \gamma_{(i,j)} \tag{4.19}$$

In the previous part, we discussed the game within a beforehand known number of VNFs. Nevertheless, the number of VNFs competing over the resources is variable and changes over time depending on the number of requests. With an increasing number of requests, the number of VNFs to be instantiated increases. Therefore, it is necessary to study this special case of MFG. In this case, we will study the MFG under an infinite number of players, which can be performed by applying a slight modification to the payoff function.

Let $\bar{\mathcal{G}}_i = \langle \mathcal{A}, \bar{r}_v \rangle$ be the game in which the number of players is infinite. A reformulation of the payoff function is required. Thus, we define the new payoff function as follows:

$$\bar{r}_v \left(a_j, m_{(v,-j)}\right) = \begin{cases} \frac{a_j^\beta}{\bar{m}^\beta} - a_j \times \gamma_{(i,j)}, m > 0 \\ 0, m = 0 \end{cases} \tag{4.20}$$

The problem of VNF placement and chaining can be defined as the optimal value of the strategy $a_j \in \mathcal{A}$ to be obtained while solving (4.20) for infinite number of VNFs and solving (4.17) for a finite number of VNFs.

4.4.2 The VNF scheduling subproblem

The objective of VNFs scheduling is to reach a stable scheduling guaranteeing that all the VNFs can get a fair share of MEC resources to accomplish their tasks. The aim is to allocate the MEC nodes resources for the VNFs in order to reduce the completion time

for each VNF. By the completion time we mean the necessary time required from a VNF to finish its processing. The VNF scheduling problem is NP-complete [7], thus, cannot be solved in polynomial time, unless $P = NP$. To efficiently solve the problem, we formulate it as a many-to-one matching game where the VNFs and the EN represent players with a list of preferences.

We consider that the access to the EN resource by the VNF starts if and only if the previous VNF for a given SFC is completely finished and forwarded to the following VNF whenever it exists. Let us consider the processing time for each VNF as p_v. We also define the starting time of VNF v at EN i as τ_v^i. The completion time ξ_v^i is simply the maximum of the sum of the values of p_v and τ_v^i.

We introduce time related parameters to express the operational assignment between the VNFs and the time slots. Let $x_{(i,v)}^{f,t}$ represents the assignment of time slot t to instance f of VNF v in EN i. $\tau_{(i,v)}^f$ is the starting time for the forward operation of the instance f of VNF v at the EN i, subsequently, $\theta_{(v,t)}^i$ is the forwarding operation delay.

The objective is to minimize the completion time,

$$\xi_v^i = \min \left\{ \max_{\substack{i \in \mathcal{E} \\ v \in \mathcal{F}_i}} \{p_v + \tau_v^i + \theta_{(v,t)}^i\} \right\} \tag{4.21}$$

considering the following constraints,

$$p_v + \tau_v^i \leq \tau_{v+1}^i \tag{4.22a}$$

$$\sum_{t=1}^{T} x_{(i,v)}^f . x_{(i,v)}^{f,t} \geq x_{(i,v)}^f \tag{4.22b}$$

$$\tau_{(i,v)}^f + \sum_{t=1}^{T} x_{(i,v)}^{f,t} . x_{(i,v)}^f \leq \tau_{(v')}^i \tag{4.22c}$$

$$\tau_v^i + \sum_{t=1}^{T} x_{(i,v)}^f . x_{(i,v)}^{f,t} \leq T_{out} \tag{4.22d}$$

$$\tau_{(i,v)}^f + \sum_{t=1}^{T} x_{(i,v)}^{f,t} . x_{(i,v)}^f \leq \tau_{(i,v')}^f \tag{4.22e}$$

$$\tau^{f}_{(i,v)} + \sum_{t=1}^{T} \theta^{i}_{(v,t)}.y^{l}_{z} \leq \tau^{i}_{v+1} \tag{4.22f}$$

$$\sum_{t=1}^{T} x^{f}_{(i,v,t)} + \theta^{i}_{(v,t)} \geq \gamma^{v}_{(i,j)} \tag{4.22g}$$

Constraint in Eq. (4.22a) ensures the order of starting time for the VNF v. Constraint in Eq. (4.22b) ensures the sufficiency of the total allocated time for the VNF v. Constraint in Eq. (4.22c) ensures that the scheduling for the VNF v is not performed earlier than it should be (*i.e.,* in cases where VNF v' is waiting to be processed/forwarded). Constraint in Eq. (4.22d) ensures that the time out of the processing/forwarding of the VNF v is respected. Constraint in Eq. (4.22e) ensures the completness of the process-forward process (*i.e.,* forwarding only the finished VNFs) Constraint in Eq. (4.22f) similar to Eq. (4.22e) ensures that a VNF will not be processed until the total forward of the previous VNF. Constraint in Eq. (4.22g) ensures that the requirement of each VNF are met.

4.5 Theoretical Game Approach solution

In this section, we will present our approach for MEC-enabled SFC resource allocation. The VNFs placement and chaining are provided through the resolution of the MFG. Then, based on the placement and chaining provided by the previous step, we provide a matching-based solution to schedule the VNFs' execution on different ENs. The proposed algorithms are executed sequentially. The first algorithm provides the second algorithm with the input, which is the placement of the different VFNs and the chaining logic between them. The second algorithm exploits this information to execute the SFC in a way that reduces latency and executes the SFC in the correct sequence.

4.5.1 The VNFs Placement and Chaining

Mean-Field Game based resource provisioning

The MFG-based solution relay on the IMLA algorithm (Algorithm 3). IMLA is performed on three stages; the learning stage, the placement stage, and the routing stage. In the learning stage (line 1-5), the algorithm trains the model to find the optimal strategy for each VNF by updating the mean-field term for several iterations. In each iteration, the controller chooses a random action, then computes the best response correspondence solution from Eq. (4.33). The obtained values are stored and the mean-field term is updated according to Eq. (4.16). At the end of this stage, the optimal strategy is obtained. This strategy is the values by which the system converges to a Nash equilibrium (NE). The

placement stage (line 6 and 7) takes place after learning the NE where the VNFs are placed accordingly by choosing the requested ones with optimal strategies, then instantiating the adequate VNFs' containers. Finally, the routing stage (line 8-10) consists in getting information about the placement of the other VNFs of the requested SFC on the other MEC nodes. Therefore, each VNF forwards the processed packets to the next VNF on the SFC.

Equilibrium analysis

Let us first introduce the best response correspondence problem (BRC) for each player (VNF in our case) when we have a fixed number of players. BRC is presented as follows; given the set of all possible strategies of all players but j, $a_{-j} = \{a_1, \ldots, a_{j-1}, a_{j+1}, \ldots\}$, find the maximum of their payoff functions, i.e. find a_j as:

$$a_j \in \arg\max_{a_{j'}} \{r_j(a_{j'}, a_{-j})\} \tag{4.23}$$

where a_j in this problem represents the best response of player j when another player asks for the same type of resources. BRC have a solution if we obtain a negative second-order derivative of the payoff $r_v^*(a_j, m_{(v,-j)})$; $\partial_{a_j}^2 r_v^*(a_j, m_{(v,-j)}) < 0$ (see appendix 6.4 for the

Algorithm 3: IMLA: Ishikawa-Mann Learning Algorithm.

Input: λ, $\mathcal{A}$, β, $\rho_{j,k}$

Learning stage;

1 **for** $t = 0$, $t < iterations$ **do**

2 | Choose random action $a_j \in \mathcal{A}$;

3 | Compute $BRC(a_j)$ from Eq. (4.33);

4 | Update the mean term in Eq. (4.16)

5 **end**

Placement stage;

6 Choose VNFs with optimal actions according to Eq. (4.23);

7 Instantiate the VNFs containers;

Routing stage;

8 Get other nodes VNFs placement;

9 Select next VNFs for the requested SFC;

10 Solve the problem of Eq. (4.21) to forward processed packets;

proof), therefore, a Nash equilibrium exists and is given by:

$$\sqrt{a_j^{\beta-1}\left(\frac{\beta\rho_i}{v\gamma_{(i,j)}}\frac{1}{v}\sum_{\substack{j,j'\in\mathcal{F}_i\\ j\neq j'}}a_{j'}^{\beta}\right)}-\frac{a_j^{\beta}}{v}-\frac{1}{v}\sum_{\substack{j,j'\in\mathcal{F}_i\\ j\neq j'}}a_{j'}^{\beta}=0 \tag{4.24}$$

We notice that the actions space and the payoff function are identical and invariant respectively, which means that this game is symmetric[110]. Therefore, NE is given as follows:

$$\sqrt{a_j^{\beta-1}\left(\frac{\beta\rho_i}{v\gamma_{(i,j)}}\frac{v-1}{v}a_{j'}^{\beta}\right)}-\frac{a_j^{\beta}}{v}-\frac{v-1}{v}a_{j'}^{\beta}=0 \tag{4.25}$$

By developing the term of Eq. (4.25), we obtain the NE value for the BRC:

$$a_j^*=\frac{\beta\rho_i\,(v-1)}{v^2\gamma_{(i,j)}} \tag{4.26}$$

In addition, whenever a equilibrium exists, all players have symmetric strategies, and, the NE payoff is given as,

$$r_v^*(a_j^*)=a_j\gamma_{(i,j)}\left[\frac{a_j^{\beta}+M}{\beta M}-1\right] \tag{4.27}$$

with M is given as,

$$M=\frac{1}{v}\sum_{\substack{j'\in\mathcal{F}_i\\ j\neq j'}}a_{j'}^{\beta} \tag{4.28}$$

For values of $\beta\in[0,1]$, we have $r_v^*\geq 0$, otherwise, if $\beta>1$, then the strategies of the players using the resources depends on others that does not use the resources and the number of VNFs executing tasks is satisfying:

$$\frac{\beta}{\beta-1}\geq v \tag{4.29}$$

Under these conditions, the solution of BRC is equal to 0. The resources are not wasted (*i.e.,* efficient usage) when the delay is equal to,

$$\gamma_{(i,j)}=\beta\times\frac{v-1}{v} \tag{4.30}$$

For $\gamma_{(i,j)} < \beta$, the requested amount of resources and the available resources are equal at the equilibrium, hence, the efficient ratio is written as

$$\nu = v \times \frac{a^*_{NE}}{\rho_i} \tag{4.31}$$

The equilibrium increases with β and the amount of requested resources and decreases with the delay $\gamma_{(i,j)}$.

Reinforcement Learning based solution

In this paper, we propose an iterative learning algorithm that converges to a NE. This algorithm is executed by the SDN controller to compute the adequate placement and chaining of the VNFs in the SFCs. The benefit of using an iterative algorithm is the low computational complexity required for reaching equilibrium. We use the following formulation based on Ishikawa-Mann iteration [111],

$$a_{(t+1)} = \lambda BRC(a_t) + a_t(1-\lambda) \tag{4.32}$$

where λ is the learning rate, and BRC is the solution function to the problem in Eq. (4.23), given as follows,

$$BRC(a_j) = \max\left\{ v\sqrt{\frac{\beta\alpha_i}{v\gamma_i}\left(m - \frac{a}{v}\right)} - \left(m - \frac{a}{v}\right), 0 \right\} \tag{4.33}$$

Eq. (4.33) is obtained through solving the equation $BRC(a^*_j) = a^*_j$.

Regarding the infinite regime (*i.e.,* cases with a huge number of VNFs competing over the EN resources), we stick with the same solution approach based on the BRC, with a small change regarding the mean-field term. In the case of a huge number of users, the mean-field term is also updated accordingly with the evolution of the BRC stages as follows,

$$a_{(i,t+1)} \in \arg\max_{a'_j} \left\{ \bar{r}(a'_j, m_t) \right\} \tag{4.34}$$

and the main objective is to maximize the payoff at the next stage, given as,

$$m_{t+1} = \lambda . BRC(m_t) + m_t(1-\lambda) \tag{4.35}$$

The complexity of the proposed Ishikawa-Mann algorithm (Algorithm 3) depends on the complexity of the BRC function. However, considering that the values of the learning

rate are smaller and fixed, the BRC converges to a fixed point ι in near-proximity of the equilibrium value a_j^*, which makes the convergence time $\log(\frac{1}{\iota})$ [110]. Therefore, we conclude that the proposed approach converges in a logarithmic time $O(\log(v))$.

4.5.2 The VNF scheduling subproblem

To illustrate the subproblem, Fig. 4.2 provides an example of SFC scheduling. The scheduling time is essentially the sum of the processing time for each VNF in the SFCs, the transmission, and the queuing delay. By processing, we mean the processing of the request by a given VNF, which could include the computational and the storage. The transmission time is considered to take part in the queuing part.

Many-To-One Matching Game

Considering the constraints on resource provisioning and the execution order of VNFs from the previous section, the SDN controller, as a supervising entity of the network, needs to find an adequate schedule to execute the VNFs of the SFCs on the different nodes of the system. The reasons behind formulating the problem of VNFs scheduling as a many-to-one matching game because multiple VNFs can be matched to one MEC node. The VNFs-resources assignment at each moment of the network operations is considered as the output of a many-to-one matching game. The players are the VNFs showing their interest to be matched to the resources of an EN and we define the requirements for a matching game, namely, the stability, blocking pairs, and preferences lists of each set of players as

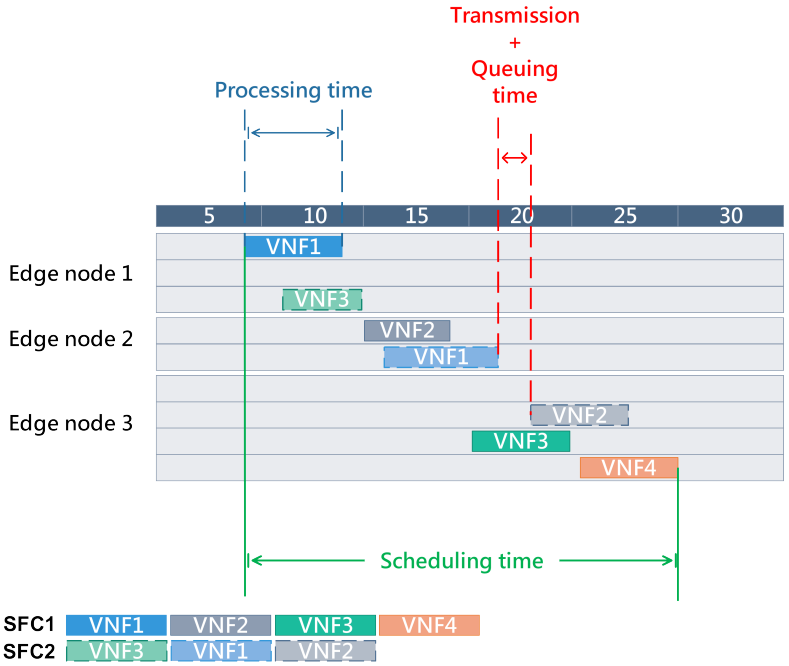

Figure 4.2 Scheduling time example for an SFC.

follows. For simplicity, the VNFs-resources assignment will be denoted as the VFNs-ENs assignment.

The preference is defined as binary relationship defined between the elements of the players sets. The relationship is complete, reflexive, and transitive. Let us denote $\succ_i$ the preferences list of the ENs and $\succ_{f_{(i,j)}}$ the preference list of the VNFs. $e_i \succ_{f_{(i,j)}} e_{i'}$ means that the VNF $f_{(i,j)}$ prefers the EN i over the EN $e_{i'}$. Using the same notation, we define the preferences of ENs over VNFs by $\succ_i$.

Definition 4.5.1 (Many-to-One matching game) *A many-to-one matching game is a two-sided assignment problem between two disjoint sets of players where the players on the first set express their interest to be matched to a player of the other set based on a preference. Let μ be a many-to-one mapping from the set of VNFs to the set of ENs. $\mu : \mathcal{F}_i \mapsto \mathcal{E}$ satisfying the following conditions:*

- $\forall f_{(i,j)} \in \mathcal{F}_i$ *we have* $\mu(f_{(i,j)}) \in \mathcal{E}$
- $\forall e_i$ *we have* $\mu(e_i) \in \mathcal{F}_i$
- $\mu(f_{(i,j)}) = e_i$ *exits, only if* $e_i \in \mathcal{F}_i$

We consider that the VNFs-ENs assignment takes into account a minimum and maximum number of VNFs to be matched to an EN to operate. These quantities are called quotas (minimal and maximal) and ensure a good resource consumption and fairness. Let q_i^{max} and q_i^{min} be the maximum and minimum quota, respectively, of the EN i. μ is feasible if and only if $q_i^{min} \leq q_i \leq q_i^{max}$ with, $q_i = |\mu(f_{(i,j)})|$ and $|\mu(f_{(i,j)})| = \{0, 1\}$. A stable matching is defined as follows:

Definition 4.5.2 (Stable matching) *A matching μ is said to be stable if there is no intention from any pair $(f_{(i,j)}, e_i)$ to deviate from μ. In other words there is no blocking pair.*

A blocking pair is defined as follows,

Definition 4.5.3 (Blocking pair) $(f_{(i,j)}, e_i)$ *is said to be a blocking pair if it satisfies the following conditions:*

- $e_i \succ_{f(i,j)} e_{i'}$, *for* $e_{i'} \in \mu(f_{(i,j)})$
- $f_{(i,j)} \succ_i f'_{(i,j)}$

Under the previous two definitions, μ is stable and guarantee stability and fairness. *a - The VNFs preference List*
As for the preferences of the VNFs, they always prefer an EN that will process/forward

Algorithm 4: eMSDA: enhanced Multi-Staged Deferred Acceptance Algorithm.

Input: $\succ_{f_{(i,j)}}$, Fp, $\succ_i$, F_k, E, q_i^{min}, q_i^{max}
Output: $\mu(f_{(i,j)})$
Initialization: $q_i^{0,min} = q_i^{min}$, $q_i^{0,max} = q_i^{max}$, $\forall i \in \mathcal{E}$

```
while Fp ≠ ∅ do
    Q^θ = Σ_{i∈E} q_i^{θ,min};
    R^θ = {f_v, ...}  with  Σ_{i∈E} α_i(t) ≤ Q;
    UM = R^{θ-1} \ R^θ;
    if UM ≠ ∅ then
        Apply classic DAA on unmatched VNFs in UM with q_i^{θ,max};
    else
        Apply classic DAA on unmatched VNFs in UM with q_i^{θ,min};
    end
    if (4.22c)-(4.22e) are met then
        Add matched VNFs to μ;
        Remove matched VNFs from Fp;
        Update the quotas according to Eqs. (4.38a) and (4.38b);
    end
end
return μ
```

them in the shortest delay, hence, with available resources (based on the results of the VNFs placement and chaining) at the current time slot. The preference list of VNFs is given as follows,

$$\tau_{f_{(i,j)}}(a_j) = t \tag{4.36}$$

b - The EN Preference List

The edge nodes, however, are more interested in executing/forwarding VFNs with smaller processing requirements.

$$\tau_{a_i}(f_{(i,j)}) = \inf_{j \in \mathcal{F}_i} [\rho_j] \tag{4.37}$$

Considering the quota constraints, the classic DAA [112] will not satisfy the feasibility of the matching (see appendix 6.4). We consider a staged deferred acceptance algorithm that supports the constraints of minimum and maximum quota. Our algorithm is called enhanced multi-stage deferred acceptance algorithm (eMSDA) and detailed in Algorithm 4. In addition, Nash equilibrium exists whenever a stable matching exists [113].

Enhanced Multistage Deferred Acceptance Algorithm

eMSDA is developed in Algorithm 4 inspired from the work in [114], which is based on a DAA algorithm performed in several stages. eMSDA works under the assumption that all the ENs share the same preference list ranking all the VNFs following their requirements in terms of resources. In addition, since the network is controlled by the SDN controller, this ensures that that the process of making the preferences list is easy. In this algorithm, Q^{θ} represents the sum of the minimum quotas of all the MEC nodes. R^{θ} and $R^{\theta-1}$ are the set of unmatched VNFs in the current stage, and the previous stage respectively. We denote by Fp the list of preferences, and $\succ_{\mathrm{Fp}}$ the shared preference relationship. Initially, we temporarily reserve a subgroup of VNFs and perform the DAA on the remaining subgroup. The assignments at a given stage θ (line 10-15) for a subset of EN and VNFs are considered final, which means that those VNFs are not destroyed until they complete their tasks. Then, accordingly, we reduce the minimum and maximum quotas as [114],

$$q_i^{\theta,min} = max \left\{ q_i^{\theta-1,min} \sum_{i \in \mathcal{E}} \alpha_i(t), 0 \right\} \tag{4.38a}$$

$$q_i^{\theta,max} = q_i^{\theta,max} - \sum_{i \in \mathcal{E}} \alpha_i(t) \tag{4.38b}$$

At every stage θ, VNFs are ranked from the most preferred and the least preferred. The number of the least preferred VNFs is the sum of the minimum quota of all the ENs. Depending on the number of preferred and non-preferred, we run the DAA with the maximum quota. Moreover, VNFs cannot be matched unless the previous VNF is finished (line 5-9). eMSDA is assumed to be online, where requests arrive in an online fashion. In the admission process of the requests, once the VNFs are placed and chained they can join the scheduling process without re-computing the scheduling. The complexity of this algorithm depends on the number V of VNFs to be matched, the number N of MEC nodes, and the number K of rounds required to match all the VNFs O(N x V x K). In the proposed solution, the number of MEC nodes is not considered in the problem, hence, M=1. This leaves us with a complexity of O(V x K). In the worst case, the proposed algorithm converges to a stable matching in a number of rounds less than the number of VNFs to be matched. Therefore, the complexity of O(KN).

4.6 Simulation Results

In this section, we present the evaluation and the performance of the proposed IMLA for the placement and chaining problem, and eMSDA for VNF scheduling problems. We

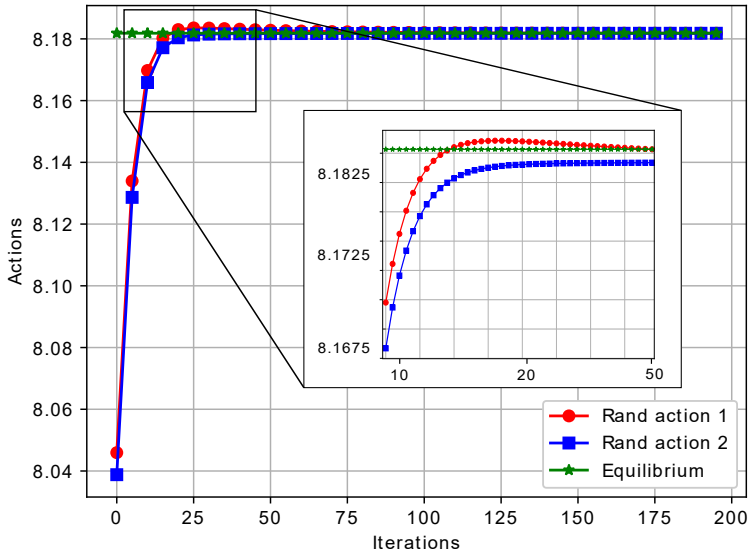

Figure 4.3 MFG Stability for finite number of VNFs. $\lambda = 0.05$, $v = 10$.

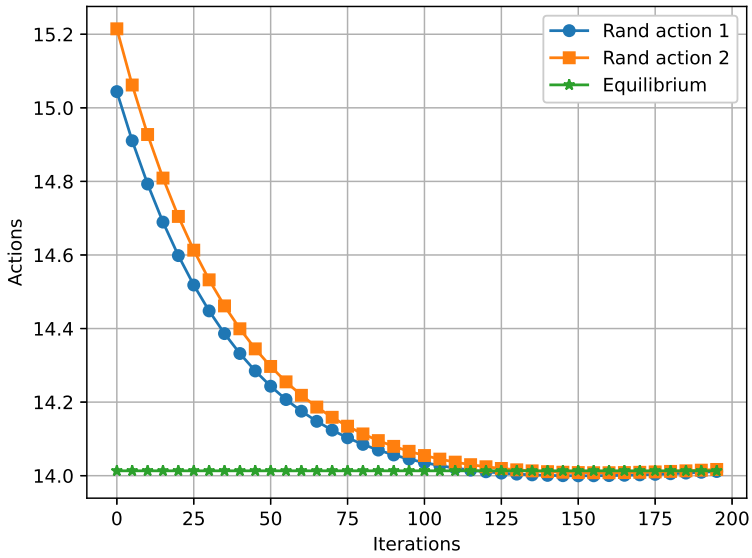

Figure 4.4 MFG Stability for infinite number of VNFs. $\lambda = 0.05$, $v = 1000$.

compare the performance of the algorithm in terms of execution delay and resource consumption to the performance of a genetic algorithm based solution and an integer linear programming based solution.

4.6.1 Games stability and convergence

To prove the existence of the equilibrium for the VNF placement and chaining problem, we simulated the BRC solution and compared it to the NE. We performed exhaustive simulations on both finite and infinite regimes. Results are illustrated in Fig. 4.3 and Fig. 4.4. For both scenarios, the actions sets were generated randomly as, $a_j \in \mathcal{A}$ and $\mathcal{A} = \mathbb{R}_+$. Fig. 4.3 illustrates the evolution of the chosen action when performing the IMLA algorithm. It is shown that both generated actions sets converges to a NE, and reached it after approximately after 35% of learning time.

IMLA algorithm converges to a NE whenever the learning rate is sufficiently small and constant. In fact, the function of Eq. (4.33) converges to a fixed point (i.e. equilibrium) with small values of λ, precisely when $\lambda \in]0, 1[$ [110]. In our case, we consider a small value of $\lambda = 0.05$.

4.6.2 System Evaluation

In this section, we present the simulation results to illustrate the performance of the proposed IMLA for the VNF placement and chaining subproblem and the proposed eMSDA for the VNFs scheduling subproblem. The simulations were performed using Python programming language and running on a i7 10th generation processor with 16Gb of RAM and a NVidia 2070 (8Gb) graphic card. We modeled the different system entities as classes having parameters related to the number of processors and their frequency, the RAM capacity, and the network capabilities. Each class has methods to perform the tasks and return different simulation parameters such as the resource consumption and the latency. A VNF is a function having a set of instructions and each instruction has a size that is assumed to be constant. Since we are interested in reducing the latency in this work, we simply considered the time to process a task, which is the fraction of the task size by the processing capacity of the MEC node. We compared the performance of the proposed algorithms in terms of execution delay and resource consumption to the performance of a genetic algorithm-based solution and an integer linear programming algorithm. Table 4.2 summarizes the significant simulation parameters.

Fig. 4.5 and Fig. 4.6 illustrate the performance of the proposed approach in terms of execution delay in function of the number of the VNFs in play. It is shown that the proposed algorithms outperforms both of other approaches. In fact, both of the proposed algorithms (*i.e.,* IMLA and MSDA) have, first, a lower complexity compared to other algorithms in terms of their implementations. Second, for instance the IMLA execution time depends on the convergence of the BCR solution, which is very low compared to the benchmark algorithms (*i.e.,* $O(\log(v))$ vs $O(Mv^2)$, with M is the number of generations

Table 4.2 Summary of the simulation parameters

Parameters	Values
MEC nodes	$[3-15]$
VNFs	$[1-1000]$
Learning rate λ	0.05
Learning iterations	200
Packets size	$[100-10000]$Kb
Processing capacities	$[1.0-2.5]$Ghz
Finite scenario number of users	10
Infinite scenario number of users	1000
Actions set $\mathcal{A}$	$\mathbb{R}_+$ randomly chosen

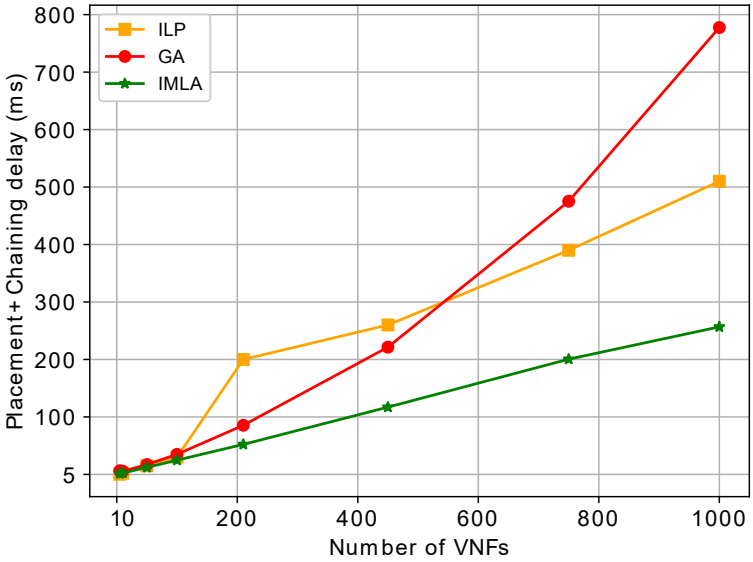

Figure 4.5 Placement and chaining delay in function of the packet size.

after which the GA converges to a solution). As for the scheduling, the GA approach show a constant behavior compared to the proposed approach and to the ILP-based approach. Although a stable behavior when it comes to a huge number of VNFs, it is not beneficial in case the number of VNFs to schedule is small. Fig. 4.5 and Fig. 4.6 show that the proposed approach based on the game theory model consumes less time for the operational delay, nearly 40% less than the GA and 25% compared to ILP in the case where the number of VNFs to manage is sufficiently big (*i.e.,* 1000 in this case).

Fig. 4.7 shows the evolution of the delay in function of the packet's size for both the eMSDA and the IMLA. In this scenario, we considered the average delay obtained from the different setups of the random scenarios through varying the number of ENs, SFCs,

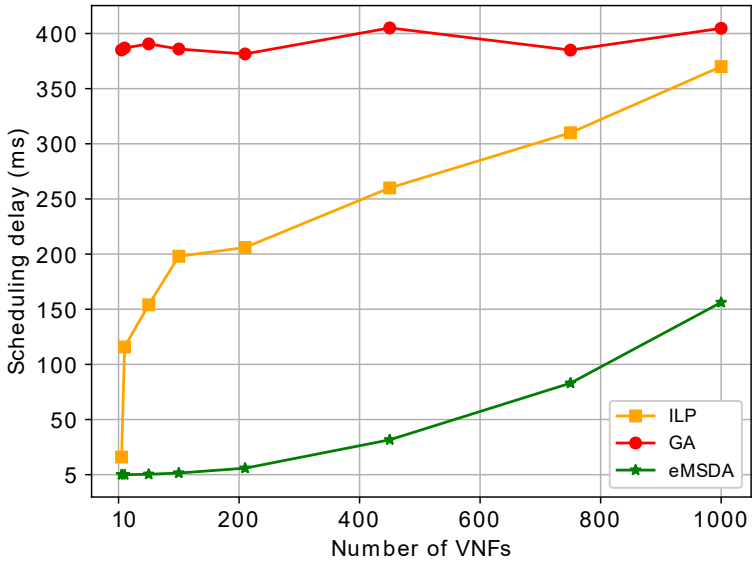

Figure 4.6 Scheduling delay in function of the number of VNFs.

and the number of VNFs per SFC and also varying the size of the packets processed by the SFCs. In most of the cases, the proposed approach shows an enhacement in terms of execution delay of the SFCs. The proposed approach offers a reduced delay (around 45%) compared to other approaches when it comes to processing different configurations of packets in terms of packet's size. Such result is justified by, first, the reduced time that both proposed algorithms are showing in terms of placing and chaining, and second, by the enhanced scheduling provided by the eMSDA, that not only provides a stable matching for the VNFs and ENs resources but also provides a low delay in terms of execution time.

Fig. 4.8 and Fig. 4.9 illustrate the resource consumption of the proposed algorithm compared to the GA and ILP approaches. It is shown that the proposed approach gives better results in terms of resource consumption (*i.e.,* CPU and memory). The GA approach shows a constant behavior also in terms of memory consumption, which can explain more the results obtained in Fig. 4.6. As for ILP, the consumption of the CPU and the memory, still higher than the proposed approach and the GA. We can conclude from Fig. 4.8 and Fig. 4.9 that the proposed approach offers enhanced results compared the benchmarked approaches. The theoretical game based approach shows around 40% lower than the ILP approach and 25% better than GA in some cases.

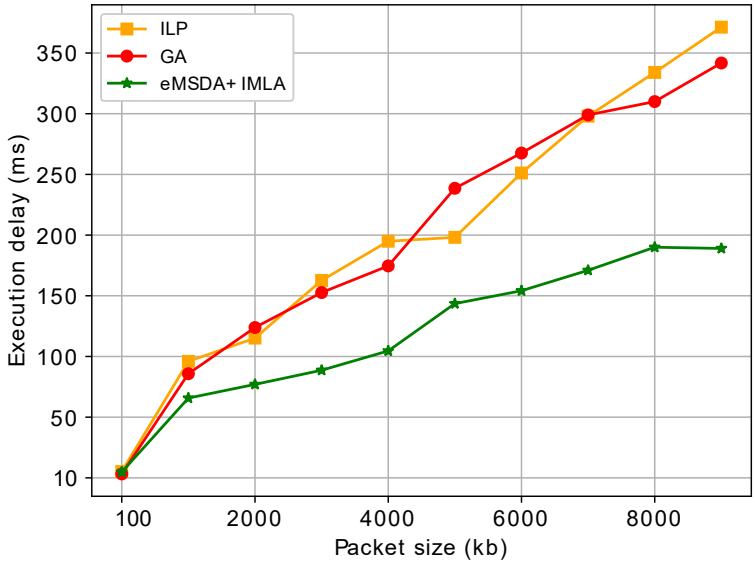

Figure 4.7 Processing time in function of the packet size.

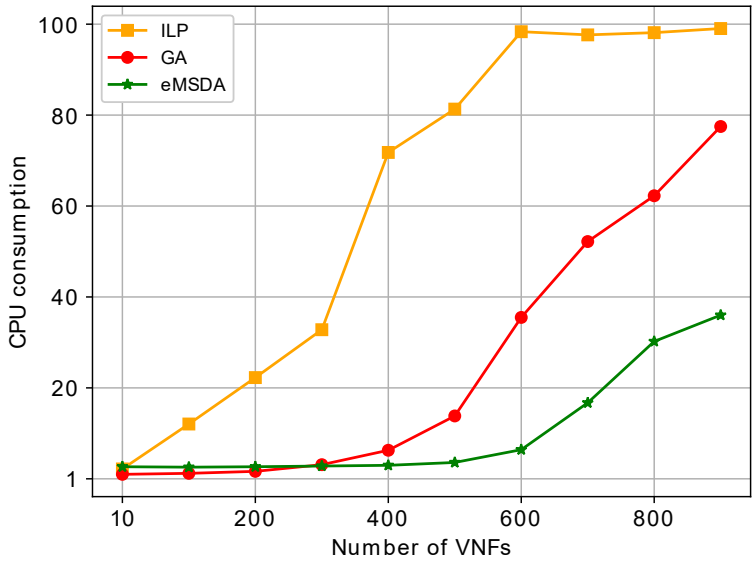

Figure 4.8 CPU consumption evolution in function of the number of VNFs.

4.7 Conclusion

In this paper, we studied VNF resource provisioning for SFC in a MEC context with the goal of reducing latency. We addressed the full problem of resource provisioning through addressing the VNF placement, the VNF routing and the VNF scheduling. As for the VNF placement and routing problem, we formulated it as a mean field game where VNFs

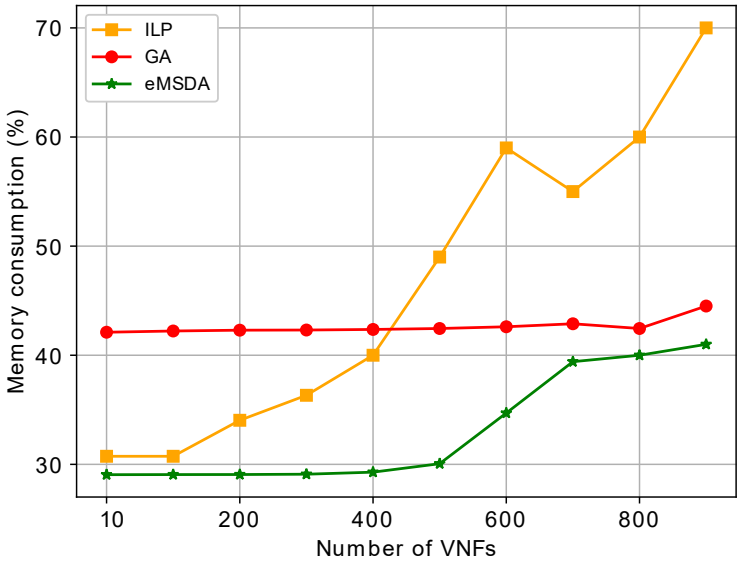

Figure 4.9 Memory consumption evolution in function of the number of VNFs.

are the players that are competing for MEC resources with the goal of reducing the resource consumption of MEC nodes and decreasing latency for end users. We exploited the Ishikawa-Mann iterated learning algorithm to place and chain the VNFs. We formulated the VNFs scheduling problem as a many-to-one matching to match multiple VFNs to MEC node resources, and we proposed a modified version of the classical deferred acceptance algorithm. We performed extensive simulations to prove the effectiveness of our proposed approaches, which were found to outperform the benchmark approaches in the studied scenarios. In future work, we will investigate the possibility of testing the proposed approach on a real testbed to compare the theoretical results with the experimental results.

4.8 Related Works

The literature review we performed on VNF/SFC resource allocation approaches led us to categories the proposed approaches under three main categories. Theoretical games-based approaches [7], linear programming variant approaches [45, 57, 54], the machine learning approaches [52, 56, 54] and the heuristic based approaches [46, 58]. The proposed approaches are proposed in the following subsections.

Work in [7] highlighted the importance of the VNF scheduling, and proposed a matching-based algorithm, namely, the one-to-one matching to cope with the scheduling problem which is NP-hard. The proposed approach guarantees the stability of the scheduling process. However, the authors in this work consider only the scheduling of one VNF at

the time, and even in the transmission phase, the whole channel is allocated to one and only one VNF. Compared to our work, we considered a many-to-one matching game to schedule multiple VNFs at the time. Regarding the channel allocation, we got inspired from the slicing techniques to allocate the channel for multiple VNFs at the time. Work in [45] proposed an online provisioning for NFV considering both unicast and multicast services with two types of VNFs, mandatory and best-effort instances. In addition, the work considered only the computational and transmission resources. Yet, it did not make any assumption about the resource state at the edge nodes and consider only two types of resources. The same work also supposed that the requests arrives in batches having the same requirements of resources which may not cover all the use cases of SFC resource provisioning while placing, chaining the VNFs. The authors also did not consider the scheduling problem and only investigated the routing and placement of the VNFs. Work in [57] addressed the problem of reliability, delay, and resource allocation for SFC in softwarized 5G networks. The work suggested an SFC sub chaining method to cope with the reliability and an Integer Linear Programming based solution for the problem of SFC placement through a matching algorithm that provides a new-optimal polynomial solution. However, in this work, the authors did not provide much detail about the format of the request, types of resources, nor the dynamic of request arrival. In addition, the proposed matching algorithm might not be efficient when the nodes do not consider its maximum capacities. From a machine learning approach, the authors in [52] investigated the problem of VNFs scheduling with the aim to minimize the overall completion time of the services under delay constraints. Authors proposed a RL based solution to learn the best scheduling scheme. In fact, the problem of scheduling is formulated as a mixed integer linear program relaxed as an Markov decision process solved using RL. However, the request arrival rate in this work is not considered stochastic and the RL is based on Q-Learning algorithm, which acts only on a set of specific fixed instance of the problem. Work in [56] models the SFC scheduling problem as a flexible job-shop scheduling problem with the objective to minimize the scheduling latency. Authors proposed a deep RL based on Q-Learning that gives the environment the advantage of performing adaptive scheduling.

Within a heuristic context, work in [59] investigated the problem of resource scheduling with the aim to enhance the usage of network resources as well as the minimization of the experienced end-to-end delay for the network services. Authors proposed an approach based the genetic algorithm through improving the crossover and mutation operations. Although genetic algorithms are good in supporting multi-objective optimization, the cost of computation in terms of time is very high. Within the same context, authors in [58] proposed a configurable service allocation scheme for VNF embedding and routing.

Due to NP-Hardness of the investigated problem, the work is considered as an integer non-linear programming model and solved it through heuristic methods, specifically a greedy algorithm. The proposed solution leveraged the properties of the different entities of the system and balance the resources consumption. Work in [46] proposed a VNF placement scheme for MEC environment and formulated of VNF placement problem under minimizing access latency and maximizing service availability constraints. To reduce the problem complexity, the authors proposed a Genetic Algorithm and compared the obtained results with a CPLEX implementation of the same problem. However, the proposed model makes a lot of assumptions mainly related to the dynamicity of the model.

In our work, we first, consider four types of resources, namely, computation, transmission, storage, and memory. Secondly, we consider a stochastic request arrival model which mimic as real as possible a real SFC deployment. Finally, we consider that the VNFs might have different resource requirement and we tackled the problem of resource allocation considering all the SFC phases (i.e. placement, chaining, and scheduling).

CHAPTER 5

A Deep Reinforcement Learning Service Migration in Slice-enabled Internet of Vehicles

Date de parution: Octobre 2021
Conférence: IEEE LCN
Titre français: Migration des services d'apprentissage par renforcement profond dans l'Internet des véhicules à base de tranches.

Resumé français: L'informatique périphérique multi-accès est un outil essentiel pour réduire la latence des réseaux de véhicules. En raison de la mobilité des véhicules, les services qu'ils demandent (par exemple, les services d'info-divertissement) doivent fréquemment être transférés sur différents serveurs MEC pour garantir leurs exigences strictes en matière de qualité de service. Dans cet article, nous étudions le problème de la migration des services dans un réseau véhiculaire MEC afin de minimiser la latence totale des services et le coût de la migration. Ce problème est formulé comme un programme non linéaire que nous avons linéarisé afin de faciliter l'obtention de la solution optimale à l'aide de solveurs standard. Ensuite, pour obtenir une solution efficace, ce problème est modélisé comme un processus de décision de Markov multi-agents (MMDP) que nous avons résolu en utilisant l'algorithme d'apprentissage profond (DQN). Le schéma DQN proposé effectue une migration proactive des services tout en assurant leur continuité sous des contraintes de mobilité élevées. Enfin, les résultats des simulations montrent que le schéma DQN proposé atteint une performance proche de l'optimal.

5.1 Abstract

MEC is a key enabler to reduce the latency of the vehicular network. Due to the mobility of the vehicle, their requested services (e.g., infotainment services) should frequently be migrated across different MEC servers to guarantee their stringent quality of service requirements. In this paper, we study the problem of service migration in a MEC-enabled vehicular network in order to minimize the total service latency and migration cost. This problem is formulated as a nonlinear integer program and is linearized to help obtain the optimal solution using off-the-shelf solvers. Then, to obtain an efficient solution, it

is modeled as a multi-agent Markov decision process and solved by leveraging the DQL algorithm. The proposed DQL scheme performs a proactive services migration while ensuring their continuity under high mobility constraints. Finally, simulations results show that the proposed DQL scheme achieves close-to-optimal performance.

5.2 Introduction

Intelligent transportation systems represent a critical component of the IoT and future smart cities. ITS will potentially provide a more secure transportation environment through effective vehicle coordination and efficient resource management [46]. In addition to safety, the ITS ecosystem will provide entertainment services such as video streaming and gaming, which can be extended to in-vehicle augmented reality [115, 116, 102]. To achieve these promising features, vehicles must be able to communicate, exchange information, and access is given services with low latency. Therefore, vehicles must operate in an environment that meets these requirements.

MEC is envisioned as a key component for 5G uRLLC services, alongside SDN [43] technology. On the one hand, MEC can be leveraged as an emerging computational paradigm that provides efficient computational capabilities to vehicles deployed in close proximity to MEC servers while ensuring low latency. On the other hand, SDN technology enables seamless, transparent, and efficient control through the separation of the data plane and the control plane, which simplifies network operation and management [95, 117]. Therefore, a MEC-enabled vehicular network can benefit from SDN to provide efficient resource management and uRLLC vehicular services [118, 51, 119]. Nevertheless, due to the limited resources of MECs and the high mobility of vehicles, there are many challenges. In particular, the requested vehicular services must be located and migrated to different MEC servers to guarantee their continuity [120, 61]. To address these challenges, we investigate the service placement and migration problem in a MEC-enabled vehicular network. We leverage SDN technologies to have efficient control of the MEC server's operations, with the objective of reducing the average service latency of the vehicles. We first, formulate the problem of service placement and migration as a nonlinear integer program that we linearize to obtain the optimal solution using off-the-shelf solvers. Second, we modeled the problem as a multi-agent Markov decision process (MMDP), in order to solve it efficiently using deep reinforcement learning (DRL) techniques, specifically, DQN. The proposed DRL-based placement and migration scheme ensures service continuity under high mobility constraints and offers a reduced total service latency as well as the additional operational costs associated with the migration. The proposed scheme performs proactive

placement of the requested services while considering the mobility of vehicles, the required amounts of computational and communication resources, and the overall migration costs.

To summarize, the main contributions of this paper are synthesized as follows:

- We formulate the service placement and migration problem as a non-linear program to minimize the total service latency (including the computing latency and the communication latency) and the cross-edge operational costs.
- We propose an MMDP framework that helps solving the problem in a distributed and scalable manner.
- We leverage DRL techniques to provide efficient solution to the MMDP model. Specifically, we propose a DQL-based solution that uses double Q network and replay buffer to improve the learning outcome.
- We evaluate the performance of the proposed DRL-based scheme and compare it to the optimal solution obtained by the CPLEX solver and we show that the proposed solution achieves close-to-optimal performance.

The remainder of this paper is structured as follows. In Section II, we present the system model and the problem formulation of the service placement and migration problem. In Section III, we present the proposed multi-agent DQL-based solution. The performance of the proposed solution is evaluated in Section IV. Last but not least, the related works are discussed in Section V. Finally, the paper is concluded in Section VI.

5.3 System Model

We consider an SDN-enabled MEC architecture covered with a set of gNodeBs (gNBs), each is equipped with a MEC server $n \in \mathcal{N} \coloneqq \{1, 2, \ldots, N\}$ that is connected to one gNB via high speed local-area network as illustrated in Fig. 5.1. There are K mobile users (or interchangeably called vehicles) demanding services from the MEC servers and are denoted by the set $\mathcal{K} \coloneqq \{1, 2, \ldots, K\}$. Each vehicle k requests some service to fulfill its requirements. Without loss of generality, we assume that all vehicles request the same vehicular service [1] (e.g., an infotainment-related service). Similar to previous works [61, 121, 122], we consider a MEC-based device-oriented service model contrary to the traditional cloud-based application-oriented service model. In other words, a dedicated container or virtual machine is assigned the vehicular service as well as the applications' environment, which are executed on each vehicle rather than on each application. An SDN controller is assumed to be placed on the cloud layer where it acts as a central controller for information

1. The case of multiple services will be considered in our future work where network slicing will be integrated into our system model.

exchange between vehicles. Time is discrete and is divided into a set of T time-slots denoted by the set $\mathcal{T} = \{1, 2, \ldots, T\}$. At each time-slot $t \in \mathcal{T}$, each vehicle $k \in \mathcal{K}$ requests the vehicular service from the MEC node $n \in \mathcal{N}$.

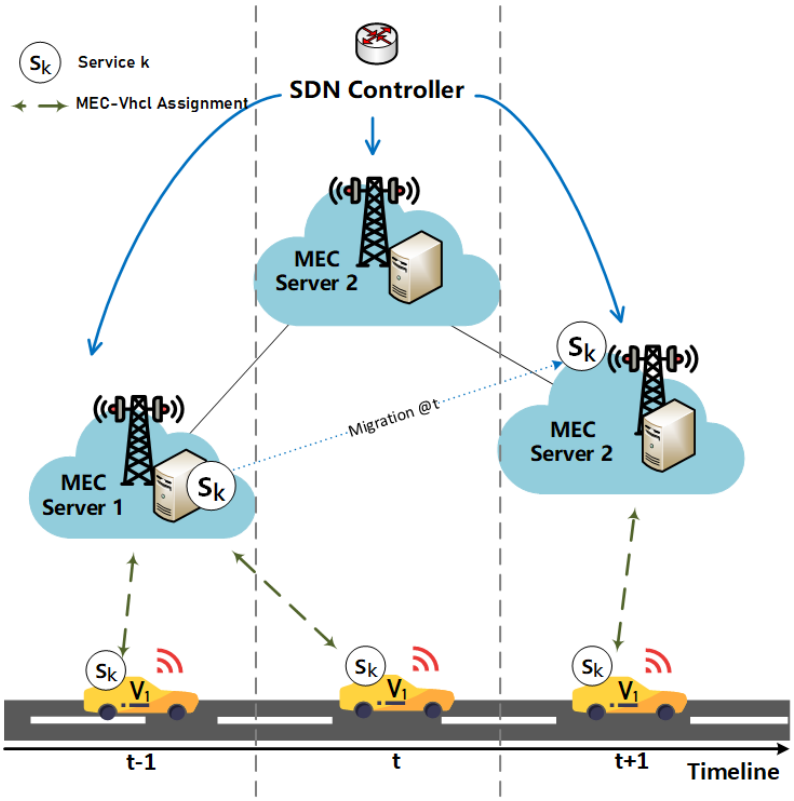

Figure 5.1 Illustration of the system model

The objective of this work is to guarantee the minimum quality of service (QoS) requirements of the vehicles while considering their erratic mobility and the computing and communication resources of MEC servers. To do so, the requested vehicular service should be placed and migrated across different MEC servers depending on the vehicles mobility patterns. In this work, we consider an hybrid centralized-distributed architecture where (i) each MEC server plays the role of an agent that makes its service placement and migration decisions independently of other MEC servers, and (ii) once each MEC agent makes its decision, it communicates it to the SDN controller that plays the role of a central agent to coordinate the decisions of all MEC servers.

The considered QoS is represented by the vehicular service latency that includes (i) the communication delay that is incurred by the transmission between a vehicle and a MEC server, and (ii) the computing delay that depends on the processing capability of the MEC server as well as the size of the vehicle's request.

Communication Delay

When a vehicle k requests from a MEC server n the vehicular service, the transmission between k and n depends mainly on the wireless environment and on the size of the requested service. The channel power gain between vehicle k and MEC server n at time-slot t is denoted by g_{kn}^t, which includes the small-scale fading as well as the large-scale fading. To simplify the analysis, we assume that the total available bandwidth is divided equally between the MEC servers and each MEC server allocates its bandwidth to the vehicles in an orthogonal manner. Accordingly, the received signal to noise ratio between MEC server n and vehicle k at time-slot t is given by

$$\gamma_{kn}^t \coloneqq \frac{p_n g_{kn}^t}{\sigma^2}, \tag{5.1}$$

where p_n is the transmit power of MEC server n and σ^2 is the power of the noise. The achieved data rate can be given as follows

$$\Gamma_{kn}^t \coloneqq w_n \lg\left(1+\gamma_{kn}^t\right), \qquad \text{[in bits/sec]} \tag{5.2}$$

where w_n is the allocated bandwidth of MEC server n. Consequently, the communication delay between MEC server n and vehicle k at time-slot t is given by:

$$d_{kn}^t \coloneqq \frac{s_k}{\Gamma_{kn}^t}, \qquad \text{[in sec]} \tag{5.3}$$

where s_k is the size of the requested service of vehicle k.

Computing Delay

The computing delay depends on the processing capacity of each MEC server n, on the total vehicles sharing MEC server n, and on the requested computing capacity of the vehicular service of vehicle k at time-slot t. More precisely, the computing delay between MEC server n and vehicle k at time-slot t is given as follows [61]:

$$c_{kn}^t \coloneqq c_k^t N_n^t / F_n, \qquad \text{[in sec]} \tag{5.4}$$

where c_k^t denotes the amount of computing capacity [in CPU cycles] required by the requested vehicular service of vehicle k at time-slot t. The computing capacity of MEC server n [in CPU cycles/sec] is given by F_n and the number of vehicles placed on MEC server n at time-slot t is given by N_n^t.

Besides the QoS requirements, placing and migrating the vehicular service across multiple MEC servers incur additional operational costs related, for example, to the energy con-

sumption and the bandwidth usage. For this reason, we consider the migration cost as an important factor into the design of our service migration solution.

Migration Cost

Due to the cross-edge migration, additional operational costs are incurred by the service migration. These costs include energy consumption, expensive wide-area network bandwidth usage, etc. [61]. To make the operational cost model general, we use $m_{n'n}^{kt}$ to denote the cost of migrating the vehicular service of vehicle k from MEC server n' to MEC node n at time-slot t. Obviously, we assume that $m_{n'n}^{kt} = 0$, for all $n' = n$ and for all k, t.

5.4 Problem Formulation

To guarantee the required QoS (communication and computing delays) and the migration cost, the optimization problem is formulated as a multi-objective optimization problem where the aim is to optimize the communication delay, the computing delay as well as the migration costs. To simplify the resolution of this multi-objective problem, we transform the multi-objective problem into a single objective one by introducing the weights λ_i for $i \in \{1, 2, 3\}$. The formulated problem is written as a nonlinear integer program (NLP) as follows.

$$\underset{\mathbf{x}}{\text{minimize}} \quad \lambda_1 C(\mathbf{x}) + \lambda_2 D(\mathbf{x}) + \lambda_3 M(\mathbf{x}) \tag{P1a}$$

$$\text{subject to} \quad x_{kn}^t \in \{0, 1\}, \quad \forall k, n, t, \tag{P1b}$$

$$\sum_{n=1}^{N} x_{kn}^t = 1, \quad \forall k, t, \tag{P1c}$$

where the variables $x_{kn}^t = 1$ if and only if the vehicular service requested by vehicle k is placed at MEC server n at time-slot t. We denote by $\mathbf{x}$ the multidimensional notation of the variables x_{kn}^t, i.e., $\mathbf{x} = [x_{kn}^t]$. The objective function in (P2a) is a linear combination of the communication delay, the computing delay, and the migration cost. Constraints (P2b) guarantee that the variables x_{kn}^t are binary. Constraints (P2c) guarantee that the vehicular service requested by vehicle k at time-slot t is placed at one and only one MEC server.

The total computing delay $C(\mathbf{x})$ is defined as follows:

$$C(\mathbf{x}) \coloneqq \sum_{n=1}^{N} \sum_{k=1}^{K} \sum_{t=1}^{T} x_{kn}^t N_n^t c_k^t / F_n, \tag{5.5}$$

where c_k^t denotes the required amount of computing capacity [in CPU cycles] of the vehicular service requested by vehicle k at time-slot t and F_n denotes the maximum computing capacity of MEC server n [in CPU cycles/sec]. The term N_n^t denotes the number of services placed at MEC server n, i.e.,

$$N_n^t \coloneqq \sum_{k=1}^{K} x_{kn}^t. \tag{5.6}$$

The total communication delay $D(\mathbf{x})$ is defined as follows:

$$D(\mathbf{x}) \coloneqq \sum_{n=1}^{N}\sum_{k=1}^{K}\sum_{t=1}^{T} x_{kn}^t d_{kn}^t, \tag{5.7}$$

where d_{kn}^t denotes the computing delay between MEC server n and the vehicle k at time-slot t (see (5.3)).

Finally, the total migration cost $M(\mathbf{x})$ is defined as follows:

$$M(\mathbf{x}) \coloneqq \sum_{n=1}^{N}\sum_{n'=1}^{N}\sum_{k=1}^{K}\sum_{t=1}^{T} x_{kn'}^{t-1} x_{kn}^t m_{n'n}^{kt}, \tag{5.8}$$

where the migration cost $m_{n'n}^{kt}$ is used to denote the cost of migrating the service k from MEC server n' to MEC server n at time-slot t. It is clear that the cost is counted inside the summation only if both $x_{kn'}^{t-1}$ and x_{kn}^t are equal to one, i.e., $x_{kn'}^{t-1} = x_{kn}^t = 1$, which means that the requested service of vehicle k is placed at MEC server n' at time-slot $t-1$ and is placed at MEC server n at time-slot t. This costs includes bandwidth costs incurred by cross-edge migration (e.g., wide-area network bandwidth usage costs) as well as energy costs caused by increased energy consumption of network devices such as routers. To make the model general, we use a general cost term $m_{n'n}^{kt}$ as done in [61].

In order to make the problem more tractable, we linearize the objective function given in (P2a). The non-linearity of (P2) comes from the functions $C(\mathbf{x})$ and $M(\mathbf{x})$ due to the multiplication of binary variables. To linearize $M(\mathbf{x})$, we introduce a new binary variable called $z_{n'n}^{kt} = x_{kn'}^{t-1} x_{kn}^t$. It is clear that this new z-variable is positive if and only if each term of the product of the x-variables is positive. In other words, $z_{n'n}^{kt} = 1 \iff x_{kn'}^{t-1} = x_{kn}^t = 1$. This means that we must add the following two constraints to force the z-variable to be

zero whenever $x_{kn'}^{t-1}$ or x_{kn}^{t} is zero:

$$z_{n'n}^{kt} \leq x_{kn'}^{t-1}, \forall k, n, n', t > 1, \tag{5.9}$$

and

$$z_{n'n}^{kt} \leq x_{kn}^{t}, \forall k, n, n', t. \tag{5.10}$$

It remains to enforce the constraints that if both $x_{kn'}^{t-1}$ and x_{kn}^{t} are equal to one, then the z-variable is one. This can be written as follows:

$$z_{n'n}^{kt} \geq x_{kn'}^{t-1} + x_{kn}^{t} - 1, \forall k, n, n', t > 1. \tag{5.11}$$

Thus, the total migration cost can be rewritten as follows:

$$M(\mathbf{z}) = \sum_{n=1}^{N} \sum_{n'=1}^{N} \sum_{k=1}^{K} \sum_{t=1}^{T} z_{n'n}^{kt} m_{n'n}^{kt}, \tag{5.12}$$

where $\mathbf{z}$ denotes the multidimensional notation of the variables $z_{n'n}^{kt}$, i.e., $\mathbf{z} = [z_{n'n}^{kt}]$.

Now, to linearize $C(\mathbf{x})$, we let y_k^t denote the quantity $\sum_{n=1}^{N} x_{kn}^{t} N_n^t c_k^t / F_n$, i.e.,

$$y_k^t \coloneqq \sum_{n=1}^{N} x_{kn}^{t} N_n^t c_k^t / F_n, \forall k, t. \tag{5.13}$$

Thus, the total computing delay $C(\mathbf{y})$ can be rewritten as follows:

$$C(\mathbf{y}) = \sum_{k=1}^{K} \sum_{t=1}^{T} y_k^t, \tag{5.14}$$

where $\mathbf{y}$ denotes the multidimensional notation of the variables y_k^t, i.e., $\mathbf{y} = [y_k^t]$. Now, we have to enforce that the following constraints:

$$x_{kn}^{t} = 1 \Rightarrow y_k^t = N_n^t c_k^t / F_n. \tag{5.15}$$

These are indicator constraints that can be easily implemented in the off-the-shelf solvers such as CPLEX or Gurobi. Nonetheless, they can be easily transformed to linear constraints using the big-M method [123].

5.5 Problem Formulation

The problem is a multi-objective optimization problem where the aim is to optimize the communication delay, the computing delay as well as the migration costs. We transform the multi-objective problem into a single objective one by introducing the weights λ_i for $i \in \{1, 2, 3\}$. The formulated problem is written as a nonlinear integer program (NLP) as follows.

$$\underset{\mathbf{x}}{\text{minimize}} \quad \lambda_1 C(\mathbf{x}) + \lambda_2 D(\mathbf{x}) + \lambda_3 M(\mathbf{x}) \tag{P2a}$$

$$\text{subject to} \quad x_i^k(t) \in \{0, 1\}, \quad \forall i, k, t, \tag{P2b}$$

$$\sum_{i=1}^{N} x_i^k(t) = 1, \quad \forall k, t, \tag{P2c}$$

The computing delay $C(\mathbf{x})$ is defined as follows:

$$C(\mathbf{x}) \coloneqq \sum_{i=1}^{N} \sum_{k=1}^{K} \sum_{t=1}^{T} x_i^k(t) n_i(t) \frac{c^k(t)}{f_i}, \tag{5.16}$$

where $c^k(t)$ denotes the required amount of computing capacity in CPU cycles of service k at time-slot t and f_i denotes the maximum computing capacity of MEC node i (in CPU cycles per second). The term $n_i(t)$ denotes the number of services placed at MEC node i, i.e.,

$$n_i(t) \coloneqq \sum_{k=1}^{K} x_i^k(t). \tag{5.17}$$

The communication delay $D(\mathbf{x})$ is defined as follows:

$$D(\mathbf{x}) \coloneqq \sum_{i=1}^{N} \sum_{k=1}^{K} \sum_{t=1}^{T} x_i^k(t) d_i^k(t), \tag{5.18}$$

where $d_i^k(t)$ characterizes the general communication delay between MEC node i and a user when its corresponding service k is placed on MEC node i at time-slot t. It mainly depends on the link bandwidth between the MEC node and the user and the amount of data transferred to the user. We use a general temr $d_i^k(t)$ to make our problem generic and independent to a specific transmission model.

Finally, the migration cost $M(\mathbf{x})$ is defined as follows:

$$M(\mathbf{x}) \coloneqq \sum_{i=1}^{N}\sum_{i'=1}^{N}\sum_{k=1}^{K}\sum_{t=1}^{T} x_{i'}^{k}(t-1)x_{i}^{k}(t)m_{i'i}^{k}(t), \tag{5.19}$$

where the migration cost $m_{i'i}^{k}(t)$ is used to denote the cost of migrating the service k from MEC node i' to MEC node i. This costs includes bandwidth costs incurred by cross-edge migration (e.g., wide-area network bandwidth costs) as well as energy costs caused by increased energy consumption of network devices such as routers. To make the model general, we use the general term $m_{i'i}^{k}(t)$ as done in...

In order to make the problem more tractable, we linearize the objective function given in (P2a). The non-linearity of (P2) comes from the functions $C(\mathbf{x})$ and $M(\mathbf{x})$ due to the multiplication of binary variables. To linearize $M(\mathbf{x})$, we introduce a new binary variable called $z_{i'i}^{k}(t) = x_{i'}^{k}(t-1)x_{i}^{k}(t)$. It is clear that this new z-variable is positive if and only if each term of the product of the x-variables is positive. In other words, $z_{i'i}^{k}(t) = 1 \iff x_{i'}^{k}(t-1) = x_{i}^{k}(t) = 1$. This means that we must add the following two constraints to force the z-variable to be zero whenever $x_{i'}^{k}(t-1)$ or $x_{i}^{k}(t)$ is zero:

$$z_{i'i}^{k}(t) \leq x_{i'}^{k}(t-1), \forall i, i', k, t > 1, \tag{5.20}$$

and

$$z_{i'i}^{k}(t) \leq x_{i}^{k}(t), \forall i, i', k, t. \tag{5.21}$$

It remains to enforce the constraints that if both $x_{i'}^{k}(t-1)$ and $x_{i}^{k}(t)$ are equal to one, then the z-variable is one. This can be written as follows:

$$z_{i'i}^{k}(t) \geq x_{i'}^{k}(t-1) + x_{i}^{k}(t) - 1, \forall i, i', k, t > 1. \tag{5.22}$$

Thus, the migration cost can be rewritten as follows:

$$M(\mathbf{z}) = \sum_{i=1}^{N}\sum_{i'=1}^{N}\sum_{k=1}^{K}\sum_{t=1}^{T} z_{i'i}^{k}(t)m_{i'i}^{k}(t), \tag{5.23}$$

Now, to linearize $C(\mathbf{x})$, we let $y^k(t)$ denotes the quantity $\sum_{i=1}^{N} x_i^k(t) n_i(t) \frac{c^k(t)}{f_i}$, i.e.,

$$y^k(t) \coloneqq \sum_{i=1}^{N} x_i^k(t) n_i(t) \frac{c^k(t)}{f_i}, \forall k, t. \tag{5.24}$$

Thus, the computing delay $C(\mathbf{y})$ can be rewritten as follows:

$$C(\mathbf{y}) = \sum_{k=1}^{K} \sum_{t=1}^{T} y^k(t). \tag{5.25}$$

Now, we have to enforce that the following constraints:

$$x_i^k(t) = 1 \Rightarrow y^k(t) = n_i(t) \frac{c^k(t)}{f_i}. \tag{5.26}$$

These are indicator constraints that can be easily implemented in the off-the-shelf solvers such as CPLEX or Gurobi. They can be easily transformed to linear constraints using the big-M method.

5.6 Proposed Solution

In this section, we propose a deep reinforcement learning (DRL) based approach to obtain an efficient solution to the service placement and migration problem defined in (P2). The proposed approach places the vehicular service requested by the vehicles in the appropriate MEC servers to ensure the continuity of services under the mobility constraint of vehicles while reducing the communication latency, the computing latency as well as the migration costs of the requested service.

We use deep Q-learning (DQL) [124]—one of the most popular DRL algorithm—to efficiently solve the service placement problem in the MEC-enabled vehicular network. DQL combines Q-learning with deep neural network (DNN). It takes as input the observed state of the environment and returns as output the Q-value of all possible actions. DQL has two main phases, namely the training phase and the inference phase. In the training phase, the agent trains a DNN, called deep Q-network (DQN), in an offline manner. In the inference phase, the agent takes actions in an online manner based on the trained DQN. Before describing each phase of the proposed DQL algorithm, we model, first, the problem as a Markov decision process (MDP).

5.6.1 The MDP Formulation

We consider a multi-agent MDP where each MEC server n acts as an independent agent, called herein after MEC agent n. At time-slot t, each MEC agent n can decide either to place and instantiate the vehicular service requested by vehicle k or not. The key elements of the multi-agent MDP are defined as follows:

The State Space

At time-slot t, the observed state by the MEC agent n, denoted by $\mathcal{S}_n^t$, mainly depends on the current vehicular environment. It includes the current positions of the vehicles, their velocities, their directions, and their service requirements (including the wireless channel gains and the distances between MEC agent n and the vehicles). Note that there are as many states as there are time-slots, i.e., every time-slot corresponds to a state. In addition, a transition from one state to the next happens according to the mobility model of the vehicles.

The Action Space

The action set of each MEC agent n at time-slot t is given by the set $\mathcal{A}_n^t \coloneqq \{0,1\}^K$. Indeed, an action $\boldsymbol{a}_n^t \in \mathcal{A}_n^t$ is given by the row vector $[a_{1n}^t, a_{2n}^t, \ldots, a_{Kn}^t]$, where each element a_{kn}^t corresponds to the decision to place the service $k \in \mathcal{K}$ at MEC server n, all happening at time-slot t. Note that the variable a_{kn}^t and x_{kn}^t defined in (P2) means essentially the same thing but to remove any possible confusion between the optimization variable x_{kn}^t and the MDP action a_{kn}^t we use two different notations. Each MEC agent n communicates its chosen action to the SDN controller to form a global action $\boldsymbol{a}^t \coloneqq [\boldsymbol{a}_1^t, \boldsymbol{a}_2^t, \ldots, \boldsymbol{a}_N^t]$. Then, the SDN controller verifies if the individual actions of the MEC agents are feasible or not according to the constraints of (P2), i.e., the individual action $\boldsymbol{a}_n^t$ of MEC agent n is considered feasible if it meets the constraints of (P2).

The Reward Function

A MEC agent n chooses an action $\boldsymbol{a}_n^t \in \mathcal{A}_n^t$ at time-slot t and receives a reward R_n^t. Since we seek to minimize the overall vehicular service latency requested by the vehicles, the objective of MEC agent n must be related to the sum-latency of the services it hosts. In other words, we define the reward R_n^t of MEC agent n at time-slot t in relation with how the placement of requested service at n affects the latency of the system. Therefore, the SDN controller calculates the individual reward of MEC agent n as follows:

$$R_n^t \coloneqq \begin{cases} \lambda_1 C_n^t + \lambda_2 D_n^t + \lambda_3 M_n^t, & \text{if } \boldsymbol{a}_n^t \text{ is feasible} \\ -1, & \text{if } \boldsymbol{a}_n^t \text{ is not feasible,} \end{cases} \tag{5.27}$$

where $C_n^t = \sum_{k=1}^{K} a_{kn}^t N_n^t c_k^t / F_n$ is the computation delay, $D_n^t = \sum_{k=1}^{K} a_{kn}^t d_{kn}^t$ is the communication delay, and $M_n^t = \sum_{\substack{n'=1 \\ n' \neq n}}^{N} \sum_{k=1}^{K} a_{kn'}^{t-1} a_{kn}^t m_{n'n}^{kt}$ is the migration cost of MEC agent n at time-slot t. If the action chosen by MEC agent n at time-slot t is not feasible, this MEC agent should be penalized with a negative reward $R_n^t = -1$ to prompt it to not choose this action in future steps.

5.6.2 The Training Phase of DQL

In general, DQN approximates the Q-values $Q(s, a, \theta)$ of each state-action pair (s, a) using a DNN, where θ represents the parameters of the Q-network. Since we propose a multi-agent MDP, the proposed DQL algorithm will be a multi-agent algorithm in which each MEC agent will have its own DQN to be approximated and trained. When there is no confusion, we omit the index n from the DQN of MEC agent n. In addition, the training process of the DNN uses the experience replay memory mechanism. This mechanism helps in creating a dataset to train the DNN once in a while by storing each MEC agent experience into a replay buffer. This experience essentially includes the current state, the next transition state, the chosen action and the received reward. Then, each MEC agent randomly chooses a set of samples from its replay buffer to perform the learning process. The experience replay memory mechanism not only allows the MEC agent to learn from the past experiences, but also to provide uncorrelated data as inputs which breaks undesirable temporal correlations. However, DQN is known to overestimate the Q-values of stat-action pairs under certain conditions, which harms the performances. To overcome this issue, double DQN (DDQN) [125] is proposed which reduces the overestimation and makes the training process faster and more reliable. Indeed, DDQN uses two DNNs, called the main Q-network and the target Q-network. The former is used to compute the Q-values $Q(s, a, \theta)$ while the latter is used to provide the target Q-values $Q(s, a, \theta^-)$ to train the parameters θ of the main Q-network. The training phase of our proposed multi-agent DQL algorithm is presented in Algorithm 5, where each MEC agent $n \in \mathcal{N}$ trains its own DDQN.

The training phase of the DQL algorithm requires as input the vehicular environment which includes the vehicles, the requested services, the MEC servers, the computing capacity of MEC servers. It returns the trained DDQN of each MEC agent as output. The DDQNs are trained simultaneously. The training begins by generating the vehicles parameters and the network parameters. The vehicles parameters include the position, the velocity and the requested service of each vehicle. The network parameters include the computing capacity of each MEC server. Then, the DQL algorithm initializes the DDQN parameters of each MEC agent. Next, it iterates the episodes. For each episode, the en-

Algorithm 5: The Training Phase of DQL

Input: Agents and environment
Output: Trained DDQNs
Initialization: Generate vehicles and network parameters;
Initialize the DDQN of each agent n;
for *Episode* e **do**
  Reset and build the agents' environment;
  for *Time-slot* t **do**
    for *each MEC* n **do**
      Observe the environment ;
      Choose an action a_n^t using ϵ-greedy;
    end
    The SDN controller obtains the global action ;
    The SDN controller calculates the individual reward of each agent;
    for *each MEC* n **do**
      Receive the individual reward from the SDN controller;
      Observe the next state of the environment ;
      Store the experience Exp_n^t in the replay buffer $\mathcal{M}_n$;
      if *batch size* **then**
        Sample a mini-batch from $\mathcal{M}_n$;
        Do a mini-batch training;
      end
      if *target step* **then**
        Update the target network parameters θ_n^-;
      end
    end
  end
end

vironment of each MEC agent is built by updating the position of the vehicles according to the mobility model and generating other network parameters. For each episode, the training continues for a period of time-slots (or steps). In each step t, each MEC agent n observes the current state of its environment and chooses an action from its action space $\mathcal{A}_n^t$. To select an action, the MEC agent uses the ϵ-greedy policy. With this policy, an action is chosen randomly with probability ϵ. Once all MEC agents select their actions, each of them communicates its action to the SDN controller to construct the global action $\boldsymbol{a}^t$. The SDN controller uses the constructed global action to verify its feasibility and calculate the individual reward of each MEC agent. Then, each MEC agent n receives its individual reward R_n^t from the SDN controller and moves to the next state. The obtained experience, denoted by Exp_n, is stored by the MEC agent n in its replay buffer $\mathcal{M}_n$. When the replay buffer contains enough experiences, i.e., a certain batch size is respected,

each MEC agent randomly samples a mini-batch to create a training dataset. The latter is used by the MEC agent to perform the training process. In the training process, each MEC agent seeks to minimize a loss function, given by:

$$L_n^t(\theta_n) = \mathbb{E}[(y_n - Q(\mathcal{S}_n^t, \boldsymbol{a}_n^t; \theta_n))^2], \tag{5.28}$$

where $Q(\mathcal{S}_n^t, \boldsymbol{a}_n^t; \theta_n)$ is the Q-value of action $\boldsymbol{a}_n^t$ given the state $\mathcal{S}_n^t$ which is calculated using the main Q-network with parameters θ_n; y_n is the target Q-value, which calculated using the target Q-network with parameters θ_n^- and it is given as follows:

$$y_n = R_n^t + \gamma Q(\mathcal{S}_n^t, \underset{\boldsymbol{a}_n^t}{max}\{Q(\mathcal{S}_n^t, \boldsymbol{a}_n^t; \theta_n)\}; \theta_n^-), \tag{5.29}$$

where $0 \leq \gamma \leq 1$ is the discount factor.

To update the parameters θ_n of the main Q-network, MEC agent n performs a gradient descent step. Finally, each MEC agent updates the parameters θ_n^- of its target Q-network at a fixed target step by copying the parameters of the main Q-network.

5.6.3 The Inference Phase of DQL

The inference phase of DQL is presented in Algorithm 6. Once the trained DDQNs are obtained, each MEC agent uses its optimal DDQN parameters to find an appropriate placement of the requested service by the vehicles. In detail, at the beginning of each episode the environment of each MEC agent is built. Then, for each step t, each MEC agent observes the current state of its environment and selects an action that maximizes its Q-value according to its trained DDQN. Based on the selected actions of all MEC agents, the SDN controller finds the overall communication delay, computing delay and migration costs and thus we obtain a solution to problem (P2).

5.7 Simulation Results

We consider a MEC-enabled vehicular network where three gNBs that are attached to three MEC servers are deployed over a highway as shown in Fig. 1. The gNBs are located randomly along the highway. We assume that the three MEC servers are deployed along the highway in a triangular fashion as depicted in Fig. 1, where the distance between MEC 1 and MEC 2 and the distance between MEC 2 and MEC 3 is equal to 2000 m and the distance between MEC 1 and MEC 3 is equal to 4000 m. The vehicles are drawn randomly in the highway that is modelled as a rectangle of length 5000 m and width 18 m with two forward lanes and two backward lanes. The vehicles move with a randomly-chosen fixed speed from the range of $[60, 110]$ km/h and once a vehicle reaches the boundary of the

Algorithm 6: The Inference Phase of DQL

Input: The trained DDQNs
Output: Placement of the vehicular service of each vehicle
Initialization: Load the DDQN of each agent n;
for *Episode* e **do**
  Reset and build the environment;
  for *Step* t **do**
    for *each MEC* n **do**
      Observe the environment ;
      Choose an action a_n^t that maximize the Q-value of the tained DDQN of n;
    end
    The SDN controller obtains the global action;
  end
  The SDN controller calculates the objective function as in (P2a);
end

highway, it reappears in the opposite side. For simplicity, all vehicles keep moving with constant speeds with acceleration, i.e., once the random speed of a vehicle is chosen, the latter keeps moving with that speed during the entire simulation period.

The proposed multi-agent DQL algorithm is trained on an computer with an Intel Core i7-10750H CPU, 16GB RAM and an nVidia GeForce GTX 2070 Super graphic card. The implementation is performed using Python and PyTorch. After performing hyper-parameters tuning, the following optimized parameters are set. Each DDQN consists of fully connected hidden neural network with two hidden layers of 256 neurons each. The discount factor is $\gamma = 0.99$. The other DDQN and vehicular network para metes are presented in Table 1. To avoid the overestimation problem of the Q-value, the parameters of each DDQN network are copied into the parameters of the corresponding target DDQN every 1000 steps.

Fig. 5.2 illustrates the average reward per episode of one MEC agent. It is clear that the reward improves with the training episodes as it increases when the number of episodes increases. This shows the effectiveness of the proposed DQL algorithm. We notice that the DQL algorithm converges at approximately 1000 episodes. In other words, the corresponding MEC agent converges to a good learning outcome, which implies that it will explore better actions. We can notice though that the reward converges while incurring large fluctuations which is mainly due to the high mobility scenario of the vehicular network.

Fig. 5.3 and Fig. 5.4 illustrate the average cost represented by the objective function (P2a) which measures the total service latency (the computing and communication latency) as

Table 5.1 Simulation parameters

Parameter	**Value**
Number of MEC servers	3
Transmit power of each gNB	30 dBm
Migration cost	UNIFORM$(0.2, 0.3)$
Number of vehicles	4
Request size	UNIFORM$(50, 300)$ Kbits
Noise variance	-174 dBm/Hz
Bandwidth	10 MHz
Learning rate	$3e-4$
Number of episodes	3000
Discount factor	0.99
Replay memory size	100000
Mini-batch size	1024
Target update interval	1000
Loss function	Minimum square error
Optimizer	Adam
Activation function	ReLU

well as the migration costs under two different configurations. The first configuration is the computational power configuration and it consists of varying the number of cores for the three MEC servers. We considered three MEC servers with 4 cores each, or 8 cores each, or 16 cores each, or 32 cores each, or 64 cores each, with a fixed clock frequency of 2.5 GHz. The second configuration is the request size configuration and it consists of varying the vehicles' request sizes, which we generate uniformly at random within a fixed interval as follows UNIFORM$(50,\ 100)$, UNIFORM$(100, 150)$, UNIFORM$(150, 200)$, UNIFORM$(200, 250)$, UNIFORM$(250, 300)$ Kbits.

Fig. 5.3 shows the objective function while considering the computational power configuration. We can notice that with increasing the computational power, the average service latency as well as the migration costs are decreasing. Regardless of different number of cores, the proposed DQL approach performs close-to-the-optimal performance. Fig. 5.4 presents the objective function while considering the request size configuration. The proposed approach is shown to approximate well the optimal solution in different scenarios of

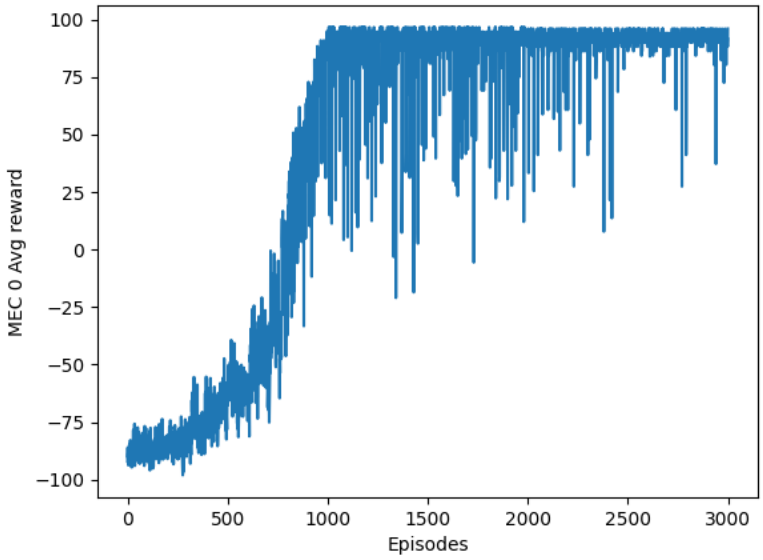

Figure 5.2 The training rewards for MEC server.

the request size configuration. The smaller the request size is, the smaller the total delay and the migration costs are. This is because (i) the DQL approach learns efficiently the appropriate placement of each vehicle's service, which helps in reducing the service latency and (ii) a small number of bits can be fulfilled easily by one MEC server without requiring to migrate to another MEC server, which helps in reducing the total migration costs. We notice that as the request sizes increase, the average service delay and the migration costs increase as well.

5.8 Related Works

The authors in [60] propose a QoE-aware scheme to ensure service continuity for the mobile cloud computing environment. The scheme relies on the buffer-occupancy threshold policy that classifies the new arriving request from the mobile users. The proposed scheme protects the migrated service from traffic fluctuation. In addition, the cloud server can change the buffer threshold dynamically for different categories of requests. In [61], the authors proposed a Follow-Me Chain algorithm to solve the problem of SFC. In particular, the work studied the problem of inter-MEC handoffs to offer a higher satisfaction for users in mobility scenarios. Such a problem is NP-hard, and the authors proposed an integer programming formulation that is solved by the Follow-Me Chain algorithm. The work in [63] investigated the relocation problem of VNF within a cloud infrastructure under mobility and resource heterogeneity constraints. The authors studied in particular the impact of the relocation operation on the service delay and the number of VNF relocations (i.e.,

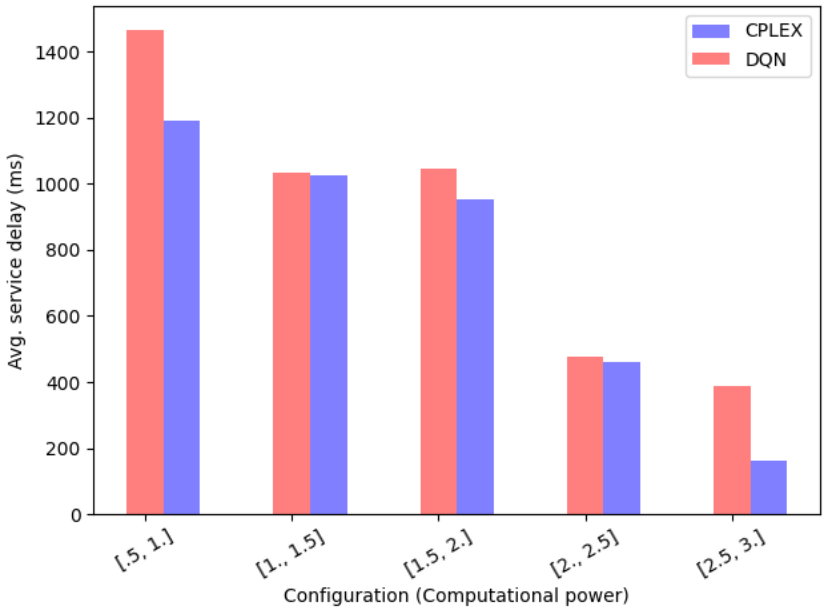

Figure 5.3 The objective function vs. the computational power of the MEC servers.

the number of times that a single VNF is being moved from a cloud to another). The problem of relocation was formulated as a mixed-integer linear programming problem and solved through a meta-heuristic approach, namely, the ant colony optimization technique. Within the same context, the authors in [64] proposed an evaluation of three container-based schemes for VNF migration as a mechanism to guarantee service continuity. In particular, the schemes consider two cases of mobility patterns, respectively, known a priori and unknown mobility patterns. For the known, a priori pattern, temporary file system, and disk-less-based migration are discussed, but the main focus was on the unknown mobility pattern, where authors proposed a solution that consists in storing the container's file system within the system images in a shared pool. The work in [126] considered two main logical slices created over the same infrastructure, namely, an autonomous driving slice for safety messages, and an infotainment slice. The authors proposed a clustering method to partition vehicles to allocate slice leaders on each cluster. A slice leader is a serving entity using vehicle-to-vehicle (V2V) links to forward safety messages, subsequently, the roadside units (RSU) forward the infotainment service using the vehicle-to-infrastructure (V2I) links. In [127], the authors proposed an offline RL-based RAN slicing solution and a low-complexity heuristic algorithm, to satisfy communication resources requirements of different slices to maximize the resource utilization. The proposed approach ensures the resource availability to meet the different requirements of the slice's traffic. The authors assume that V2V communications are either in cellular (through gNBs) or inside link

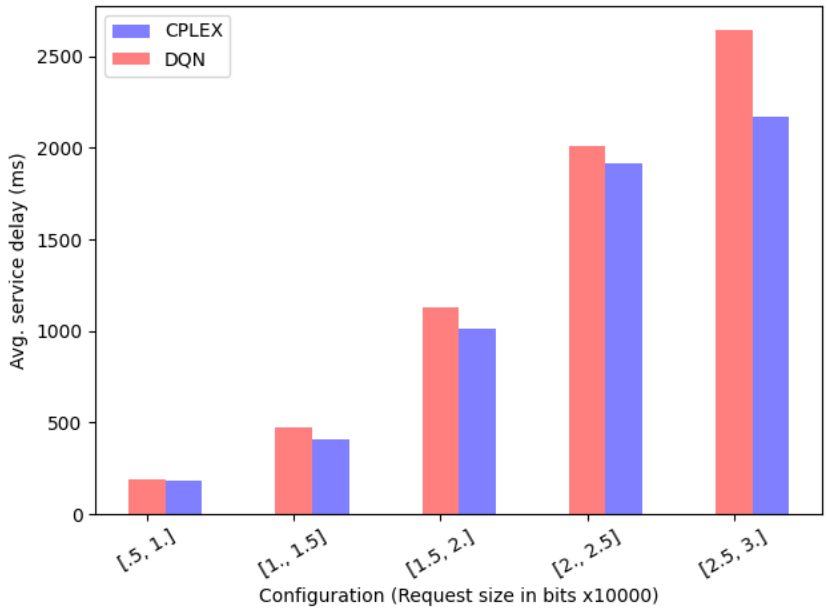

Figure 5.4 The objective function vs. the request sizes of the vehicles.

mode (through PC5 communication). In addition, in the side link mode, each vehicle can multicast to multiple vehicles within the same cluster. Finally, the proposed RL approach is executed separately for each communication mode (i.e., uplink and downlink), which means that the RL is being executed twice.

Most of the literature studied hereabove, focus on the resource provisioning at the MEC sides independently with no consideration to the services migration problem in the vehicular network, where factors such as the mobility patterns, the services migration costs, and the requirements of the services need to be considered for a more efficient service placement schemes. Further, previous works consider only heuristic or meta-heuristic methods that focus on solving the placement problem to minimize only the latency without studying the cost of migration. The works that considered the service latency, as well as the migration costs, leverage simple algorithmic solutions without considering advanced machine learning approaches such as the one proposed in this paper. In this paper, we fill these gaps and we propose a service migration scheme based on DRL techniques in a MEC-enabled vehicular network aiming to minimize the total service latency and migration cost.

5.9 Conclusion

In this paper, we proposed a DRL-based approach to solving the problem of vehicular service placement and migration in a MEC-enabled vehicular network. First, we formulated the problem as a nonlinear integer optimization problem where the objective is to mini-

mize the total vehicular service latency plus migration costs. The service latency modeled the quality of service of the vehicles which includes (i) the communication delay, and (ii) the computing delay. The migration cost, on the other hand, models the cost of migrating services across different MEC servers and is related to the energy consumption and bandwidth usage. To solve the optimization problem optimally using off-the-shelf solvers such as CPLEX, we linearize it and transform it to an integer linear optimization problem. Next, to obtain an efficient and non-complex solution, we modeled the problem as a multi-agent Markov decision process and we developed a DRL-based method by leveraging the DQL algorithm. The DQL algorithm used double DQN and replay memory strategies to increase the training accuracy and solve the Q-value overestimation problem. Finally, we demonstrated, through extensive simulations, that the proposed DQL algorithm has close-to-optimal performance compared to the CPLEX solution.

CHAPTER 6

Conclusions and Future Works

The main objective of this thesis was to create innovative and efficient schemes for resource allocation at the network edge. To achieve this goal, we proposed several new schemes that address the challenges of network edge resource provisioning. In this chapter, we provide conclusions for each proposed approach and suggest some perspectives for future work.

6.1 Conclusion

In Chapter III, we proposed a resource provisioning scheme for EC under latency and resource consumption requirements. To successfully reduce the latency, we have studied the incurred delays at the different tiers of the architecture. In particular, we studied the queuing dynamics at the edge node level through a Lyapunov optimization model. In order to achieve efficient resource provisioning, we have proposed a resource representation for edge devices. This resource representation allows the exposure of information concerning the edge devices' resources at any moment through the ETSI standard for edge computing applications. Edge node supervisors use the gathered information concerning the resource availability of the supervised edge devices to set the optimal resource provisioning scheme using the drift plus penalty of the Lyapunov optimization model. We also investigated the reallocation frequency of resources and composed an algorithm based on the workload at each edge device of the edge node to minimize the number of times a resource allocation operation is performed. Through the performed theoretical and experimental simulations, the proposed approaches have been shown to be effective, and the results showed that the proposed approach outperformed the benchmark approaches.

In Chapter IV, we investigated SFC resource provisioning within an MEC environment with the objective of lowering the request response time. We have addressed the entire resource provisioning problem, including placement, routing, and scheduling on the distributed MEC nodes, and we proposed a joint game theory and machine learning approach to the problem. We leveraged mean-field games to formulate the VNF placement and chaining problem, with the aim of reducing the overall resource consumption while decreasing both the request response time and the placement and routing operation time. We used the Ishikawa-Mann iterative learning algorithm to solve the game and thus to

properly place and chain the VNFs. Concerning the VNF scheduling problem, we leveraged the matching games framework, specifically, many-to-one matching, to match the VNFs to MEC node resources. We proposed a modified version of the classic deferred acceptance algorithm called the enhanced multi-stage deferred acceptance algorithm, which takes into account the specific constraints of the problem. Through the extensive simulations performed, it is shown that the proposed approaches perform better than the benchmarked approaches under the studied scenarios.

Finally, in Chapter V, we proposed a DRL-based scheme to address the placement and migration of vehicular services in an MEC-based vehicular network. First of all, we have formulated the problem as a nonlinear integer optimization problem that aims to minimize the total response time (i.e., communication and computation delays) as well as the migration costs in terms of the power consumption and bandwidth utilization. To solve the optimization problem, we used standard solvers, specifically, those from CPLEX, and linearized it to transform it into an integer linear optimization problem. Then, we formulated the problem as a multi-agent Markov decision process and developed a DRL-based method by leveraging the DQL algorithm to obtain an efficient and non-complex solution. The DQL algorithm utilizes dual DQN and replays memory strategies to augment the learning accuracy and solve the Q-value overestimation problem. Finally, we have demonstrated through comprehensive simulations that the proposed DQL algorithm achieves a near-optimal performance in comparison to the CPLEX solution.

6.2 Future Works

In this section, we suggest some propositions for future studies with the aim of enhancing the previous contributions:

- The network edge architecture is essentially built on distributed heterogeneous devices located outside the operators' premises, which means that the links and network structure are exposed to changes during execution. As a result, the network edge is exposed to failures. In future work, we propose to study the problem of fault tolerance at the network edge under the constraints of high reliability. Such a perspective would improve the results of the resource provisioning scheme studied in Chapter III.
- To improve the performance of the resource provisioning scheme for services proposed in Chapter IV, we propose to study in a future work its integration into the testbed proposed in Chapter III. Indeed, it will be interesting to compare the ob-

tained theoretical results with the results of the testbed in terms of the resource consumption and response time.

– Concerning the service migration problem in Chapter V, we propose integrating network slicing into our system model, as well as migrating multiple services across the vehicular networks.

Conclusions et Travaux Futurs

L'objectif principal de cette thèse était de créer des schémas innovants et efficaces pour l'allocation des ressources à la périphérie du réseau. Pour atteindre cet objectif, nous avons proposé plusieurs nouveaux schémas qui répondent aux défis de l'allocation des ressources à la périphérie du réseau. Dans ce chapitre, nous fournissons des conclusions pour chaque approche proposée et suggérons quelques perspectives pour les travaux futurs.

6.3 Conclusion

Dans le chapitre III, nous avons proposé un schéma d'approvisionnement en ressources pour l'EC sous des exigences de latence et de consommation de ressources. Pour réussir à réduire la latence, nous avons étudié les délais encourus aux différents niveaux de l'architecture. En particulier, nous avons étudié la dynamique de la mise en file d'attente au niveau du nœud de périphérie par le biais d'un modèle d'optimisation de Lyapunov. Afin de parvenir à un approvisionnement efficace en ressources, nous avons proposé une représentation des ressources pour les équipements de périphérie. Cette représentation des ressources permet d'exposer les informations concernant les ressources des dispositifs de périphérie à tout moment par le biais de la norme ETSI pour les applications informatiques de périphérie. Les superviseurs des nœuds de périphérie utilisent les informations recueillies concernant la disponibilité des ressources des dispositifs de périphérie supervisés pour définir le schéma optimal d'approvisionnement en ressources en utilisant la dérive plus la pénalité du modèle d'optimisation de Lyapunov. Nous avons également étudié la fréquence de réaffectation des ressources et composé un algorithme basé sur la charge de travail de chaque dispositif périphérique du nœud périphérique afin de minimiser le nombre de fois où une opération d'affectation des ressources est effectuée. Grâce aux simulations théoriques et expérimentales réalisées, les approches proposées se sont avérées efficaces et les résultats ont montré que l'approche proposée surpasse les approches de référence.

Dans le chapitre IV, nous avons étudié l'approvisionnement en ressources pour les fonctions des services chainées dans la périphérie afin de réduire le temps de réponse aux requêtes. Nous avons abordé le problème de l'approvisionnement en ressources pour les services, du placement à l'ordonnancement en passant par le routage sur les nœuds de périphérie distribués. Ensuite, nous avons proposé une étude conjointe de la théorie des jeux et de l'apprentissage automatique pour résoudre ce problème. Nous avons utilisé les jeux à champ moyen pour formuler le problème de placement et de chaînage des fonc-

tions réseau en vue de réduire la consommation globale des ressources tout en diminuant à la fois le temps de réponse aux requêtes des utilisateurs. Nous avons développé un algorithme d'apprentissage itératif à base de l'algorithme d'Ishikawa-Mann pour résoudre le jeu, et ainsi, placer et enchaîner correctement les fonctions réseau. En ce qui concerne le problème d'ordonnancement des fonctions réseau, nous avons utilisé les jeux de correspondance, et plus particulièrement la correspondance de plusieurs-à-un pour faire correspondre les fonctions réseau aux ressources des nœuds de périphérie. Nous avons proposé une version modifiée de l'algorithme classique d'acceptation différée qui prend en compte les contraintes spécifiques du problème. Les simulations approfondies montrent que les approches proposées sont meilleures que les approches de référence dans les scénarios étudiés.

Enfin, dans le chapitre V, nous avons proposé un schéma basé sur l'apprentissage automatique profond pour traiter le placement et la migration des services véhiculaires dans un réseau véhiculaire en périphérie. Tout d'abord, nous avons formulé le problème comme un problème d'optimisation non-linéaire en nombres entiers qui vise à minimiser le temps de réponse total (c'est-à-dire les délais de communication et de calcul) ainsi que les coûts de migration en termes de consommation d'énergie et d'utilisation de la bande passante. Pour résoudre le problème d'optimisation, nous avons utilisé des solveurs standard, notamment CPLEX. Ensuite, nous avons formulé le problème sous la forme d'un processus de décision de Markov multi-agents et développé une méthode basée sur l'apprentissage automatique profond en exploitant l'algorithme DQL pour obtenir une solution efficace et non complexe. L'algorithme DQL utilise des stratégies de mémoire double DQN et de répétition pour augmenter la précision de l'apprentissage et résoudre le problème de la surestimation de la valeur Q. Enfin, nous avons démontré par des simulations approfondies que l'algorithme DQL proposé atteint des performances quasi-optimales par rapport à la solution CPLEX.

6.4 Travaux Futurs

– L'architecture de périphérie du réseau est essentiellement construite à base des dispositifs hétérogènes distribués situés en dehors des prémisses des opérateurs, ce qui rend les liens et la structure du réseau exposés aux changements pendant l'exécution. Par conséquent, la périphérie du réseau est exposée aux pannes. Dans le cadre de travaux futurs, nous proposons d'étudier le problème de la tolérance aux pannes à la périphérie du réseau sous les contraintes de haute fiabilité. Une telle perspective

permettrait d'améliorer les résultats du schéma d'approvisionnement en ressources étudié au chapitre III.

- Pour avoir une meilleure performance du schéma d'approvisionnement en ressources pour les services, proposé au chapitre IV, nous proposons d'étudier dans un travail futur son intégration dans le banc d'essai proposé au chapitre III. Il sera en effet intéressant de comparer les résultats théoriques obtenus aux résultats du banc d'essai du point de vue de la consommation des ressources et du temps de réponse.
- Concernant le problème de la migration des services du chapitre V, nous proposons pour un travail futur l'intégration du découpage du réseau dans notre modèle de système, ainsi que la migration de services multiples dans les réseaux de véhicules.

List of publications

- A. Abouaomar, Z. Mlika, A. Filali, S. Cherkaoui, and A. Kobbane. A Deep Reinforcement Learning Approach for Service Migration in MEC-enabled Vehicular Networks. (Accepted) IEEE LCN, 1-8, 2021.
- A. Abouaomar, Z. Mlika, S. Cherkaoui, and A. Kobbane. Mean-Field Game and Reinforcement Learning MEC Resource Provisioning for SFC. (Accepted) IEEE GLOBECOM, 1-6, 2021.
- A. Abouaomar, S. Cherkaoui, Z. Mlika, and A. Kobbane. Service Function Chaining in MEC: A Mean-Field Game and Reinforcement Learning Approach. arXiv preprint arXiv:2105.04701, 2021.
- A. Abouaomar, S. Cherkaoui, Z. Mlika, and A. Kobbane. Resource provisioning in edge computing for latency sensitive applications.IEEE Internet of Things Journal, Issue 4, Vol 8, 2021.
- A. Filali, A. Abouaomar, S. Cherkaoui, A. Kobbane and M. Guizani, "Multi-Access Edge Computing: A Survey," in IEEE Access, vol. 8, pp. 197017-197046, 2020.
- A. Abouaomar, S. Cherkaoui, A. Kobbane, and O. A. Dambri. A resources representation for resource allocation in fog computing networks. In IEEE Global Communications Conference (GLOBECOM), pages 1–6.IEEE, 2019.
- A. Abouaomar, A. Kobbane, and S. Cherkaoui. Matching-game for user-fog assignment. In IEEE GLOBECOM Conference, pages 1–6, 2018.

Appendices

A. Proof of BRC second-order derivative.

Here we prove the existence of NE by proving that the second-order derivative of the BRC is negative.

$$\begin{aligned}\partial_{a_i} r_v^*(a_j, m_{(v,-j)}) &= \frac{\alpha_i}{v}\left(\frac{\beta a_{\beta-1}^j m_v^\beta - a_j^\beta\left(\frac{\beta}{v}a_j^{\beta-1}\right)}{m_v^{2\beta}}\right) - \delta_{(i,j)} \\ &= \beta a_j^{\beta-1} \times \frac{\alpha_i}{v}\left(\frac{m_v^\beta - \frac{a_j^\beta}{v}}{m_v^{2\beta}}\right) - \delta_{(i,j)}\end{aligned} \tag{1}$$

from (4.16), we can write:

$$m_v^\beta - \frac{a_j^\beta}{v} = \sum_{\substack{j'\in\mathcal{F}_i \\ j\neq j'}} a_{j'}^\beta \tag{2}$$

since $a_{j'}$ is independent of a_j, we can develop the second derivative of the payoff function $r_v^*\left(a_j, m_{(v,-j)}\right)$ as follows:

$$\partial_{a_j}^2 r_v^*\left(a_j, m_{(v,-j)}\right) =$$

$$\begin{aligned}&\frac{\alpha_i}{v}\left(\frac{\beta\left(m_v^\beta - a_j^\beta\right)\left[(\beta-1)\,a_j^{(\beta-2)}m_v^{2\beta} - a_j^{2(\beta-1)}m_v^\beta\left(\frac{2\beta}{v}\right)\right]}{m_v^{4\beta}}\right) \\ &\quad = \frac{\alpha_i}{v}\left(m_v^\beta - \frac{a_j^\beta}{v}\right) \times \frac{a_j^{(\beta-2)}}{m_{v-}^{3\beta}} \times \left((\beta-1)\,m_v^\beta - \frac{2\beta}{v}a_j^\beta\right)\end{aligned} \tag{3}$$

For given values of $\beta \in [0,1]$, the second derivative $\partial_{a_j}^2 r_v^*\left(a_j, m_{(v,-j)}\right)$ is negative.

B. Minimum and Maximum Quota constraints

Let us consider the following scenario with the set of VNFs $F_i = \{f_1, f_2, f_3\}$ and the set of ENs $\mathcal{E} = \{e_1, e_2, e_3\}$. And let us assume that the maximum quota for all the ENs, $q_i^{max} = 2$ and the minimum quota, $q_i^{min} = 1$. If we consider that all the ENs share the same preference list $\succ_f$, and if we consider that $e_1 \succ_f e_2 \succ_f e_3$ and $f_1 \succ_{f_i} f_2 \succ_{f_i} f_3$, applying the classical DAA gives that (1) $\mu(f1) = \{e_1, e_2\}$; (2) $\mu(f2) = \{e_3\}$; (3) $\mu(f3) = \emptyset$. This breaks the minimum quota rules.

LIST OF REFERENCES

[1] J. Halpern, C. Pignataro, *et al.*, "Service function chaining (sfc) architecture," in *RFC 7665*, 2015.

[2] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary, "Piecing together the nfv provisioning puzzle: Efficient placement and chaining of virtual network functions," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 98–106, IEEE, 2015.

[3] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE pervasive Computing*, vol. 8, no. 4, pp. 14–23, 2009.

[4] A. Filali, A. Abouaomar, S. Cherkaoui, A. Kobbane, and M. Guizani, "Multi-access edge computing: A survey," *IEEE Access*, vol. 8, pp. 197017–197046, 2020.

[5] I. Alam, K. Sharif, F. Li, Z. Latif, M. M. Karim, S. Biswas, B. Nour, and Y. Wang, "A survey of network virtualization techniques for internet of things using sdn and nfv," *ACM Computing Surveys (CSUR)*, vol. 53, no. 2, pp. 1–40, 2020.

[6] J. G. Herrera and J. F. Botero, "Resource allocation in nfv: A comprehensive survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 518–532, 2016.

[7] C. Pham, N. H. Tran, and C. S. Hong, "Virtual network function scheduling: A matching game approach," *IEEE Communications Letters*, vol. 22, no. 1, pp. 69–72, 2017.

[8] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[9] V. Bahl, "Emergence of micro datacenter (cloudlets/edges) for mobile computing," *Microsoft Devices & Networking Summit 2015*, vol. 5, 2015.

[10] S. Yi, C. Li, and Q. Li, "A survey of fog computing: concepts, applications and issues," in *Proceedings of the 2015 workshop on mobile big data*, pp. 37–42, 2015.

[11] I. Sittón-Candanedo, R. S. Alonso, J. M. Corchado, S. Rodríguez-González, and R. Casado-Vara, "A review of edge computing reference architectures and a new global edge proposal," *Future Generation Computer Systems*, vol. 99, pp. 278–294, 2019.

[12] ETSI, "Mec in 5g networks," white paper, ETSI, June 2018.

[13] ETSI, "ETSI GR MEC 027 Multi-access Edge Computing (MEC); Study on MEC support for alternative virtualization technologies," white paper, ETSI, Nov. 2019.

[14] T. Qiu, J. Chi, X. Zhou, Z. Ning, M. Atiquzzaman, and D. O. Wu, "Edge computing in industrial internet of things: Architecture, advances and challenges," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2462–2488, 2020.

[15] CISCO, "What is edge computing?," Mar 2021.

[16] J. Zhang, B. Chen, Y. Zhao, X. Cheng, and F. Hu, "Data security and privacy-preserving in edge computing paradigm: Survey and open issues," *IEEE Access*, vol. 6, pp. 18209–18237, 2018.

[17] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[18] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere, "Edge-centric computing: Vision and challenges," 2015.

[19] D. Liu, B. Chen, C. Yang, and A. F. Molisch, "Caching at the wireless edge: design aspects, challenges, and future directions," *IEEE Communications Magazine*, vol. 54, no. 9, pp. 22–28, 2016.

[20] A. S. Gomes, B. Sousa, D. Palma, V. Fonseca, Z. Zhao, E. Monteiro, T. Braun, P. Simoes, and L. Cordeiro, "Edge caching with mobility prediction in virtualized lte mobile networks," *Future Generation Computer Systems*, vol. 70, pp. 148–162, 2017.

[21] C. Fricker, F. Guillemin, P. Robert, and G. Thompson, "Analysis of an offloading scheme for data centers in the framework of fog computing," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)*, vol. 1, no. 4, pp. 1–18, 2016.

[22] J. Plachy, Z. Becvar, and E. C. Strinati, "Dynamic resource allocation exploiting mobility prediction in mobile edge computing," in *2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pp. 1–6, IEEE, 2016.

[23] L. F. Bittencourt, J. Diaz-Montes, R. Buyya, O. F. Rana, and M. Parashar, "Mobility-aware application scheduling in fog computing," *IEEE Cloud Computing*, vol. 4, no. 2, pp. 26–35, 2017.

[24] J. Wang, J. Pan, and F. Esposito, "Elastic urban video surveillance system using edge computing," in *Proceedings of the Workshop on Smart Internet of Things*, pp. 1–6, 2017.

[25] A. Singh, N. Auluck, O. Rana, A. Jones, and S. Nepal, "Rt-sane: Real time security aware scheduling on the network edge," in *Proceedings of the10th International Conference on Utility and Cloud Computing*, pp. 131–140, 2017.

[26] M. Aazam, M. St-Hilaire, C.-H. Lung, and I. Lambadaris, "Pre-fog: Iot trace based probabilistic resource estimation at fog," in *2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pp. 12–17, IEEE, 2016.

[27] M. Aazam and E.-N. Huh, "Fog computing micro datacenter based dynamic resource estimation and pricing model for iot," in *2015 IEEE 29th International Conference on Advanced Information Networking and Applications*, pp. 687–694, IEEE, 2015.

[28] T. Penner, A. Johnson, B. Van Slyke, M. Guirguis, and Q. Gu, "Transient clouds: Assignment and collaborative execution of tasks on mobile devices," in *2014 IEEE Global Communications Conference*, pp. 2801–2806, IEEE, 2014.

[29] H. Sun, H. Yu, G. Fan, and L. Chen, "Energy and time efficient task offloading and resource allocation on the generic iot-fog-cloud architecture," *Peer-to-Peer Networking and Applications*, vol. 13, no. 2, pp. 548–563, 2020.

[30] X. Deng, J. Li, L. Shi, Z. Wei, X. Zhou, and J. Yuan, "Wireless powered mobile edge computing: Dynamic resource allocation and throughput maximization," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2020.

[31] T. Taleb and A. Ksentini, "Follow me cloud: interworking federated clouds and distributed mobile networks," *IEEE Network*, vol. 27, no. 5, pp. 12–19, 2013.

[32] H. Liao, Z. Zhou, X. Zhao, L. Zhang, S. Mumtaz, A. Jolfaei, S. H. Ahmed, and A. K. Bashir, "Learning-based context-aware resource allocation for edge-computing-empowered industrial iot," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4260–4277, 2019.

[33] Y. Liu, M. J. Lee, and Y. Zheng, "Adaptive multi-resource allocation for cloudlet-based mobile cloud computing system," *IEEE Transactions on Mobile Computing*, vol. 15, no. 10, pp. 2398–2410, 2015.

[34] K. Wang, K. Yang, X. Wang, and C. S. Magurawalage, "Cost-effective resource allocation in c-ran with mobile cloud," in *2016 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2016.

[35] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato, "Hybrid method for minimizing service delay in edge cloud computing through vm migration and transmission power control," *IEEE Transactions on Computers*, vol. 66, no. 5, pp. 810–819, 2016.

[36] A. R. Zamani, M. Zou, J. Diaz-Montes, I. Petri, O. Rana, A. Anjum, and M. Parashar, "Deadline constrained video analysis via in-transit computational environments," *IEEE Transactions on Services Computing*, vol. 13, no. 1, pp. 59–72, 2017.

[37] W. Tärneberg, A. Mehta, E. Wadbro, J. Tordsson, J. Eker, M. Kihl, and E. Elmroth, "Dynamic application placement in the mobile cloud network," *Future Generation Computer Systems*, vol. 70, pp. 163–177, 2017.

[38] S. Yi, Z. Hao, Q. Zhang, Q. Zhang, W. Shi, and Q. Li, "Lavea: Latency-aware video analytics on edge computing platform," in *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, pp. 1–13, 2017.

[39] Q. Zhang, L. Gui, F. Hou, J. Chen, S. Zhu, and F. Tian, "Dynamic task offloading and resource allocation for mobile-edge computing in dense cloud ran," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3282–3299, 2020.

[40] C.-F. Liu, M. Bennis, M. Debbah, and H. V. Poor, "Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing," *IEEE Transactions on Communications*, vol. 67, no. 6, pp. 4132–4150, 2019.

[41] R. Cziva, C. Anagnostopoulos, and D. P. Pezaros, "Dynamic, latency-optimal vnf placement at the network edge," in *Ieee infocom 2018-ieee conference on computer communications*, pp. 693–701, IEEE, 2018.

[42] D. Bhamare, R. Jain, M. Samaka, and A. Erbad, "A survey on service function chaining," *Journal of Network and Computer Applications*, vol. 75, pp. 138–155, 2016.

[43] A. Abouaomar, S. Cherkaoui, Z. Mlika, and A. Kobbane, "Resource provisioning in edge computing for latency-sensitive applications," *IEEE Internet of Things Journal*, vol. 8, no. 14, pp. 11088–11099, 2021. Early Access.

[44] A. Abouaomar, S. Cherkaoui, A. Kobbane, and O. A. Dambri, "A resources representation for resource allocation in fog computing networks," in *2019 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, 2019.

[45] O. Alhussein and W. Zhuang, "Robust online composition, routing and nf placement for nfv-enabled services," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 1089–1101, 2020.

[46] L. Yala, P. A. Frangoudis, and A. Ksentini, "Latency and availability driven vnf placement in a mec-nfv environment," in *2018 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7, IEEE, 2018.

[47] S. Agarwal, F. Malandrino, C.-F. Chiasserini, and S. De, "Joint vnf placement and cpu allocation in 5g," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, pp. 1943–1951, 2018.

[48] L. Liu, S. Guo, G. Liu, and Y. Yang, "Joint dynamical vnf placement and sfc routing in nfv-enabled sdns," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2021.

[49] N. Toumi, O. Bernier, D.-E. Meddour, and A. Ksentini, "On using physical programming for multi-domain sfc placement with limited visibility," *IEEE Transactions on Cloud Computing*, 2020.

[50] H.-W. Tseng, T.-T. Yang, and F.-T. Hsu, "An mec-based vnf placement and scheduling scheme for ar application topology," in *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, IEEE, 2021.

[51] A. Abouaomar, S. Cherkaoui, Z. Mlika, and A. Kobbane, "Service Function Chaining in MEC: A Mean-Field Game and Reinforcement Learning Approach," 2021.

[52] J. Li, W. Shi, N. Zhang, and X. Shen, "Delay-aware vnf scheduling: A reinforcement learning approach with variable action set," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 1, pp. 304–318, 2021.

[53] J. Li, W. Shi, N. Zhang, and X. S. Shen, "Reinforcement learning based vnf scheduling with end-to-end delay guarantee," in *2019 IEEE/CIC International Conference on Communications in China (ICCC)*, pp. 572–577, 2019.

[54] J. Li, W. Shi, P. Yang, and X. Shen, "On dynamic mapping and scheduling of service function chains in sdn/nfv-enabled networks," in *2019 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2019.

[55] I. Sarrigiannis, K. Ramantas, E. Kartsakli, P.-V. Mekikis, A. Antonopoulos, and C. Verikoukis, "Online vnf lifecycle management in an mec-enabled 5g iot architecture," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4183–4194, 2020.

[56] T. Wang, J. Zu, G. Hu, and D. Peng, "Adaptive service function chain scheduling in mobile edge computing via deep reinforcement learning," *IEEE Access*, vol. 8, pp. 164922–164935, 2020.

[57] P. Kaliyammal Thiruvasagam, V. J. Kotagi, and S. R. Murthy, "A reliability-aware, delay guaranteed, and resource efficient placement of service function chains in softwarized 5g networks," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2020.

[58] G. Wang, S. Zhou, S. Zhang, Z. Niu, and X. Shen, "Sfc-based service provisioning for reconfigurable space-air-ground integrated networks," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 7, pp. 1478–1489, 2020.

[59] Q. Li, X. Wang, T. Zhao, Y. Wang, Z. Li, and L. Rui, "An improved genetic algorithm for the scheduling of virtual network functions," in *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pp. 1–4, 2019.

[60] Y.-R. Haung, "A qoe-aware strategy for supporting service continuity in an mcc environment," *Wireless Personal Communications*, vol. 116, no. 1, pp. 629–654, 2021.

[61] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2333–2345, 2018.

[62] Y.-T. Chen and W. Liao, "Mobility-aware service function chaining in 5g wireless networks with mobile edge computing," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2019.

[63] P. Roy, A. Tahsin, S. Sarker, T. Adhikary, M. A. Razzaque, and M. M. Hassan, "User mobility and quality-of-experience aware placement of virtual network functions in 5g," *Computer Communications*, vol. 150, pp. 367–377, 2020.

[64] R. A. Addad, D. L. C. Dutra, M. Bagaa, T. Taleb, and H. Flinck, "Towards a fast service migration in 5g," in *2018 IEEE Conference on Standards for Communications and Networking (CSCN)*, pp. 1–6, IEEE, 2018.

[65] R. A. Addad, T. Taleb, M. Bagaa, D. L. C. Dutra, and H. Flinck, "Towards modeling cross-domain network slices for 5g," in *2018 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7, IEEE, 2018.

[66] R. A. Addad, T. Taleb, H. Flinck, M. Bagaa, and D. Dutra, "Network slice mobility in next generation mobile systems: Challenges and potential solutions," *IEEE Network*, vol. 34, no. 1, pp. 84–93, 2020.

[67] M. Abu Sharkh, A. Shami, and M. Kalil, *The Era of the Personal Cloud: What Does It Mean for Cloud Providers?: From Hype to Reality*, pp. 1–15. 01 2019.

[68] K. Toczé and S. Nadjm-Tehrani, "A taxonomy for management and optimization of multiple resources in edge computing," *Wireless Communications and Mobile Computing*, vol. 2018, 2018.

[69] ETSI, "Multi-access edge computing (MEC) MEC management ETSI GS MEC 010-2, part 2: Application lifecycle, rules and requirements management," white paper, ETSI, Nov. 2019.

[70] ETSI, "Multi-access edge computing (MEC) ETSI GS MEC 011: Edge platform application enablement," white paper, ETSI, Nov. 2019.

[71] ETSI, "Multi-access edge computing (MEC) ETSI GS MEC 021: Application mobility service API," white paper, ETSI, Jan. 2020.

[72] Q. Zhang, L. Gui, F. Hou, J. Chen, S. Zhu, and F. Tian, "Dynamic task offloading and resource allocation for mobile-edge computing in dense cloud ran," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3282–3299, 2020.

[73] M. Merluzzi, P. D. Lorenzo, and S. Barbarossa, "Dynamic joint resource allocation and user assignment in multi-access edge computing," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4759–4763, 2019.

[74] S. Guo, X. Hu, G. Dong, W. Li, and X. Qiu, "Mobile edge computing resource allocation: A joint stackelberg game and matching strategy," *International Journal of Distributed Sensor Networks*, vol. 15, no. 7, p. 1550147719861556, 2019.

[75] Y. Du, J. Li, L. Shi, T. Liu, F. Shu, and Z. Han, "Two-tier matching game in small cell networks for mobile edge computing," *IEEE Transactions on Services Computing*, pp. 1–1, 2019.

[76] Z. Zhou, P. Liu, J. Feng, Y. Zhang, S. Mumtaz, and J. Rodriguez, "Computation resource allocation and task assignment optimization in vehicular fog computing: A contract-matching approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3113–3125, 2019.

[77] B. Gu and Z. Zhou, "Task offloading in vehicular mobile edge computing: A matching-theoretic framework," *IEEE Vehicular Technology Magazine*, vol. 14, no. 3, pp. 100–106, 2019.

[78] H. R. Arkian, A. Diyanat, and A. Pourkhalili, "Mist: Fog-based data analytics scheme with cost-efficient resource provisioning for iot crowdsensing applications," *Journal of Network and Computer Applications*, vol. 82, pp. 152–165, 2017.

[79] O. Skarlat, M. Nardelli, S. Schulte, and S. Dustdar, "Towards qos-aware fog service placement," in *2017 IEEE 1st international conference on Fog and Edge Computing (ICFEC)*, pp. 89–96, IEEE, 2017.

[80] F. Z. Yousaf and T. Taleb, "Fine-grained resource-aware virtual network function management for 5g carrier cloud," *IEEE Network*, vol. 30, no. 2, pp. 110–115, 2016.

[81] B. P. Rimal, M. Maier, and M. Satyanarayanan, "Experimental testbed for edge computing in fiber-wireless broadband access networks," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 160–167, 2018.

[82] A. Abouaomar, A. Kobbane, and S. Cherkaoui, "Matching-game for user-fog assignment," in *2018 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, 2018.

[83] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," *Linux journal*, vol. 2014, no. 239, p. 2, 2014.

[84] D. R.-J. G.-J. Rydning, "The digitization of the world from edge to core," *Framingham: International Data Corporation*, 2018.

[85] J. Xu, K. Ota, and M. Dong, "Saving energy on the edge: In-memory caching for multi-tier heterogeneous networks," *IEEE Communications Magazine*, vol. 56, no. 5, pp. 102–107, 2018.

[86] J. Zhang, X. Hu, Z. Ning, E. C. . Ngai, L. Zhou, J. Wei, J. Cheng, B. Hu, and V. C. M. Leung, "Joint resource allocation for latency-sensitive services over mobile edge computing networks with caching," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4283–4294, 2019.

[87] M. J. Neely, "Stability and probability 1 convergence for queueing networks via lyapunov optimization," *Journal of Applied Mathematics*, vol. 2012, 2012.

[88] W. Lin, J. Z. Wang, C. Liang, and D. Qi, "A threshold-based dynamic resource allocation scheme for cloud computing," *Procedia Engineering*, vol. 23, pp. 695–703, 2011.

[89] A. Kaehler and G. Bradski, *Learning OpenCV 3: computer vision in C++ with the OpenCV library.* " O'Reilly Media, Inc.", 2016.

[90] N. Muslim and S. Islam, "Face recognition in the edge cloud," in *Proceedings of the International Conference on Imaging, Signal Processing and Communication*, pp. 5–9, 2017.

[91] C.-F. Liu, S. Samarakoon, M. Bennis, and H. V. Poor, "Fronthaul-aware software-defined wireless networks: Resource allocation and user scheduling," *IEEE Transactions on Wireless Communications*, vol. 17, no. 1, pp. 533–547, 2017.

[92] J. Navarro-Ortiz, P. Romero-Diaz, S. Sendra, P. Ameigeiras, J. J. Ramos-Munoz, and J. M. Lopez-Soler, "A survey on 5g usage scenarios and traffic models," *IEEE Communications Surveys Tutorials*, vol. 22, no. 2, pp. 905–929, 2020.

[93] A. Ksentini and P. A. Frangoudis, "Toward slicing-enabled multi-access edge computing in 5g," *IEEE Network*, vol. 34, no. 2, pp. 99–105, 2020.

[94] B. Nour, S. Mastorakis, and A. Mtibaa, "Compute-Less Networking: Perspectives, Challenges, and Opportunities," *IEEE Network*, vol. 34, no. 6, pp. 259–265, 2020.

[95] A. Filali, Z. Mlika, S. Cherkaoui, and A. Kobbane, "Preemptive sdn load balancing with machine learning for delay sensitive applications," *IEEE Transactions on Vehicular Technology*, pp. 1–1, 2020.

[96] J. Gil Herrera and J. F. Botero, "Resource allocation in nfv: A comprehensive survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 518–532, 2016.

[97] A. Abouaomar, A. Filali, and A. Kobbane, "Caching, device-to-device and fog computing in 5 th cellular networks generation: Survey," in *2017 International Conference on Wireless Networks and Mobile Communications (WINCOM)*, pp. 1–6, IEEE, 2017.

[98] A. Tak and S. Cherkaoui, "Federated edge learning: Design issues and challenges," *IEEE Network*, 2020.

[99] O. Alhussein and W. Zhuang, "Robust online composition, routing and nf placement for nfv-enabled services," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 1089–1101, 2020.

[100] L. Yala, P. A. Frangoudis, and A. Ksentini, "Latency and Availability Driven VNF Placement in a MEC-NFV Environment," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, pp. 1–7, 2018.

[101] S. Lhazmir, M.-A. Koulali, A. Kobbane, and H. Elbiaze, "Performance analysis of uav-assisted ferrying for the internet of things," in *2019 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1–6, IEEE, 2019.

[102] M. Azizian, S. Cherkaoui, and A. S. Hafid, "Vehicle software updates distribution with sdn and cloud computing," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 74–79, 2017.

[103] R. W. Wolff, "Poisson arrivals see time averages," *Operations Research*, vol. 30, no. 2, pp. 223–231, 1982.

[104] K. Gopalan, L. Huang, G. Peng, T.-C. Chiueh, and Y.-J. Lin, "Statistical admission control using delay distribution measurements," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 2, no. 4, pp. 258–281, 2006.

[105] G. A. Marin, X. Mang, E. Gelenbe, and R. O. Onvural, "Statistical call admission control," Apr. 2001. US Patent 6,222,824.

[106] A. Gupta, G. Guruganesh, B. Peng, and D. Wajc, "Stochastic online metric matching," *arXiv preprint arXiv:1904.09284*, 2019.

[107] S. Dehghani, S. Ehsani, M. Hajiaghayi, V. Liaghat, and S. Seddighin, "Stochastic k-server: How should uber work?," *arXiv preprint arXiv:1705.05755*, 2017.

[108] N. Thakral, "Matching with stochastic arrival," in *AEA Papers and Proceedings*, vol. 109, pp. 209–12, 2019.

[109] P. E. Caines, M. Huang, and R. P. Malhamé, "Mean field games.," 2015.

[110] A. F. Hanif, H. Tembine, M. Assaad, and D. Zeghlache, "Mean-field games for resource sharing in cloud-based networks," *IEEE/ACM Transactions on Networking*, vol. 24, no. 1, pp. 624–637, 2015.

[111] S. Ishikawa, "Fixed points by a new iteration method," *Proceedings of the American Mathematical Society*, vol. 44, no. 1, pp. 147–150, 1974.

[112] D. Gale and L. S. Shapley, "College admissions and the stability of marriage," *The American Mathematical Monthly*, vol. 120, no. 5, pp. 386–391, 2013.

[113] Z. Han, D. Niyato, W. Saad, T. Başar, and A. Hjørungnes, *Game theory in wireless and communication networks: theory, models, and applications*. Cambridge university press, 2012.

[114] D. Fragiadakis, A. Iwasaki, P. Troyan, S. Ueda, and M. Yokoo, "Strategyproof matching with minimum quotas," *ACM Transactions on Economics and Computation (TEAC)*, vol. 4, no. 1, pp. 1–40, 2016.

[115] S. Wang, V. Charissis, J. Campbell, W. Chan, D. Moore, and D. Harrison, "An Investigation Into the Use of Virtual Reality Technology for Passenger Infotainment in a Vehicular Environment," in *Proc. IEEE Int. Conf. Adv. Mater. Sci. Eng. (ICAMSE)*, pp. 404–407, 2016.

[116] H. Khan, S. Samarakoon, and M. Bennis, "Enhancing Video Streaming in Vehicular Networks via Resource Slicing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 3513–3522, 2020.

[117] P. A. Frangoudis and A. Ksentini, "Service migration versus service replication in multi-access edge computing," in *2018 14th International Wireless Communications Mobile Computing Conference (IWCMC)*, pp. 124–129, 2018.

[118] Z. Mlika and S. Cherkaoui, "Network Slicing with MEC and Deep Reinforcement Learning for the Internet of Vehicles," *IEEE Network*, pp. 1–7, 2021. Early Access.

[119] A. Aissioui, A. Ksentini, A. M. Gueroui, and T. Taleb, "On enabling 5g automotive systems using follow me edge-cloud concept," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 6, pp. 5302–5316, 2018.

[120] T. Ouyang, R. Li, X. Chen, Z. Zhou, and X. Tang, "Adaptive User-Managed Service Placement for Mobile Edge Computing: An Online Learning Approach," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, pp. 1468–1476, 2019.

[121] P. Mach and Z. Becvar, "Mobile Edge Computing: A Survey on Architecture and Computation Offloading," *IEEE Commun. Surv. Tutor.*, vol. 19, no. 3, pp. 1628–1656, 2017.

[122] R. Urgaonkar, S. Wang, T. He, M. Zafer, K. Chan, and K. K. Leung, "Dynamic Service Migration and Workload Scheduling in Edge-Clouds," *Performance Evaluation*, vol. 91, pp. 205–228, 2015.

[123] Z. Mlika, M. Goonewardena, W. Ajib, and H. Elbiaze, "User–Base-Station Association in HetSNets: Complexity and Efficient Algorithms," *IEEE Trans. Veh. Technol.*, vol. 66, no. 2, pp. 1484–1495, 2017.

[124] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-Level Control Through Deep Reinforcement Learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[125] H. Van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-Learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 30, 2016.

[126] H. Khan, P. Luoto, S. Samarakoon, M. Bennis, and M. Latva-Aho, "Network Slicing for Vehicular Communication," *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 1, p. e3652, 2021.

[127] H. D. R. Albonda and J. Pérez-Romero, "An Efficient RAN Slicing Strategy for a Heterogeneous Network With eMBB and V2X Services," *IEEE Access*, vol. 7, pp. 44771–44782, 2019.