Artificial intelligence for resource allocation in multi-tenant edge computing

Ayoub Ben Ameur

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2023IPPAS019

Thèse de doctorat

Artificial Intelligence for Resource Allocation in Multi-Tenant Edge Computing

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Télécom SudParis

École doctorale n°626 Dénomination (EDIPP)
Spécialité de doctorat: Informatique, Données et Intelligence Artificielle

Thèse présentée et soutenue à Palaiseau, le 30/11/2023, par

Ayoub Ben Ameur

Composition du Jury :

Antonio Capone Professor, Politecnico di Milano	Rapporteur (Président)
Anastasios Giovanidis Researcher, Ericsson, CR CNRS mise-en-disponibilité	Rapporteur
Ana Bušić Research Scientist, Inria (Paris research unit) and ENS	Examinateur
Daniel Kofman Professor, Télécom Paris	Examinateur
Andrea Araldo Associate Professor, Télécom SudParis (SAMOVAR)	Co-directeur de thèse
Tijani Chahed Professor, Télécom SudParis (SAMOVAR)	Directeur de thèse

Declaration

I hereby declare that this thesis represents my own work which has been done after registration for the degree of PhD at Institut Polytechnique de Paris, and has not been previously included in a thesis or dissertation submitted to this or any other institution for a degree, diploma or other qualifications.

I have read the University's current research ethics guidelines, and accept responsibility for the conduct of the procedures in accordance with the University's Committee on the Use of Human and Animal Subjects in Teaching and Research (HASC). I have attempted to identify all the risks related to this research that may arise in conducting this research, obtained the relevant ethical and/or safety approval (where applicable), and acknowledged my obligations and the rights of the participants.

Ayoub Ben Ameur

Acknowledgements

Completing this thesis has been a journey filled with challenges, growth, and moments of inspiration, and I am profoundly grateful to all those who have played a significant role in this academic endeavor. Their unwavering support, guidance, and encouragement have been invaluable in shaping this work.

First and foremost, I am deeply indebted to my thesis advisors, Professor Tijani Chahed and Professor Andrea Araldo whose expertise, dedication, and patience guided me throughout this research journey. Your insightful feedback and unwavering support were instrumental in shaping the direction of this thesis. I have learned immeasurable lessons from you, both as scholars and as mentors.

I would also like to express my sincere appreciation to the members of my thesis committee, Antonio Capone, Anastasios Giovanidis, Ana Bušić and Daniel Kofman for their valuable insights, critical feedback, and constructive criticism. Your collective expertise enriched this study and provided valuable perspectives that have contributed to its overall quality.

I owe a profound debt of gratitude to my family for their endless love and unwavering belief in my abilities. Mom and Dad, your sacrifices and encouragement have been my driving force, and I dedicate this achievement to you. To my wife Fatma, thank you for your understanding and support, even when I was buried in research. To my sister Chaima and her beloved family and my brother Soltane, thank you for your daily support despite the distance that separated us.

My heartfelt thanks go to my friends and peers who have been my pillars of strength throughout this journey. Your camaraderie, late-night study sessions, and words of encouragement made the challenges bearable and the successes sweeter.

I would like to acknowledge the financial support provided by Carnot Institute and Telecom SudParis, which made this research possible. Your investment in my education is deeply appreciated. I would like to thank the support of Erasmus+ and the French Embassy in Sweden | the French Institute in Sweden, which made my doctoral mobility to the Royal Institute of Technology (KTH), Stockholm, Sweden, possible.

To Professor György Dán who generously shared his time and his insights for this study and hosted me for 6 months at KTH, I extend my heartfelt appreciation. Your contributions were integral to the completion of this research, and your willingness to participate was

truly remarkable. I deeply thank my fellow at KTH Feridun Tütüncüoglu as well, with whom I had a great collaboration and enjoyed working in Sweden.

This research would not have been possible without the unwavering support of my colleagues at Institut Polytechnique de Paris. Your intellectual contributions and camaraderie have enriched my academic journey. In particular, I deeply thank Rosario Patane, Alessandro Spallina, Wei Huang, Massinissa Ait Aba, Maxime Elkael, Mohammad Abdullah, Marco Di Prima and Maya Kassis for the great time we spent together in Palaiseau offices.

Last but not least, I want to acknowledge the lessons learned from both successes and failures along the way. Each challenge and setback has been a valuable source of growth and learning, contributing to my personal and academic development.

In conclusion, this thesis is the culmination of the collective efforts, support, and encouragement of many individuals and institutions. I am profoundly grateful to each and every one of you for your contributions to this academic achievement. While it is not possible to name everyone who has been a part of this journey, your influence is deeply appreciated and will be remembered with gratitude.

Thank you all for being a part of this significant chapter in my academic life.

With heartfelt appreciation,

Ayoub Ben Ameur

DOCTORAL SCHOOL

Titre : L'Intelligence artificielle pour l'allocation des ressources dans le Multi-tenant Edge Computing

Mots clés : Edge computing, Optimisation orientée données, Réseaux nouvelle génération, Intelligence artificielle, Allocation des ressources, Multi-tenancy

Résumé : Dans cette thèse, nous considérons le Edge Computing (EC) comme un environnement multi-tenant où les Opérateurs Réseau (NOs) possèdent des ressources en périphérie déployées dans les stations de base, les bureaux centraux et/ou les boîtiers intelligents, les virtualisent, et permettent aux Fournisseurs de Services tiers (SPs) - ou tenants - de distribuer une partie de leurs applications en périphérie afin de répondre aux demandes des utilisateurs. Les SPs aux besoins hétérogènes coexistent en périphérie, allant des Communications Ultra-Fiables à Latence Ultra-Basse (URLLC) pour le contrôle des véhicules ou des robots, à la Communication de Type Machine Massive (mMTC) pour l'Internet des Objets (IoT) nécessitant un grand nombre de dispositifs connectés, en passant par les services multimédias tels que la diffusion vidéo et la Réalité Augmentée/Virtuelle (AR/VR), dont la qualité d'expérience dépend fortement des ressources disponibles. Les SPs orchestrent indépendamment leur ensemble de microservices, exécutés dans des conteneurs, qui peuvent être facilement répliqués, migrés ou arrêtés. Chaque SP peut s'adapter aux ressources allouées par le NO, en décidant s'il doit exécuter des microservices sur les appareils, les nœuds en périphérie ou dans le cloud. L'objectif de cette thèse est de promouvoir l'émergence de déploiements réels du "véritable" EC dans de vrais réseaux, en montrant l'utilité que les NOs peuvent tirer de l'EC. Nous croyons que cela peut contribuer à encourager l'engagement concret et les investissements des NOs dans l'EC. À cette fin, nous proposons de concevoir de nouvelles stratégies basées sur les données qui allouent efficacement les ressources entre les SPs hétérogènes, en périphérie, appartenant au NO, afin d'optimiser ses objectifs pertinents, tels que la réduction des coûts, la maximisation des revenus et l'amélioration de la Qualité de Service (QoS) perçue par les utilisateurs finaux, en termes de latence, de fiabilité et de débit, tout en répondant aux exigences des SPs. Cette thèse présente une perspective sur la manière dont les NOs, les seuls propriétaires de ressources en périphérie, peuvent extraire de la valeur grâce à la mise en œuvre de l'EC dans un environnement multi-tenant. En promouvant cette vision de l'EC et en la soutenant par des résultats quantitatifs et une analyse approfondie, cette thèse fournit principalement aux NOs des conclusions susceptibles d'influencer les stratégies de décision concernant le déploiement futur de l'EC. Cela pourrait favoriser l'émergence de nouvelles applications à faible latence et à forte intensité de données, telles que la réalité augmentée haute résolution, qui ne sont pas envisageables dans le cadre actuel du Cloud Computing (CC). Une autre contribution de la thèse est qu'elle propose des solutions basées sur des méthodes novatrices exploitant la puissance de l'optimisation basée sur les données. En effet, nous adaptons des techniques de pointe issues de l'Apprentissage par Renforcement (RL) et de la prise de décision séquentielle au problème pratique de l'allocation des ressources en EC. Ce faisant, nous parvenons à réduire le temps d'apprentissage des stratégies adoptées à des échelles compatibles avec la dynamique de l'EC, grâce à la conception soignée de modèles d'estimation intégrés au processus d'apprentissage. Nos stratégies sont conçues de manière à ne pas violer les garanties de confidentialité essentielles pour que les SPs acceptent d'exécuter leurs calculs en périphérie, grâce à l'environnement multi-tenant.

Title : Artificial Intelligence for Resource Allocation in Multi-Tenant Edge Computing

Keywords : Edge computing, Data-driven optimization, Next generation networks, Artificial intelligence, Resource allocation, Multi-tenancy

We consider in this thesis Edge Computing (EC) as a multi-tenant environment where Network Operators (NOs) own edge resources deployed in base stations, central offices and/or smart boxes, virtualize them and let third party Service Providers (SPs) - or tenants - distribute part of their applications in the edge in order to serve the requests sent by the users. SPs with heterogeneous requirements coexist in the edge, ranging from Ultra-Reliable Low Latency Communications (URLLC) for controlling cars or robots, to massive Machine Type Communication (mMTC) for Internet of Things (IoT) requiring a massive number of connected devices, to media services, such as video streaming and Augmented/Virtual Reality (AR/VR), whose quality of experience is strongly dependant on the available resources. SPs independently orchestrate their set of microservices, running on containers, which can be easily replicated, migrated or stopped. Each SP can adapt to the resources allocated by the NO, deciding whether to run microservices in the devices, in the edge nodes or in the cloud. We aim in this thesis to advance the emergence of real deployments of the “true” EC in real networks, by showing the utility that NOs can collect thanks to EC. We believe that this can contribute to encourage concrete engagement and investments engagement of NOs in EC. For this, we point to design novel data-driven strategies that efficiently allocate resources between heterogeneous SPs, at the edge owned by the NO, in order to optimize its relevant objectives, e.g., cost reduction, revenue maximization and better Quality of Service (QoS) perceived by end users, in terms of latency, reliability and throughput, while satisfying the SPs requirements. This thesis presents a perspective on how NOs, the sole owners of resources at the far edge (e.g., at base stations), can extract value through the implementation of EC within a multi-tenant environment. By promoting this vision of EC and by supporting it via quantitative results and analysis, this thesis provides, mainly to NOs, findings that can influence decision strategies about the future deployment of EC. This might foster the emergence of novel low-latency and data-intensive applications, such as high resolution augmented reality, which are not feasible in the current Cloud Computing (CC) setting. Another contribution of the thesis it that it provides solutions based on novel methods that harness the power of data-driven optimization.We indeed adapt cutting-edge techniques from Reinforcement Learning (RL) and sequential decision making to the practical problem of resource allocation in EC. In doing so, we succeed in reducing the learning time of the adopted strategies up to scales that are compatible with the EC dynamics, via careful design of estimation models embedded in the learning process. Our strategies are conceived in order not to violate the confidentiality guarantees that are essential for SPs to accept running their computation at the EC, thanks to the multi-tenant setting.

Institut Polytechnique de Paris
91120 Palaiseau, France

Abstract

We consider in this thesis Edge Computing (EC) as a multi-tenant environment where Network Operators (NOs) own edge resources deployed in base stations, central offices and/or smart boxes, virtualize them and let third party Service Providers (SPs) - or tenants - distribute part of their applications in the edge in order to serve the requests sent by the users. SPs with heterogeneous requirements coexist in the edge, ranging from Ultra-Reliable Low Latency Communications (URLLC) for controlling cars or robots, to massive Machine Type Communication (mMTC) for Internet of Things (IoT) requiring a massive number of connected devices, to media services, such as video streaming and Augmented/Virtual Reality (AR/VR), whose quality of experience is strongly dependant on the available resources. SPs independently orchestrate their set of microservices, running on containers, which can be easily replicated, migrated or stopped. Each SP can adapt to the resources allocated by the NO, deciding whether to run microservices in the devices, in the edge nodes or in the cloud. We aim in this thesis to advance the emergence of real deployments of the "true" EC in real networks, by showing the utility that NOs can collect thanks to EC. We believe that this can contribute to encourage concrete engagement and investments engagement of NOs in EC. For this, we point to design novel data-driven strategies that efficiently allocate resources between heterogeneous SPs, at the edge owned by the NO, in order to optimize its relevant objectives, e.g., cost reduction, revenue maximization and better Quality of Service (QoS) perceived by end users, in terms of latency, reliability and throughput, while satisfying the SPs requirements.

This thesis presents a perspective on how NOs, the sole owners of resources at the far edge (e.g., at base stations), can extract value through the implementation of EC within a multi-tenant environment. By promoting this vision of EC and by supporting it via quantitative results and analysis, this thesis provides, mainly to NOs, findings that can influence decision strategies about the future deployment of EC. This might foster the emergence of novel low-latency and data-intensive applications, such as high resolution augmented reality, which are not feasible in the current Cloud Computing (CC) setting.

Another contribution of the thesis it that it provides solutions based on novel methods that harness the power of data-driven optimization.We indeed adapt cutting-edge techniques from Reinforcement Learning (RL) and sequential decision making to the practical problem of resource allocation in EC. In doing so, we succeed in reducing the learning

time of the adopted strategies up to scales that are compatible with the EC dynamics, via careful design of estimation models embedded in the learning process. Our strategies are conceived in order not to violate the confidentiality guarantees that are essential for SPs to accept running their computation at the EC, thanks to the multi-tenant setting.

Résumé

Dans cette thèse, nous considérons le Edge Computing (EC) comme un environnement multi-tenant où les Opérateurs Réseau (NOs) possèdent des ressources en périphérie déployées dans les stations de base, les bureaux centraux et/ou les boîtiers intelligents, les virtualisent, et permettent aux Fournisseurs de Services tiers (SPs) - ou tenants - de distribuer une partie de leurs applications en périphérie afin de répondre aux demandes des utilisateurs. Les SPs aux besoins hétérogènes coexistent en périphérie, allant des Communications Ultra-Fiables à Latence Ultra-Basse (URLLC) pour le contrôle des véhicules ou des robots, à la Communication de Type Machine Massive (mMTC) pour l'Internet des Objets (IoT) nécessitant un grand nombre de dispositifs connectés, en passant par les services multimédias tels que la diffusion vidéo et la Réalité Augmentée/Virtuelle (AR/VR), dont la qualité d'expérience dépend fortement des ressources disponibles. Les SPs orchestrent indépendamment leur ensemble de microservices, exécutés dans des conteneurs, qui peuvent être facilement répliqués, migrés ou arrêtés. Chaque SP peut s'adapter aux ressources allouées par le NO, en décidant s'il doit exécuter des microservices sur les appareils, les nœuds en périphérie ou dans le cloud.

L'objectif de cette thèse est de promouvoir l'émergence de déploiements réels du "véritable" EC dans de vrais réseaux, en montrant l'utilité que les NOs peuvent tirer de l'EC. Nous croyons que cela peut contribuer à encourager l'engagement concret et les investissements des NOs dans l'EC. À cette fin, nous proposons de concevoir de nouvelles stratégies basées sur les données qui allouent efficacement les ressources entre les SPs hétérogènes, en périphérie, appartenant au NO, afin d'optimiser ses objectifs pertinents, tels que la réduction des coûts, la maximisation des revenus et l'amélioration de la Qualité de Service (QoS) perçue par les utilisateurs finaux, en termes de latence, de fiabilité et de débit, tout en répondant aux exigences des SPs.

Cette thèse présente une perspective sur la manière dont les NOs, les seuls propriétaires de ressources en périphérie, peuvent extraire de la valeur grâce à la mise en œuvre de l'EC dans un environnement multi-tenant. En promouvant cette vision de l'EC et en la soutenant par des résultats quantitatifs et une analyse approfondie, cette thèse fournit principalement aux NOs des conclusions susceptibles d'influencer les stratégies de décision concernant le déploiement futur de l'EC. Cela pourrait favoriser l'émergence de nouvelles applications à faible latence et à forte intensité de données, telles que la réalité augmentée

haute résolution, qui ne sont pas envisageables dans le cadre actuel du Cloud Computing (CC).

Une autre contribution de la thèse est qu'elle propose des solutions basées sur des méthodes novatrices exploitant la puissance de l'optimisation basée sur les données. En effet, nous adaptons des techniques de pointe issues de l'Apprentissage par Renforcement (RL) et de la prise de décision séquentielle au problème pratique de l'allocation des ressources en EC. Ce faisant, nous parvenons à réduire le temps d'apprentissage des stratégies adoptées à des échelles compatibles avec la dynamique de l'EC, grâce à la conception soignée de modèles d'estimation intégrés au processus d'apprentissage. Nos stratégies sont conçues de manière à ne pas violer les garanties de confidentialité essentielles pour que les SPs acceptent d'exécuter leurs calculs en périphérie, grâce à l'environnement multi-tenant.

Contents

List of Tables

List of Figures

List of Algorithms

List of Acronyms

AI Artificial Intelligence

AR Augmented Reality

AWS Amazon Web Services

BNN Bayesian Neural Network

BO Bayesian Optimization

c.d.f cumulative distribution function

CC Cloud Computing

CDN Content Delivery Network

CMAB-SC Combinatorial Multi-Armed Bandit with Switching Cost

CPI Cycles per Instruction

DMDP Deterministic Markov Decision Process

DP Dynamic Programming

DQN Deep Q Network

EC Edge Computing

FaaS Function as a Service

GP Gaussian Process

HiP-MDP Hidden Parameter Markov Decision Process

ILP Integer Linear Program

IoMT Internet of Medical Things

IoT Internet of Things

ISP Internet Service Provider

LO Lyapunov Optimization

MAB Multi-Armed Bandit

MAB-SC Multi-Armed Bandit with Switching Cost

MAR Mobile Augmented Reality

MB-RL model-based Reinforcement Learning

MC Markov Chain

MDP Markov Decision Process

MEC Multi-access Edge Computing

MF-RL model-free Reinforcement Learning

mMTC massive Machine Type Communication

NO Network Operator

Open-RAN Open Radio Access Network

p.d.f probability density function

POMDP Partially Observable MDP

QoS Quality of Service

RAN Radio Access Network

RL Reinforcement Learning

SC Stochastic Control

SGD Stochastic Gradient Descent

SLA Service Level Agreement

SOCO Smoothed Online Convex Optimization

SP Service Provider

SPSA Simultaneous Perturbation Stochastic Approximation

SPSO Simultaneous Perturbation Stochastic Optimization

TL Transfer Learning

URLLC Ultra Reliable Low Latency Communication

VR Virtual Reality

WD Wireless Device

Chapter 1

General Introduction

1.1 Edge Computing

5G is pushing toward "Intelligent Networks", which are not only able to transfer flows of data but also to serve computational needs. In previous decades, cloud services reshaped the landscape of computing power and data storage, moving them from users and organizations locations to data centers. However, there is now a noticeable shift that might appear as a reversal, in the sense of bringing resources back to the local scale, driven by the surging popularity of Edge Computing (EC) in the scientific community.

Over the past several years, organizations have been flocking to the cloud in pursuit of its numerous benefits, propelling the rapid expansion of Cloud Computing (CC). The industry value surged from 90 billion dollar in 2015 to a staggering 312 billion dollars in 2020 [2]. Yet, in parallel, EC has been steadily gaining ground in the research community [3].

To those unfamiliar with the concept, this may appear as a regressive move. After all the excitement surrounding the cloud, why would organizations turn to semi-local edge servers? However, the rationale behind this choice is straightforward and readily apparent: edge servers are not meant to replace the cloud. Instead, they serve distinct purposes requiring smaller, decentralised servers with smaller response times, a characteristic not typically associated with the cloud.

As CC gained widespread adoption [4], it allowed companies to focus on their core business functions while entrusting computing tasks to specialized firms skilled in data management, storage, and analysis, such as Google, Amazon and Microsoft. Nevertheless, over time, the limitations of CC became increasingly evident. In the research literature authors started to pinpoint that latency from CC is too high for new services, such as high-resolution augmented reality [5]. Another concern related to the very high traffic that CC generates [6]. Moreover, privacy concerns in cloud computing are a pressing issue as more individuals and organizations entrust their sensitive data to remote servers [7, 8].

This growing demand for a solution gave rise to what we now recognize as EC. The term "edge" in its name reflects the strategic positioning of servers at the edge of the network, e.g., base stations, access points, central offices. This is in stark contrast to conventional computing paradigms, conducted on centralized servers and data centers. This is also in contrast with CC, which relies on distant servers located sometimes thousands of miles away from end users. Edge servers differ from their conventional counterparts found in data centers in that they are smaller in scale and, instead of being concentrated in central facilities, they are strategically dispersed across various locations. This distribution brings them into closer proximity to the devices that either generate data or require rapid access to it.

Overall, EC has high potential utility in various domains, including:

- Autonomous Driving: Autonomous vehicle technology, while still in its nascent stages, relies heavily on onboard computers that make decisions within a narrow scope. To achieve a level of autonomy where human intervention is not needed, interconnected networks of autonomous vehicles working in unison to prevent collisions and accidents are necessary. Until now, the only concrete example of autonomous cars having "Full Self-Driving Capability" is Tesla [9] (yet, not all Tesla cars come with this feature). Tesla self-driving cars rely on in-car computers, which is impractical due to their high cost for the manufacturer and their maintenance-upgrade cost for drivers, i.e., drivers are asked to replace the hardware whenever an upgrade available [10]. Edge servers, on the other hand, could be a good candidate for reducing such costs in autonomous driving [11].

- Next Generation Smart Grid: To eradicate the issues from Internet of Things (IoT) in conventional smart grid, EC is a technology of great potential for an next generation smart grid. The IoT devices for smart grid collect massive datasets [12] that are difficult to process because the cloud servers are situated in a distant geographic area. For instance, 1 million smart meters installed in the smart grid would result in 35.04 billion records, equivalent to 2920 Tb [13, Table 1]. The networking system is stressed when raw data collected from IoT devices are transmitted to the cloud because of the increases in latency and reaction time. The data collected from an SG may contain private data, and as the data are sent to a third-party cloud server it may pose the risk of privacy breach. The EC solution shows huge potential to remove these problems which are presented by current smart grid systems. Another perk of EC is that it can reduce the network load to a great extent by shrinking the volume of transmitted data.

- Healthcare Systems: Healthcare providers are rapidly digitizing their operations and increasingly relying on IoT devices and connected medical equipment [14]. The proliferation of the Internet of Medical Things (IoMT) and growing use of wearables for the collection of physiological data and bio-signals is leading to an emergence of new

distributed computing paradigms, such as EC, that combines wearable devices with IoMT for scalable remote tele-treatment and telecare [15, 16, 17].

- Augmented Reality (AR)/Virtual Reality (VR): EC might be the perfect match for AR/VR applications due to its ability to minimize latency, to reduce upstream traffic to/from the Internet, to ensure scalability, and to offer offline functionality (no need to browse the Internet since resources are installed at the edge of the network). Low latency is the key factor of reliable AR/VR application. In fact, for general industrial applications, video streams with a frame rate greater than 60 Hz and 1280×720 High Definition (HD) resolution are desirable [18]. While CC manages to achieve 50 to 100 ms latency in best case scenarios [5], EC in the other hand, could run at < 20 ms [19, 18] and meet the requirements mentioned above. These advantages collectively contribute to delivering smoother, more immersive, and responsive AR/VR experiences, making EC an essential technology for the growth and success of AR/VR applications in the near future.

In summary, EC has attracted a lot of attention in research community during the last decade (see Figure 1 of [20]) and is starting to have commercial interest of technology big players.

1.2 Barriers Towards the "True" Edge

A deployment recently labeled as "Edge Computing" comes from Amazon with "AWS for the Edge" and "Lambda@Edge" services [21, 22]. These services are currently deployed by many businesses, such as Hulu [23], an American subscription video-on-demand service, and Volkswagen Group [24], Europe's largest car maker. Nevertheless, this "edge" is still far from the edge of our vision. In fact, it nothing but a closer cloud to end users. Although it is claimed that the mentioned deployments reduce latency, they still require passing through the Internet.

In our vision, the"true" edge exists in the closest possible point to the end user's device that enables services without accessing the Internet at every data exchange.Such points are Central Offices, cellular network Base Stations, WiFi access points, the Internet "boxes" deployed by operators in households' premises. Such network locations are generally owned by Network Operators (NOs). Even big actors, such as Amazon or Google, cannot arrive, at least up to now, so close to the users. For this reason, we believe **Network Operators have the key to the Edge**. We also believe this is an unprecedented business opportunity for them. Unfortunately, we do not have concrete elements to claim that NOs are actually exploiting such an opportunity.

The main barrier for the deployment of our vision of EC is the huge cost, for NOs, to equip thousands of edge locations with computational capabilities, such as CPU, memory, GPUs and the huge maintenance cost. The prohibitive cost of EC is a significant consideration for organizations looking to harness the potential benefits of decentralized processing.

The expenses associated with infrastructure, hardware, networking, maintenance, and scalability challenges can be substantial. Setting up and maintaining distributed edge servers, ensuring reliable connections, and managing security measures all contribute to the financial burden [25]. The need for customization and integration further add to the overall expenditure. While the advantages of reduced latency and enhanced data privacy are compelling, organizations must carefully weigh these benefits against the higher upfront and ongoing costs to make informed decisions about adopting EC solutions. Therefore, NOs will engage in EC investment only if they can collect corresponding revenues. For this reason, we focus in this thesis in strategies devoted to maximize the utility of NOs, in the form of upstream traffic reduction, revenue from pricing EC resources or increase Quality of Service (QoS) for the customers.

Collecting utility is however not trivial. Network operators are "in the middle" between users and Service Providers (SPs), where the former wish to consume the services provided by the latter. We believe that to get value from EC, network operators need to "open their egde" to third party service providers, via virtualization techniques. In this framework, the main decision of the NO is how to partition the limited resources at the edge between service providers.

1.3 General Setting

In this thesis we consider, in general, a scenario where a Network Operator (NO) owns edge resources deployed in base stations, central offices and/or smart boxes (as depicted in Figure 1.1), virtualizes them and lets third party SPs - or tenants - distribute part of their applications in the edge in order to serve the requests sent by the users. SPs with heterogeneous requirements coexist in the edge, ranging from Ultra Reliable Low Latency Communication (URLLC) for controlling cars or robots, to massive Machine Type Communication (mMTC) for IoT requiring a massive number of connected devices, to media services, such as video streaming and AR/VR, whose quality of experience is impacted by the available resources. SPs independently orchestrate their set of microservices, running on containers, which can be easily replicated, migrated or stopped (for instance, Netflix launches hundreds of thousands of containers daily [26]). This enables service elasticity [27], i.e., each SP can adapt to the resources allocated by the NO, deciding whether to run microservices in the devices, in the edge nodes or in the cloud.

In this general setting, we consider that end users pay the NO if their requests are processed at the edge (as in the classic business model of the NO, end users are charged for communication services, i.e., calls, messages and data). We also assume that users pay SPs to consume their services (either by explicit fees, or by enduring advertisement). Since such payment typically does not involve the NO, on which we focus in this thesis, we do not consider such a payment. Moreover, we do not consider any payment made by SPs to the NO. This is motivated by the fact that in some cases, i.e., when resource in question

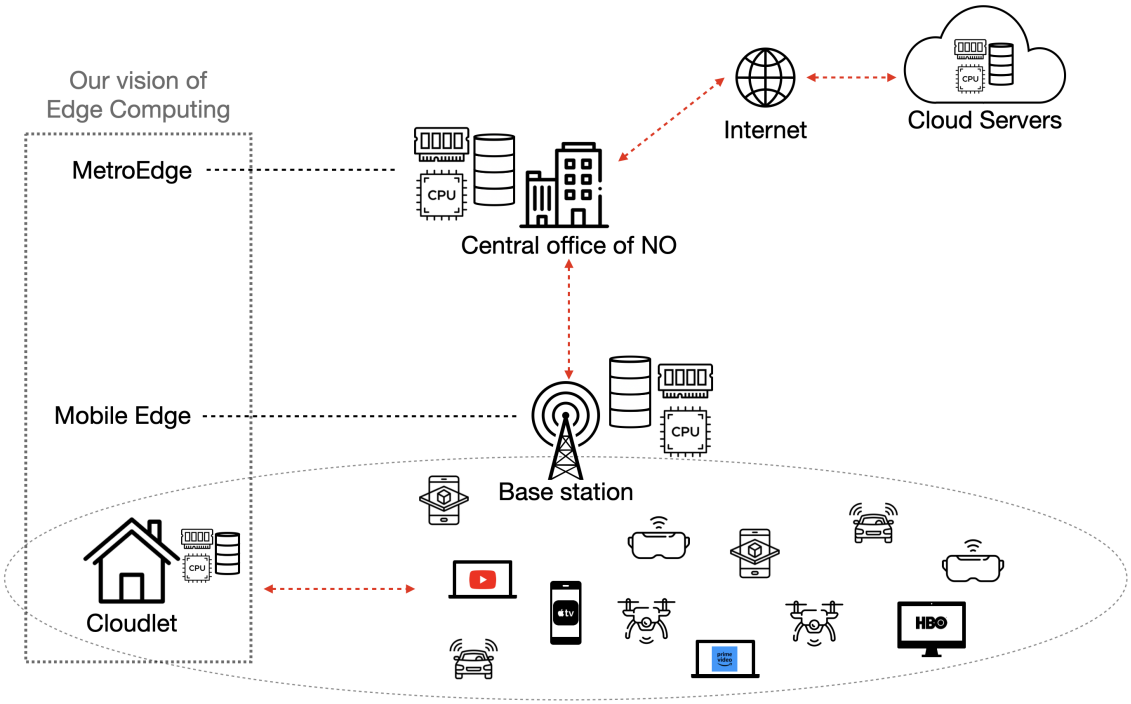


Figure 1.1: General setting of the thesis

is storage, the NO has interest in providing this resource for free. Indeed, by letting SPs cache their most popular content at the edge, the NO can reduce upstream traffic (we develop this motivation in more detail in Section 3.1). There is another reason why we believe payment from SPs should not be the main criterion for allocating edge resources among them, and it concerns "edge democracy". In fact, payment made by SPs to the NO may result in few giant SPs monopolising the market and taking over all resources at the edge. We instead aim to adapt resource allocation to the traffic generated by users.

In this context, our scenario can be described as a bi-level optimization problem, where in the upper level the NO only decides the amount of resources (memory, storage, bandwidth, CPU cycles) to allocate to each SP and in the lower level each SP adapts to the granted resources by orchestrating its microservices accordingly. In other words, to a certain action (allocation decision) of the NO, a relative reaction (orchestration or decision of resource partitioning between content or users) of SPs follows. This reaction depends on SPs' requirements and on information inaccessible to the NO. Indeed, we assume the data and the processing of each SP to be confidential: a SP does not want to share them with the other SPs, nor with the NO. Therefore, the NO does not know the requests received by each SP, its status, its configuration, etc. Otherwise stated, SPs are "black boxes" for the NO, which receive and send encrypted traffic. The NO can only base its allocation decisions on monitoring information, resource usage [28] or metrics inferred from the encrypted traffic [29]. For this reason, the classical approach of first-model-then-optimize is inapplicable in our work, which brings us to the use of Artificial Intelligence. The bi-level optimization problem description above holds for Chapter 3 and Chapter 4. In Chapter 5, the setting is very similar, but to the decision of the NO, the reaction of wireless devices follow (instead of the reaction of SPs).

1.4 Novelty

While most related work assumes a single service running at the edge, one of the points of novelty of this thesis is that we consider instead EC as a **multi-tenant environment**. Note that that multi-tenancy is more cost-effective than single-tenancy, for the resource owners, i.e., NOs and for the tenants.

First, by sharing edge resources, resource owners can reduce capital and operational expenses by avoiding the cost of setting up and maintaining separate edge deployments for each individual tenant.

Second, the multi-tenant edge environment will save the SPs the effort and the cost to deploy and maintain themselves the resources at the edge. This effort is currently born by few SPs, such as Netflix and Google.[1] They place some of their servers directly at Internet Service Providers (ISPs)' access networks. But smaller ISPs, with less economic power and

[1] See Netflix Open Connect [30] and Google Global Cache [31].

bargaining power faced with NOs, would surely not be able to deploy such a widespread infrastructure.

Third, multi-tenancy enables higher utilization of the resources in edge servers (as in cloud servers) and offers greater flexibility for tenants, i.e., they can dynamically adapt the usage of the allocated resources as they wish, based on their specific needs and their observed workload, making it easier to adapt to changing workloads and requirements.

Fourth, multi-tenancy allows guaranteeing confidentiality to the SPs, as the NO does not deal directly with user requests, which instead are handled by SPs, as if computation were running in private premises. The NO just needs to observe aggregated information, such as upstream traffic.

This multi-tenant setting may involve the establishment of Service Level Agreements (SLAs) between the NO and the SPs that specify QoS guarantees, in terms of parameters, such as latency, throughput, and reliability. These SLAs may also introduce some constraints on the flexibility of resource allocation for the NO. For instance, certain resources may be reserved for specific SPs, limiting the dynamic allocation of those resources based on immediate demand. This compels NOs to adapt to changing demands while providing SPs with the assurance of minimum service levels.

While CC is inherently multi-tenant, it is essential to emphasize that resource allocation in a multi-tenant context within EC presents unique challenges and nuances that have not received comprehensive exploration. At first glance, it might appear that all the necessary groundwork has been laid in the cloud environment. However, it is of paramount importance to underscore that multi-tenancy in EC represents a relatively uncharted territory in terms of research and development. What sets multi-tenant resource allocation in EC apart from its cloud counterpart are the distinctive characteristics and requirements of EC. These include factors like the geographically distributed nature of edge devices, the limited resources at the edge (contrary to practically "unlimited" resources in the cloud), their varying computational capabilities, stringent latency constraints, and the need for real-time processing. Elucidating these differentiating factors is crucial to conveying the novelty of this thesis and significance of addressing multi-tenant resource allocation in the context of EC.

In this vision, multi-tenant EC and Open Radio Access Network (Open-RAN) [32] can be strategically linked to create a dynamic and flexible ecosystem. Open-RAN is a network architecture which enables the deployment of multi-operator radio access network. While multi-tenant EC provides the resources in the access network for multiple tenants, Open-RAN complements this by extending the concept of disaggregation to the radio access network, enabling the flexible integration of diverse and inter-operable Radio Access Network (RAN) components from different vendors. This integration of multi-tenant EC and Open-RAN may enhance the agility and responsiveness of the network. It allows for the deployment of edge applications and services that require low-latency processing, while simultaneously optimizing the radio access network through the open and modular architecture of Open-RAN. The collaboration between these two paradigms could enable

NOs to deliver customized and efficient solutions tailored to the specific requirements of various SPs.

1.5 Goals of the PhD

The overarching aim of this thesis is to advance the emergence of real deployments of the "true" Edge Computing (EC) (as we define it in Section 1.2) in real networks, by showing the utility that Network Operators (NOs) can collect thanks to EC. We believe that this can contribute to encourage concrete engagement and investments engagement of NOs in EC, which go beyond press announcement or marketing messages [33].

The goal we set to fulfill the overarching aim is to design novel **data-driven strategies** that efficiently allocate resources between heterogeneous SPs, at the edge owned by the NO, in order to optimize its relevant objectives, e.g., cost reduction (Chapter 3), revenue maximization (Chapter 5) and better QoS perceived by end users, in terms of latency, reliability and throughput (Chapter 4), while satisfying the SPs requirements.

Achieving this goal is particularly challenging when dealing with encrypted traffic because encryption conceals the nature and content of the data being transmitted. This opacity limits the NO's ability to discern the type of traffic, impeding informed decisions regarding resource allocation. Therefore, we rely in our solutions on Artificial Intelligence (AI) methods.

1.6 Methodology

1.6.1 Optimization Problems and Methods

Our approach is bottom-up. As depicted in Figure 1.2, when proposing a resource allocation strategy, we usually start by formalizing the scenario and the objective in an optimization framework. In Chapters 3 and 5 we use the framework of Markov Decision Process (MDP). In Chapter 4 we first use the framework of submodular optimization and then cast the problem as a MDP.

Our resource allocation problem is studied incrementally.

At first, in Chapter 3, we suppose that the NO owns one single resource, namely storage, and aims to allocate it among multiple video streaming SPs on an up and running system, for the sake of one objective: reduce upstream traffic under encrypted traffic. Hence, the use of Reinforcement Learning (RL).

Second, we raise the challenge to two resources in Chapter 4, where the NO allocates memory and CPU between heterogeneous Mobile Augmented Reality (MAR) SPs in order to maximize the QoS perceived by end users. Note that in these two first chapters, we consider a relation between the NO and the SPs while users are assumed to be just the source of data (traffic) and we do not model their relation between the users and the NO,

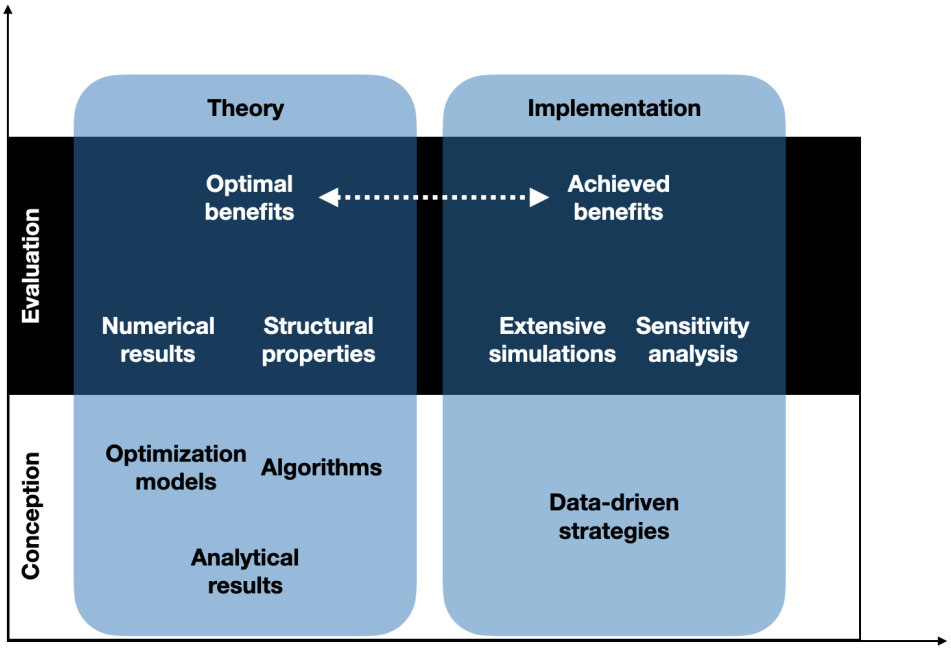


Figure 1.2: Evaluation methodology used in this thesis

nor between the users and the SPs. Moreover, we consider no payments made by the SPs to the NO, and hence no pricing strategies are provided (as motivated in Section 1.3).

Finally, we extend our formulation to include pricing in Chapter 5. We model the relation between the end users and the NO: end users pay the NO for memory and CPU they are using at the edge. We develop a ready-to-use strategy based on RL and Bayesian Neural Networks (BNNs), that maximizes the NO revenue taking into consideration the users response to such strategy. We consider in this chapter that the end users are executing applications (an application is a set of functions) provided by certain SPs and the resources allocated by the NO to users running the same application are aggregated as if they were allocated to the SP offering the application.

During the thesis, we observe that the main disadvantage of RL, which we principally propose as a solution, is its slow training phase. Therefore, when training RL online, on a

system that is up and running, learning a strategy only on actually observed system transitions, i.e., jumping from a state to another, is not sufficient. Indeed (i) such transitions happen at every time slot (for instance 1/4 of second) and (ii) RL training usually requires thousands of transitions. This means that training a good RL policy online would require too much time, due to the exploration required during training.

In both Chapter 3 and Chapter 5 we remove this barrier using the same modeling "trick": we use actually observed transitions to train a model of the system (in Chapter 3 this model is based on simple regression, while in Chapter 5 this model is composed by a pair of Bayesian Neural Networks). Then, we train RL on "fictional transitions" generated by such a model. This has the great advantage that we can now generate thousands of fictional transitions in a small span of time, as such transitions do not need to be applied to the real system. This allows boosting the sample efficiency of our learning process, i.e., relatively few observations are needed to learn a good policy.

1.6.2 Evaluation

We evaluate our proposed strategies by means of theoretical analysis and simulation.

In Chapter 3 we compare the results of our data-driven approach with the theoretical optimum that could be achieved in the ideal assumption of having perfect knowledge about request load.

In Chapter 4, the considered multiple-resource allocation problem is proved to be NP-hard. We formulate it as a sub-modular maximization problem with multi-dimensional knapsack constraint, and we solve it with the so-called streaming algorithm, in the case where all system parameters are known. We then propose a learning approach and compare it with the streaming algorithm.

The same methodology is followed for resource allocation and pricing problem in Chapter 5. We then show that the problem can be cast as a Hidden Parameter Markov Decision Process (HiP-MDP) and proposed a dual BNN approximator as a solution. We compare our pricing policy with state-of-the-art strategies present in the literature.

Overall, in our work we start from theory to end in simulation. The final goal is, in any case, to propose implementable strategies, which we evaluate by means of simulation of realistic scenarios.

1.7 Contribution of the Thesis

This thesis gives a vision of how NOs, the only entities owning the resources in the far edge, can get value from the deployment of EC in a multi-tenant setting. In such a setting the NO makes edge resources available to third party SPs and intelligently allocate resources among them. The relevant objectives of the NO we optimize in this thesis are upstream traffic reduction, revenue maximization and better QoS perceived by end users.

We believe that, by promoting this vision of EC and by supporting it via quantitative results and analysis, this thesis provides, mainly to NOs, findings that can impact (and hopefully encourage) decision strategies about the future deployment of EC. This might foster the emergence of novel low-latency and data-intensive applications, such as high resolution augmented reality, which are not feasible in the current CC setting.

We consider that another contribution of the thesis it that it applies novel methods that harness the power of data-driven optimization. We indeed adapt cutting-edge techniques from RL and sequential decision making to the practical problem of resource allocation in EC. In doing so, we succeed in reducing the learning time of the adopted strategies up to scales that are compatible with the EC dynamics, via careful design of estimation models embedded in the learning process (Chapters 3 and 5). We also prove important analytical properties of our strategies (e.g., convergence in Chapter 3).

We also emphasize that our strategies are conceived in order not to violate the confidentiality guarantees that are essential for SPs to accept running their computation at the EC, thanks to the multi-tenant setting.

1.8 Thesis Organization

The remainder of this thesis is organized as follows:

- **State of the Art**. Chapter 2 reviews the resource allocation and pricing strategies conceived in the literature and implemented in production networks.
- **Cache Allocation for Multi-tenant EC**. Chapter 3 solves the problem of cache allocation at the edge among several SPs, where the aim is not only to minimize the cost, in terms of miss rate, but also to optimize the way to achieve that, through minimizing perturbations, assuming encrypted, not all cacheable content: a major challenge of in-network caching. We introduce a model-based RL algorithm designed for real-time cache allocation at the edge, on an up and running system. We start with the establishment of theoretical foundations, demonstrating the algorithm convergence towards an absorbing discrete optimal state. Subsequently, we conduct an extensive set of simulations, comparing our dynamic allocation approach to other strategies, including a method from the state of the art and model-free RL. Our simulations clearly illustrate that our algorithm consistently converges to a configuration close to the theoretical optimal solution. Convergence is much faster than the compared allocation strategies thus substantially reducing overall system costs.
- **Multiple-resource Allocation for Multi-tenant EC**. Chapter 4 tackles the resource allocation at EC between SPs competing over multiple, limited resources, e.g., CPU and memory. We model the users dynamics in terms of an Erlang-type queuing model, we formulate resource allocation problem as a sub-modular maximization problem subject to multiple knapsack constraints and solve it via an approximation algorithm with provable optimality gap, under perfect knowledge of system parameters. We formulate the problem as a sequential decision making problem, when system parameters are unknown, and we cast it as MDP and propose deep RL to solve it. Our numerical results quantify the performance of the deep RL algorithm and the approximation algorithm in terms of the probability that users get served by the Edge, as opposed to being blocked and re-directed towards the Cloud which entails larger delay and hence lesser QoS.
- **Resource Pricing for Serverless EC** Chapter 5 considers the problem of pricing in serverless EC under dynamic workloads of individual users, as opposed to the previous two chapters where the interaction was between SPs and NO. We first formulate the problem of maximizing the revenue of the operator as a sequential decision making problem under uncertainty. We then show that the problem can be cast as a HiP-MDP and proposed a dual BNN approximator as a solution. The proposed solution is a form of transfer learning; after pre-training on synthetic traces, it adapts fast to previously unseen workloads. Our results show that the proposed solution accelerates learning and achieves superior performance compared to the state of the art.

Furthermore, our results show that non-linear pricing models could benefit edge deployment, as they encourage users to request computational resources sparingly, and thereby effectively increasing the number of concurrent users that can be served. The work contained in this chapter results from a collaboration with Prof. György Dán from the Royal Institute of Technology (KTH), Sweden, and Feridun Tütüncüoglu, PhD student at KTH where I carried out this work during my doctoral mobility in KTH in Spring 2023.

1.9 List of Publications

- Ben-Ameur, A., Araldo, A. and Chahed, T. (2022, May). Cache allocation in multi-tenant edge computing via online reinforcement learning. In ICC 2022-IEEE International Conference on Communications.
- Ben-Ameur, A., Araldo, A. and Chahed, T. (2023, May). Multiple Resource Allocation in Multi-Tenant Edge Computing via Sub-modular Optimization. In ICC 2023-IEEE International Conference on Communications.
- Ben-Ameur, A., Araldo, A. and Chahed, T. (Submitted to journal). Online Model-based Reinforcement Learning for Multi-tenant Edge Cache Allocation.
- Ben-Ameur, A., Tütüncüoğlu, F., Dán, G., Araldo, A. and Chahed, T. (Submitted to conference). Dynamic Time-of-use Pricing for Serverless Edge Computing with Hidden Parameter Markov Decision Processes.

Chapter 2

Context and State of the Art

2.1 Edge Computing Ecosystem

Within the intricate ecosystem of Edge Computing (EC), several key actors play pivotal roles, contributing to its dynamic growth and development. First and foremost are the technology giants and cloud providers such as Amazon Web Services (AWS) [34, 35, 21, 22], Microsoft Azure and Google Cloud, who are investing heavily in EC infrastructure and services [36], [37]. Lastly, startups and innovative tech companies are driving experimentation and pushing the boundaries of EC applications. For instance, Akamai [38] offers, among other services, Edge Content Delivery Network (CDN) that lower overhead with pre-built edge applications that run capabilities integrated on CDN, serverless computing that boost performance and user experience by enabling developers to build web applications closer to end users.

However, apart from the "marketing" appeal, the exact meaning of the term "edge" employed by the aforementioned companies (and others) is not clear. In fact, these deployments are far from the very edge, i.e., the last point before the device of the end user. Without downgrading the efforts done by Amazon, Microsoft, Google and Akamai, we can describe their deployments as "closer cloud", as they require anyways going through the Internet, since their servers generally sit not far from the users (e.g., behind Internet Exchange Points) but not very close (e.g., in cellular base stations or WiFi access points). [1]

Observe that the far edge is owned by the NO as it is the only actor in the network that owns base stations, central offices and smart boxes (the ones that come with household Internet subscriptions).

[1] We discuss here only solutions that allow third party SPs to run their computation in the Edge. Therefore, Netflix Open Connect and Google Global Cache are outside of the scope of our discussion here, as they only carry content of Netflix and Google, respectively. We have already discussed the limitations related to these solutions in Section 1.4.

Another key player, in this vision of the edge, is tower companies.[2] In fact, they can use some of their existing tower sites or infrastructure to host edge servers. Since tower companies often have experience in site acquisition (identifying and evaluating suitable locations for towers), zoning (classifying areas for towers placements), and permitting (obtaining the necessary approvals) processes. They can help identify suitable locations for edge servers, navigate regulatory requirements, and streamline the deployment of edge infrastructure. Since, in our vision, tower companies have a role in EC similar to NOs, we will just focus on NOs in this thesis. For what concerns the methods that we propose here, NOs and tower companies are interchangeable, *mutatis mutandis*.

Collaborations between cloud providers and telecommunications companies are becoming increasingly common to offer seamless edge services, for instance Google Cloud and Orange have recently announced a strategic partnership, part of which is dedicated to edge computing [39]. However the limit of this agreement is that only Google will use that edge and not all the third party SPs, which is an issue we raised in Section 1.4.

Furthermore, hardware manufacturers, ranging from chip makers like Intel and NVIDIA to device manufacturers like Dell and HP, are designing specialized hardware optimized for EC workloads [40], [41], [42], [43]. Such hardware is characterized by its compactness, energy consumption efficiency and high processing power (GPU, CPU and memory). Moreover, this kind of hardware is optimized for data analytics, deep learning and machine learning (for instance Nvidia Jetson Nano [43] is CUDA-X-native which makes it suitable for AI and robotics).

In the software domain, open-source communities are actively contributing to EC ecosystems by developing edge-focused platforms and frameworks. Projects like Kubernetes K3s [44] and OpenStack are adapting to the demands of edge environments. In fact, K3s is designed to be minimalistic and resource-efficient. It has a smaller memory and CPU footprint compared to K8s, making it better suited for edge devices with limited computing resources. Moreover, K3s supports offline installation, which can be essential in edge scenarios where internet connectivity may be unreliable or unavailable. This ensures that edge clusters can be set up without reliance on external repositories.

That being said, some parts of the research community are still doubting the potentials of EC. For instance, authors of [20] observe that cloud providers are expanding their data centers to many countries. Furthermore, cloud providers are establishing (and incorporating) specialized facilities to tackle edge needs, such as CloudFront, and thus there is no real need for EC. They claim that the effectiveness of EC is limited to a few applications, such as traffic monitoring, gaming, AR/VR, etc. Assuming that the measurements provided in this survey are precise, we believe in any case that the mentioned applications are not

[2] Tower companies, also known as cell tower companies or telecom tower companies, are specialized firms that own, operate, and manage telecommunications infrastructure, specifically cell towers and related assets. These companies play a crucial role in the wireless telecommunications industry by providing infrastructure to NOs.

limited, as they have their place in the market (for instance the size of AR market is 42.20 billion in 2022 [45]).

The ecosystem of EC forms a dynamic and intricate web of interconnected devices and services at the edge of traditional cloud infrastructure. However, the efficient utilization of resources in this environment is crucial to ensure optimal performance, as resource at the edge are scarce (contrary to resources in the cloud, which are so abundant that they can be considered by the cloud users as practically infinite). This necessitates resource allocation strategies particularly tailored for the limited resources and low latency requirements that characterize EC. This is the subject of this thesis.

High data rate is a key requirement for the successful deployment and operation of 5G [46] and future 6G [47]. Using edge servers, rather than the cloud reduces the traffic amount across the connection between small cells and the core network. Hence, (i) the bandwidth of the connection can be increased to prevent bottleneck; and (ii) the traffic amount in the core network is reduced.

2.2 Terminology

The terminology surrounding Edge Computing has not yet converged to a well established vocabulary, upon which all agree. Many researchers consider that Edge Computing refers to the practice of processing data closer to the source, reducing latency and bandwidth usage [48]. This definition is not wrong, but very general and lacks precision. In fact if you replace Edge by "Fog", "Mobile Edge" or "Cloudlet" in this definition, it will hold. This section is here to eliminate this ambiguity and answer the question *where is the Edge?*, at least for us, for what concerns this thesis. As we mention in Section 1.3, in our vision, the edge is owned by the NO and exists in central offices, base stations and smart boxes. When resources are deployed in base stations, we can use the term "Mobile Edge Computing" (MEC) [49]. "Cloudlet" corresponds to the case where the resources are deployed in the smart boxes delivered by the NO [50]. Fog computing [51] extends this idea, introducing a hierarchical structure where data processing occurs not only at this edge, the way we define it above, but also in intermediary fog nodes, between edge and cloud. In this thesis, when we use the term Edge Computing, we will mean MEC or Cloudlet, interchangeably.

2.3 State-of-the-art in the Scientific Literature

We give in this section a comprehensive overview of the current landscape within the field of resource allocation in EC with a focus on (i) cache allocation: as we start the thesis by solving the problem of caching at the edge in Chapter 3, (ii) multiple-resource allocation in EC: as it is very relevant to Chapter 4 and Chapter 5 where we study the case of multiple resources, (iii) data-driven methods for resource allocation: as we base our solutions on

data-driven optimization and (iv) resource pricing: since Chapter 5 solves the problem of dynamic pricing in EC.

2.3.1 Cache Allocation

Edge caching has been extensively studied in the literature as experts recognize its potential to reduce data latency and enhance traffic efficiency. By deploying caches closer to users and devices, this approach promises faster content delivery, more responsive data retrieval and reduction of upstream traffic. Cache space is often perceived as a single resource, where either (i) all stored objects belong exclusively to one Service Provider (SP), or (ii) when multiple SPs are involved, objects are stored indiscriminately without differentiation between SPs. However, we advocate for an alternative approach known as cache partitioning. Cache partitioning involves allocating distinct portions of cache space to individual SPs. This approach provides several advantages, including the fact that each SP is isolated from the others, thus guaranteeing to the SP that its data remain confidential.

Hence, instead of describing research in caching strategies in general, we only focus here in the research concerning cache partitioning problems.

In the approach of [52], each cache can be partitioned into slices with each slice dedicated to a content provider. However, they need information about the system conditions in order to solve it (e.g., request rate for each content provider). We assume instead that no information is available and that optimization is done by observing the changes in upstream traffic induced by perturbing the allocation.

In [53], authors formulate collaborative joint resource allocation problem as an Integer Linear Program (ILP) that minimizes the backhaul network cost, subject to capacity constraints. Multiple edge nodes collaborate to orchestrate the allocation of cache and computation resource (jointly) for only one SP. However, in our work MEC is a multi-tenant environment as we motivate in Section 1.4.

Authors of [54] study optimal content caching problem in EC, assuming content popularity is unknown and the instantaneous demands are observed only for those contents stored in the cache memory. They model this problem as a Combinatorial Multi-Armed Bandit with Switching Cost (CMAB-SC) problem. Similar to our RL framework, in MAB problems, an agent with partial system knowledge repeatedly takes actions to maximize accumulated rewards over time, while acquiring new knowledge. In our case, due to encryption, the NO can only allocate cache slots to SPs without knowing their content, allowing SPs to decide what to store. Our assumption is more realistic, as today SPs generally require that their traffic stays encrypted and unknown to the NO.

In [55], the authors propose a resource pricing framework for one NO and several SPs, knowing the demand of the users. We instead do not know what users request. Also, our focus is on resource allocation and not pricing, we assume that SPs do not pay for the resources (as motivated in Section 1.3).

In [56], the MEC network is assumed to have multiple cache servers to assist SPs, each with its own set of users. Each cache server acts as a rational selfish player, in a bargaining game, aiming to maximize its utility by making strategies of local self caching. In [57], authors also consider sharing cache between SPs, by applying coalitional game theory. They assume that the cache resources at the edge are owned by a central office equipped as a data center connecting multiple NOs and each NO pays for these resources to make them available for SPs. The NO also pays SPs to convince them to use the cache at the edge and lets them decide how much resources they get. This may seem counter-intuitive, as we expect that the NO will get paid when it provides certain resource. They justify this by the fact that the interest of the NO is to minimize its expenses: the authors show that if the NO does not pay SPs to use the cache, the NO will end up paying more, in order to carry the large amount of traffic from SPs. However, our allocation decision is centralized by the NO, who owns the resources, and we do not require any payment (as explained in Section 1.3).

Recently, authors of [58] considered the problem where the NO is aiming to allocate applications images at the edge, in a cache with limited capacity. The NO decides the amount of cache to give to each application in order to maximize its profit. The problem is solved via a Stackelberg game. Each placement has a cost for the NO. Users have computational tasks to be executed, each associated to one application. A user device can decide to offload a task at the edge, if the correspondent image is cached there, and in this case, it pays the NO. Otherwise, it executes the task locally, with a certain energy cost. Users want to execute their tasks within latency constraints, while minimizing their payments. The authors work under complete information assumption, i.e., the system parameters and utilities are assumed to be known by the NO, while in our work this information is unknown.

2.3.2 Multi-resource Allocation at the Edge

In Chapter 4, we extend our work from one resource to multiple resource allocation, i.e., from cache to memory and CPU.

In [59], the authors consider an EC system under network slicing in which the wireless devices generate latency sensitive computational tasks. The allocation of wireless and computing resources to a set of autonomous wireless devices in an EC system is considered in [60]. They model the interaction between the NO, which manages the allocation of wireless and computing resources, and devices. In order to minimize completion time, the latter decide autonomously whether to use shared resources for offloading computing tasks so as to minimize their own completion times or to compute tasks locally.

In [61], authors establish a software-defined networking based architecture for edge/cloud computing services in 5G networks to manage on-demand computing resource in order to satisfy time-varying computational tasks.

A dynamic provisioning of computing resources is considered in [62]. Computing resources are provisioned as VM instances on the fly. The authors propose two allocation and pricing mechanisms based on greedy algorithm and linear programming based approximation.

A main common assumption of [59, 60, 61, 62] is that user devices submit tasks to the NO and such tasks consume resources to be executed and transmitted. Contention for resources is then modeled among user devices. However, we consider that these models are not appropriate for MEC in our vision (Section 1.3), since all traffic between devices and SPs is encrypted to maintain confidentiality and the NO does not have control over it. Therefore the contention for resources is, in our vision, between SPs and not between tasks submitted by users. In our assumption, the NO can only decide how to allocate resources among SPs and then users devices interact directly with SPs, outside the control of the NO.

In [63], the authors assume that each SP explicitly requests a certain amount of resources (e.g., memory, CPU and link capacity) and consider the difference between the resources requested and the resources actually allocated to them. The decision maker (the NO) in their work aims to maximize the fairness of the resources allocation. We instead assume that the NO allocates its mobile edge resources so as to satisfy its own goals (upstream traffic minimization in our case), without requiring to receive explicit resource requests from SPs.

An explicit request of SPs of the amount of resources they are willing to consume is also required in [27] and [64]. The former relies on a heuristic for resolution and assumes that SPs are truthful and declare the resources they really need. The latter makes use of Monte-Carlo Tree Search and SPs pay proportionally to the resources they are granted. We do not need such assumptions in our work.

Similar to us, but in the context of network slicing, in [65] the NO jointly allocates CPU and bandwidth to several tenants, one per slice, in order to minimize an objective function relevant to the NO (energy, in their case). However, they assume the NO knows the expected load of requests of each tenant and the requirements of each request, while we do not require this assumption in our work (Chapter 4).

In [66], the authors propose a market-based framework for efficiently allocating resources of heterogeneous capacity-limited edge nodes to multiple competing SPs. Each SP is a player. Given a price vector of the resources, each SP aims to maximize its utility subject to a certain budget constraint. The authors assume that this utility function is known (see Section 4.2 of [66]). However, in our case the utility (opposite to cost) function is unknown. Moreover, we maximize the utility of the NO (not SPs), who owns the resources.

In [67], multiple edge servers and only one SP are considered. The resources on these edge servers are managed by multiple NOs. The SP has a budget to use resources at the edge. The end-users subscribed to this SP submit computation jobs subject to delay con-

straints. An algorithm is proposed to allocate resources at each time-slot to the submitted jobs. While their work considers only one SP, we consider multiple SPs and one NO.

2.3.3 Data-driven Methods for Resource Allocation

The main limit of most of the aforementioned work is that they assume that an exact characterization of the system is known, i.e., all the quantities and dependencies involved in the resource allocation problem are known in advance. This is not the case for this thesis, where we generally assume the NO does not know the expression of the cost (or reward) function (upstream traffic in Chapter 3, blocking probability in Chapter 4 and revenue in Chapter 5) that we want to minimize (or to maximize). We thus need to resort to data-driven approaches, which can drive cost functions toward the minimum in the absence of a known characterization of the system, based solely on monitoring information.

RL has been used for resource allocation in the context of MEC in, for instance, [68, 69, 70, 71, 72]. Joint management of the communication and computation resources using deep RL is considered in [68]. In [69], the authors consider a cluster with multiple resource types and use deep RL to choose one or more of the waiting jobs to schedule at each time step. In [70], the authors solve the problem of allocating GPU at the edge to run deep neural networks to maximize the QoS of the prediction model. Authors of [71] use a RL approach to allocate CPU time, virtual CPUs and memory, to Virtual Machines (VMs). In [72], authors propose deep RL for allocating resources in a network slicing scenario. Contrary to our approach, the authors of the works mentioned above pre-train the RL algorithm offline on a simulated system before using it on the actual one. We instead assume that no information to build a simulator is available and we train our algorithm online, directly acting on the hot and running system in Chapter 3 ad Chapter 5. This imposes on us a more parsimonious learning strategy as we need to ensure that the system is not heavily perturbed during training.

In [73], the authors present a RL algorithm for resource auto-scaling in clouds: resources are assumed to be unlimited, however the goal is to allocate to each SP an amount of resources that does not exceed its needs. In our case, instead, resources are scarce, allocating resources to one SP means allocating less for another. This implies that we cannot decide how much resource to allocate to each SP in isolation. Therefore, our solutions need to always take allocation decisions for all SPs at the same time.

To the best of our knowledge, the only data-driven method we can compare against for partitioning a finite amount of cache among several SPs with encrypted content allocation (Chapter 3) is Simultaneous Perturbation Stochastic Approximation (SPSA) [28]. The authors do so based on stochastic optimization. However, they need to continuously perturb the allocation, generating spurious upstream traffic that may be non-negligible. We instead include traffic perturbation into the cost function (Section 3.2.3), thus managing to keep it low, which allows us to outperform [28], as shown in Section 3.6.

2.3.4 Resource Allocation and Pricing

As opposed to Chapter 3 and Chapter 4, where we treated the resource allocation problem without payment, we focus in Chapter 5 on resources in EC via pricing strategies. The commercial adoption of EC will require pricing schemes that cater for the financial interests of the operators and of the users. Pricing in EC is particularly challenging as it has to take into account the limited amount of edge resources (contrary to the cloud where resources are "infinite") as well as the stochasticity of user workloads due to location-specific workload characteristics and differences in user activity. Note that by statistically multiplexing requests coming from different locations, the peaks from one location may be compensated by off-peak in others. In EC instead, all requests are local, and there is no aggregation that can limit variation, as it happens in the cloud.

Several approaches have been proposed for the network operator for allocating and pricing its limited resources at the edge to different tenants.

Auctions [74, 75, 62] lack transparency, as the price at which resources are allocated is not announced in advance, but it is determined by the set of bids. Similar to our setting, in [76] users queue if all EC resources are currently occupied, and the operator needs to find a trade off between accepting more users, thus increasing revenue, and avoiding long waiting times, affecting user QoS. However, [76] assumes that the operator uses admission control by suggesting users to join or balk, while prices are considered to be given. We instead explore this trade-off by means of dynamic pricing, which can clearly provide higher revenue than just admission control.

Pricing is often studied via game theory (Stackelberg [77, 78, 79, 80] or coalitional games [81]) where the network operator sets prices and Wireless Devices (WDs) take offloading decisions. In [77, 78, 81] and [80] calculating the optimal price requires prior information about EC traffic characteristics, which is unrealistic. More realistic is the setting of [82], where the operator has incomplete information, i.e., it only knows the distribution of WDs' traffic characteristics. However, in reality, not even such distributions are known, and they would vary over time. In [79] pricing decisions are taken under the assumption that if too many users use an edge server, such a server will fail and nobody can use it anymore. Linear pricing is learned via RL in [83] for stationary traffic, which our proposed solution outperforms.

Recently proposed dynamic pricing schemes are based on learning and require no a-priori knowledge about user traffic [84, 85, 86], but suffer from three main limitations. First, the training phase is very long, which makes these approaches infeasible in practice, since training requires exploration of prices, which can lead to revenue loss for a long period of time. The "Spot" pricing proposed in [84] requires training on 10^4 servers, while the number of training epochs (i.e., days) is 400 in [85, Figure 1]. The number of training epochs are not reported for the RL agent of [86] and the Multi-agent RL of [87] (which also performs price discrimination, when the NO charges different prices for the same resources

to different users based on their willingness to pay, such pricing scheme is unfair and lead to economic inequality).

Second, pricing is linear, which we show to result in reduced revenue in EC. Third, the RL algorithm in [85] does not include the current utilization of edge resources into the state. Therefore, the algorithm cannot capture the congestion on resources and thus, the only trend that the algorithm can learn is that of a repetitive workload, whose dynamic is the same from a day to another. The algorithm has no means to perform well on unseen workloads, which our proposed approach is capable of (resource utilization is directly correlated to the dynamics of the workload because the level of resource utilization depends on how the workload behaves and how it fluctuates over time).

The RL-based resource load-aware dynamic pricing in [86] needs only one day of training data, but adopts a simple linear pricing structure with only 8 possible prices. A mean field approximation, assuming an infinite number of edge servers, is used to obtain asymptotic results about load-aware dynamic pricing [88], which are however of limited practical use in the problem we consider with a single edge node.

2.4 Position of the Thesis in the State-of-the-art

While previous research has mostly focused on resource allocation in single-tenant scenarios or disregarded the complexities introduced by encrypted traffic and uncertain workload patterns, this thesis develops resource allocations strategies that address the challenges posed by coexisting tenants with diverse resource demands while ensuring the confidentiality of data through encryption. Furthermore, the incorporation of uncertainty factors, such as varying workload dynamics brings practical relevance to our strategies based on AI methods.

Chapter 3

Cache Allocation for Multi-tenant Edge Computing

3.1 Introduction

Caching plays a pivotal role in addressing the imminent challenge of data generation surpassing current Internet capacities [89]. As we look to the future, the priority is to efficiently fulfill (mobile) user requests directly at the network's edge. This approach significantly diminishes the need for upstream traffic and minimizes latency associated with requests to and from remote server locations. Edge Computing (EC) emerges as a crucial strategy, involving the deployment of storage to nodes strategically positioned at the edge of the network [90].

Since more than 80% of the Internet traffic might be represented by content delivery, and in particular video [91], EC might be particularly relevant for these applications. This is testified by the fact that big players in video content delivery are already deploying EC solution. The most notable example is represented by Netflix Open Connect Appliance (OCA) [26]: Netflix installs its own hardware servers into the access networks of some Network Operators (NOs). By remotely controlling such servers, Netflix can place there the most popular content, which can then be served directly by the servers, without producing upstream Internet traffic. Google Global Cache employs a similar system [92]. While this solution seems today very effective, it is costly in the sense that only big players can afford installing their own piece of hardware into access networks. Hence, multi-tenant EC (Section 1.3) is particularly interesting for all the others, as it is probably the only way for small or medium Service Providers (SPs) to reach the edge of the network. Moreover, it is impossible to install one hardware server per service in the edge nodes very close to final users, e.g., in their WiFi access points or home "smart boxes".

Indeed, the very edge of the network is owned by the NO, which is the only one that can deploy storage close to the users. On the other hand, the services consumed by the latter,

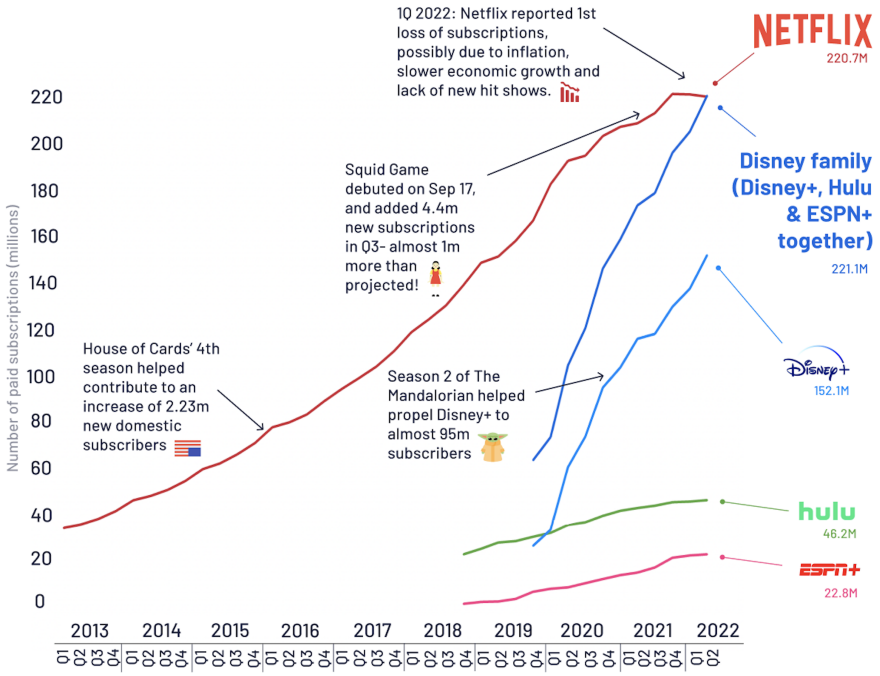


Figure 3.1: Evolution of the number of users of major video streaming SPs [1]

e.g., video streaming, are typically provided by third party SPs, such as Netflix, Amazon Prime Video, HBO, etc. Hence, the question is how to let SPs run their applications close to the users if the storage is not owned by them, but by the NO. We position our work in the framework of multi-tenant EC [27]: the NO virtualizes the storage resources at the edge, partitions them and allocates them to the SPs (tenants).

Storage is becoming increasingly precious in the face of the current and future traffic deluge (see the evolution of the number of users of major video streaming SPs in Figure 3.1). Upstream traffic can be very costly for the NO: it requires (i) network infrastructure to carry it and (ii) specific arrangement with other Internet Providers to receive such traffic from the Internet.[1] Therefore, reducing upstream traffic can help the NO save on expenses. In fact, the cost of bandwidth is the cost of all transport and routing equipment dimensioned to carry busy period traffic between given points in the network with adequate quality of service. This cost is proportional to peak demand [57].

[1]In some cases, the NO could be connected to the Internet via a Transit Internet Provider, which requires to be paid [93]. In other cases, the NO could have peering agreements with an Internet Provider, where traffic can be exchanged for free, but cannot anyways exceed certain limits specified in the agreement [94]. Another alternative for the NO is to join Internet Exchange Points (IXPs) [95]. In this case, the price the NO pays increases with the requested "port capacity", which increases with the upstream traffic [96, Section VI.4].

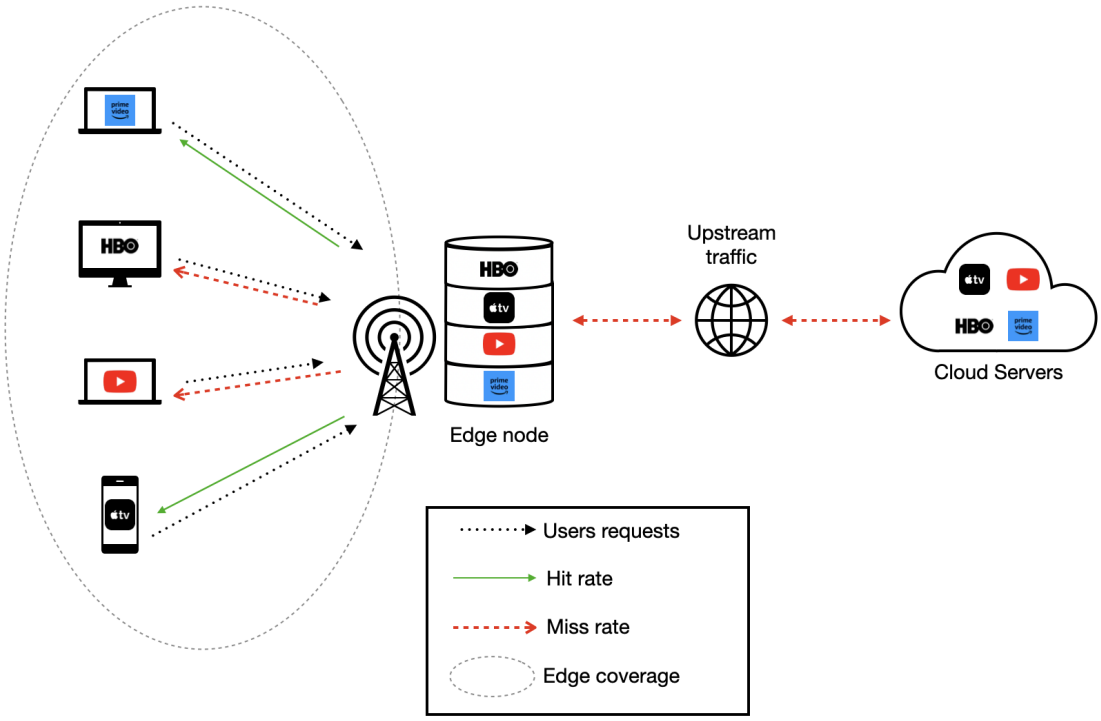


Figure 3.2: Cache allocation and upstream traffic (Origin servers → Edge) with multiple Service Providers (SPs).

Providing cache resources at the edge for free to SPs helps address this issue, otherwise the NO would have to invest more money in more bandwidth to carry remote SPs traffic, which is proven to be more costly in [57]. The authors of [57] suggest that it is a good financial incentive created by the NOs for the SPs to place their content in the free allocated cache deployed at the edge of the network.

We assume that the NO owns storage at the edge nodes and uses it as cache. However, the NO cannot operate caching directly, as classically assumed, as all the traffic is encrypted by the SPs, and so it is impossible for the NO to know which objects are requested, for instance which ones are the most popular ones, nor whether they are cacheable (for instance online video broadcast is not cacheable as chunks become obsolete few seconds after they are produced). We thus assume that the NO allocates storage among SPs and lets each SP decide what to cache within the allocated space, as depicted in Figure 3.2.[2] Our aim is to

[2]As in classic content caching, we assume the SPs do not pay the NO for the cache: cache is used by SPs for free, and the NO compensates the initial storage deployment cost with upstream traffic reduction.

solve the problem for the NO to optimally allocate cache storage among several SPs, i.e., deciding how many *cache slots* should be allocated to each SP, in order to minimize the upstream traffic, i.e., the traffic from the Internet to the edge node, without knowing the nature of the traffic of the SPs. Other objectives, which are out of the scope of this chapter, could be improving user Quality of Experience [29], reducing power consumption [97], maximizing throughput and/or fairness [98], etc.

Since traffic is encrypted (the confidentiality of the data and the processing of each SP can be guaranteed by state-of-the art memory encryption technologies, like Intel SGX [99]), the NO can only base its allocation decision on data-driven strategies consisting in *trial and error*: the NO continuously perturbs cache allocation and observes induced variation on the upstream traffic. We formulate this problem as a Markov Decision Process (MDP) and we use *online* model-based Reinforcement Learning (MB-RL) to solve it: while RL is usually trained offline and then applied to a real system, we instead train RL on the system while it is up and running. Therefore, we are not only interested in finding a good cache allocation, but also in *how* to find it. Indeed, while the only way for the NO to learn how to optimize the allocation is to continuously perturb it, we also need to keep the cost of such perturbations small. Our contributions can be summarized as follows:

1. We propose a MB-RL agent for cache allocation among third-party SPs. Starting from no knowledge of the system, the agent constructs a model of the upstream traffic based on observations. Then, the agent learns an optimal policy on such a model, while at the same time continuously updating the model-based on observed traffic. Our RL agent includes the following features: memory replay, learning rate scheduling and decaying exploration probability (epsilon-greedy).

2. We analytically prove that the allocation converges to a state close to the optimum and stays there with probability 1 for an infinite horizon.

3. We show in simulation the performance of our method under different scenario parameters and show that it outperforms the state of the art Simultaneous Perturbation Stochastic Approximation (SPSA) [28]. We also show the benefits of using MB-RL over model-free Reinforcement Learning (MF-RL).

The remainder of this chapter is organized as follows. In Section 3.2, we present our system model. In Section 3.3, we present the MDP formulation of our problem. Section 3.4 and Section 3.5 discuss the theoretical properties and motivate the use of RL, respectively. In Section 3.6, we show our simulation results. Our simulation code is available as open source.[3] Section 3.7 concludes the chapter.

Notation	Description
P	Number of SPs (Section 3.2)
K	Cache total capacity (Section 3.2)
N_p	Catalog size of SP p (Section 3.2.1)
ζ_p	Cacheability of SP p (Section 3.2.1)
ω	Exogenous conditions (Section 3.2.2)
$\boldsymbol{\theta}$	Cache allocation (Section 3.2)
$\mathbf{a}$	Action of the NO (Section 3.2)
θ_p	Number of slots given to SP p (Section 3.2)
$\boldsymbol{\theta}^*$	Optimal allocation (Section 3.2.3)
Δ	Perturbation (Section 3.3.1)
$\boldsymbol{\theta}^{\text{prop}}$	Proportional allocation (Section 3.6.2)
λ	Total request rate (Section 3.2.1)
f_p	Probability that a request is for SP p (Section 3.2.1)
$\rho_{c,p}$	Popularity of object c of SP p (Section 3.2.1)
$\lambda_{c,p}$	Request rate for object c of SP p (Section 3.2.1)
$C_{\text{nom},p}(\theta_p, \omega)$	Nominal cost for SP p (Section 3.2.2)
$C_{\text{nom}}(\boldsymbol{\theta}, \omega)$	Nominal cost (Section 3.2.2)
$C_{\text{pert}}(\mathbf{a})$	Perturbation cost (Section 3.2.3)
$C^{(k)}$	Instantaneous cost (Section 3.2.3)
$\mathcal{S}$	State space (Section 3.3.1)
$\mathcal{A}_{\boldsymbol{\theta}}$	Action space (Section 3.3.1)
$\alpha^{(k)}$	Learning rate at time slot k (3.12)
$\gamma^{(k)}$	Discount factor at time slot k (3.12)
$\epsilon^{(k)}$	Epsilon at time slot k (Section 3.3.3)
$\hat{C}_{\text{nom},p}(\theta_p)$	Model of nominal cost for SP p (Section 3.3.2)
$\hat{C}_{\text{nom}}(\boldsymbol{\theta})$	Model of total nominal cost (Section 3.3.2)
$\mathcal{M}^{(k)}$	Memory at time slot k (Section 3.3.3)
N_{mem}	Mini-batch size for memory (Section 3.3.3)
N_{model}	Mini-batch size for model (Section 3.3.2)

Table 3.1: Frequently used notations in Chapter 3

3.2 System Model

We consider a system that consists of a NO that owns cache space K at an edge node, for instance at the base station, and of P SPs providing services such as video streaming service. Time is slotted. The NO can share its cache space among the P SPs, and we denote by $\theta_p^{(k)}$ the cache space allocated to SP p in time slot k. We consider that cache space is an integer K and each slot can store one object. We denote by $\boldsymbol{\theta}^{(k)} = (\theta_1^{(k)},...,\theta_P^{(k)})$ the allocation at time slot k and we define the set of feasible allocations

$$\mathcal{T} \triangleq \left\{ \boldsymbol{\theta} | \sum_{p=1}^{P} \theta_p \leq K, \theta_p \in \mathbb{Z}^+ \right\} \tag{3.1}$$

Table 3.1 summarizes the most frequently used notations in this chapter.

3.2.1 Request pattern

We consider that each SP p has a catalog of N_p cacheable objects. We use the tuple (c, p), $c = 1, 2, \ldots, N_p$ to refer to object c of SP p. Requests for objects arrive with rate λ. We denote by f_p the probability that a given request is for an object offered by SP p, and hence the request arrival rate for objects of SP p is $\lambda \cdot f_p$. To capture the fact that not all objects of an SP may be cacheable (e.g., live streams and broadcasts), we denote by ζ_p the probability that a request to SP p is for a cacheable object, and we refer to this as its *cacheability*. For a cacheable object (c, p), we denote by $\rho_{c,p}$ its popularity, i.e., the probability that, among the requests for all cacheable objects of SP p, the request is for object c. Under this model, a cacheable object (c, p) receives requests at rate $\lambda_{c,p} = \lambda \cdot f_p \cdot \zeta_p \cdot \rho_{c,p}$.

We adopt the common assumption in the literature that all objects have the same size [57, 28, 52]. Objects may represent, for instance, chunks of videos. We consider that the arrival process is stationary. In practice, object popularity and request rate change smoothly over time, and as we will show in Section 3.6.2 our algorithm can converge fast enough (15 minutes, see Figure 3.6) to be able to consider the arrival process to be stationary.

3.2.2 Cost model

We consider the cost of the NO due to upstream traffic, which could be incurred for two reasons. First, for a given cache partitioning $\boldsymbol{\theta}$, the cache misses cause upstream traffic: if a request arrives, which is for an object that is not cached, such an object must be downloaded from the Internet. We call the resulting cost the *nominal cost*. Observe that for any allocation $\boldsymbol{\theta}$, the nominal cost is a random variable parameterized by parameters that are unknown/not observable by the NO, i.e., the requests of users for video objects

[3] https://github.com/Ressource-Allocation/Cache-Allocation-Project

that could changes over time. We denote these unknown parameters, which represent exogenous conditions that are not under the control of the NO, by ω. We express this dependence, for any $\boldsymbol{\theta}$, by using the notation $C_{\text{nom},p}(\boldsymbol{\theta}, \omega)$ for the nominal cost due to SP p. We denote the total nominal cost due to all SPs by $C_{\text{nom}}(\boldsymbol{\theta}, \omega)$.

$$\begin{aligned} C_{\text{nom}} :& \mathbb{R}^P \times \mathcal{X} \to \mathbb{R} \\ & (\boldsymbol{\theta}, \omega) \to C_{\text{nom}}(\boldsymbol{\theta}, \omega) \end{aligned}$$

where $\mathcal{X}$ is any topological space.

The total nominal cost due to all SPs, at time slot k, is

$$C_{\text{nom}}(\boldsymbol{\theta}^{(k)}, \omega) = \sum_{p=1}^{P} C_{\text{nom},p}(\boldsymbol{\theta}^{(k)}, \omega) \tag{3.2}$$

The second source of upstream traffic is due to changing the cache partitioning between the SPs. We refer to the corresponding cost as *perturbation cost*. To express the perturbation cost, let us denote by $\mathbf{a}^{(k)} = \boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}^{(k)}$ the *perturbation vector*, i.e., the change in cache partitioning of the NO at time slot k. If $\theta_p^{(k+1)} > \theta_p^{(k)}$ then SP p will download $\theta_p^{(k+1)} - \theta_p^{(k)} > 0$ objects from the Internet to its allocated storage. We can thus express the perturbation cost as:

$$C_{\text{pert}}(\mathbf{a}^{(k)}) = \sum_{p=1}^{P} [\theta_p^{(k+1)} - \theta_p^{(k)}]^+, \tag{3.2bis}$$

where $[.]^+$ denotes the $\max(., 0)$. The *instantaneous cost* $C^{(k)}$ at time slot k is then

$$C^{(k)} \triangleq C_{\text{nom}}(\boldsymbol{\theta}^{(k)}, \omega) + C_{\text{pert}}(\mathbf{a}^{(k)}), \tag{3.3}$$

and the *cumulative cost* over Z time slots as:

$$C_{\text{cum}}(Z) = \sum_{k=1}^{Z} C^{(k)} \tag{3.4}$$

3.2.3 Problem formulation

Since for any $\boldsymbol{\theta}$, the total nominal cost $C_{\text{nom}}(\boldsymbol{\theta}, \omega)$ is a random variable, whose randomness comes from ω. The NO aims to minimize its expected value:

$$\boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta} \in \mathcal{T}} \mathbb{E}_\omega \big[C_{\text{nom}}(\boldsymbol{\theta}, \omega) \big] \tag{3.5}$$

We emphasize that $\mathbb{E}_\omega\big[C_{\text{nom}}(\boldsymbol{\theta},\omega)\big]$ is never observable directly, only a realization $C_{\text{nom}}(\boldsymbol{\theta},\omega)$ is. The latter can be considered as a noisy observation of $\mathbb{E}_\omega\big[C_{\text{nom}}(\boldsymbol{\theta},\omega)\big]$, i.e., $\forall\boldsymbol{\theta}\in\mathcal{T}$

$$C_{\text{nom}}(\boldsymbol{\theta},\omega)=\mathbb{E}_\omega\big[C_{\text{nom}}(\boldsymbol{\theta},\omega)\big]+\eta, \tag{3.6}$$

where η is a random variable.

The problem (3.5) is a contextual bandit problem, where ω is the context. Algorithms for solving contextual bandit problems rely on experimenting with different cache allocations, and hence they will involve perturbation cost that they do not take into account. Hence it is more reasonable to consider the following optimization problem:

$$\pi^*=\arg\min_{\pi\in\Pi}\lim_{Z\to\infty}\frac{1}{Z}\sum_{k=1}^{Z}\mathbb{E}[C^{(k)}] \tag{3.7}$$

where Π is is the set of causal allocation policies, i.e., policies that are designed to make decisions based on a causal relationships between actions, states of the environment, and the resulting rewards.

An allocation policy π is a function $\pi(\mathbf{a}|\boldsymbol{\theta})$ defining the decisions of the NO: whenever the NO observes state $\boldsymbol{\theta}$, it will choose an action $\mathbf{a}$ with probability $\pi(\mathbf{a}|\boldsymbol{\theta})$. During training, the NO starts with a certain policy $\pi^{(0)}(\cdot)$ and then adjusts it, based on the measured cost, in order to approach the optimal policy π^*.

Note that, despite the fact that the spurious traffic generated by perturbations adds to the cost, perturbations are the only way for the NO to discover how to optimize the "black-box" function $\boldsymbol{\theta}\to\mathbb{E}_\omega\big[C_{\text{nom}}(\boldsymbol{\theta},\omega)\big]$. Indeed, by observing the effects of perturbations on the nominal cost, the NO can accumulate knowledge that can be used to drive the system close to the optimal allocation $\boldsymbol{\theta}^*$. Therefore, in our data-driven approach, rather than directly solving (3.5), which would be infeasible for the reasons stated above, our aim is to find a sequence of perturbations $\{\mathbf{a}^{(k)}\}$ in order to minimize the expected mean cumulative cost (3.4).

Observe that the NO problem is a sequential decision making problem under uncertainty, where the uncertainty is due to the randomness of the users requests. In what follows we propose an allocation policy based on RL. We will show that, by applying it, we converge close to the optimal allocation (3.5). Note that, for any initial allocation $\boldsymbol{\theta}^{(0)}$, the sequence $\{\mathbf{a}^{(k)}\}$ deterministically induces a sequence of states $\{\boldsymbol{\theta}^{(k)}\}$:

$$\boldsymbol{\theta}^{(k+1)}=\boldsymbol{\theta}^{(k)}+\mathbf{a}^{(k)} \tag{3.8}$$

We refer to $\{\boldsymbol{\theta}^{(k)},\mathbf{a}^{(k)}\}$ as a *state-action sequence*. Adopting the standard terminology from the literature [100, Section 4.1], the observations of the NO are based on a *bandit feedback model*, in that at every time-slot k the NO observes only the cost (3.3) of the state-action pair visited in that time-slot and not the others.

3.3 Data-driven Optimization

3.3.1 MDP formulation

Our cache allocation problem can be formulated as a deterministic MDP. The set of **states** $\mathcal{S}$ consists of all the allocation vectors that we can visit. To reduce the complexity of the problem, we adopt a discretization step $\Delta \in \mathbb{N}$, i.e., the amount by which we change the allocation, and define $\mathcal{S}$ as:

$$\mathcal{S} = \left\{ \boldsymbol{\theta} = (\theta_1, \ldots, \theta_P) | \sum_{p=1}^{P} \theta_p \leq K, \theta_p \text{ multiple of } \Delta \right\} \tag{3.9}$$

The discretization step Δ constitutes a precision/complexity trade-off. A smaller value of Δ increases the precision of the allocation since it allows converging to a discrete solution closer to the optimal one (Section 3.6.1); it however increases the complexity of the problem since it expands the space of states. Observe that $\mathcal{S} \subset \mathcal{T}$ (3.1). When in state $\boldsymbol{\theta}$, the NO can pick an action from the following **action space**:

$$\mathcal{A}_{\boldsymbol{\theta}} = \left\{ \mathbf{a} = \Delta \cdot (\mathbf{e}_p - \mathbf{e}_{p'}) | \boldsymbol{\theta} + \mathbf{a} \in \mathcal{S}, p, p' = 1, \ldots, P \right\} \tag{3.10}$$

where $\mathbf{e}_p$ is the p-th element of the standard basis of $\mathbb{R}^p$.

We will use the terms allocation/state and action/perturbation interchangeably. Therefore, an action $\mathbf{a}$ of the NO consists in adding Δ units of storage to a certain SP p and removing the same amount from another SP p'. The null action corresponds to not changing the allocation (which happens in (3.10) when $p = p'$). Thanks to (3.8), the transition from a state to another is deterministic.

Our objective function accounts for both nominal cost as well as perturbation cost and is given by:

$$C^{\gamma}_{\text{cum}} = \lim_{Z \to \infty} \mathbb{E} \left[\sum_{k=0}^{Z} \gamma^{(k)} \cdot \underbrace{\left(C_{\text{nom}}(\boldsymbol{\theta}^{(k)}, \omega) + C_{\text{pert}}(\mathbf{a}^{(k)}) \right)}_{\text{Instantaneous cost } C^{(k)}} \right]_{\substack{\boldsymbol{\theta}^{(k)} \in \mathcal{S} \\ \mathbf{a}^{(k)} \in \mathcal{A}}} \tag{3.11}$$

where $0 < \gamma < 1$ is a hyper-parameter called *discount factor*.

3.3.2 Online model-based Reinforcement Learning

To properly characterize MB-RL, we first need individual definitions of *planning* and *Reinforcement Learning*. We can distinguish them based on the assumptions related to the

knowledge about the system: planning methods assume that a *model* of the system (nominal cost $\mathbb{E}_\omega\big[C_{\text{nom}}(\boldsymbol{\theta},\omega)\big]$ vs. allocation $\boldsymbol{\theta}$, in our case) is available (a-priori or via learning), which allows the agent to repeatedly plan forward from any state $\boldsymbol{\theta}$ using the model. In contrast, for RL in its simplest form (i.e., MF-RL) this model is not present, so the agent can have information about the instantaneous cost of a certain allocation, only by physically visiting it. *MB-RL* [101, Section 8] combines both approaches. In MB-RL, "learning" happens at two locations in the decision algorithm: 1) to learn the model of the system, and 2) to learn the policy telling us which actions to take from any state.

Since we do not know the form of the function $\boldsymbol{\theta} \to \mathbb{E}_\omega\big[C_{\text{nom}}(\boldsymbol{\theta},\omega)\big]$, the first step of model-based RL involves learning it from observed data. For that, we construct for each SP p a model $\hat{C}^{(k)}_{\text{nom,p}}(\theta_p)$ that will approximate the expected nominal cost $\mathbb{E}_\omega\big[C_{\text{nom,p}}(\theta_p,\omega)\big]$ for any θ_p, at time slot k. We thus obtain a model $\hat{C}^{(k)}_{\text{nom}}(\boldsymbol{\theta}) \triangleq \sum_{p=1}^{P} \hat{C}^{(k)}_{\text{nom,p}}(\theta_p)$ that approximates the nominal cost $\mathbb{E}_\omega\big[C_{\text{nom}}(\boldsymbol{\theta},\omega)\big]$, at time slot k. Note that the estimation model $\boldsymbol{\theta} \to \hat{C}^{(k)}_{\text{nom}}(\boldsymbol{\theta}), \forall \boldsymbol{\theta} \in \mathcal{S}$, is updated at each time slot k.

At each time slot k, we perform the Bellman Optimality Equation update:

$$Q^{(k)}(\boldsymbol{\theta}^{(k)},\mathbf{a}^{(k)}) = (1-\alpha^{(k)}) \cdot Q(\boldsymbol{\theta}^{(k)},\mathbf{a}^{(k)}) + \alpha^{(k)} \cdot \left(C^{(k)} + \gamma^{(k)} \min_{\mathbf{a} \in \mathcal{A}_{\boldsymbol{\theta}^{(k+1)}}} Q^{(k)}(\boldsymbol{\theta}^{(k+1)},\mathbf{a}) \right) \tag{3.12}$$

Then, we take the last measured value $C_{\text{nom,p}}(\theta_p^{(k)},\omega)$ and do the following:

- We construct model $\hat{C}^{(k)}_{\text{nom}}(\boldsymbol{\theta})$.
- Then, we sample N_{model} random state-action pairs from the state space $\mathcal{S}$ and action space $\mathcal{A}$. Let us call $\big\{(\boldsymbol{\theta}^{[i]},\boldsymbol{a}^{[i]}), \boldsymbol{\theta}^{[i]} \in \mathcal{S}, \boldsymbol{a}^{[i]} \in \mathcal{A}_{\boldsymbol{\theta}^{[i]}}\big\}_{i=1,\dots,N_{\text{model}}}$ such samples. For each i-th sample, we predict a cost $\hat{C}^{[i]} = \hat{C}_{\text{nom}}(\boldsymbol{\theta}^{[i]}) + C_{\text{pert}}(\boldsymbol{a}^{[i]})$. Note that in this way we are able to make predictions on state-action pairs that have not been visited yet, exploiting "similar" pairs observed in the past. Note that $C_{\text{pert}}(\boldsymbol{a}^{[i]})$ is a deterministic constant (3.2bis) while $\hat{C}_{\text{nom}}(\boldsymbol{\theta}^{[i]})$ is an estimation obtained with our model.
- We finally perform N_{model} Q-table updates, as in (3.12), using $\hat{C}^{[i]}$ calculated above in place of $C^{(k)}$ of formula (3.12).

Algorithm 1 represents our approach to solve the considered cache allocation problem. In line 20 of Algorithm 1, we estimate a model of the nominal cost from the collected observations. Such a model is then used to perform additional updates of the Q-table. It will be required in the proof of Theorem 7.1.2.3 (Appendix 7) that this model has to be unbiased, in order to avoid driving Q-table updates in the wrong direction. A simple

empirical average of the observed nominal costs would be an unbiased model, but we need to collect many samples before empirical averages are sufficiently close to the expected value of the nominal cost. Moreover, if we only use empirical averages, at the beginning of the learning process, we would have a lot of allocations that have not yet been visited, and thus empirical averages would not be available. For this reason, in the first iterations we perform regression on the collected observations. By doing so, we can also estimate the nominal cost of the allocations not yet visited, by exploiting observations of similar allocations. This mechanism "accelerates" the updates of the Q-table at the beginning. To obtain unbiasedness, we then gradually abandon the regression model and we adopt the empirical averages.

We now formally describe how we estimate the cost model in Line 20 of Algorithm 1. Up to a certain time slot K_{reg}, model $\hat{C}^{(k)}_{\text{nom},p}(\theta_p)$ is obtained by regression using $\mathcal{D}_p$ as dataset. Let us denote $\hat{C}^{(k)}_{\text{nom},p,reg}(\theta_p)$ the model obtained by regression. Then, we gradually replace $\hat{C}^{(k)}_{\text{nom},p,reg}(\theta_p)$ with the empirical mean:

$$\overline{C}^{(k)}_{\text{nom},p}(\theta_p) \triangleq \frac{1}{|\mathcal{K}^{(k)}_{p,\theta_p}|} \sum_{k' \in \mathcal{K}^{(k)}_{p,\theta_p}} C^{(k')}_{\text{nom},p}(\theta_p, \omega^{(k')}) \tag{3.13}$$

where $\mathcal{K}^{(k)}_{p,\theta_p}$ is the set of time-slots $k' \leq k$, in which SP p has been allocated θ_p slots.

We define our model at time slot k for any $\theta_p \in [0, K]$ as:

$$\hat{C}^{(k)}_{\text{nom},p}(\theta_p) \triangleq \begin{cases} \hat{C}^{(k)}_{\text{nom},p,\text{reg}}(\theta_p) & \text{if } k \leq K_{\text{reg}} \\ & \text{or } \mathcal{K}^{(k)}_{p,\theta_p} = \emptyset \\ \frac{1}{|\mathcal{K}^{(k)}_{p,\theta_p}|} \hat{C}^{(k)}_{\text{nom},p,\text{reg}}(\theta_p) & \\ + \left(1 - \frac{1}{|\mathcal{K}^{(k)}_{p,\theta_p}|}\right) \bar{C}^{(k)}_{\text{nom},p}(\theta_p) & \text{otherwise} \end{cases} \tag{3.14}$$

Similar to (3.2), the cost estimated by the model is $\hat{C}^{(k)}_{\text{nom}}(\boldsymbol{\theta}) \triangleq \sum_{p=1}^{p} \hat{C}^{(k)}_{\text{nom},p}(\theta_p)$ and the empirical mean of the nominal cost measured is $\bar{C}^{(k)}_{\text{nom}}(\boldsymbol{\theta}) \triangleq \sum_{p=1}^{p} \bar{C}^{(k)}_{\text{nom},p}(\theta_p)$.

The following theorem ensures that the model (3.14) is an unbiased estimator of the nominal cost.

Theorem 3.3.2.1. *For any SP p and allocated cache slots θ_p our model converges uniformly to the expected value of the nominal cost, i.e.,*

$$\lim_{k \to \infty}^{u} \hat{C}^{(k)}_{nom}(\cdot) = \mathbb{E}_{\omega}\big[C_{nom}(\cdot, \omega)\big] \tag{3.15}$$

where symbol $\lim^{u}$ *indicates that*

$$\lim_{k \to \infty} \|\hat{C}^{(k)}_{nom}(\cdot) - \mathbb{E}_{\omega}\big[C_{nom}(\cdot, \omega)\big]\|_{\infty} = 0 \tag{3.16}$$

Proof. Let us consider a SP p and a number θ_p of allocated slots and prove point-wise convergence first. As k goes to ∞, the number of times SP p has been allocated θ_p increases, i.e., $\lim_{k\to\infty} |\mathcal{K}^{(k)}_{p,\theta_p}| = \infty$. For $k > K_{\text{reg}}$,

$$\hat{C}^{(k)}_{\text{nom},p}(\theta_p) = \frac{1}{|\mathcal{K}^{(k)}_{p,\theta_p}|}\hat{C}^{(k)}_{\text{nom},p,\text{reg}}(\theta_p) + \left(1 - \frac{1}{|\mathcal{K}^{(k)}_{p,\theta_p}|}\right)\bar{C}^{(k)}_{\text{nom},p}(\theta_p)$$
$$\xrightarrow{k\to\infty} \bar{C}^{(k)}_{\text{nom},p}(\theta_p)$$

Using the law of large numbers (Definition 7.1 of [102]), we can write for any SP p, $\forall\theta_p$

$$\lim_{k\to\infty} \overline{C}^{(k)}_{\text{nom},p}(\theta_p) = \mathbb{E}_\omega\big[C_{\text{nom},p}(\theta_p,\omega)\big]. \tag{3.17}$$

Hence,

$$\lim_{k\to\infty} \hat{C}^{(k)}_{\text{nom},p}(\theta_p) = \mathbb{E}_\omega\big[C_{\text{nom},p}(\theta_p,\omega)\big],$$

which easily implies that

$$\lim_{k\to\infty} \hat{C}^{(k)}_{\text{nom}}(\boldsymbol{\theta}) = \mathbb{E}_\omega\big[C_{\text{nom}}(\boldsymbol{\theta},\omega)\big], \forall\boldsymbol{\theta}\in\mathcal{S}.$$

Since $\mathcal{S}$ is a finite set, point-wise convergence implies [103, Proposition 1] uniform convergence. □

3.3.3 Additional enhancements

We now report some enhancements that considerably improve the performance of our algorithm.

Learning rate scheduling

The hyper-parameter $\alpha^{(k)}$ in (3.12) tells the magnitude of step that is taken towards the solution. $\alpha^{(k)}$ should not be too big a number as it may continuously oscillate around the minima and it should not be too small of a number else it will take a lot of time and iterations to reach the minima. As in [28], we decrease it slowly, because initially when we are at a totally random point in solution space we need to take big leaps towards the solution and later when we come close to it, we make small jumps and hence small improvements to finally reach the minima.

$$\alpha^{(k)} = \alpha^{(k-1)} \cdot \left(1 - \frac{1}{1+M+k}\right)^{\frac{1}{2}+\xi} \tag{3.18}$$

where M and ξ are positive constants, used to tune the slope of decrease.

Experience replay

In the simplest implementation of Q-learning, the measurement made in a certain time-slot is used to update the Q-table in that time-slot only and is never used again. However, the set of previous measurements (i.e., the past "experience") could be further exploited to improve the Q-table update in future time-slots. To this aim, Experience Replay has been proposed [104]. At any time-slot k, in addition to using the measured instantaneous cost $C^{(k)}$ to update the Q-table in (3.12), we also store this measurement in the form of a triplet $\left(\boldsymbol{\theta}^{(k)}, \mathbf{a}^{(k)}, C^{(k)}\right)$, which we call *experience*. The set of experiences accumulated in this way is called *memory*. Formally, let us define the memory as:

$$\mathcal{M}^{(k)} = \{(\boldsymbol{\theta}^{(0)}, \mathbf{a}^{(0)}, C_0), \ldots, (\boldsymbol{\theta}^{(k)}, \mathbf{a}^{(k)}, C^{(k)})\}$$

Whenever we update the Q-table, additionally to performing (3.12) using the current observation, we also sample the memory randomly for a mini-batch of experiences of size N_{mem} and we use these random samples as entries for the Q-table by applying (3.12).

ϵ stretched exponential decay

The value of $\epsilon^{(k)}$ is the probability of taking a random action (exploration) instead of the best so far, at any time-slot k. We impose, motivated by [105], the following decay:

$$\epsilon^{(k)} = \begin{cases} \epsilon_0 - \left[\frac{0.9 \cdot \epsilon_0}{\cosh(e^{-\frac{k - A \cdot Z}{B \cdot Z}})} + \frac{k \cdot C}{Z}\right] & \text{if } k \leq Z \\ \frac{\epsilon^{(Z)}}{k - Z} & \text{otherwise} \end{cases} \tag{3.19}$$

where ϵ_0 is the initial value of ϵ, A, B and C are hyper parameters and Z is a time horizon, after which the behavior of the decrease change. Indeed, bigger value of Z (longer horizon) implies that the value of ϵ goes smaller in the exploitation phase. This decay provides:

- sufficient time for exploration at the beginning.
- preference to exploitation (with respect to exploration) in the end (quasi-deterministic policy).
- smooth transition while switching from exploration to exploitation.

The parameter A decides whether to spend more time on Exploration or on Exploitation. For values of A below 0.5, RL would be spending less time exploring and more time exploiting. For values of A above 0.5, we force RL to explore more. The parameter B decides the slope of transition region from Exploration to Exploitation. Parameter C controls the steepness of left (exploration) and right (exploitation) tails of the curve of the $\epsilon^{(k)}$ evolution. The higher the value of C, the steeper are the left and right tails of the curve.

Algorithm 1 describes how we combine model-based RL with the enhancements cited above.

Algorithm 1: k-th step of Model-based RL

$\alpha^{(k)} \leftarrow$ calculate the value of α via formula (3.18);

$\epsilon^{(k)} \leftarrow$ calculate the value of ϵ via formula (3.19);

with probability $\epsilon^{(k)}$: $\mathbf{a}^{(k)} \leftarrow$ random action ; // ϵ-greedy policy

with probability $1-\epsilon^{(k)}$: $\mathbf{a}^{(k)} \leftarrow$ best action from $Q^{(k)}(\boldsymbol{\theta}^{(k)}, \mathbf{a})$;

$\boldsymbol{\theta}^{(k+1)} \leftarrow \boldsymbol{\theta}^{(k)} + \mathbf{a}^{(k)}$;

$C^{(k)} \leftarrow C_{\text{nom}}(\boldsymbol{\theta}^{(k)}, \omega) + C_{\text{pert}}(\mathbf{a}^{(k)})$;

$Q^{(k)}(\boldsymbol{\theta}^{(k)}, \mathbf{a}^{(k)}) \leftarrow$
$(1-\alpha^{(k)}) \cdot Q^{(k)}(\boldsymbol{\theta}^{(k)}, \mathbf{a}^{(k)}) + \alpha^{(k)} \cdot \left(C^{(k)} + \gamma \min_{\mathbf{a} \in \mathcal{A}_{\boldsymbol{\theta}^{(k+1)}}} Q^{(k)}(\boldsymbol{\theta}^{(k+1)}, \mathbf{a}) \right)$;
// update $Q^{(k)}$

////////////////

/// Memory replay

$\mathcal{M}^{(k)} \leftarrow \mathcal{M}^{(k-1)} \cup \{(\boldsymbol{\theta}^{(k)}, \mathbf{a}^{(k)}, C_{\text{nom}}(\boldsymbol{\theta}^{(k)}))\}$;

for N_{mem} *times ;* // Simulate state-action pairs from the memory

do

  $(\boldsymbol{\theta}^{\text{rd}}, \mathbf{a}^{\text{rd}}, C^{\text{rd}}_{\text{nom}}) \leftarrow$ random element from $\mathcal{M}^{(k)}$;

  $\boldsymbol{\theta}'^{\text{rd}} \leftarrow \boldsymbol{\theta}^{\text{rd}} + \mathbf{a}^{\text{rd}}$;

  $Q^{(k)}(\boldsymbol{\theta}^{\text{rd}}, \mathbf{a}^{\text{rd}}) \leftarrow$
  $(1-\alpha^{(k)}) \cdot Q^{(k)}(\boldsymbol{\theta}^{\text{rd}}, \mathbf{a}^{\text{rd}}) + \alpha^{(k)} \cdot \left(C^{\text{rd}}_{\text{nom}} + C_{\text{pert}}(\mathbf{a}^{\text{rd}}) + \gamma \min_{\mathbf{a} \in \mathcal{A}_{\boldsymbol{\theta}'^{\text{rd}}}} Q^{(k)}(\boldsymbol{\theta}'^{\text{rd}}, \mathbf{a}) \right)$
  ; // update $Q^{(k)}$

end

////////////////

/// Model training and inference

$\mathcal{D}_p^{(k)} \leftarrow \mathcal{D}_p^{(k-1)} \cup \{(\theta_p^{(k)}, C_{\text{nom,p}}(\theta_p^{(k)})\}, \forall$ SP p; // collect realization of $C_{\text{nom,p}}(\theta_p^{(k)})$

$\hat{C}^{(k)}_{\text{nom},p}(\theta_p) \leftarrow$ estimate model from $\mathcal{D}_p^{(k)}$;

for N_{model} *times ;* // Simulate state-action pairs taken randomly from state space and action space

do

  $\boldsymbol{\theta}^{\text{rd}} \leftarrow$ random state from $\mathcal{S}$;

  $\mathbf{a}^{\text{rd}} \leftarrow$ random action from $\mathcal{A}_{\boldsymbol{\theta}^{\text{rd}}}$;

  $\boldsymbol{\theta}'^{\text{rd}} \leftarrow \boldsymbol{\theta}^{\text{rd}} + \mathbf{a}^{\text{rd}}$;

  Compute $\hat{C}^{(k)}_{\text{nom,p}}(\theta_p^{\text{rd}}), \forall$ SP p ; // predict the nominal cost using the model

  $\hat{C} \leftarrow \sum_{\text{p}=1}^{P} \hat{C}_{\text{nom,p}}(\theta_p^{\text{rd}})) + C_{\text{pert}}(\mathbf{a}^{\text{rd}})$;

  $Q^{(k)}(\boldsymbol{\theta}^{\text{rd}}, \mathbf{a}^{\text{rd}}) \leftarrow (1-\alpha^{(k)}) \cdot Q^{(k)}(\boldsymbol{\theta}^{\text{rd}}, \mathbf{a}^{\text{rd}}) + \alpha^{(k)} \cdot \left(\hat{C} + \gamma \min_{\mathbf{a} \in \mathcal{A}_{\boldsymbol{\theta}'^{\text{rd}}}} Q^{(k)}(\boldsymbol{\theta}'^{\text{rd}}, \mathbf{a}) \right)$
  ; // update $Q^{(k)}$ based on fictitious transitions

end

3.4 Theoretical Properties

In this section we will present the memory complexity of our algorithm (Section 3.4.1) and then we will prove that the system will converge to a discretely optimal state and will stay in that state with probability 1 as time goes to infinity (Section 3.4.2).

3.4.1 Memory complexity

Definition 3.9 of state space and definition 3.10 of action space imply that the size of the state space $\mathcal{S}$ is $O((K/\Delta)^P)$ and of the action space $\mathcal{A}$ is $O(P^2)$. Therefore, the memory required to store the Q-table is $O((K/\Delta)^P \cdot P^2)$. Increasing the number of SPs P will increase the complexity of the system in terms of state space size and action space size. The computational complexity will also increase, but not proportionally. In fact, the convergence speed is less affected by the sizes of the state and action spaces because the model will allow the agent to update the policy based on fictional allocations that are not really visited in the real environment. In what follows, we assume that the number of SPs getting a cache slice is limited (3 or 4). The dependence on P is a limited issue under this assumption. Observe that we can counter-fight the linear increase of memory complexity with respect to the cache size K by increasing the elementary allocation Δ.

3.4.2 Convergence

Since we are using a model that approximates the expected nominal cost $\mathbb{E}_\omega\big[C_{\text{nom}}(\boldsymbol{\theta}, \omega)\big]$ (Section 3.3.2) alongside Q-Learning, with other enhancements used, mentioned in Section 3.3.3, we cannot simply rely on the property of convergence of classical Q-learning (Theorem 2 of [106] or [107]) to justify the convergence of our algorithm. The following theorem characterizes the main property of our algorithm: we provide in this section a sketch of the proof, and we present the full proof in a separate section, due to its length (see Appendix 7). In our algorithm we make use of a discretization constant Δ, which may be larger than 1. The latter limits the state space $\mathcal{S}$ (3.9) and consequently prevents us from reaching the optimal allocation $\boldsymbol{\theta}^*$ defined by (3.5). We can thus only reach

$$\hat{\boldsymbol{\theta}}^* \in \arg\min_{\boldsymbol{\theta} \in \mathcal{S}} \mathbb{E}_\omega\big[C_{\text{nom}}(\boldsymbol{\theta}, \omega)\big], \tag{3.20}$$

which we call *discretely optimal allocation*.

The following theorem, proved in Appendix 7, shows that our allocation converges to the discretely optimal one.

Theorem 3.4.2.1. *If the discount factor γ is sufficiently close to 1*

$$\lim_{k \to \infty} \boldsymbol{\theta}^{(k)} = \hat{\boldsymbol{\theta}}^* \text{ with probability } 1.$$

Sketch of the proof. The main steps to prove the above theorem are:

- We prove that our Q-table $Q^{(k)}$, resulting from Algorithm 1, converges to the optimal Q-table Q^* with probability 1 (see Section 7.1.2).
- We prove that the sequence of actions and states induced by Q^* has an absorbing state that is the discretely optimal state $\hat{\boldsymbol{\theta}}^*$ (see Section 7.1.3).
- We prove that the sequence of states and actions induced by our Q-table $Q^{(k)}$ (assuming no more exploration) also follows Q^* (see Section 7.1.4).
- We prove that the sequence of states and actions $\{\boldsymbol{\theta}^{(k)}, \mathbf{a}^{(k)}\}$ that we take online, based on Algorithm 1, converges with probability 1 to the sequence induced by our Q-table $Q^{(k)}$ (assuming no more exploration)(see Section 7.1.5).
- Finally, we show that this sequence $\{\boldsymbol{\theta}^{(k)}, \mathbf{a}^{(k)}\}$, taken online based on Algorithm 1, converges with probability 1 to a sequence induced by Q^*.

□

We now establish a theoretical bound to describe how fast or accurate our algorithm converge to approach the optimal allocation. For this, let us define the *discretization gap* G_Δ as the loss induced by the fact that we do not allocate cache slot by slot, but only in multiples of the discretization step Δ:

$$G_\Delta = \mathbb{E}_\omega\big[C_{\text{nom}}(\hat{\boldsymbol{\theta}}^*, \omega)\big] - \mathbb{E}_\omega\big[C_{\text{nom}}(\boldsymbol{\theta}^*, \omega)\big] \tag{3.21}$$

Intuitively, the larger the discretization step Δ, the larger the discretization gap G_Δ. A consequence of Theorem 3.4.2.1 is the following:

Corollary 3.4.2.2. *Let us denote by $C_{cum}(Z)$ and $C^*_{cum}(Z)$ the cumulative cost* (3.4) *obtained with our algorithm and with the optimal theoretical allocation $\boldsymbol{\theta}^*$, respectively. The difference between the two converges in expectation to the discretization gap:*

$$\lim_{Z\to\infty} \frac{1}{Z}\mathbb{E}\left[C_{cum}(Z) - C^*_{cum}(Z)\right] = G_\Delta$$

Therefore, a trade-off emerges: one the one hand, discretization allows us to reduce the memory complexity, i.e., state space and action space (Section 3.4.1), on the other hand it keeps allocations at a finite distance from the theoretical optimal cost. We will however see in the numerical results (Section 3.6) that this distance is in practice very small.

The following bound shows this dependence between Δ and G_Δ.

Proposition 3.4.2.3. *Suppose objects of SP p are ordered from the most to the least popular. Hence,*

$$G_\Delta \leq \sum_{p=1}^{P}\sum_{c=1}^{\Delta} \lambda_{c,p}.$$

3.5 Discussion on the Use of RL

We now briefly discuss why we preferred our RL setting over other possible methodologies. First of all, we rule out all the static optimization techniques that require full information, due to the online and stochastic nature of the problem at hand.

SPSA has been applied in [108] for instance to optimize systems in a stochastic environment. However, the objective of such method is just to converge towards the optimum and no cost is considered for perturbation. Therefore, SPSA continuously perturbs the system along multiple directions. In real situations, perturbing a system can prevent it from working properly or engenders high cost (as in our problem). Our RL formulation allows to naturally include the cost of perturbation in the optimization problem and to consider the trade-off between converging fast toward the optimum and limiting the extent and frequencies of perturbations.

We could also interpret our allocation problem at hand as a "black-box optimization": we have an unknown system whose cost function is unknown ($C_{\text{nom}}(\boldsymbol{\theta}, \omega)$ in our case) and we aim to find the optimal solution $\boldsymbol{\theta}^*$. In such problems, Bayesian Optimization techniques [109] trade off exploration against exploitation to pick the states to visit, observe the cost at those states and decide the next state to visit, up to finding the optimum. However, such techniques suffer from the same limitation of SPSA: they are meant for offline problems, where the objective is to retrieve the minimum of the cost function *at the end of the optimization* and the cost of jumping from one state to another is not quantified nor directly minimized. Our RL framework not only allows us to reach an allocation close to the optimum at the end, but also implicitly optimizes the path of states visited *during* the optimization.

Lyapunov Optimization (LO) has also been used for allocation problems [97, 98]. However, these works assume that the expression of the cost or reward function ($C_{\text{nom}}(\boldsymbol{\theta}, \omega)$ in our case) is known, the only unknown information is the sequence of future realizations ω of the environment. We do not need to rely on such a strong assumption, as RL allows to optimize the system even if the expression of the cost function remains unknown. We indeed estimate such an expression online, based on observation (see Section 3.3.2).

The Markov Decision Process (MDP) underlying our RL method is a Deterministic Markov Decision Process (DMDP), as the transition from one state to another is deterministic in our case. In [110], transitions are unknown (although deterministic) and authors use model-free Q-learning to solve the MDP. However, we focus on an infinite-horizon deterministic control system with an approximation model for the reward (cost).

Our problem can be described as a Multi-Armed Bandit with Switching Cost (MAB-SC) [111, 112]. In that context, we would need to interpret each allocation vector as an arm and the perturbation cost as a switching cost. We indeed solve in our case this MAB-SC problem via model-based RL and as any randomized strategy, our algorithm is subject to the bounds derived in this context (Theorem 1 of [112]). We resort to model-based RL instead of other algorithms proposed in the MAB context, like EXP3 [113], as

we can naturally embed in our algorithm the model $\boldsymbol{\theta} \to \hat{C}_{\text{nom}}(\boldsymbol{\theta})$ of the nominal cost $\boldsymbol{\theta} \to \mathbb{E}_{\omega}\big[C_{\text{nom}}(\boldsymbol{\theta}, \omega)\big]$, inferred via simple regression, which would not be as immediate to do using other MAB algorithms. Observe that, as in [114], our problem is not a contextual MAB, as the actions determine the states.

We do not frame our problem as an adversarial MAB [115] either. Indeed, the focus of adversarial MAB is on worst-case analysis: performance bounds are provided under the assumptions that the environment (the parameter ω in our case) or even the actual cost function expression $C_{\text{nom}}(\boldsymbol{\theta}, \omega)$ are chosen by an adversary in order to "hurt" the decision maker (the NO in our case) and to increase the cost of the system as much as possible. In such a setting, [116, Theorem 3.1] shows that it is impossible to effectively optimize a DMDP such as ours. For our case, adversarial analysis is too pessimistic, as the worst case (in our case it would be for instance users generating a huge amount of requests only for unpopular non-cached objects) would have a negligible probability to occur. We instead adopt a stochastic setting, where the realizations ω of the environment are taken from an unknown probability distribution that is independent of the NO decisions and we study the "average" behavior of the system, i.e., the expected value of the cost and the probability to converge close to the optimum.

Online decision problems have been presented in an adversarial setting: relevant to our case is Smoothed Online Convex Optimization (SOCO) [117, 118], which aims to optimize the cost of a system by also taking into account the perturbation cost to jump from one state to another. However, such algorithms are studied in an adversarial setting, while we are in a stochastic setting, as mentioned above.

While we propose model-based RL by constructing a model that approximates the nominal cost (Section 3.3.2), one could wonder why not use Dynamic Programming (DP). We motivate our approach by the fact that model-based RL can be more sample-efficient than DP. In DP, we typically need to perform a complete backup of the value function for all states in each iteration, which can be computationally expensive and require a large number of samples. In contrast, model-based RL can use the learned model of the nominal cost function to plan and make decisions, often requiring fewer samples to achieve good performance. Furthermore, in this chapter, model-based RL is used in conjunction with model-free method, allowing for the exploitation of the strengths of both approaches. It uses the learned model for planning and exploration, while still using model-free updates for fine-tuning and policy improvement.

We have performed some performance analysis using R-learning instead of Q-learning. However, we observed worse achieved cost (which confirms previous findings from the literature [119]) and we omit the obtained results.

3.6 Numerical Results

We now evaluate the performance of our RL allocation $\boldsymbol{\theta}^{(k)}$ through simulations developed in Python and compare it to two static allocations, i.e.,

- the theoretical optimal allocation $\boldsymbol{\theta}^*$, which would ideally be computed by an oracle who knows exactly the content popularity and thus the expression of function $\boldsymbol{\theta} \rightarrow \mathbb{E}_\omega\big[C_{\text{nom}}(\boldsymbol{\theta}, \omega)\big]$.
- the proportional allocation $\boldsymbol{\theta}^{\text{prop}}$ where θ_p is proportional to the rate of requests λ_p directed to SP p.

We also compare it to two dynamic allocation strategies, namely

- model-free RL (with all the enhancements mentioned in Section 3.3.3).
- SPSA [28].

We consider a network with 3 SPs. We set the overall request arrival rate to $\lambda = 4000$ requests per second (in the same order of magnitude of requests supported in one edge location of Amazon CloudFront [35]). Each of these requests is directed to SP 1, 2 or 3 with probabilities 0.75, 0.25 and 0.05, respectively. We set the cacheability (Section 3.2.1) of SP_1, SP_2 and SP_3 to $\zeta_1 = 0.4$, $\zeta_2 = 0.9$ and $\zeta_3 = 0.9$, respectively. Each SP has a catalog of $N_1 = N_2 = N_3 = 10^7$ cacheable objects. Content popularity in each catalog follows Zipf's law with exponents $\beta_1 = 1.2$, $\beta_2 = 0.4$ and $\beta_3 = 0.2$, respectively (Figure 3.5). The total cache size is $K = 5 \cdot 10^6$. For the discretization step Δ, we found out that a good complexity vs. precision trade-off was to set it to $K/50$. To limit perturbations, we give a higher "weight" to the null action. Indeed, when we take a random action, we set the probability of choosing any non-null action to only $1/P^2$ and all the remaining probability is for the null-action. The simulation time is set to $Z = 6$ hours. The length of a time-slot is 0.25 second. We choose such value to have 1000 requests per time-slot. We found that this value is a good compromise since very large values can lead to overfitting and delayed learning and small values may lead to high uncertainty and slower learning.

We plot a normalized cost, i.e., the amount of objects downloaded from the Internet (either as a result of an edge cache miss or of an allocation perturbation) divided by the total amount of objects requested by the users. All curves are averaged with a sliding window of 10 min.

3.6.1 Hyper-parameters pre-tuning

We now discuss some preliminary tuning that we performed in experimentation not shown in this chapter on the features indicated in Section 3.3.3.

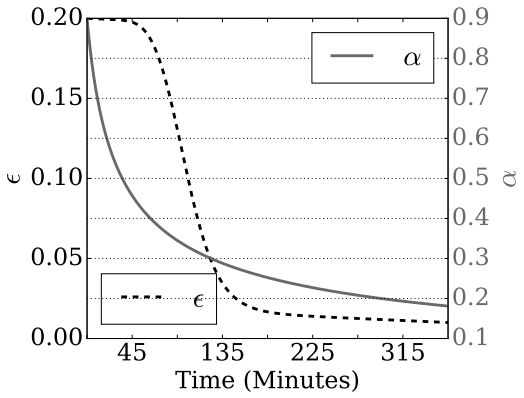


Figure 3.3: Evolution of ϵ and α vs. time

1. We set the discount factor γ to 0.99, i.e., very close to 1 to give importance to future rewards and prevent myopic decisions.

2. For α, the learning rate, we found that convergence was slow when it was fixed. Therefore, we adopt learning rate scheduling, which starts at 0.9 and decreases following (3.18) to 0.2, with $M = 3600$ and $\xi = 0.01$. The variation of α is illustrated in Figure 3.3.

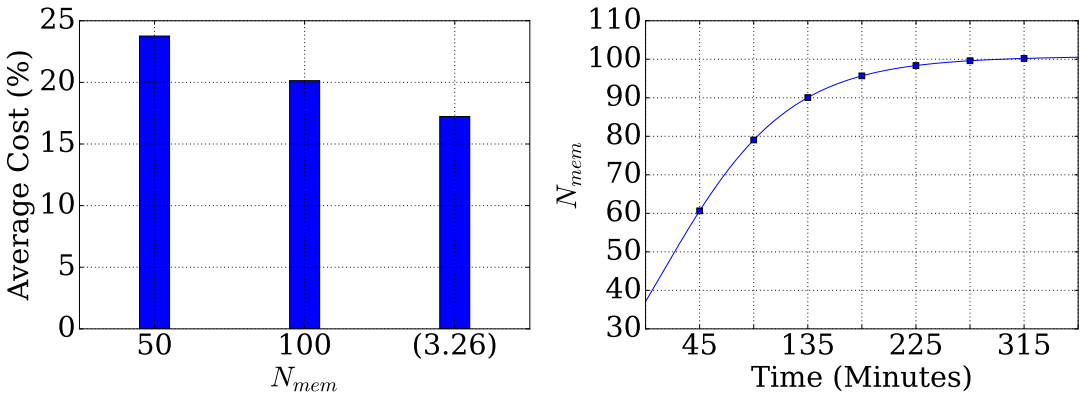
3. We make ϵ decay as in (3.19) with $A = 0.3$, $B = 0.1$ and $C = 0.01$. Z = 6 hours. These hyper-parameters have been chosen empirically after experimentation and provide a good compromise between exploration and exploitation. Figure 3.3 shows the decay of ϵ.

4. Regarding the size N_{mem} of the mini-batch of experiences, we found that small fixed values were not allowing to exploit past experience, on the other hand, with large values past experience was dominating too much the updates. We obtained the best performance by scheduling N as follows:

$$N_{\text{mem}}^{(k)} = \frac{N_{\max}}{\cosh(e^{-\frac{k-A\cdot Z}{B\cdot Z}})} + \frac{k \cdot C}{Z} \tag{3.22}$$

   where $N_{\max} = 100$, $A = 0.15$, $B = 0.3$, $C = 0.7$, $Z = 6$ hours. The choice of N_{mem} is illustrated in Figure 3.4.

5. For N_{model}, we take at each time-slot $N_{\text{model}} = 50$ samples on which we will apply the model $\hat{C}_{\text{nom}}(\boldsymbol{\theta})$.



(a) Average cost with various N_{mem} values

(b) Variation of N_{mem} over time

Figure 3.4: Choice of N_{mem}

3.6.2 Convergence close to the optimum

The behavior of our algorithm is well illustrated by Figure 3.6: Our MB-RL algorithm learns the system in only 15 minutes and converges to a cost close to the theoretical optimum. This rapid convergence is achieved thanks to the accurate model we obtain by regression (Figure 3.7) and that we use to estimate the nominal cost for any given allocation (even if that allocation is not visited). Although this chapter focuses on a stationary scenario, the results suggest that the fast convergence of our MB-RL algorithm could allow us to adapt even if the system parameters change over time. For instance, even if the request pattern is non stationary, our model can adapt to any new request pattern in 15 minutes. For model-free Reinforcement Learning (MF-RL), instead, we do not construct a model while learning the system behavior and we update the Q-table solely by perturbing the system via relatively many suboptimal actions in a first phase, in order to learn it. For this reason, perturbation cost is high up to 135 minutes. (see Figure 3.9a) After that, we start to exploit the collected knowledge and we limit perturbation.

Furthermore, our RL algorithm outperforms SPSA used in [28], which converges to the optimal allocation in 45 minutes but never stays at the optimal allocation due to the continuous perturbations it has to apply to estimate the sub-gradient of the objective function.

Note that if we apply "brute force" solution to find the optimal allocation $\boldsymbol{\theta}^*$, assuming optimistically that only 1 second is enough to compute the expected nominal cost for a given allocation, we would need

$$\left(\frac{K}{\Delta}\right)^P = \left(\frac{K}{K/50}\right)^P = \begin{cases} 50^3 \text{ seconds} \approx 34 \text{ hours, if } P = 3 \\ 50^4 \text{ seconds} \approx 1736 \text{ hours} \approx 2 \text{ months, if } P = 4 \end{cases}$$

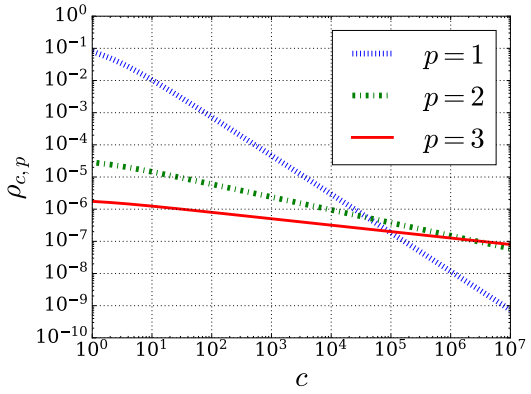


Figure 3.5: Zipf distribution of the different SPs

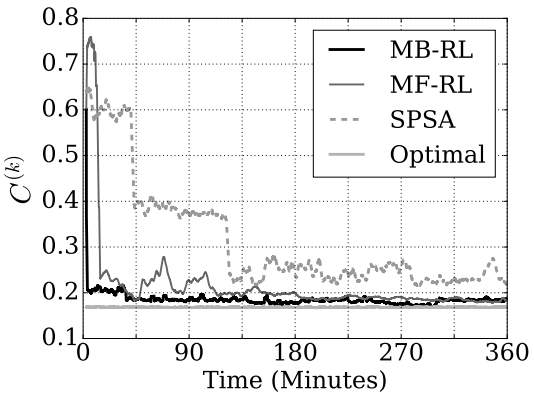


Figure 3.6: Evolution of the total instantaneous cost $C^{(k)}$

We now compare the cost $C^{(k)}$ induced by our policy with the cost of the static proportional allocation $\boldsymbol{\theta}^{\text{prop}}$ (proportional to the rate of requests directed to each SP). Note that while our method deals with both nominal and perturbation costs (3.3), the static $\boldsymbol{\theta}^{\text{prop}}$ does not apply any perturbation to the system. We define the gain of our policy with respect to $\boldsymbol{\theta}^{\text{prop}}$ as:

$$G^k_{\text{prop}} = \frac{C_{\text{nom}}(\boldsymbol{\theta}_{\text{prop}}, \omega) - C^{(k)}}{C_{\text{nom}}(\boldsymbol{\theta}_{\text{prop}}, \omega)} \tag{3.23}$$

Figure 3.8 shows that our solution reaches a gain of 60% in less than 45 minutes with respect to $\boldsymbol{\theta}^{\text{prop}}$.

To confirm our findings, we plot in Figure 3.9a the perturbation cost $C_{\text{pert}}(\mathbf{a}^{(k)})$ for the model-based and model-free versions of our algorithm and for SPSA. Results confirm that after 15 minutes, $C_{\text{pert}}(\mathbf{a}^{(k)})$ is practically null on an average 10 minute window for the model-based RL which means that we no longer drastically change the allocation. In the case of model-free RL, instead, we have to wait about 3 hours for the system to stabilize and the perturbations to be negligible. This validates two findings: (i) our MB-RL algorithm converges to a cache configuration and moves from there with a small probability, according to Theorem 3.4.2.1 and (ii) model-based RL converges 10 times faster than model-free RL. Observe that for SPSA, $C_{\text{pert}}(\mathbf{a}^{(k)})$ is always higher than in our RL algorithms (both model-free and model-based), which is expected since SPSA consists in continuously perturbing the allocation.

We map in Figure 3.9b each value of $C^{(k)}$ with the nature of the action taken at the time-slot k: each black point represents a random action which means the agent is exploring the environment. The large number of black points in the beginning confirms that the first phase is an exploration phase and it corresponds to the high value of ϵ, then the number of perturbations starts to decrease as the value of ϵ decreases in order to limit the exploration

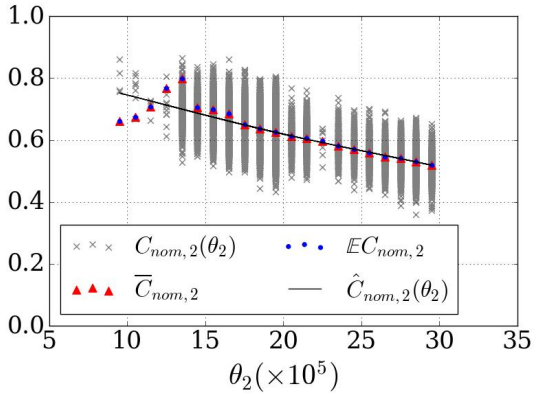


Figure 3.7: Model $\hat{C}_{\text{nom,p}}(\theta_p), p = 2$

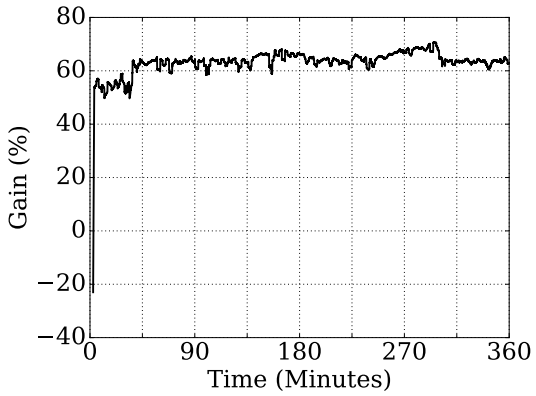
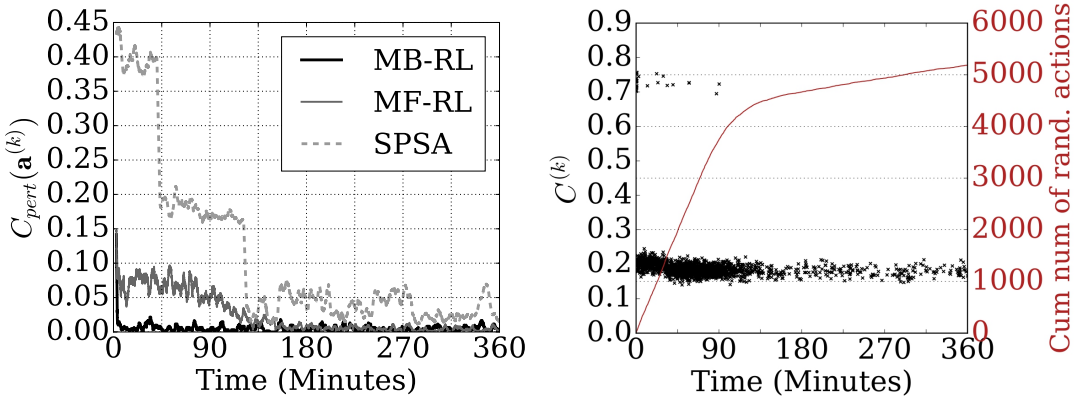


Figure 3.8: Gain of MB-RL with respect to proportional allocation $\boldsymbol{\theta}^{\text{prop}}$

as stated in Section 3.3.3. We plot in the same figure the cumulative number of random actions: we observe that it increases rapidly in the first 90 minutes and then starts to stabilize in the rest of the simulation.

We plot in Figure 3.10 the evolution of the allocation $\boldsymbol{\theta}^{(k)}$ of the MB-RL algorithm over time (center), the proportional allocation $\boldsymbol{\theta}^{\text{prop}}$ (left) and the optimal allocation $\boldsymbol{\theta}^*$ (right). Note that we start by $\boldsymbol{\theta}^0 = \boldsymbol{\theta}^{\text{prop}}$. The results show that we converge to an allocation $\hat{\boldsymbol{\theta}}^*$ close to $\boldsymbol{\theta}^*$ and we almost stay in this allocation, which matches our theoretical result stated in Theorem 3.4.2.1.

In Figure 3.10b, we plot, per each timeslot k, how many times the current state $\boldsymbol{\theta}^{(k)}$ has been visited in the past. We see that between 0 and 15 minutes a lot of states are visited few times, which corresponds to the exploration phase: the agent keeps jumping from one state to another to learn the optimal policy. After 15 minutes, the time in which



(a) Perturbation Cost $C_{\text{pert}}(\mathbf{a}^{(k)})$

(b) Random actions

Figure 3.9: System perturbation

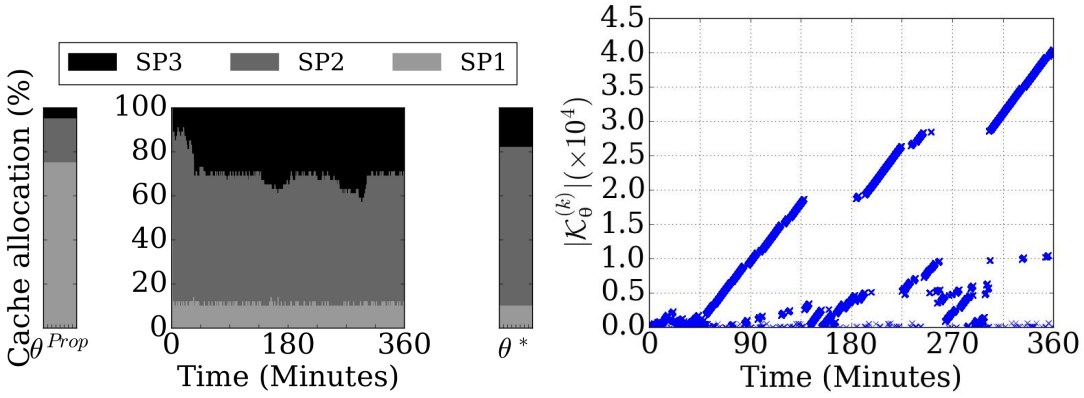
our algorithm converges, we observe an almost linear behaviour: the agent keeps visiting the same allocation $\hat{\boldsymbol{\theta}}^*$ (the absorbing state). The few points that we observe out of the linear behaviour represent the few random actions taken by the agent because $\epsilon > 0$. As long as the value of ϵ is non zero, the agent will choose a random action at some point, even after reaching $\hat{\boldsymbol{\theta}}^*$. The figure shows that our RL agent exploits more and more the good states, but never completely ceases visiting other states for exploration purposes.

3.6.3 Fairness

Let us denote with $x_p = \frac{\theta_p}{\zeta_p \cdot \lambda \cdot f_p}$ the slots given to SP p, normalized to its amount of cacheable requests. We compute the fairness of the system with the Jain's fairness index [120] as follows:

$$\mathcal{J}(x_1, \ldots, x_P) \triangleq \frac{(\sum_{p=1}^{P} x_p)^2}{P \cdot \sum_{p=1}^{P} x_p^2} \tag{3.24}$$

Our results show that cache sharing strategy with our MB-RL allocation (0.7 fairness) is much fairer than the optimal allocation $\boldsymbol{\theta}^*$ (0.36 fairness), at almost the same total cost (see Figure 3.6). It is also close to that of the proportional allocation $\boldsymbol{\theta}^{\text{prop}}$ (0.85 fairness) albeit being much better in terms of cost. Note that we are also close to the ideal maximum fairness achieved by the proportional allocation not taking into account cacheability, i.e., if all contents were cacheable (i.e., $\zeta_p = 1, p = 1, .., P$). The latter is 1, by construction, as it is proportional to the rate of requests directed to each SP; on the other hand, it is an artificial measure, as it ignores cacheability.



(a) Evolution of the allocation vs. time

(b) Number of times current $\boldsymbol{\theta}$ is visited

Figure 3.10: System evolution for MB-RL

3.6.4 Sensitivity analysis

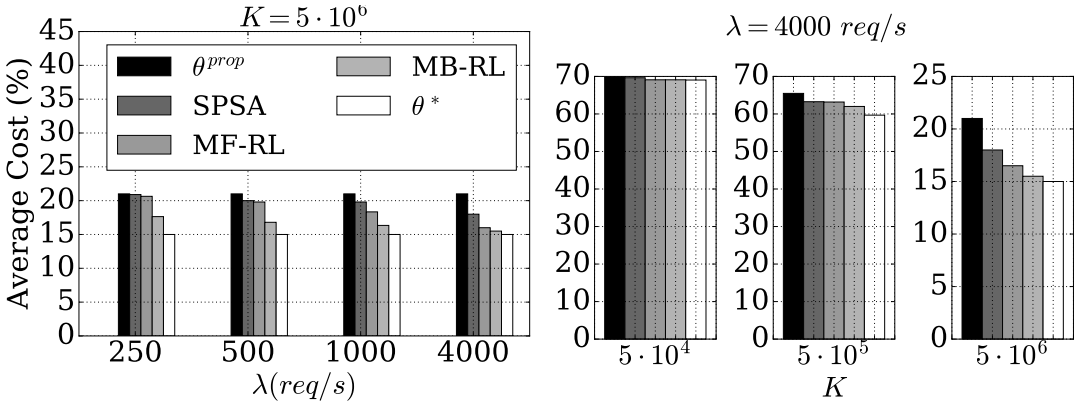
We next study how the performance of our solution is affected by the request rate λ and the cache capacity K. In Figure 3.11 we plot the average cost $\frac{1}{Z}C_{\text{cum}}(Z)$ (3.4) of MB-RL and MF-RL algorithm, after $Z = 6$ hours, and compare it to the static proportional and optimal allocations.

Let us first focus on the request rate λ. A small λ implies that only few requests are observed in each time slot, which may result in a high noise, as defined in (3.6), and ultimately affects the accuracy of the update of the Q-table and slows down the convergence. We thus expect any data-driven approach to perform best with large λ. This is evident for MF-RL and SPSA (Figure 3.11a), whose cost is far from the optimum for $\lambda \leq 1000$ req/s. It is however interesting to observe that MB-RL performs relatively well even with few requests per second. This shows that embedding the inferred model into the RL agent increases the sample efficiency of our method.

Figure 3.11b shows the average cost measured over $Z = 6$ hours for various cache sizes $K \in \{5 \cdot 10^4, 5 \cdot 10^5, 5 \cdot 10^6\}$ and a fixed request rate $\lambda = 4 \cdot 10^3 req/s$. It confirms that the gains of our MB-RL algorithm hold for different cache sizes, and shows that gain increases for larger caches. Indeed, for a small cache size there is not much to optimize: the cost is high with both proportional and optimal allocations, so even if MB-RL and MF-RL position themselves between the two, the improvement in cost is negligible.

Compared to SPSA, we observe in Figure 3.11 that our algorithm performs better in any configuration of the system.

We finally verify that the good performance of MB-RL is maintained when increasing the number of SPs. We simulate a scenario in the same conditions as in Section 3.6.2 but we change the number of SPs to $P = 4$. Each of the requests is directed to SP 1, 2, 3



(a) To request rate λ (b) To cache size K

Figure 3.11: Sensitivity of the system

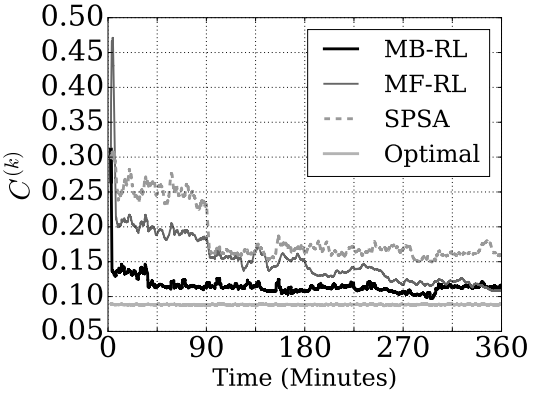


Figure 3.12: Total instantaneous cost $C^{(k)}$ for 4 SPs

or 4 with probabilities 0.60, 0.20, 0.10 and 0.10, respectively. We set the cacheability of SP_1, SP_2, SP_3 and SP_4 to $\zeta_1 = 0.5$, $\zeta_2 = 0.7$, $\zeta_3 = 0.9$ and $\zeta_4 = 0.9$, respectively. Each SP has a catalog of $N_1 = N_2 = N_3 = N_4 = 10^7$ cacheable objects. Content popularity in each catalog follows Zipf's law with exponents $\beta_1 = 2.2$, $\beta_2 = 0.4$, $\beta_3 = 0.1$ and $\beta_4 = 0.1$, respectively. The total cache size is $K = 5 \cdot 10^6$. The simulation time is maintained at $Z = 6$ hours. The length of a time-slot is maintained at 0.25 second and total request rate at $\lambda = 4000$ requests per second.

As in Section 3.6.2, we plot in Figure 3.12 the total cost $C^{(k)}$ of our MB-RL algorithm, the optimal allocation $\boldsymbol{\theta}^*$, MF-RL and SPSA for 4 SPs. The results show that our algorithm rapidly converges close to optimal cost, outperforming SPSA and MF-RL.

3.7 Conclusion

This chapter proposes a model-based RL algorithm for online cache allocation at the edge between several SPs, where the aim is not only to minimize the cost, in terms of miss rate, but also to optimize the way to achieve that, through minimizing perturbations, assuming encrypted, not all cacheable content: a major challenge of in-network caching. We first proved that our algorithm converges to an absorbing discrete optimal state with probability 1 in an infinite horizon. We then compared via extensive simulations our dynamic allocation to two static allocations: (i) theoretical optimal one, which would be computed by an oracle who knows exactly the content popularity and thus the expression of the cost function, and (ii) proportional one where proportionality is with respect to the probabilities of requesting content from each SP. We also compared our model-based RL algorithm to two dynamic allocation strategies: (iii) model-free RL and (iv) state of the art SPSA. Simulations in several scenarios show that our algorithm converges to a configuration close to the optimal one, much faster than the compared allocation strategies. This allows to drastically reduce overall system cost. In the next chapter, we will extend our work to multiple resource allocation in EC.

Chapter 4

Multiple-resource Allocation for Multi-tenant Edge Computing

4.1 Introduction

Mobile Augmented Reality (MAR) has become one of the most emerging applications, accompanied by the development of mobile devices and wireless communication. In MAR, the human perception of the world can be enhanced by merging virtual information (generated from object detection, classification, or tracking) with the real environment via mobile devices [121]. However, it is difficult for a mobile device to offer the abundant computation and energy required by MAR applications.

Big Tech players started already to develop their own AR devices since 2013 when Google announced its open beta of Google Glass [122] followed by the HoloLens [123] and HoloLens 2 [124] from Microsoft in 2016 and 2019, respectively. Meta Group (formerly Facebook) joined the competition with Virtual Reality (VR) headsets: the Oculus Quest [125] in 2019 and Oculus Quest 2 [126] in 2020.

While the production of AR/VR dedicated hardware seems very effective to run AR/VR applications properly, it is costly in the sense that only big players can afford producing their own devices. Hence, multi-tenant EC is particularly interesting for all the other players, as it is probably the only way for small or medium AR Service Providers (SPs) to run their applications at the edge of the network. The development of EC and 5G can eliminate the obstacle to deploying the MAR service. In the concept of EC [49], computing and storage resources are deployed at the edge of the access network. Several MAR clients on mobile devices can send MAR requests that contain original data captured by sensors and cameras to the Edge Computing (EC) server. Furthermore, dedicated computing hardware (e.g., Graphics Processing Unit (GPU) and Central Processing Unit (CPU)) and software (e.g., computer vision-based algorithms) can process these data in the EC server

and then return the results, such as object classification or space coordinate information, to the mobile devices.

The use of the EC for MAR has attracted extensive attention from the research community and industry recently [127, 128], which mainly focus on architecture design and deployment. However, scheduling the MAR requests received from several competing MAR clients on one EC server is critical and challenging. We address in this chapter the issue of resource allocation of a Network Operator (NO) to competing, heterogeneous SPs in the case of multiple, limited resources at the Edge. We first model the arrivals and service dynamics of the flows using Erlang queuing model. We then formulate the resource allocation problem, when all system parameters are known, as a sub-modular maximization under Knapsack constraints problem and we propose an implementation of the so-called streaming algorithm to solve it. We obtain a $(\frac{1}{1+2d} - \epsilon)$-approximate optimal value, where d is the number of resource types and ϵ is a controllable error term. When system parameters are unknown by the NO, we cast the problem as a sequential decision making problem. We formulate it as a Markov Decision Process (MDP) and use state-of-the-art deep Reinforcement Learning algorithm, namely Deep Q Network (DQN) [129] to solve it. We eventually provide numerical results to show that the performance of the RL algorithm significantly approaches the performance of the streaming algorithm and outperforms baseline resource allocation policy.

The remainder of this chapter is organized as follows. We describe in Section 4.2 our system model. We formulate the sub-modular maximization problem under Knapsack constraints in Section 4.3 and describe the proposed algorithm to solve it. In Section 4.4, we formulate the problem as a MDP and describe the proposed algorithm to solve this MDP. In Section 4.5, we show our simulation results. We draw conclusions in Section 4.6.

4.2 System Model and Problem Formulation

We consider a setting with one NO, owning a set of resources $\mathcal{R} = \{\text{memory, CPU}\}$ and willing to share them between P different SPs (Figure 4.1). Each SP can then use its assigned share as if it had a dedicated hardware deployed in the edge.

Table 4.1 summarizes the frequently used notations in this chapter.

4.2.1 Request Pattern and Augmented Reality Setting

MAR users of SP $p, p = 1, ..., P$ arrive to the EC server following a Poisson process with rate λ_p expressed in $users/s$. Once a user of any SP p is connected to the edge server, a session is created. This session is valid for a period of time denoted by T_p during which the user can perform a sequence of interactions within that MAR application. A single session can contain multiple activities running while the user is connected. Each SP runs in the edge a virtual server, e.g., a Kubernetes POD [130]. A MAR user establishes a session with the virtual server of the respective SP. Within that session, it sends a stream of image

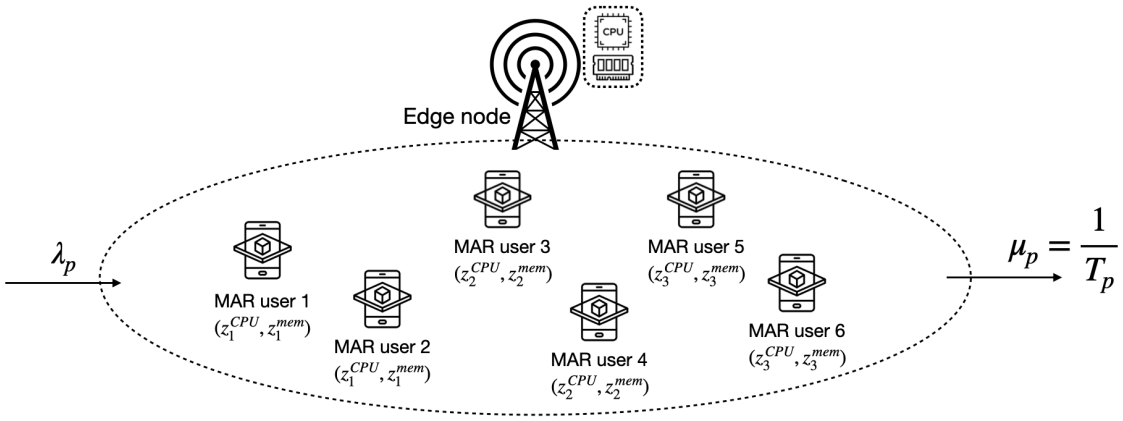


Figure 4.1: Considered scenario

processing requests. When users point their MAR device toward an object, pre-processed video frames are sent from the MAR device cameras to the edge server. These frames are delivered to the AR tracker to determine the user's position with respect to the physical surroundings. Given the tracking results, virtual coordinates of the environment can be established by the mapper. Then, the internal objects in video frames are identified by the object recognizer with robust features, running in the edge server. The MAR device finally downloads information about the object from the edge server. The AR information is presented in a 3-D "experience" superimposed on the object. What users see, then, is part real and part virtual. Since MAR needs high data rates, ultra-low latency and the possible use of lightweight devices, performing processing at the edge of 5G mobile networks can help guarantee the requirements of MAR applications (Section III-F of [121]).

We assume that a session of a single user of SP p requires a certain amount z_p^r of resource for every type $r \in \mathcal{R}$, e.g., CPU and memory. If the SP does not have at the edge such amount of resources available, the user will establish a session with the cloud, suffering longer delay. Once a user of SP p completes their service at the edge, his session will be closed and he leaves the EC system. Please note that users can leave the system when they decide, this does not deny that we can define an average service rate for SP p expressed in $users/s$ denoted by $\mu_p = \frac{1}{T_p}$.

4.2.2 Resource Partitioning

The NO owns CPU and memory at the edge of the network, for instance, in a server co-located with a (micro) base station or central offices at the metropolitan scale. It allocates a total capacity K^{CPU} of CPU and a total capacity K^{mem} of memory among the P SPs.

Parameter	Definition
$\mathcal{R}$	Set of resources (Section 4.2)
P	Number of SPs (Section 4.2)
T	Number of time slots (Section 4.4.1)
K^{CPU}	Total capacity of CPU (Section 4.2.2)
K^{mem}	Total capacity of memory (Section 4.2.2)
λ_p	Users arrival rate for SP p (Section 4.2.1)
μ_p	Users service rate for SP p (Section 4.2.1)
z_p^r	Amount of resource r required by one user of SP p (Section 4.2.1)
N_p	Number of users of SP p if it has all the resources (Section 4.2.3)
B_p	Blocking probability of SP p (Section 4.2.4)
U_E	Utility of a user who establishes a session directly in the edge (Section 4.2.4)
U_C	Utility of a user who establishes a session directly in remote cloud (Section 4.2.4)
$\mathcal{V}_p$	Set of users of SP p if it has all the resources on the edge (Section 4.3)

Decision Variable	Definition
$\vec{\boldsymbol{\theta}}$	Allocation vector (Section 4.2.2)
θ_p^r	Amount of resource r given to SP p on the edge node (Section 4.2.3)
n_p	Maximum number of users of SP p served at the edge given resources $(\theta_p^r)_{r\in\mathcal{R}}$ (Section 4.2.3)
$\mathcal{S}_p$	Set of users of SP p served at the edge given resources $(\theta_p^r)_{r\in\mathcal{R}}$ (Section 4.3)

Table 4.1: Frequently used notations in Chapter 4

The allocation is a vector $\vec{\boldsymbol{\theta}} = (\boldsymbol{\theta}^{\vec{\text{CPU}}}, \boldsymbol{\theta}^{\vec{\text{mem}}})$ where each vector $\vec{\boldsymbol{\theta}^r}$ is the allocation of resource $r \in \mathcal{R}$. For simplicity, in this chapter we consider $\mathcal{R} = \{\text{CPU, mem}\}$. Therefore, the allocation has a form as follows:

$$\vec{\boldsymbol{\theta}} = (\theta_1^{\text{CPU}}, \ldots, \theta_P^{\text{CPU}}, \theta_1^{\text{mem}}, \ldots, \theta_P^{\text{mem}}) \tag{4.1}$$

We define the set of all possible allocations as:

$$\mathcal{T} \triangleq \left\{ \vec{\boldsymbol{\theta}} | \sum_{p=1}^{P} \theta_p^r \leq K^r, \theta_p^r \in \mathbb{Z}^+, r \in \mathcal{R} \right\} \tag{4.2}$$

4.2.3 Service Model

We model our system as an Erlang queue [131] which models Poisson arrivals, exponentially distributed service time, and a number of servers equal to the number of places in the system, i.e., users are either directly served at the edge or directed to the cloud. In our case, users of SP p arrive to the edge according to a Poisson distribution with mean arrival rate λ_p, they remain in the system for an exponentially distributed duration, T_p. The number of servers in our case refers to the maximum number of sessions that the edge can accommodate for each SP, as determined next. Each user of SP p has fixed requirements $(z_p^r)_{r\in\mathcal{R}}$ and fixed allocation $(\theta_p^r)_{r\in\mathcal{R}}$ during service. We denote by $n_p(\vec{\boldsymbol{\theta}})$ the maximum number of users that can be served at the edge for a SP p when the resource allocation decided by the NO is $\vec{\boldsymbol{\theta}} = (\theta_p^r)_{p=1..P,r\in\mathcal{R}}$. Each user of each SP p will receive an amount z_p^r of the resource r for their session. Hence the maximum number of sessions $n_p(\vec{\boldsymbol{\theta}})$ each SP p can establish at the edge when the allocation from the NO is $\vec{\boldsymbol{\theta}}$ must satisfy:

$$n_p(\vec{\boldsymbol{\theta}}) \cdot z_p^r \leq \theta_p^r, p = 1 \dots P, r \in \mathcal{R}. \tag{4.3}$$

Therefore, $n_p(\vec{\boldsymbol{\theta}})$ is:

$$n_p(\vec{\boldsymbol{\theta}}) = \left\lfloor \min_{r\in\mathcal{R}} \left(\frac{\theta_p^r}{z_p^r} \right) \right\rfloor, p = 1 \dots P \tag{4.4}$$

where $\lfloor . \rfloor$ is the floor function giving as output the greatest integer less than or equal to $\left(\frac{\theta_p^r}{z_p^r}\right)$.

Let us denote by N_p the number of users of SP p served at the edge if all the resources are allocated only to this SP p.

$$N_p = \left\lfloor \min_{r\in\mathcal{R}} \left(\frac{K^r}{z_p^r} \right) \right\rfloor, p = 1 \dots P \tag{4.5}$$

4.2.4 Utility Model

A user of SP p is served directly by the edge if the latter can satisfy the requirements z_p^{mem} and z_p^{CPU}. Otherwise, the corresponding session is not accepted (we say that it is "blocked", following the terminology from queuing theory) and directed to a remote cloud server. Using Erlang (equation (3.45) of [131]), the probability for a user of SP p to be blocked is

$$B_p(\vec{\boldsymbol{\theta}}) = \frac{\frac{A_p^{n_p(\vec{\boldsymbol{\theta}})}}{n_p(\vec{\boldsymbol{\theta}})!}}{\sum_{i=0}^{n_p(\vec{\boldsymbol{\theta}})} \frac{A_p^i}{i!}}, p = 1 \dots P \tag{4.6}$$

where $A_p = \frac{\lambda_p}{\mu_p}$. The probability for a user of SP p to have his/her session established with the edge is thus:

$$\bar{B}_p(\vec{\boldsymbol{\theta}}) = 1 - B_p(\vec{\boldsymbol{\theta}}). \tag{4.7}$$

The utility perceived by a user who establishes a session directly in the edge is U_E, while if the session is with the cloud, the utility is U_C. Such utilities take into account the impact on the Quality of Service (QoS) of the delay to process every user request, accounting for a larger delay to reach the cloud. Hence, $U_E > U_C > 0$. For simplicity, we assume that U_E and U_C are the same for all SPs. Since $1 - B_p$ indicates the fraction of users of SP p establishing sessions with the edge, the expected value of the utility perceived by a user of SP p is, by the theorem of total probability:

$$\begin{aligned}
\mathbb{E}U_p(\vec{\boldsymbol{\theta}}) &= \mathbb{P}(\text{session established with the edge}) \cdot U_E \\
&\quad + \mathbb{P}(\text{session established with the cloud}) \cdot U_C \\
&= \bar{B}_p(\vec{\boldsymbol{\theta}}) \cdot U_E + (1 - \bar{B}_p(\vec{\boldsymbol{\theta}})) \cdot U_C \\
&= (U_E - U_C) \cdot \bar{B}_p(\vec{\boldsymbol{\theta}}) + U_C
\end{aligned} \tag{4.8}$$

By the theorem of total expectation, the utility perceived by a generic user is

$$\begin{aligned}
\mathbb{E}U(\vec{\boldsymbol{\theta}}) &= \sum_{p=1}^{p} \mathbb{E}U_p(\vec{\boldsymbol{\theta}}) \cdot \mathbb{P}(\text{new user is for SP}p) \\
&= \sum_{p=1}^{p} w_p \cdot \mathbb{E}U_p(\vec{\boldsymbol{\theta}})
\end{aligned} \tag{4.9}$$

where

$$w_p = \frac{\lambda_p}{\sum_{p'=1}^{P} \lambda_{p'}}.$$

4.2.5 Optimization Problem

The NO aims to maximize the expected value of the utility perceived by a generic user:

$$\begin{aligned}
\max_{\vec{\boldsymbol{\theta}}} \quad & \mathbb{E}U(\vec{\boldsymbol{\theta}}) \\
\text{s.t.} \quad & \sum_{p=1}^{P} \theta_p^r \leq K^r, \forall r \in \mathcal{R}
\end{aligned} \tag{4.10}$$

Replacing $\mathbb{E}U_p(\vec{\boldsymbol{\theta}})$ with its value found in (4.8) and observing that $(U_E - U_C)$ and U_C are positive constants, the optimization problem becomes:

$$\begin{aligned} \max_{\vec{\boldsymbol{\theta}}} \quad & \sum_{p=1}^{P} w_p \bar{B}_p(\vec{\boldsymbol{\theta}}) \\ \text{s.t.} \quad & \sum_{p=1}^{P} \theta_p^r \leq K^r, \forall r \in \mathcal{R} \end{aligned} \tag{4.11}$$

Thanks to (4.3) and (4.6), we can express the problem in terms of $\vec{\mathbf{n}} = (n_1, \ldots, n_P)$ instead of $\vec{\boldsymbol{\theta}}$:

$$\begin{aligned} \max_{\vec{\boldsymbol{n}}} \quad & f(\vec{\boldsymbol{n}}) = \sum_{p=1}^{P} w_p \bar{B}_p(\vec{\boldsymbol{n}}) \\ \text{s.t.} \quad & \sum_{p=1}^{P} n_p \cdot z_p^r \leq K^r, \forall r \in \mathcal{R} \end{aligned} \tag{4.12}$$

$$\text{where} \quad \bar{B}_p(\vec{\boldsymbol{n}}) \triangleq 1 - \frac{\frac{A_p^{n_p}}{n_p!}}{\sum_{i=0}^{n_p} \frac{A_p^i}{i!}}, p = 1 \ldots P \tag{4.13}$$

Observe that $f(\vec{\mathbf{n}})$ is the probability for a generic user to be served with a session with the edge node. This shows that improving the expected user utility (4.10) is equivalent to maximizing the probability of establishing a session with the edge (4.12).

4.3 Sub-modular Optimization

In this section we will solve the problem of memory-CPU allocation as if the NO has the knowledge of all the system parameters. We will describe our problem (4.12) in terms of sub-modular optimization problem, by interpreting a user session established with the edge node as an item (we will use this terminology in (4.19)). Let

$$\mathcal{V}_p = \{1, 2, \ldots, N_p\} \tag{4.14}$$

be the set of *candidate sessions* of SP p that could coexist in the edge if all resources were given to this SP p. Since in reality resources at the edge are not given to one SP only, we need to choose a subset of sessions $\mathcal{S}_p \subseteq \mathcal{V}_p$ to allocate to each SP p. This choice induces a certain probability of establishing a session with the edge:

$$\bar{B}_p(\mathcal{S}_p) = 1 - \frac{\frac{A_p^{|\mathcal{S}_p|}}{|\mathcal{S}_p|!}}{\sum_{i=0}^{|\mathcal{S}_p|} \frac{A_p^i}{i!}} \tag{4.15}$$

With slight abuse of notation, in the formula above we use the notation $\bar{B}_p(\cdot)$ as in (4.13), to emphasize that the two quantities are conceptually the same thing, by setting $n_p = |\mathcal{S}_p|$. Let

$$\mathcal{V} \triangleq \bigcup_{p=1}^{P} \mathcal{V}_p \tag{4.16}$$

be the set of all candidate sessions, hence, $|\mathcal{V}| = N = \sum_{p=1}^{P} N_p$. And let

$$\mathcal{S} = \bigcup_{p=1}^{P} \mathcal{S}_p \subseteq \mathcal{V} \tag{4.17}$$

be the set of sessions allocated, which will represent our decision variable.

For each SP p, we define a non-negative set function f_p, taking as input all possible subsets $\mathcal{S}$ of $\mathcal{V}$, as follows:

$$f_p(\mathcal{S}) \triangleq w_p \cdot \bar{B}_p(\mathcal{S} \cap \mathcal{V}_p) \in [0, 1]$$

Function f_p represents the probability, for a user that arrives, to be of SP p and to be served with a session at the edge. We define $f(\mathcal{S}) \triangleq \sum_{p=1}^{P} f_p(\mathcal{S})$. It indicates, for any arriving user, the probability to be served with a session at the edge.

For any subset $\mathcal{S}$ of $\mathcal{V}$, we denote the characteristic vector of $\mathcal{S}$ by:

$$\boldsymbol{x}_{\mathcal{S}} = (x_{\mathcal{S}_1,1}, \ldots, x_{\mathcal{S}_1,N_1}, \ldots, x_{\mathcal{S}_P,1}, \ldots, x_{\mathcal{S}_P,N_P})^T, \tag{4.18}$$

where for any $j \in [1, N_p]$ and $p = 1, ..., P$:

$$x_{\mathcal{S}_p,j} = \begin{cases} 1, & \text{if the } j\text{-th item of } \mathcal{V}_p \text{ is in } \mathcal{S}_p \\ 0, & \text{otherwise} \end{cases} \tag{4.19}$$

For $\mathcal{S} \subseteq \mathcal{V}$ and $v \in \mathcal{V}$, the marginal gain in f when adding v to set $\mathcal{S}$ is defined as:

$$\Delta_f(v|\mathcal{S}) \triangleq f(\mathcal{S} \cup \{v\}) - f(\mathcal{S}), \tag{4.20}$$

which quantifies the increase in $f(\mathcal{S})$ when v is added into subset $\mathcal{S}$.

We introduce now the d-knapsack constraint where $d = |\mathcal{R}|$. Let $\boldsymbol{k} = (K^1, \ldots, K^d)^T$ be the resource capacity vector and $\boldsymbol{Z}_p = (z^r_{p,j})$ denote a $d \times N_p$ matrix, whose (r, j)-th entry $z^r_{p,j} > 0$ is the weight of the j-th item of $\mathcal{V}_p$ in terms of resource r. Since we have assumed in Section 4.2.3 that all users of SP p require the same amount of each resource, $z^r_{p,j} = z^r_p$ for all the items in $\mathcal{V}_p$, the constraint in (4.12) can be expressed as follows:

$$\boldsymbol{Z} \cdot \boldsymbol{x}_{\mathcal{S}} \leq \boldsymbol{k}, \tag{4.21}$$

where $\boldsymbol{Z} = (\boldsymbol{Z}_1, \ldots, \boldsymbol{Z}_P) \in \mathbb{R}^{d \times \sum_p N_p}$ and $\boldsymbol{x}_{\mathcal{S}} \in \{0,1\}^{\sum_p N_p \times 1}$.

Problem (4.12) becomes:

$$\begin{aligned} \max_{\mathcal{S}} \quad & f(\mathcal{S}) = \sum_{p=1}^{P} f_p(\mathcal{S}_p) \\ \text{s.t.} \quad & \boldsymbol{Z}\boldsymbol{X}_{\mathcal{S}} \leq \boldsymbol{k} \end{aligned} \tag{4.22}$$

Without loss of generality, for $1 \leq i \leq d, 1 \leq j \leq N$, we assume that $z_p^r \leq K^r$. That is, no item has a larger weight than the corresponding knapsack budget, since otherwise such an item would never be selected into $\mathcal{S}$.

We are now ready to study the properties of formulation (4.22). To do so, we recall two common definitions from set-function theory [132].

Definition 4.3.0.1. *A function f is sub-modular if it satisfies that $\Delta_f(v|\mathcal{B}) \leq \Delta_f(v|\mathcal{A})$, for any $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{V}$ and $v \in \mathcal{V} \setminus \mathcal{B}$.*

Definition 4.3.0.2. *A function f is monotone if for any $\mathcal{S} \subseteq \mathcal{V}$ and $v \in \mathcal{V}$, $\Delta_f(v|\mathcal{S}) \geq 0$.*

Theorem 4.3.0.3. *Function f in (4.22) is monotone and sub-modular.*

Proof. Let $\mathcal{S} \subseteq \mathcal{V}$ and $v \in \mathcal{V}$. Suppose in particular that $v \in \mathcal{V}_{p'}$.

$$\begin{aligned} \Delta_f(v|\mathcal{S}) =& f(\mathcal{S} \cup \{v\}) - f(\mathcal{S}) \\ =& \sum_{p \neq p'} f_p(\mathcal{S}_p) + f_{p'}(\mathcal{S}_{p'} \cup \{v\}) - \sum_{p=1}^{P} f_p(\mathcal{S}_p) \\ =& f_{p'}(\mathcal{S}_{p'} \cup \{v\}) - f_{p'}(\mathcal{S}_{p'}) \\ =& w_{p'} \cdot \bar{B}_{p'}(\mathcal{S}_{p'} \cup \{v\}) - w_{p'} \cdot \bar{B}_{p'}(\mathcal{S}_{p'}) \\ \geq& 0, \end{aligned}$$

where the last inequality can be obtained by simple calculus from (4.15). This shows that function f is monotone.

Let us consider sets $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{V}$ and a vector $v \in \mathcal{V} \setminus \mathcal{B}$.

$$\begin{aligned} \Delta_f(v|\mathcal{B}) - \Delta_f(v|\mathcal{A}) =& [f(\mathcal{B} \cup \{v\}) - f(\mathcal{B})] - [f(\mathcal{A} \cup \{v\}) - f(\mathcal{A})] \\ =& [f(\mathcal{B} \cup \{v\}) - f(\mathcal{A} \cup \{v\})] + [f(\mathcal{A}) - f(\mathcal{B})] \end{aligned}$$

Having $\mathcal{A} \subseteq \mathcal{B}$, we can write $\exists \mathcal{Q} \subseteq \mathcal{V}/\mathcal{B} = \mathcal{A} \cup \mathcal{Q}$. Hence:

Algorithm 2: Streaming Algorithm for sub-modular maximization problem under Knapsack constraints

Data: $d, z_p^r, K^r, \lambda_p, \mu_p$
Result: $\mathcal{S}^*$

$m \leftarrow 0$;
$\mathcal{Q} \leftarrow \{[1+(1+2d)\epsilon]^l | l \in \mathbb{Z}\}$;
for $v \in \mathcal{Q}$ **do**
  $\mathcal{S}_v \leftarrow \emptyset$;
end
for $1 \leq j \leq N$ **do**
  for $1 \leq i \leq d$ **do**
    $m \leftarrow \max\{m, f(\{j\})/z_{i,j}\}$;
  end
  $\mathcal{Q} \leftarrow \{[1+(1+2d)\epsilon]^l | l \in \mathbb{Z}, \frac{m}{1+(1+2d)\epsilon} \leq [1+(1+2d)\epsilon]^l \leq 2Km\}$;
  for $v \in \mathcal{Q}$ **do**
    if $\exists i \in [1,d], z_{i,j} \geq \frac{K}{2}$ ***and*** $\frac{f(\{j\})}{z_{i,j}} \geq \frac{2v}{K^{(1+2d)}}$ **then**
      $\mathcal{S}_v \leftarrow \{j\}$;
      break;
    end
    if $\forall i \in [1,d], \sum_{l \in \mathcal{S} \cup \{j\}} z_{i,l} \leq K$ ***and*** $\frac{\Delta_f(j|\mathcal{S})}{z_{i,j}} \geq \frac{2v}{K^{(1+2d)}}$ **then**
      $\mathcal{S}_v \leftarrow \mathcal{S}_v \cup \{j\}$;
    end
  end
end
$\mathcal{S}^* \leftarrow \arg\max_{\mathcal{S}_v, v \in \mathcal{Q}} f(\mathcal{S}_v)$;

$$
\begin{aligned}
&[f(\mathcal{B} \cup \{v\}) - f(\mathcal{A} \cup \{v\})] + [f(\mathcal{A}) - f(\mathcal{B})] \\
&= [f(\mathcal{A} \cup \mathcal{Q} \cup \{v\}) - f(\mathcal{A} \cup \{v\})] + [f(\mathcal{A}) - f(\mathcal{A} \cup \mathcal{Q})] \\
&\leq [f(\mathcal{A}) + f(\mathcal{Q} \cup \{v\}) - f(\mathcal{A}) - f(\{v\})] \\
&+ [f(\mathcal{A}) - f(\mathcal{A} \cup \mathcal{Q})] \\
&\leq [f(\mathcal{Q} \cup \{v\}) - f(\{v\})] + [f(\mathcal{A}) - f(\mathcal{A}) - f(\mathcal{Q})] \\
&= f(\mathcal{Q} \cup \{v\}) - [f(\{v\}) + f(\mathcal{Q})] \leq 0
\end{aligned}
$$

Therefore, the function f is sub-modular. □

Now that we have proved that our objective function f is monotone and sub-modular, we can use well known results from sub-modular optimization. In particular, we adopt the algorithm proposed in [133], which we report in Algorithm 2. The main idea of the algorithm is that for every potential new user for each SP p, we compare the increase in f when we add this user to the set of users $\mathcal{S}$. We then add the user which yields the most increase in f. The algorithm guarantees the following sub-optimality gap (Theorem 1 of [133]).

Theorem 4.3.0.4. *Algorithm 2 outputs $\mathcal{S}$ that satisfies $f(\mathcal{S}) \geq (\frac{1}{1+2d} - \epsilon)OPT$ and has $O(\frac{\log(K_{\max})}{\epsilon})$ computational complexity per element, d being the number of resources, $0 < \epsilon < \frac{1}{1+2d}$, $K_{\max} = \max_{1 \leq i \leq d} K^i$ and OPT the value of f obtained by the optimal solution.*

Note that the hyper-parameter ϵ impacts the behavior of the algorithm as well as the quality of the optimality gap. The smaller is ϵ, the larger is $f(\mathcal{S})$.

4.4 Data-driven Optimization

Recall that problem (4.10) faced by the NO, if the latter does not have any information about the system parameters, can be cast as a sequential decision making problem under uncertainty. We formulate the problem as a MDP and use RL for solving it. In this section, we consider that time is slotted and we index it by k. The NO chooses an action periodically, at time instants $t_0, t_1 = t_0 + \Delta, \ldots, t_k, \ldots$; the action is constant during each period $T_k = [t_k, t_{k+1})$.

4.4.1 MDP Formulation

Our memory-CPU allocation problem can be formulated as a MDP. The **state** space $\mathcal{S}$ consists of the resource utilization, i.e.,

$$\mathcal{S} = \left\{ \vec{\boldsymbol{\theta}} = (\theta_1^{\text{CPU}}, \ldots, \theta_P^{\text{CPU}}, \theta_1^{\text{mem}}, \ldots, \theta_P^{\text{mem}}) | \sum_{p=1}^{P} \theta_p^r \leq K^r, \theta_p^r \in \mathbb{Z}^+, r \in \mathcal{R} \right\} \tag{4.23}$$

The action of the NO is, as in the sub-modular maximization problem (Section 4.3), the set of users of each SP to accept at the edge at each time slot. We can thus define the **action** space as:

$$\mathcal{A} = \left\{ \mathbf{a}^{(k)} = (n_1^{(k)}, .., n_P^{(k)}) \in \mathbb{N}^P \right\}, \tag{4.24}$$

where $n_p^{(k)}$ is the value of n_p defined by (4.4) at time slot k, $p = 1, ..., P$.

We define the **reward** as the utility U^k, perceived by a generic user during the period T_k, defined previously in Section 4.2.4. Hence, we re-write the optimization problem (4.10) as follows:

$$\pi^* = \arg\max_{\pi \in \Pi} \lim_{T \to \infty} \frac{1}{T} \sum_{k=1}^{T} \mathbb{E}[U^k] \tag{4.25}$$

where Π is the set of causal allocation policies.

4.4.2 Proposed Solution

We propose to address the above problem with model-free Deep Q Network (DQN). DQN is an extension of Q-learning that combines deep neural networks with Q-learning to handle high-dimensional state spaces.[1] DQN and Q-learning share a common objective of learning the optimal action-value function in reinforcement learning. While Q-learning maintains a tabular representation of Q-values, which becomes infeasible in such huge state space environment, DQN approximates Q-values using neural networks. This allows DQN to generalize across similar states, making it suitable for problems with large and continuous state spaces. Additionally, DQN uses a target network, i.e., a duplicate of the main Q-network used for estimating the Q-values of actions, to improve sample efficiency and stabilize training, addressing some of the limitations of Q-learning's online updates. These characteristics make DQN applicable to our problem.

Note that DQN makes use of experience replay buffer (as in Section 3.3.3), which stores past experiences (state, action, reward, next state) and samples from it to train the neural network that approximates the Q-values.

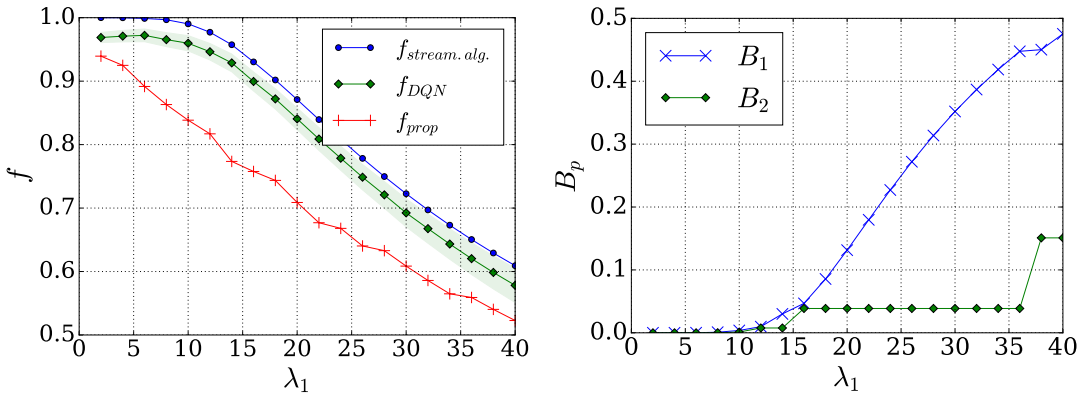
4.5 Numerical Results

We now evaluate the performance of the streaming, approximate algorithm, i.e., Algorithm 2 and DQN via a numerical model developed in Python and compare it to the proportional allocation where θ_p^r is proportional to the arrival rate λ_p of users of each SP p. We set $\epsilon = 0.01$.

4.5.1 Setting

We focus on an edge node co-located with a central office serving 2 SPs. We set arrival rates λ_1 and λ_2 at 20 and 5 $users/s$, respectively and departure rates μ_1 and μ_2 at 1 and 10 $users/s$, respectively. Motivated by Amazon EC2 instances, such as G4dn [34], designed to support machine learning inference for applications like adding metadata to an image, object detection, recommendation systems, automated speech recognition, and language translation, we consider an edge server similar to the G4dn.metal with $K^{\text{mem}} = 384$ GB

[1] Note that the size of the state space in Chapter 3 was $O((K/\Delta)^P)$ and in this chapter the size of the state space is $O(\prod_{r \in \mathcal{R}} K^{r \times P})$.



(a) Objective function f vs. λ_1

(b) Blocking probability B_p vs. λ_1

Figure 4.2: Performance of the streaming algorithm and DQN w.r.t λ_1

of total memory capacity and a 2nd Generation Intel Xeon Scalable CPU: Cascade Lake P-8259L with total capacity of CPU $K^{\text{CPU}} = 96$ vCPU. Taking in consideration AR applications similar to Pokemon GO [134], we set memory and CPU requirements for SP 1 and SP 2 at: $z_1^{\text{mem}} = 2$ GB, $z_1^{\text{CPU}} = 1$ vCPU, $z_2^{\text{mem}} = 0.5$ GB and $z_2^{\text{CPU}} = 4$ vCPU, respectively.

For training the DQN (Section 4.4.2), we set the number of episodes $= 40$, one episode $= 1$ day and $\Delta = 1$ hour.

4.5.2 Results

We plot in Figure 4.2a the performance of the streaming algorithm and DQN in terms of the objective function f, which is the probability for a user to establish a session with the edge (4.12) and we compare them with the baseline f_{prop}, i.e., the probability of establishing sessions with the edge obtained when allocating resources to SPs proportionally to their users arrival rates. The results show that DQN significantly approaches the streaming algorithm and outperforms the baseline solution for all values of λ_1, which means that DQN managed to find a policy close to the one obtained with the streaming algorithm. Note that the streaming algorithm is only valid under perfect knowledge of system parameters are known, which is not true in real world scenarios. DQN as a first step toward a practical algorithm, as we are using under partial information (users requirements are assumed to be known and stationary).

In Figure 4.2b, we show the blocking probabilities for each SP when varying λ_1, using the streaming algorithm. The increase in λ_1 results in higher blocking probability for SP 1, which is expected as more users will consume more resources given to SP 1 at the edge and

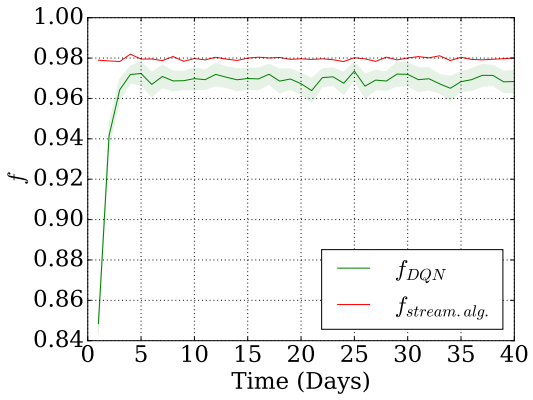
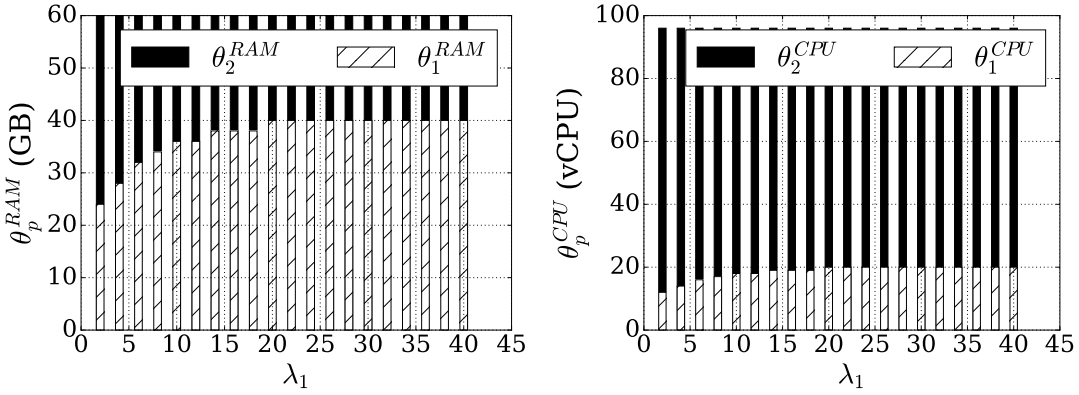


Figure 4.3: Learning curve of DQN

less resources are left. Higher λ_1 will also affect SP 2 but much less significantly (higher λ_1 would result in higher resources allocated to SP 1 and hence less resources left for SP 2).

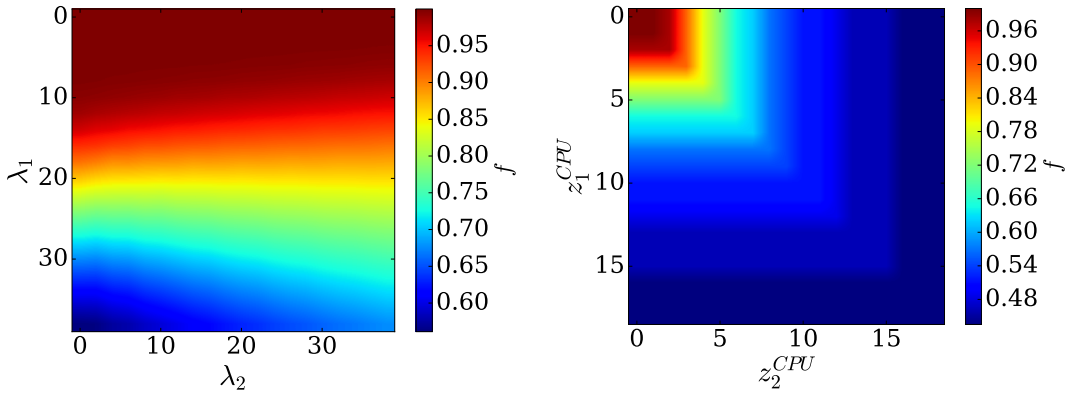
We plot in Figure 4.3 the learning curve of our proposed DQN-based solution along with the value of the objective function f obtained by the streaming algorithm. The results confirm that DQN approaches the streaming algorithm. Observe that DQN converges after 3 days of training. The training process is relatively long, which is expected as the RL agent learns directly from interactions with the environment. This typically involves trial-and-error learning, where the agent explores different actions and observes the induced utility. Exploring the environment can be time-consuming, especially in environments with large state space. This slow training is not suitable for online resource allocation when directly dealing with an up and running system. Indeed, in a real system, arrivals are not stationary, and the rate change in a time-scale of hours. We thus need an algorithm that learns the dynamics in a matter of minutes (and not days) to cope up with the varying load. In order to accelerate learning and convergence, one could make use of a model as we did in Chapter 3 for the cache allocation and as we will show next in Chapter 5 for the case of multiple resources. We plan to work on accelerating learning also for the case of this chapter in our future work.

As for resource utilization, the results illustrated in Figure 4.4, using the streaming algorithm, show that the CPU is totally utilized by the two SPs (Figure 4.4b), while the memory is not fully exploited (less than 20% as shown in Figure 4.4a). Despite having more than 80% of memory free, we cannot expect better performance since the blocking comes always from the CPU, which is the scarcer resource. Even at higher arrival rate, the streaming algorithm does not allow SP 1 to have more CPU as this resource is almost 80% used by SP 2. We can explain this by looking at the values of z_1^{CPU} and z_2^{CPU}, we can see that SP 2 is CPU-greedy: users of SP 2 consume 4 times more CPU than users of SP 1.



(a) Memory allocation vs. λ_1 (b) CPU allocation vs. λ_1

Figure 4.4: Resource utilization vs. λ_1



(a) Objective function f w.r.t λ_1 and λ_2 (b) f w.r.t z_1^{CPU} and z_2^{CPU}

Figure 4.5: Sensitivity of Algorithm 2 w.r.t λ_p and $z_p^{\text{CPU}}, p = 1, 2$

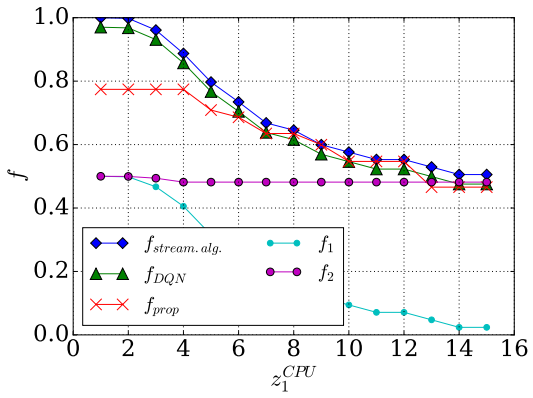


Figure 4.6: Sensitivity of the system w.r.t z_1^{CPU}

We now investigate the sensitivity of the objective function f with respect to the arrival rates and CPU requirements.

In Figure4.5a, we plot a heat-map describing the global objective function f, obtained with streaming algorithm, with respect to the variations of the two arrival rates. Obviously, the performance of the algorithm under lower arrival rates is better (dark upper region $f \geq 0.95$). But what is more interesting in the figure, is that even for high arrival rates for SP 2 ($\lambda_2 \geq 35$), the algorithm keeps performing well up to $\lambda_1 = 20$ (horizontal middle region $f \geq 0.85$), no matter the arrival rate λ_2 of SP 2. The opposite is not true: for all values of λ_2, even small ones, the performance highly depends on λ_1. We can explain this by the fact that the users of SP 2 consume a lot of CPU (the blocking resource) which means every new admission of SP 1 user would degrade the performance of the algorithm.

In Figure 4.5b, we plot the heat-map describing the global objective function f obtained with the streaming algorithm with respect to the CPU requirements. The algorithm maintains a satisfying performance (upper left region) up to requirements around 5 vCPU at most and then the performance rapidly decreases with the higher CPU requirements.

Very similar heat-map plots are obtained using DQN.

We plot in Figure 4.6 the objective functions: $f_{stream.alg.}$, f_1 and f_2 obtained by the streaming algorithm, f_{DQN} obtained by the DQN algorithm and f_{prop}. The results show that DQN approaches the streaming algorithm for all values of z_1^{CPU} outperforming the baseline allocation in the majority of the cases. The figure shows that the utility of SP 1 decreases with the increase in z_1^{CPU}, while the utility of SP 2 remains constant, i.e., does not depend on z_1^{CPU}. At high values of z_1^{CPU}, the performance of SP 1 degrades significantly with f_1 tending to 0. Overall, f becomes equal to f_2 using the three appraoches.

4.6 Conclusion

We tackled in this chapter resource allocation at EC between heterogeneous, MAR-oriented SPs competing over multiple, limited resources. We modeled the users dynamics and blocking in terms of an Erlang-type queuing model, we formulated the resource allocation problem as a sub-modular maximization problem subject to multiple knapsack constraints and solved it via an approximation algorithm, called streaming algorithm, with provable optimality gap, when all system parameters are known by the NO. In the case where the NO does not know information about the system, we formulated the problem as a sequential decision making problem. We showed that the problem can be cast as a MDP and we used deep RL, namely DQN, to solve it. Our numerical results quantified the performance of the streaming algorithm and DQN in terms of the probability that users get served by the edge, as opposed to being blocked and re-directed towards the cloud which entails larger delay and hence lesser QoS. We showed the resulting resource partitioning between the SPs. We showed that DQN outperforms the baseline resource allocation proportional to users arrival rates and approaches the streaming algorithm. Finally, we included a sensitivity analysis with respect to arrival rates and individual user requirements of a given resource. Note that while the training process of DQN is relatively slow in this chapter, we consider it as a preliminary work, it could be accelerated using a model as in Chapter 3 and Chapter 5. Another possible approach is to use maximum likelihood estimation to estimate the users arrival rates λ_p and service times T_p, for each SP p, and then use the streaming algorithm to solve the problem.

Chapter 5

Resource Pricing for Serverless Edge Computing

5.1 Introduction

Despite its potential, Edge Computing (EC) is rather far from commercial deployment [135]. Existing edge deployments consist of cloud resources deployed at the periphery of the core network (not as far as base stations) and are targeted at corporate users, as a localized version of cloud computing [136]. The slow deployment is due to the lack of a versatile programming abstraction and due to the lack of an appropriate business model, including appropriate pricing schemes.

A promising abstraction for edge computing could be serverless computing, which allows the execution of functions, relieving the users from managing compute and memory resources [137]. Serverless computing has found adoption in Cloud Computing (CC), but the static pricing models that made it popular would not suit edge deployments for several reasons. First, compared to CC, where resources are practically unlimited [138], edge nodes have scarce resources. Hence, some Wireless Devices (WDs) may not get the resources they request immediately, and they may actually leave the edge service area before they would get served. Moreover, cloud nodes benefit from statistical multiplexing [139], i.e., requests coming from broad geographical regions tend to reciprocally compensate fluctuations. This is not true in EC, hence demand for compute resources is more dynamic.

For these reasons, while practically all the big CC providers offer static, usage-based pricing, pricing in EC will have to be *dynamic*, fulfilling two main criteria. First, pricing should be *adaptive*, i.e., it should *learn* how to maximize revenue given information about the workload and the available resources over time. Adaptation should be *fast*, at the time scale of the workload dynamics. Second, pricing should be *transparent* to users, in the sense that the costs and charges associated with the service are presented in a clear and straightforward manner without any hidden fees, ambiguous pricing structures, or complex

calculations, so that they can incorporate pricing information in the long term decisions about whether to rely on EC for executing their tasks. Finding a pricing scheme that is adaptive, transparent and computationally efficient is, however, extremely challenging. In fact, an adaptive pricing scheme would require a perfect knowledge of the workload and of the users behaviour (request rates and prices they are willing to pay). Moreover, price discrimination, where businesses charge different prices to different users based on their willingness to pay or other factors, could complicate pricing transparency efforts. In most optimistic scenarios, where these two criteria were satisfied, it would be computationally heavy as it requires learning all the factors impacting the pricing scheme.

In this chapter, we address this challenge and make the following main contributions:

- We formulate the problem of maximizing the revenue of a serverless edge operator as a sequential decision making problem under uncertainty. In our formulation, prices are piecewise constant over time, and enable non-linear costs in the resources requested.
- We provide a novel Hidden Parameter Markov Decision Process (HiP-MDP) formulation of the problem, and use it to propose a learning scheme that uses a dual Bayesian Neural Network (BNN) approximator for fast and accurate transfer learning, i.e., transferring knowledge gained through learning on several problem instances to a new problem instance, without the need to train the algorithm from scratch. The proposed *Hidden Parameter Edge* (HiPE) pricing algorithm learns *latent variables* that capture the parameters of the dynamics of the problem instance and uses them to parametrize the BNNs used for training a state-of-the-art RL algorithm.
- We use extensive simulations on synthetic and real traces for evaluating the proposed scheme and show that it outperforms state-of-the-art solutions by up to 50% in terms of revenue as well as user received value in using the service. Our pricing scheme overcomes the three challenges mentioned above. First, it adapts itself to the changes of environment parameters (we trained on synthetic traces and we tested directly on real traces). Second, it is transparent, as all the users observe directly the price and are charged with the same price in a pricing period. Third, our pricing scheme is 3 times faster (and then more computationally efficient) than existing learning algorithms.

The rest of this chapter is organized as follows. We describe the system model and the problem formulation in Section 5.2. In Section 5.3, we provide analytical results under simplifying assumptions. In Section 5.4 we propose a HiP-MDP formulation of the problem and a dual BNN approximator based solution for the pricing problem. We provide numerical results in Section 5.5. We conclude the chapter in Section 5.6.

Recall that the work contained in this chapter results from a collaboration with Prof. György Dán from KTH, Sweden, and Feridun Tütüncüoglu, PhD student at KTH, that I carried out during my doctoral mobility in KTH in Spring 2023. The optimal pricing

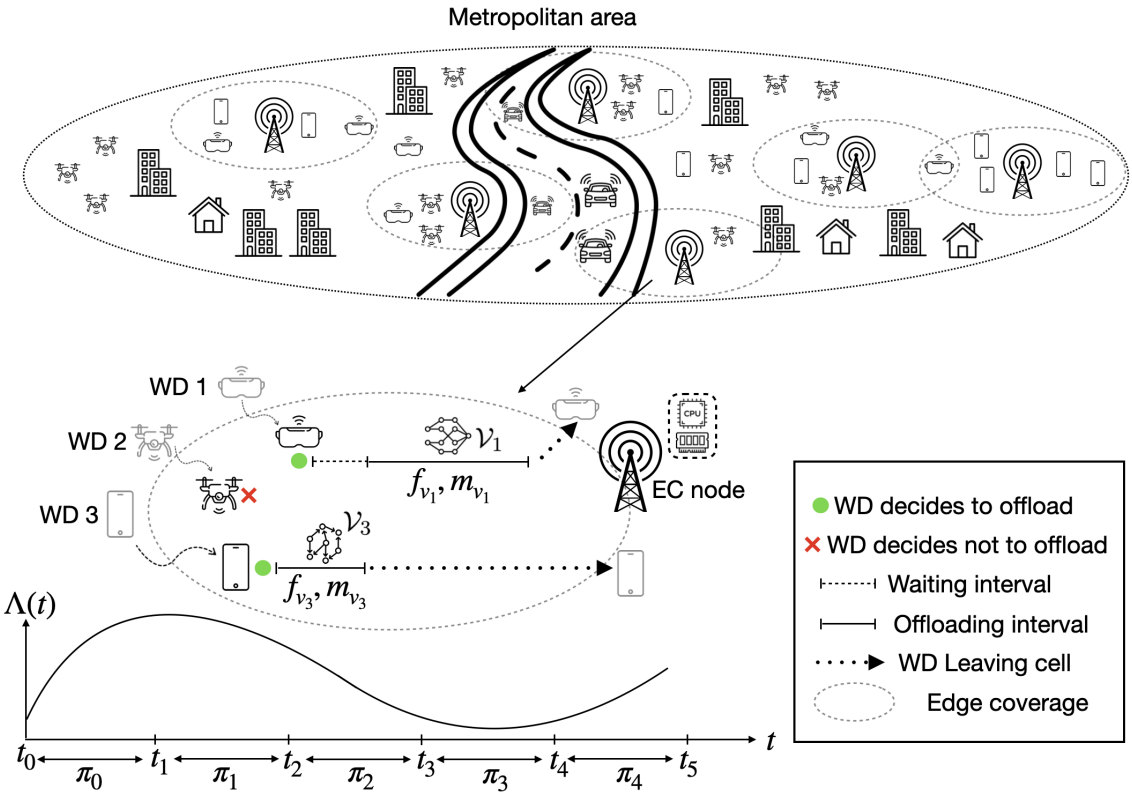


Figure 5.1: Example of a serverless edge service with time-dependent arrival rate $\Lambda(t)$ and piecewise constant price $\boldsymbol{\pi}_k$. Three WDs arrive in pricing periods T_1 (WD 2 and WD 3) and T_2 (WD 1): WD 3 receives service immediately upon arrival at price $\boldsymbol{\pi}_1$, while WD 2 decides not to offload at price $\boldsymbol{\pi}_1$. WD 1 has to wait for receiving service, at price $\boldsymbol{\pi}_2$.

model contained in Section 5.3 was derived by Feridun Tütüncüoglu and will serve as a lower bound to the proposed solution based on HiP-MDP.

5.2 System Model

We consider a Multi-access Edge Computing (MEC) system that provides Function as a Service (FaaS) (also known as serverless) computing [140] to a dynamic population of WDs. Table 5.1 summarizes the notations used in this chapter.

Notation	Definition
i	Index of the user (Section 5.2.1)
α_i	Tasks generation rate of the user i (Section 5.2.1)
D_i	Dwell time of the user i (Section 5.2.1)
t_i^a	Arrival time of user i from the edge area (Section 5.2.1)
t_i^d	Departure time of user i from the edge area (Section 5.2.1)
$\mathcal{F}$	Set of possible CPU frequency allocations (Section 5.2.2)
f_v	Requested CPU frequency allocation for function v (Section 5.2.1)
m_v	Requested memory allocation for function v (Section 5.2.1)
$\tau_v(f_v)$	Execution time of the function v (Section 5.2.1)
$\overline{\tau}_v$	Delay bound of the function v (Section 5.2.1)
$\mathcal{V}_i$	Set of functions of the user i (Section 5.2.1)
F	Total CPU clock frequency capacity (Section 5.2.2)
M	Total Versatile memory capacity (Section 5.2.2)
$\Lambda(t)$	Intensity of users arrival at time t (Section 5.2.1)
T_k	Pricing period $[t_k, t_{k+1})$ (Section 5.2.2)
Δ	Length of $T_k, \forall k$ (Section 5.2.2)
$\pi_k^f(f)$	Price function of CPU frequency f in pricing period T_k (Section 5.2.2)
$\pi_k^m(m)$	Price function of memory m in pricing period T_k (Section 5.2.2)
π_k^r	Unit price per request in pricing period T_k (Section 5.2.2)
$\boldsymbol{\pi}_k$	Pricing action vector in pricing period T_k (Section 5.2.2)
$C_i^{\boldsymbol{\pi}_k}$	Expected unit task offloading cost of user i (Section 5.2.2)
$\overline{C}_i$	Reservation cost of user i (Section 5.2.2)
$C_{i,\Sigma}^{\boldsymbol{\pi}_k}$	Cost of offloading of user i (Section 5.2.2)
o_i	Offloading decision of user i (Section 5.2.3)
Q	Number of servers (Section 5.3.2)

Table 5.1: Frequently used notations in Chapter 5

5.2.1 User Model

WDs arrive to the system following an inhomogeneous Poisson process with intensity $\Lambda(t)$ as shown in Figure 5.1. WD i remains in the service area of the edge service for D_i amount of time, which we refer to as the dwell time of WD i. We assume that D_i is a random variable that follows a certain distribution $\overline{D}$.[1] We denote by t_i^a and $t_i^d = t_i^a + D_i$ the arrival and departure times of WD i, respectively. WD i generates requests at rate $\alpha_i > 0$ while it is in the MEC service area. We assume that α_i is a random variable following a certain distribution $\overline{\alpha}$. Following the FaaS model, we model the task of user i by a set $\mathcal{V}_i$ of functions that need to be executed to perform the task, and we denote by n_v the average number of invocations of function $v \in \mathcal{V}_i$.

User i can request CPU allocation f_v and memory allocation m_v for function $v \in \mathcal{V}_i$, which determine the processing power (in Hz) and memory capacity (in GB) allocated for the function, respectively. The expected execution time $\tau_v(f_v)$ of function $v \in \mathcal{V}_i$ is a convex non-increasing function of the computing power f_v allocated to it [141]. This model generalizes the relation $\tau_v(f_v) = \frac{\mathbb{E}[L_v]}{f_v}$ widely used in edge computing, where $\mathbb{E}[L_v]$ is the expected computational complexity of the function measured in CPU cycles. We include in $\tau_v(f_v)$ the potential impact of storage access latency [142, 143].

Tasks have finite average execution time, i.e., $\tau_i(f_{\mathcal{V}_i}) < \infty$, $f_{\mathcal{V}_i} = \sum_{v \in \mathcal{V}_i} f_v$, and WD i has a constraint $\tau_v(f_v) \leq \overline{\tau}_v$ on the expected execution time of function $\forall v \in \mathcal{V}_i$, determined by the delay bound $\overline{\tau}_i$ of its task.

In order to be focused on the pricing of computational resources and not on the wireless channel, we assume users can send and receive information without paying at each packet transmission (as in nowadays contracts). We also assume good channel conditions, such that packet loss is negligible. We focus on computation-intensive scenarios, where the wireless bandwidth is not the bottleneck (in Section 5.5 the bandwidth consumed is 20 times smaller than the available channel capacity). Observe that in reality what matters to the user is the total offload delay (transmission + execution). We thus implicitly assume that delay bound $\overline{\tau}_i$ is obtained by subtracting the transmission time from the maximum tolerable total offload time.

5.2.2 Edge Resources, Pricing and Offloading

We consider that the operator maintains an edge cloud with CPU capacity F and memory M in the service area [144]. Aligned with common FaaS offerings (e.g AWS Lambda, Google Cloud Functions), we consider that the operator offers a set $\mathcal{F} = \{f_{(1)}, f_{(2)}, \ldots, \overline{f}\}$ of possible amounts of CPU (in Hz) that can be allocated a to function. When requesting to execute a function v at the edge, the WD chooses one of the values contained in $\mathcal{F}$ [145].

[1] Observe that in practice, a WD could enter the service area, exits and then enters again. For the sake of simplicity, we will consider this case as there were two different WDs, entering the service area at different time.

Similarly, WDs can choose from the set $\mathcal{M} = \{m_{(1)}, m_{(2)}, \ldots, \overline{m}\}$ of memory allocations for each function. Naturally, $\overline{f} \leq F$ and $\overline{m} \leq M$.

Similar to existing FaaS pricing models, we consider that pricing is based on the execution time, on the amount of used resources and on the number of function invocations. The operator sets the prices periodically, at time instants $t_0, t_1 = t_0 + \Delta, \ldots$; the price is constant during pricing period $T_k = [t_k, t_{k+1})$ so as to make the cost of using the edge service more predictable for the users than under user specific pricing schemes considered in previous works [146, 147], where users are charged based on different prices in a same pricing period and prices change for one user from one pricing period to another.

Contrary to existing FaaS pricing models, we consider a general, non-linear pricing model. The price of compute capacity for CPU allocation $f \in \mathcal{F}$ in pricing period T_k is $\pi_k^f(f) = f^{\gamma_k}\pi_k^f$ and the price of memory allocation $m \in \mathcal{M}$ is $\pi_k^m(m) = m^{\gamma_k}\pi^m$, where π_k^f and π_k^m are the unit cost of compute power and memory, respectively, and $\gamma_k \geq 1$ is an exponent. Observe that for $\gamma_k = 1$ pricing is linear, as in current FaaS offerings. We denote by π_k^r the price paid at each function invocation during pricing period T_k. Such price π_k^r is independent from the nature of functions invoked and the resources they consume. We use the shorthand notation $\boldsymbol{\pi}_k = (\gamma_k, \pi_k^f, \pi_k^m, \pi_k^r)$ to denote the pricing discipline that the operator imposes during time slot k.

If WD i arrives during pricing period T_k then it will be charged based on the price $\boldsymbol{\pi}_k$ throughout its dwell time D_i, should it decide to offload. The expected *task unit* cost of WD i that arrives in pricing period k for offloading a task is then

$$C_i^{\boldsymbol{\pi}_k}((f_v)_{v\in\mathcal{V}_i}, (m_v)_{v\in\mathcal{V}_i}) = \sum_{v\in\mathcal{V}_i} n_v \cdot \Big(\pi_k^r + \tau_v(f_v)(\pi_k^f(f_v) + \pi_k^m(m_v))\Big) \tag{5.1}$$

We denote by $\overline{C}_i$ the reservation cost of WD i, i.e., its valuation for offloading its task. $\overline{C}_i$ is unknown to the operator, and we model it as a random variable with distribution $\overline{C}$. WD i decides to offload if the unit cost satisfies

$$C_i^{\boldsymbol{\pi}_k}((f_v)_{v\in\mathcal{V}_i}, (m_v)_{v\in\mathcal{V}_i}) \leq \overline{C}_i, \tag{5.2}$$

otherwise the WD executes the computation within the device or discards its tasks. We denote by $o_i \in \{0, 1\}$ the decision of WD i.

Due to edge resource constraints, even if WD i decides to offload, it may not be able to do so immediately, but it may have to wait in a FIFO queue until resources become available. A waiting WD does not offload, hence it does not have to pay for offloading during waiting, and if its dwell time D_i expires during the waiting phase, the WD will leave the system without ever offloading. We denote by t_i^o the time instant when WD i can start offloading, when enough memory and CPU are available at the edge, hence $t_i^a \leq t_i^o \leq t_i^d$ if $o_i = 1$, and we define $t_i^o = t_i^d$ if $o_i = 0$. We use these to define the active time of WD i during pricing period T_k as:

$$\mathcal{T}_{i,k}^a = \min(t_i^d, t_{k+1}) - \max(t_i^o, t_k) \tag{5.3}$$

and its total active time as:

$$\mathcal{T}_i^a = (t_i^d - t_i^o) \tag{5.4}$$

The active time, the task arrival rate and the unit task offloading cost together determine the expected cost of offloading of WD i:

$$C_{i,\Sigma}^{\boldsymbol{\pi}_k}((f_v)_{v\in\mathcal{V}_i},(m_v)_{v\in\mathcal{V}_i}) = \mathcal{T}_i^a \alpha_i C_i^{\boldsymbol{\pi}_k}((f_v)_{v\in\mathcal{V}_i},(m_v)_{v\in\mathcal{V}_i}) \tag{5.5}$$

5.2.3 Problem Formulation

We consider that the WDs and the operator are profit maximizing entities. The goal of WD i is to minimize its cost of offloading subject to the task latency constraint $\overline{\tau}_i$, i.e.,

$$\max_{(m_v,f_v)_{v\in\mathcal{V}_i}} o_i(t_i^a)(\overline{C}_i - C_i^{\boldsymbol{\pi}_k}((f_v)_{v\in\mathcal{V}_i},(m_v)_{v\in\mathcal{V}_i})) \tag{5.6}$$

$$s.t \quad o_i(t_i^a)\tau_v(f_v,m_v) \leq \overline{\tau}_v, \forall v \in \mathcal{V}_i, \tag{5.7}$$

i.e., WD i chooses the CPU frequencies $(f_v)_{v\in\mathcal{V}_i}$ that minimize its cost (5.1), while it chooses the smallest amount of memory that allows function execution.

In pricing period k the operator collects revenue from the WDs that offload in the period. Recall that if $t_i^a \in T_k$ then WD i is charged based on the price $\boldsymbol{\pi}_k$ upon its arrival. We can thus express the revenue, in pricing period T_k, as

$$\rho_k^\theta = \sum_{k'=0}^{k} \sum_{i\in\mathcal{N}_{k'}} \mathcal{T}_{i,k}^a \alpha_i C_i^{\boldsymbol{\pi}_{k'}}((f_v)_{v\in\mathcal{V}_i},(m_v)_{v\in\mathcal{V}_i}), \tag{5.8}$$

where $\mathcal{N}_{k'} = \{i | t_i^a \in T_{k'}\}$, i.e., the set of users arrived in pricing period $T_{k'}$, $\mathcal{T}_{i,k}^a$ is the active time of user i in pricing period T_k (5.3) and θ is the pricing policy until time slot k, i.e., $\theta = \{\boldsymbol{\pi}_{k'}\}_{k'=1,\ldots,k}$. Observe that ρ_k^θ is a random variable, as it depends on the workload.

The operator's objective is to maximize its expected mean revenue by finding a policy

$$\theta^* = \arg\max_{\theta\in\Theta} \lim_{K\to\infty} \frac{1}{K} \sum_{k=0}^{K} \mathbb{E}[\rho_k^\theta], \tag{5.9}$$

where Θ is the set of causal pricing policies. Observe that the operator's problem is a sequential decision making problem under uncertainty, where the uncertainty is due to the randomness of the arrivals, departures, reservation costs and resource requirements of the WDs. In what follows we first provide analytical results under simplifying assumptions (Section 5.3). We then propose a pricing policy based on a semi-parametric approach as a solution to the general case (Section 5.4).

5.3 Analytical Results

To obtain insight into the structure of optimal policies, we start with characterizing the best response of the users for a given pricing policy, we then turn to the analysis of pricing policies. The result contained in this section were obtained by Feridun Tütüncüoglu. I report it here to make the manuscript self-contained. Hence, I will omit the proof.

5.3.1 User Best Response Characterization

Observe that the execution time is independent of the memory allocation, hence WD i requests the smallest amount of memory m_v^* that allows function execution. Nonetheless, the dependence of the cost on the amount of memory makes the optimal choice of the CPU frequency non-trivial.

Lemma 5.3.1.1. *Assume a non-linear pricing model where $\pi_k^f = f_v^\gamma \pi^f$ and $\pi_k^m = m_v^\gamma \pi^m$. The CPU frequency that minimizes the cost $C_i^{\boldsymbol{\pi}_k}(f_v, m_v)$ of WD i for function $v \in \mathcal{V}_i$ is*

$$f_v^* = \begin{cases} \arg\min_{f_v \in \{\tilde{f}_v^-, \tilde{f}_v^+\}} C_i^{\boldsymbol{\pi}_k}(f_v, m_v^*) & \text{if } \tilde{f}_v^- \geq \frac{\mathbb{E}[L_v]}{\tau_v} \\ \min\{\ f_v \geq \frac{\mathbb{E}[L_v]}{\tau_v}, f_v \in \mathcal{F}\} & \text{otherwise.} \end{cases} \tag{5.10}$$

where $\tilde{f}_v^-, \tilde{f}_v^+$ are the two adjacent values in $\mathcal{F}$, such that

$$\tilde{f}_v^- \leq \left(\tilde{f}_v^* = \begin{cases} m_v^* \left(\frac{\pi^m}{(\gamma-1)\pi^f} \right)^{\frac{1}{\gamma}} & \text{if } \gamma > 1, \\ \overline{f} & \text{if } \gamma = 1, \end{cases} \right) \leq \tilde{f}_v^+ \tag{5.11}$$

5.3.2 Optimal Pricing and Reward in Steady State

We now turn to the analysis of the optimal price, under assumptions that allow analytical tractability. We will abandon these assumptions in Section 5.4.

Assumption 5.3.2.1. *WD arrivals follow a homogeneous Poisson process with intensity Λ. Dwell times D_i are exponentially distributed with the same mean $1/\mu$.*

Assumption 5.3.2.1 enables us to streamline a simple user queue model and achieve a stationary solution for that queue.

Assumption 5.3.2.2. *WD i has a single function, i.e., $|\mathcal{V}_i| = 1$. All WDs request the same CPU frequency $\overline{f}$ and memory $\overline{m}$.*

Assumption 5.3.2.2 assumes a homogeneous resource allocation among arriving users, which simplifies the queue model in our analysis.

Assumption 5.3.2.3. *Let $Q = \lfloor F/\overline{f} \rfloor$. Then $M \geq Q\overline{m}$, i.e., the edge system is not memory constrained.*

Assumption 5.3.2.4. *Reservation costs $\overline{C}_i$ are uniformly distributed on $[0, b], b > 0$.*

Assumption 5.3.2.5. *The operator has the complete knowledge of system parameters, i.e., Λ, $\frac{1}{\mu}$, $\overline{f}$, $\overline{m}$ and b.*

Proposition 5.3.2.6. *Assume pricing is linear, i.e., $\gamma_k = 1, \forall k$. Under the above assumptions, the optimal price can be written as follows:*

$$\pi^{f*} = \pi^{m*} = \pi^{r*} = \pi^* = b\frac{\mathcal{L}_0(e^{\chi+1}) - 1}{B\chi}, \tag{5.12}$$

where $\mathcal{L}_0(.)$ is the principal branch of the Lambert function [148], $\chi = \frac{\Lambda}{Q\mu}$ and $B = \left(\frac{\mathbb{E}[L]}{\overline{f}} \cdot (\overline{f} + \overline{m}) + 1\right)$.

5.4 Data-driven Optimization

Recall that problem (5.9) faced by the operator is a sequential decision making problem under uncertainty. A straightforward approach would be to formulate the problem as a MDP and use model-free reinforcement learning (RL) for solving it. This approach may work for a single edge deployment with a stationary workload, but a new policy would have to be learned for each edge deployment or for each workload evolution.

5.4.1 Hidden Parameter MDP Formulation

We propose to address this challenge by following a semi-parametric approach, formulating the operator's problem as a HiP-MDP. A HiP-MDP is a class of MDPs represented by a tuple $\langle \mathcal{S}, \mathcal{A}, W, T, R, \tilde{\gamma}, P_W \rangle$, where $\mathcal{S} \subseteq \mathbb{R}^N, N \in \mathbb{N}^+$, $\mathcal{A} \subseteq \mathbb{R}^N$ and $\tilde{\gamma}$ are the state space, the action set and the discount factor, respectively, as in a MDP.

The transition function $s_{k+1} \sim T(s_{k+1}|s_k, a_k, w_g)$ and the reward function $r_k \sim R(s_k, a_k, w_g^r)$ are, however, parametrized by w_g and w_g^r, respectively, which are drawn from prior distribution P_W and are not observable. g is called environment instance, it represents one single MDP. Observe that while in previous HiP-MDP formulations, the reward was given in closed form [149, 150], in our formulation reward is stochastic. A HiP-MDP defines a class of problems; a particular problem instance (a MDP) is obtained once the parameters w_g and w_g^r are drawn. A HiP-MDP is different from a Partially Observable MDP (POMDP) as the state is observable, but the environment dynamics and reward are parametrized and the learning agent has to estimate the parameters based on interaction with the environment, while maximizing its reward.

We argue that this semi-parametric approach is a powerful abstraction for the considered EC pricing problem. HiP-MDP leverages the intuition that the state transition and the reward in the underlying queueing system can be approximated by a family of

functions, parametrized by w_g and w_g^r. This allows learning a family of policies, which is valid for a large set of scenarios, each corresponding to an edge deployment and workload profile. When dealing with a new scenario, HiP-MDP allows to "transfer" the already learned policies to this new scenario. This is obtained by adapting the policies to the new scenario by adjusting w_g and w_g^r.

In fact, the considered problem can be modeled as a HiP-MDP, as we show next.

Proposition 5.4.1.1. *Under Assumptions 5.3.2.1, 5.3.2.2 and 5.3.2.3 problem* (5.9) *is a HiP-MDP, with:*

- *queue state* $(N_a^k + N_w^k)$ *where* N_a^k *is the number of active users and* N_w^k *is the number of waiting users, for each time slot* k*.*
- *action* $a_k = (\pi_k^f, \pi_k^m, \pi_k^r, \gamma_k)$*, i.e., pricing decision, for each time slot* k*.*
- *reward* $R_k^{\boldsymbol{\pi}_k} = \sum_{i \in \mathcal{N}_k} C_{i,\Sigma}^{\boldsymbol{\pi}_k}((f_v)_{v \in \mathcal{V}_i}, (m_v)_{v \in \mathcal{V}_i})$*, i.e., revenue from WDs accepted during pricing period* T_k*.*
- *and latent (hidden) parameter vector* $w = f(\overline{D}, \Lambda(t), \overline{\alpha}, \overline{C}, Q)$*.*

Proof. The system with the considered state representation is Markovian due to Assumptions 5.3.2.1, 5.3.2.2 and 5.3.2.3. The transition function depends on the state, the action, and the system parameters $(\overline{D}, \Lambda(t), \overline{\alpha}, \overline{C}, Q)$ [151]. Nonetheless, the reward ρ_k^θ defined in (5.8) during pricing period T_k does not only depend on the state and the action, but also on past actions, and is hence not Markovian. To provide a Markovian formulation, we have defined the revenue from WDs accepted during pricing period T_k

$$R_k^{\boldsymbol{\pi}_k} = \sum_{i \in \mathcal{N}_k} C_{i,\Sigma}^{\boldsymbol{\pi}_k}((f_v)_{v \in \mathcal{V}_i}, (m_v)_{v \in \mathcal{V}_i}). \tag{5.13}$$

Observe that $R_k^{\boldsymbol{\pi}_k}$ only depends on the state, the action and the system parameters. Furthermore, maximizing $\lim_{K \to \infty} \frac{1}{K} \sum_{k=1}^{K} \mathbb{E}[R_k^{\boldsymbol{\pi}_k}]$ is equivalent to solving (5.9). Hence problem (5.9) is a HiP-MDP. □

5.4.2 Dual Bayesian Neural Network (BNN) Approximation

For the above HiP-MDP problem formulation, we propose to use two BNNs as function approximators,

$$s' \sim \hat{T}^{(BNN)}(s, a, w_g) + \epsilon_t \tag{5.14}$$

$$r \sim \hat{R}^{(BNN)}(s, a, w_g^r) + \epsilon_r \tag{5.15}$$

$$\epsilon_t, \epsilon_r \sim \mathcal{N}(0, \sigma_n^2). \tag{5.16}$$

where ϵ_t and ϵ_r are approximation errors for transition function approximator $\hat{T}^{(BNN)}$ and reward function approximator $\hat{R}^{(BNN)}$, respectively. Note that both the approximators for the transition (5.14) and the reward (5.15) functions are parametrized by environment instance g. Furthermore, the latent parameters w_g and w_g^r are used as input to the functions approximators $\hat{T}^{(BNN)}$ and $\hat{R}^{(BNN)}$, respectively, and are continuously updated during training. Importantly, we allow parameters w_g and w_g^r used for the two approximators to be different. Indeed, they are low dimensional representations of system latent parameter vector $w = f(\overline{D}, \Lambda(t), \overline{\alpha}, \overline{C}, Q)$, which, for a same environment instance g, may be different.

BNNs, are a type of neural network where the weights (parameters) are treated as random variables with a certain prior distribution P_W. The use of probabilistic models, such as BNNs, can help alleviate the overfitting problem commonly associated with traditional neural networks when dealing with limited amounts of data [152, 153]. In the context of accelerating the learning of a good policy with minimal interaction with the real environment, BNNs are employed. For instance, in [150], a feed-forward BNN is utilized to model the transition function $T(s_{k+1}|s_k, a_k, w_g)$. However, in contrast to the algorithm proposed in [150], where the reward is assumed to be known by the operator, an additional BNN is employed in our scenario to infer the dynamics of the stochastic reward function $R(s_k, a_k, w_g^r)$, which is not directly observed by the operator (delayed reward). We set the weights of the BNNs as random variables with some prior P_W and we place independent Gaussian prior on each weight, i.e., $P_W(\mathcal{W}) = \prod_{w \in \mathcal{W}} \mathcal{N}(w; \mu, \sigma)$.

Algorithm 3 shows the procedure for learning a policy using the proposed approach. We assume a set $\mathcal{G}_{tra}$ of pre-training problem instances is available. For every problem instance g, we collect a replay buffer $\mathcal{D}_g$ with the observed transitions and rewards (s, a, s', r). The global replay buffer is $\mathcal{D} = \bigcup_{g \in \mathcal{G}_{tra}} \mathcal{D}_g$. We also learn a posterior distribution of the weights $\mathcal{W}^t$ and $\mathcal{W}^r$ of the BNNs using $\mathcal{D}$ (Line 1). In particular, for any problem instance g, the algorithm first aims at determining the latent embeddings w_g and w_g^r, based on observations $(s, a, s', r) \in \mathcal{D}_g$ (Lines 6-9). It does so, by minimizing the α-divergence[2] of the observed transitions and rewards and the ones predicted by $\hat{T}^{(BNN)}(s, a, w_g)$ and $\hat{R}^{(BNN)}(s, a, w_g^r)$, respectively [154] (Lines 10-12). The algorithm then uses functions (5.14)-(5.15) parametrized by w_g and w_g^r for generating fictional transitions and rewards, which are collected in replay buffer $\mathcal{D}_g^f$. Such fictional transitions and rewards represent the environment on which the RL agent $\hat{\pi}_g$ is trained (Lines 13-15). Figure 5.2 is a graphical representation of the workflow of our proposed algorithm.

It is worth emphasizing that Algorithm 3 involves two phases of training to learn the policy, for any new environment instance g:

[2] α-divergence is a family of mathematical functions to measure the dissimilarity or divergence between two probability distributions. It is a generalization of the well-known divergence measure the Kullback-Leibler divergence ($\alpha = 1$).

Algorithm 3: Hidden Parameter Edge (HiPE) pricing algorithm

Compute weights of the BNNs $\mathcal{W}^t, \mathcal{W}^r$ using $\mathcal{G}_{tra}$;
Draw $w_g, w_g^r \sim P_W$ for the new environment;
Randomly initialize policy $\hat{\pi}_g$;
Initialize model, replay and fictional buffers $\mathcal{D}_g, \mathcal{D}_g^f$;
for t ***from*** 1 ***to*** N_T // N_T is the total number of episodes
do
  while $t < t^{update}$ // the algorithm do not reach the time of fictional update yet
  do
    for i ***from*** 0 *to* $\frac{episode\_length}{\Delta}$ **do**
      Take action $a \leftarrow \hat{\pi}_g(s)$;
      $\mathcal{D}_g \leftarrow (s, a, r, s', w_g, w_g^r)$;
    end
  end
  if $\hat{T}_g^{(BNN)}$ *and* $\hat{R}_g^{(BNN)}$ *are inaccurate* // α-divergence minimization
  then
    TRAIN-BNN$(\mathcal{D}_g, \mathcal{W}_g^r, \mathcal{W}_g^s, w_g, w_g^r)$ // Procedure 4
  end
  if $t \,\%\, t^{update} == 0$ **then**
    FICTIONAL-TRAIN$(\mathcal{D}_g^f, \mathcal{W}_g^r, \mathcal{W}_g^s, w_g, w_g^r)$ // Procedure 5
  end
end

Procedure 4: TRAIN-BNN

Data: $\mathcal{D}_g, \mathcal{W}_g^t, \mathcal{W}_g^r, w_g, w_g^r, N_{obs}$
for k ***from*** 0 *to* N_{obs} // for tuning the BNNs for the new instance g, $N_{obs} = (t^{update} \times \frac{\text{episode\_length}}{\Delta})$
do
  Update w_g using $\mathcal{D}_g$;
  Update w_g^r using $\mathcal{D}_g$;
  Update $\mathcal{W}_g^s, \mathcal{W}_g^r$ using $\mathcal{D}_g$;
end

Procedure 5: FICTIONAL-TRAIN

Data: $\mathcal{D}_g^f, \mathcal{W}_g^t, \mathcal{W}_g^r, w_g, w_g^r, N_f$

for t ***from*** 0 *to* N_f *episodes* **do**
  for i ***from*** 0 *to* $\frac{episode\_length}{\Delta}$ **do**
    Take action $a \leftarrow \hat{\pi}_g(s)$;
    Estimate next state $\hat{s}' \leftarrow \hat{T}(s, a, w_g)$;
    Estimate reward $\hat{r} \leftarrow \hat{R}(s, a, w_g^r)$;
    Store $\mathcal{D}_g^f \leftarrow (s, a, \hat{r}, \hat{s}')$;
    if $t \,\%\, t^{update} == 0$ **then**
      Update $\hat{\pi}_g$ using $\mathcal{D}_g^f$;
    end
  end
end

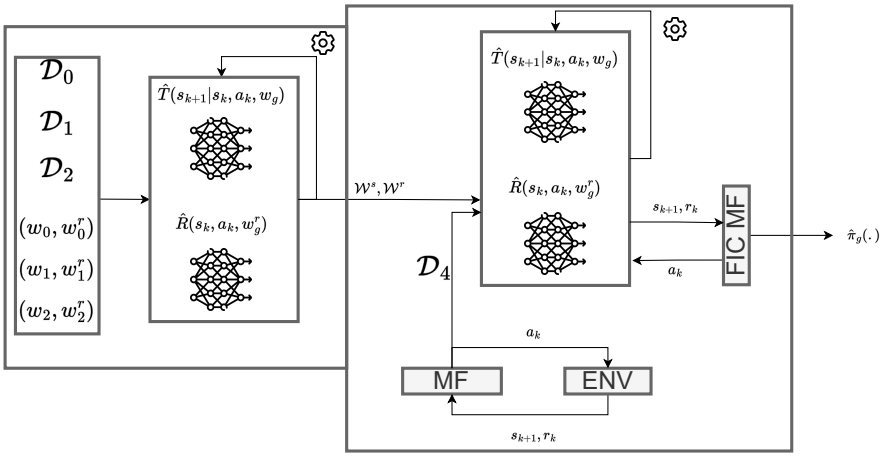


Figure 5.2: The workflow of proposed Hidden Parameter Edge (HiPE) pricing algorithm

1. The first phase of training happens before t^{update} episodes with direct interaction with the real environment. This training is as if we are updating the policy using state-of-the-art model-free RL.

2. The second phase of training happens after t^{update} episodes based on fictional transitions that are not happening in the real environment, generated by $\hat{T}_g^{(BNN)}$ and $\hat{R}_g^{(BNN)}$, for N_f episodes.

Note that the two BNNs $\hat{T}^{(BNN)}$ and $\hat{R}^{(BNN)}$ are pre-trained before this above process. After t^{update} episodes, $\hat{T}^{(BNN)}$ and $\hat{R}^{(BNN)}$ are tuned based on the real transitions that happened until the $(t^{update})^{-th}$ episode resulting in $\hat{T}_g^{(BNN)}$ and $\hat{R}_g^{(BNN)}$, the transition and reward functions approximators corresponding to the new environment instance g. Observe that simulated environments can generate a vast amount of transitions very fast and at a lower cost compared to transitions from real-world interactions. This allows the RL agent to learn more efficiently, especially in our case where we aim to maximize the NO revenue during real environment transitions. This process allows the RL agent to experience and learn from a large number of episodes in a shorter amount of time. We will show in Section 5.5.4 that this accelerated learning is crucial for rapid convergence.

5.5 Numerical Results

We use simulations on synthetic and on measured traces for evaluating the performance of the proposed dual BNN HiP-MDP (Section 5.4).

5.5.1 Evaluation Scenario

Table 5.2 summarizes the parameters used for the evaluation. Observe that in the scenarios considered, on average ∼350 bits of information would be sent in the air for each function invocation. Moreover, the maximum number of users active in a cell is 120 across all considered scenarios. In the worst case in which users offload 4 functions each, each invoked 3 times,[3] offloading would occupy a wireless capacity of 500 Kbps, which is well below the minimum data rate 10 Gbps of 5G [161]. For synthetic traces, we use in our simulations two dwell time distributions: deterministic and exponential. For measured traces, we choose 3 cells from the Greater Shanghai metropolitan area traces [162, 163, 164] (see Figure 5.3):

- Cell 1 is in the city center with high offered load
- Cell 2 is in a suburban area with medium offered load
- Cell 3 is in a rural area with low offered load

[3]These are the maximum number of functions and the maximum number of function invocations in all the scenario.

Parameter	Value	Motivation
Capacities	Compute: $\overline{f} = 120$ GHz Memory: $\overline{m} = 300$ GB	Cluster of 10 compact edge servers [155, 156]
CPU allocations	$\mathcal{F} = \{1, 1.3, 1.6, .., 4\}$ Ghz	Set of possible values (Section 5.2.2)
Memory required	Uniformly at random from $\mathcal{M} = \{1, 1.2, 1.4, .., 3\}$GB	See Section 5.2.2
Requested functions	$\lvert\mathcal{V}_i\rvert$ generated unif. at random in {1, 2, 3, 4}	Number of functions requested by user i
Invocations	n_v uniformly distributed in $[1, 3]$	Average number of function invocations
Computation complexity	L_v exponentially distributed with mean 0.01 GCycles	If 2 instr/cycle [157] $\Rightarrow$ 10^7 instr per function (as in [158, Figure 4])
Compute density	30 Kcycles/bit ([159, Table II])	Offloading is viable [160, Figure 4]
Delay bound	$\overline{\tau}_v$ generated uniformly at random in [5, 10] ms	Augmented reality applications [128]
Reservation cost	$\overline{C}_i$ from a unif. distr. on $[0, 0.001]$ \$ or from a truncated Gaussian on $[0, 0.001]$ \$	Values typical of Amazon serverless offerings

Table 5.2: Summary of evaluation parameters.

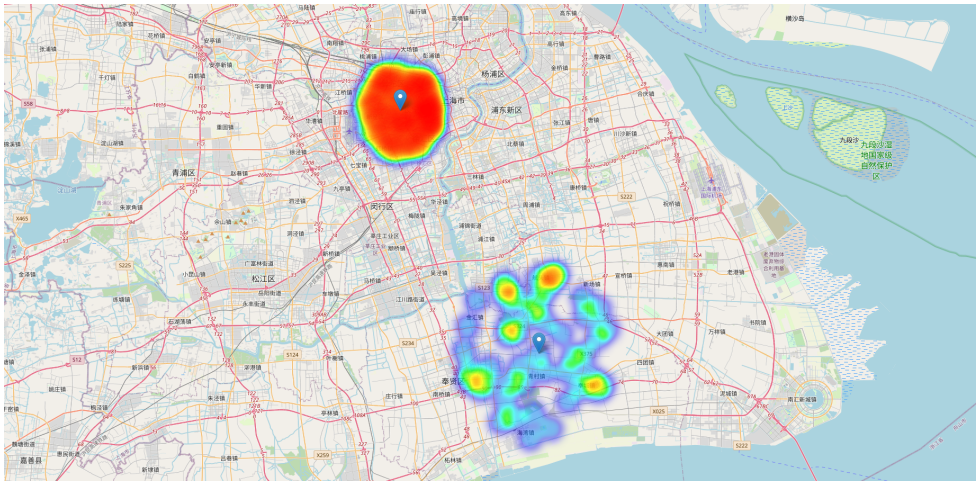


Figure 5.3: User locations (heatmap) and base station markers visualized on a map of Shanghai, collected over a 6-month period from June 1, 2017, to December 1, 2017

We consider three variants of the pricing model:

1. Univariate pricing ($\pi_k^f = \pi_k^m = \pi_k^r = \pi_k, \gamma_k = 1$)
2. Multivariate linear pricing ($\pi_k^f, \pi_k^m, \pi_k^r, \gamma_k = 1$)
3. Non-linear pricing ($\pi_k^f, \pi_k^m, \pi_k^r, \gamma_k \geq 1$)

We set the length of the pricing periods to $\Delta = 1$ hour. For the HiP-MDP, we use $s_k = \{(\rho_{\text{CPU}}^k, \rho_{\text{mem}}^k, k)\} \in \mathcal{S} \subseteq [0,1]^2 \times \mathbb{N}^+$ as the state, where ρ_{CPU}^k and ρ_{mem}^k are the CPU and memory utilization at time slot k, respectively, and k is the index of pricing period T_k in a day. The choice of the state is motivated by that ρ_{CPU}^k and ρ_{mem}^k capture the congestion on computation and memory resources, which in turn determine the reward in a pricing period T_k. The use of the time index of the period in a day is motivated by the periodicity of user arrivals observed in real data. We found this to be a concise state representation that allows fast convergence. We define the action to be $a_k = \{\pi_k^f, \pi_k^m, \pi_k^r, \gamma_k\} \in \mathcal{A} \subseteq \mathbb{R}_+^4$, and the reward $R(s_k, a_k, w_g^r)$ as the revenue collected from WDs accepted during pricing period T_k. In the proposed algorithm we use Soft Actor-Critic (SAC) to learn the policy (Procedure 5) and we use $t^{update} = 25$ days to tune the BNNs in the unseen environment, where we assume one episode is 1 day, and apply $N_f = 60$ episodes of fictional updates. We use the default hyper-parameters in the stable baselines library [165].

To approximate the transition and reward dynamics for an environment instance g, we use a 2-layer neural network architecture, where each layer contains 25 neurons with Gaussian priors on the weights. For pre-training the BNNs, we collect transition samples from synthetic traces with exponentially distributed dwell times with mean

$\{180, 720, 1200, 1800, 2400, 3600, 5400\}$, uniformly distributed reservation cost, while the rest of the parameters are as shown in Table 5.2. We use $\mathbb{R}^5$ for the latent parameter space, i.e., $w_g, w_g^r \in \mathbb{R}^5$. Increasing the dimension of the latent variables increases the computational complexity, whereas choosing low dimensionality results in limited representation of the environment instances g, which negatively affects transferability. We learn the latent parameters w_g and w_g^r and the network weights by minimizing the α-divergence using ADAM with $\alpha = 1$ (we aim to globally fit $\hat{T}^{(BNN)}$ and $\hat{R}^{(BNN)}$) [166].

We use four baselines for comparison:

- The first baseline is the pricing scheme for the online Knapsack problem proposed in [147], which is a user-specific pricing scheme based on the instantaneous system load. This pricing scheme performs price discrimination since it charges different prices for the same resources to different users based on their willingness to pay, in order to maximize the operator profit. Such strategy violates one of our desiderata, i.e., transparency.

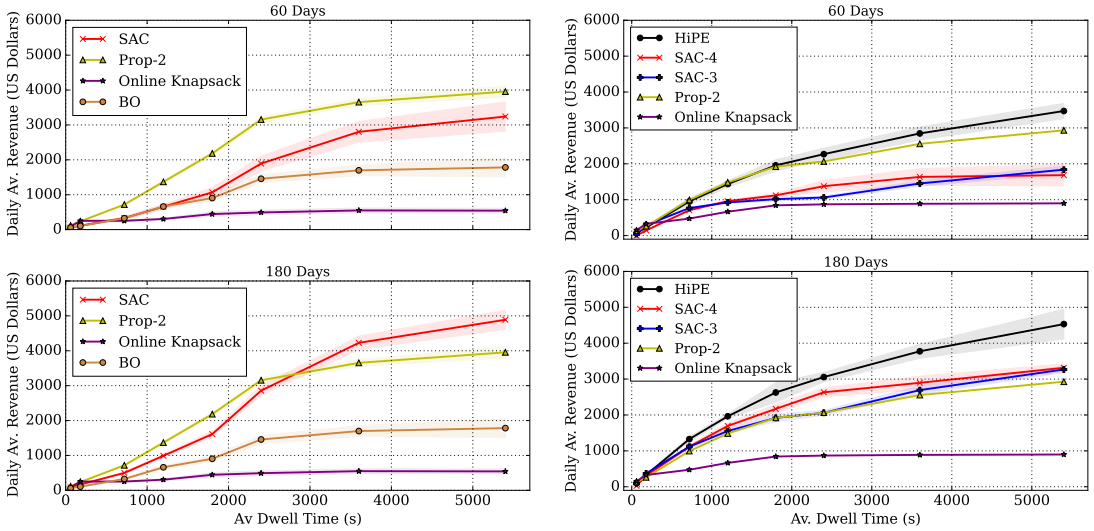
- The second baseline is using Bayesian Optimization (BO) for each time index of a day, as proposed in [167], where we assign an agent to each time index of a day, and each agent maximizes the expected revenue using a Gaussian process approximation.

- The third baseline is based on the bound in Proposition 5.3.2.6 where we use (5.12) as a starting price and implement a gradient ascent algorithm (labeled as Prop-2 in the figures). Recall that this baseline is an ideal baseline that is valid only under the idealistic assumptions 5.3.2.1, 5.3.2.2, 5.3.2.3, 5.3.2.4 and 5.3.2.5 hold, which is not true in real world scenario.

- The fourth baseline is a model-free RL agent using the SAC algorithm [168] with state $s_k = \{(\rho_{\text{CPU}}^{(k)}, \rho_{\text{mem}}^{(k)}, k)\} \in \mathcal{S} \subseteq [0,1]^2 \times \mathbb{N}^+$, action $a_k = \{\pi_k^f, \pi_k^m, \pi_k^r, \gamma_k\} \in \mathcal{A} \subseteq \mathbb{R}_+^4$ and reward $R_k^{\boldsymbol{\pi}_k}$ defined by (5.13).

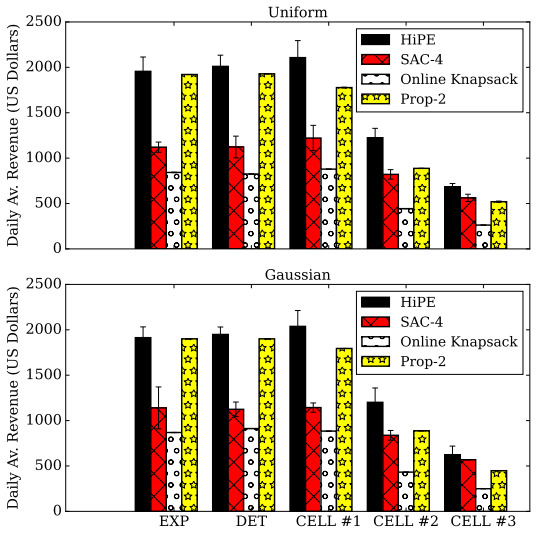
5.5.2 Operator Revenue

Figure 5.4a shows the daily average revenue of the edge operator as a function of average dwell time over a period of 60 and 180 days based on synthetic traces that satisfy Assumptions 5.3.2.1, 5.3.2.2, and 5.3.2.3 (homogeneous resource allocation): $\overline{f} = 2$ GHz, $\overline{m} = 1$ GB and $Q = 10$. We use the univariate pricing model and uniform reservation cost distribution to evaluate the accuracy of the analytical approximation (Prop-2). We observe that over 60 days the analytical approximation has the best revenue and thus it is an accurate approximation. For 180 days, it performs best for low average dwell times, when the mixing times (the amount of time it takes for the system to reach a state where it closely approximates its steady-state behavior) are short and the steady state approximation is accurate, but its revenue is not far from that of SAC even for long dwell times. The



(a) Average daily revenue vs. average dwell time under homogeneous resource allocation.

(b) Average daily revenue vs. average dwell time under heterogeneous resource allocation.



(c) Average daily revenue for two synthetic dwell time distributions ($\mathbb{E}[D_i] = 1800$ Section) and trace-based distributions, and for two reservation cost distributions (Uniform and truncated Gaussian) for 60 days of training.

Figure 5.4: Average daily revenue for synthetic and real traces with various dwell time distributions.

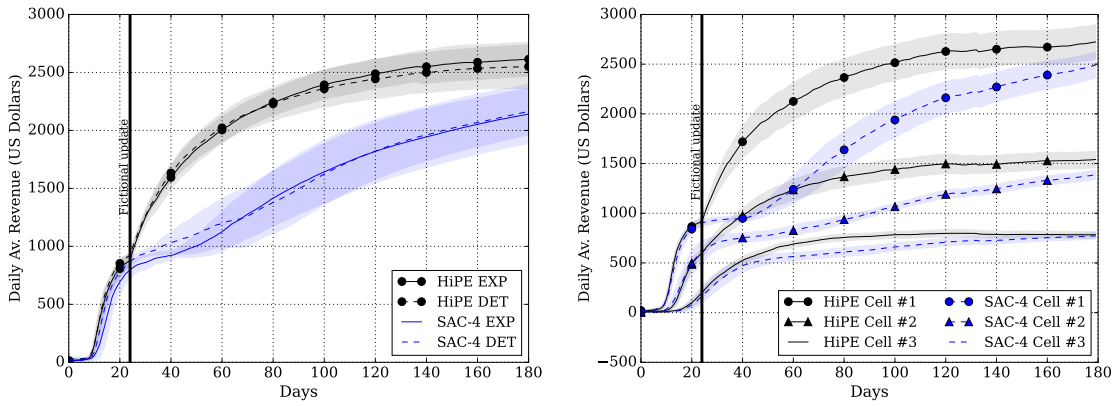
BO scheme, which learns a price for each pricing period, fails to find an effective pricing policy. We can also observe that the online knapsack algorithm does not work well either; it consistently offers high prices with increasing system load, resulting in the rejection of too many WDs (see Section 5.5.5). These results show that pricing based on the steady state approximation works rather well when the modeling assumptions are satisfied, but its advantage is mainly due to that it does not have to learn the system parameters. We now turn to more complex scenarios.

Since BO has to learn a price for each pricing period, involving training an agent for each pricing period, we omit the comparison with this approach in the following results.

Figure 5.4b shows the daily average revenue of the operator as a function of the average dwell time for 60 days and 180 days of simulation, for heterogeneous resource allocation. The figure shows results for multivariate (SAC-3) and non-linear (SAC-4) pricing using *SAC* (the numbers denote the dimension of the action of the *SAC*), to assess the advantage of non-linear pricing. The figure shows that the proposed *HiPE* pricing scheme outperforms all baselines, with an increasing margin as the average dwell time increases, providing up to 80% higher revenue than *SAC*-4. Interestingly, over 60 days even the analytical approximation outperforms *SAC*, even though it uses univariate pricing and the average resource requirement of the WDs, indicating that *SAC* suffers from slow learning, which is detrimental to the average revenue, unlike the proposed *HiPE* pricing algorithm. Comparing linear (*SAC*-3) and non-linear (*SAC*-4) pricing we can observe that non-linear pricing is most beneficial for moderate average dwell times, and allows up to 40% higher revenue than linear pricing, as WDs tend to request less resources and resource intensive functions can be charged more aggressively. We can also observe that the online knapsack algorithm exhibits consistently low revenue similar to the case of homogeneous resource allocation.

5.5.3 Sensitivity Analysis

Figure 5.4c shows the daily average revenue for sythetic traces using exponentially and deterministically distributed dwell times with mean $\mathbb{E}[D_i] = 1800$ sec, and for real traces based on Cells #1, #2 and #3; reservation costs follow uniform and truncated Gaussian distributions. The figure shows that even though the *HiPE* algorithm was pre-trained on synthetic traces with exponentially distributed dwell times and uniformly distributed reservation costs, and then applied to the real trace environment, the data collected from the real environment within the initial 25 days of the evaluation is sufficient to fine-tune the BNNs for this previously unseen environment, and achieves superior performance compared to all baselines. The figure shows subtle revenue differences among the results obtained using different dwell time and reservation cost distributions, indicating that the revenue is predominantly influenced by the average dwell time. Comparing the performance achieved on the real traces from Cell #1, Cell #2 and Cell #3, we observe that the highest gain is achieved in Cell #1, which has the highest offered load. This observation emphasises the



(a) For synthetic traces

(b) For Cells #1, #2, and #3

Figure 5.5: Learning curves of SAC and HiPE

difficulty of learning an effective policy using a model-free approach in such scenarios and highlights the advantages of the proposed *HiPE* pricing algorithm. Overall, Figure 5.4c shows the robustness and adaptability of the proposed *HiPE* algorithm, which makes it well-suited for real-world environments.

5.5.4 Learning Curves

Figure 5.5a and Figure 5.5b show the daily average revenue achieved using the *HiPE* and *SAC*-4 pricing schemes, for synthetic traces with exponentially and deterministically distributed dwell times and for real traces, respectively.

The figures show that *HiPE* learns significantly faster than *SAC*; it takes *HiPE* approximately 60 days to achieve the average revenue that *SAC*-4 achieves after 180 days of learning. The learning curves also confirm that *HiPE* is most advantageous under high traffic load (Cell #1), which has the highest potential revenue. The ability of *HiPE* to learn fast makes it particularly appealing for real deployments, and emphasises the necessity to use transfer learning for the purpose of pricing in EC.

5.5.5 Consumer Surplus and Service Probability

Let us define the daily users surplus, which corresponds to the added value of the edge service as perceived by the WDs (the difference of the reservation cost and the actual cost of using the edge service) and denote it ϕ^+,

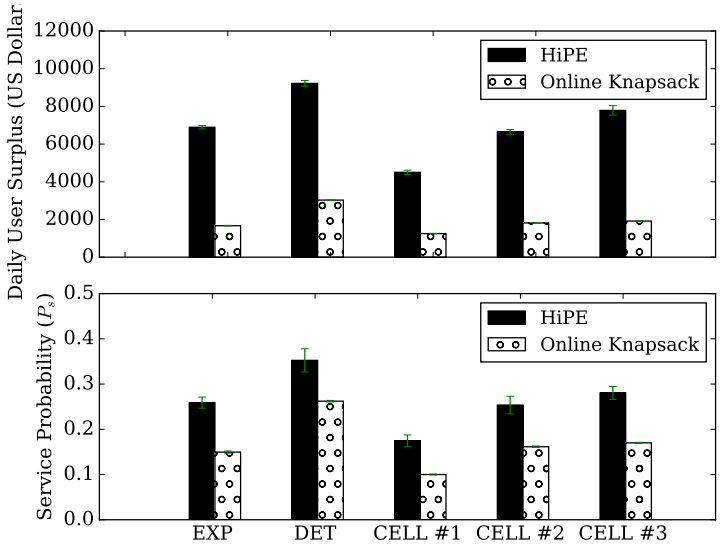


Figure 5.6: Consumer surplus and P_s for two synthetic dwell time distributions ($\mathbb{E}[D_i] =$ 1800 Section) and trace-based distributions, for uniform reservation cost distribution.

$$\phi^{+} = \sum_{k=1}^{K=24} \sum_{i \in \mathcal{N}_k} \left(\overline{C}_i - C_{i,\Sigma}^{\boldsymbol{\pi}_k} \right),$$

and the probability P_s of a WD receiving service, i.e., the fraction of served WDs,

$$P_s = \frac{\sum_{k=1}^{K=24} |\mathcal{N}_k^o|}{\sum_{k=1}^{K=24} |\mathcal{N}_k|},$$

where $\mathcal{N}_k^o = \{i | t_i^o \in T_k\}$.

Figure 5.6 shows the results obtained for synthetic traces with exponentially and deterministically distributed dwell times with mean $\mathbb{E}[D_i] = 1800$ seconds and for real traces. The figure shows that the *HiPE* pricing algorithm yields up to 5 times higher consumer surplus and up to 2 times higher probability of receiving service compared to the baseline, online knapsack. This shows that the proposed *HiPE* pricing algorithm is not only superior in terms of operator revenue but it is also preferable from the perspective of WDs in terms of the received added value. In technical terms, our proposed approach effectively addresses admission control through pricing and at the same time offers a favorable edge service for users, combining adaptivity with transparency. While our proposed *HiPE* pricing algorithm is better than online Knapsack in terms of service probability, it is still not optimal since the service probability is always $< 50\%$. This is predictable as our main

objective was to maximize the NO revenue by providing a transparent and efficient pricing scheme which results in high resource utilization.

5.6 Conclusion

In this chapter, we considered the problem of pricing in serverless EC under dynamic workloads. We first formulated the problem of maximizing the revenue of the operator as a sequential decision making problem under uncertainty. We then showed that the problem can be cast as a HiP-MDP and proposed a dual BNN approximator as a solution. The proposed solution is a form of transfer learning; after pre-training on synthetic traces, it adapts fast to previously unseen workloads. Our results show that the proposed solution accelerates learning and achieves superior performance, in terms of revenue generation, compared to the state of the art and to the steady state approximation, but without the need for prior information about the parameters of the system. Furthermore, our results show that non-linear pricing models could benefit edge deployments, as they encourage users to request computational resources sparingly, and thereby effectively increasing the number of concurrent users that can be served, compared to the state-of-the-art online Knapsack .

Chapter 6

General Conclusion

6.1 Summary of our Contribution

Edge Computing (EC) has emerged as a computing paradigm in technological landscape. Its evolution is driven by the increasing demand for fast processing and minimal latency in our interconnected world.

This thesis presents a vision for Network Operators (NOs), who are the only owners of the far edge, to extract value from the implementation EC in a multi-tenant environment. In such a scenario, the NO extends edge resources to third-party Service Providers (SPs) while intelligently managing resource allocation among them. The key objectives we address in this thesis for the NO are optimizing upstream traffic reduction, maximizing revenue, and enhancing Quality of Service (QoS) as perceived by end users.

We believe that by advocating this EC vision and substantiating it with quantitative findings and analysis, this thesis offers insights, particularly for NOs, that can influence and, hopefully, bolster decision-making strategies regarding future EC deployment. This approach may catalyze the emergence of innovative low-latency and data-intensive applications, such as high-resolution Augmented Reality (AR), which are currently impractical in the existing Cloud Computing (CC) paradigm.

Another contribution of this thesis lies in the application of novel methods harnessing the potential of data-driven optimization. We adapt state-of-the-art techniques from Reinforcement Learning (RL) and sequential decision-making to the practical challenge of resource allocation in EC. By carefully designing estimation models integrated into the learning process, we succeed in reducing the learning time of the adopted strategies to scales compatible with the dynamics of EC. Additionally, we establish important analytical properties of our strategies.

It is important to emphasize that our strategies are purposefully designed to uphold the confidentiality guarantees essential for SPs to be willing participants in running their computations at the EC within the multi-tenant framework.

6.1.1 Cache Allocation for Multi-tenant Edge Computing

We solved in Chapter 3 the problem of cache allocation at the edge among several SPs, where the aim is not only to minimize the cost, in terms of miss rate, but also to optimize the way to achieve that, through minimizing perturbations, assuming encrypted, not all cacheable content: a major challenge of in-network caching. We introduced a model-based RL algorithm designed for real-time cache allocation at the edge. We started with the establishment of theoretical foundations, demonstrating the algorithm convergence towards an absorbing discrete optimal state with a probability of 1 for an infinite horizon. Subsequently, we conducted an extensive set of simulations, comparing our dynamic allocation approach to two static strategies: (i) an optimal allocation, that we computed under the assumption of complete knowledge of content popularity, and (ii) a proportional allocation based on the probabilities of content requests from each SP. Additionally, we assessed our model-based RL algorithm against two dynamic allocation strategies: (iii) a model-free variant of our own algorithm and (iv) the state-of-the-art SPSA method. Our simulations, conducted across various scenarios, clearly illustrate that our algorithm consistently converges to a configuration closely aligned with the optimal solution, accomplishing this feat significantly faster than the compared allocation strategies. This accelerated convergence not only holds promise for enhancing system performance but also carries the potential to substantially reduce overall system costs.

6.1.2 Multiple-resource Allocation for Multi-tenant Edge Computing

In Chapter 4, we tackled the resource allocation at EC between heterogeneous, MAR-oriented SPs competing over multiple, limited resources. We modeled the users dynamics in terms of an Erlang-type queuing model, we formulated the resource allocation problem as a sub-modular maximization problem subject to multiple knapsack constraints, under perfect knowledge of system parameters, and solved it via an approximation algorithm with provable optimality gap. We then formulated the problem as a sequential decision making problem, when system parameters are unknown by the NO, and we solved it via deep RL. Our numerical results quantified the performance of both algorithms in terms of the probability that users get served by the Edge, as opposed to being blocked and re-directed towards the Cloud which entails larger delay and hence lesser QoS. We showed the resulting resource partitioning between the SPs. We showed that deep RL approaches the approximation algorithm and outperforms a baseline resource allocation, proportional to users arrival rates, albeit at slow training phase which can be accelerated using a model as in the other chapters. We eventually performed a sensitivity analysis of the objective function with respect to arrival rates and users requirements.

6.1.3 Resource Pricing for Serverless Edge Computing

Chapter 5 considered the problem of pricing in serverless EC under dynamic workloads of individual users, as opposed to the previous two chapters where the interaction was between SPs and NO. We first formulated the problem of maximizing the revenue of the operator as a sequential decision making problem under uncertainty. Second, we showed that, under Markovian assumptions, a pricing policy can be obtained analytically that serves as a lower bound for the operator revenue. We then showed that the problem can be cast as a Hidden Parameter Markov Decision Process (HiP-MDP) and proposed a dual Bayesian Neural Network (BNN) approximator as a solution. The proposed solution is a form of transfer learning; after pre-training on synthetic traces, it adapts fast to previously unseen workloads. Our results show that the proposed solution accelerates learning and achieves superior performance compared to the state of the art, on par with the steady state approximation, but without the need for prior information about the parameters of the system. Furthermore, our results show that non-linear pricing models could benefit edge deployments, as they encourage users to request computational resources sparingly, and thereby effectively increasing the number of concurrent users that can be served.

6.2 Discussion and Future Work

The research conducted in this thesis has laid the foundation for addressing critical challenges in resource allocation within multi-tenant edge computing environments. However, there are numerous avenues for further investigation and improvement. The following sections outline potential directions for future research following this thesis:

6.2.1 Cache Allocation in EC

In Chapter 3, we solved the problem of caching at the edge under stationary popularity of the SPs' objects. One interesting direction to take is to solve the problem for time-varying popularity of the content. Indeed, in real world, users preferences and behaviors are not the same at each period of the day. A strategy recognizing that what is popular at one moment may not be the same in the next, ensures that the cached content remains relevant to users over time. Another possible direction is when cache is partitioned across multiple edge nodes. Caching across multiple edge nodes enhances performance by reducing the load on primary edge node and increasing the overall availability of cached data. However, it also introduces more challenges related to cache consistency, load balancing, synchronization, and cache eviction strategies, which must be carefully managed to ensure the integrity of the cached data and the performance of the caching system.

6.2.2 Multiple-resource Allocation in EC

In Chapter 4, we solve the problem of memory-CPU allocation in multi-tenant EC based on the assumption that we are always in adequate wirless channel conditions. However, this assumption is too optimistic. It is reasonable to consider integrating wireless channel modeling into our resource allocation framework. In fact, wireless channel conditions can significantly impact the performance of edge computing applications. By accounting for channel characteristics like signal strength, interference, and channel fading, we could optimize resource allocation decisions for better real-time performance.

6.2.3 Resource Pricing in EC

We could consider resource pricing strategies that take into account the energy efficiency of edge devices. Optimize pricing not only for performance but also for minimizing energy consumption, which is critical for resource-constrained edge environments, would make it more adaptable, efficient, and responsive to the evolving demands of EC environments. Another perspective could be investigating user-centric pricing models that take into account individual user preferences and behavior patterns. Personalized pricing may improve user satisfaction and resource utilization. On the methodology level, implementing Autoencoders, instead of BNNs used in Chapter 5, could be a promising improvement. In fact, Autoencoders are effective at capturing meaningful features and reducing the dimensionality of input data, which can be beneficial when dealing with high-dimensional state spaces.

6.2.4 Beyond Considered Problems

Stating that this thesis has completely addressed all the outstanding challenges of resource allocation in EC would be an overstatement. The following challenges, among others, could be a potential continuation of this thesis.

1. Multi-objective optimization: resource allocation is often a multi-objective problem with conflicting goals, such as minimizing latency, maximizing resource utilization, and reducing costs. Future research can explore multi-agent reinforcement learning techniques to strike a balance between these competing objectives and provide more holistic resource allocation solutions.

2. Energy efficiency: it is a critical concern in resource allocation, particularly in resource-constrained edge environments. Future work can delve into optimizing resource allocation not only for performance but also for energy efficiency. This could involve the incorporation of power-aware reinforcement learning algorithms and hardware-level optimizations.

3. Environmental considerations: the fact that EC is "better" than CC from an ecological perspective depends on the specific context and how these technologies are

implemented. EC has the potential to be more ecologically friendly in scenarios where reduced data transmission, lower latency, and offline operation are critical. However, a holistic evaluation should consider the entire lifecycle of edge devices and their environmental implications, including resource extraction, manufacturing, energy efficiency, and end-of-life disposal. Additionally, the optimal solution may involve a combination of both edge and cloud computing, depending on the specific requirements of the ecological application.

4. Security and privacy considerations: edge environments are inherently distributed and often involve sensitive data processing. Future research should address security and privacy concerns associated with resource allocation. This may involve developing reinforcement learning models that consider security as an additional objective or exploring encryption and access control mechanisms to protect data at the edge.

Chapter 7

Appendix

7.1 Proof of the Convergence Theorem 3.4.2.1

In what follows, we aim to prove that if the discount factor γ is sufficiently close to 1, then

$$\lim_{k\to\infty} \boldsymbol{\theta}^{(k)} = \hat{\boldsymbol{\theta}}^* \text{ with probability 1.}$$

This means that the system will converge to a discretely optimal state (Section 3.4.2) and will stay in that state with probability 1 as time goes to infinity.

Definition 7.1.0.1. *Given a Q-table $Q(\boldsymbol{\theta}, \mathbf{a})$, $\forall(\boldsymbol{\theta}, \mathbf{a}) \in \mathcal{S} \times \mathcal{A}_{\boldsymbol{\theta}}$, we say that a sequence of states and actions is induced by Q, if and only if,*

$$\mathbf{a}^{(k)} \in \arg\min_{\mathbf{a}} Q(\boldsymbol{\theta}^{(k)}, \mathbf{a}), \forall k > 0$$

i.e., if and only if it greedily follows Q.

Definition 7.1.0.2. *Given a sequence $Q^{(k)}$ of Q-tables , we say that a sequence of state and actions is induced by a sequence of Q-tables $Q^{(k)}$, if and only if,*

$$\mathbf{a}^{(k)} \in \arg\min_{\mathbf{a}} Q^{(k)}(\boldsymbol{\theta}^{(k)}, \mathbf{a}), \forall k > 0$$

i.e., if and only if it greedily follows $Q^{(k)}$.

We decompose the proof as follows: In Section 7.1.1 we deeply describe the process of updating the Q-table $Q^{(k)}$. In Section 7.1.2 we prove that our Q-table $Q^{(k)}$ converges to the optimal Q-table Q^* with probability 1. In Section 7.1.3 we prove that the sequence of actions and states induced by Q^* has an absorbing state that is the discretely optimal state $\hat{\boldsymbol{\theta}}^*$. In Section 7.1.4 we prove that the sequence of actions and states induced by our Q-table $Q^{(k)}$ (assuming no more exploration) is also induced by Q^*. In Section 7.1.5 we prove

that the sequence of states and actions $\{\boldsymbol{\theta}^{(k)}, \mathbf{a}^{(k)}\}$, that we take online, converges with probability 1 to the sequence induced by our Q-table $Q^{(k)}$ (assuming no more exploration). Finally, we show that $\{\boldsymbol{\theta}^{(k)}, \mathbf{a}^{(k)}\}$, taken online, converges with probability 1 to the sequence induced by Q^*.

Definition 7.1.0.3. *We say that $Q^{(k)}$ converges with probability 1 to Q^*, if and only if [169]*

$$\mathbb{P}(\lim_{k\to\infty} |Q^{(k)}(\boldsymbol{\theta}, \mathbf{a}) - Q^*(\boldsymbol{\theta}, \mathbf{a})| < \epsilon) = 1, \forall \epsilon > 0, \forall (\boldsymbol{\theta}, \mathbf{a})$$

7.1.1 Consistency Q-table updates

Observe that in Algorithm 1, we update at every time-slot k the Q-table $N_u = 1 + N_{\text{memory}} + N_{model}$ times. For simplicity of notation, up to now we have only referred to Q-table $Q^{(k)}(\cdot,\cdot)$, but actually the Q-table has changed N_u times in one single iteration of Algorithm 1. In this proof, we need to distinguish all these different versions. Let us denote with $Q^{\{j\}}(\cdot,\cdot)$ the j-th version of the Q-table. In time-slot k, versions $Q^{\{j\}}(\cdot,\cdot)$, for $j = k \cdot N_u, \ldots, (k+1) \cdot N_u - 1$ are created. Updates of lines 7, 15 and 28 of Algorithm 1 can be described in a unified way: when computing version $Q^{\{j+1\}}(\cdot,\cdot)$, a state action pair $(\boldsymbol{\theta}^{\{j\}}, \mathbf{a}^{\{j\}})$ is chosen, and we apply an update rule of the following form:

$$\begin{aligned} Q^{\{j+1\}}(\boldsymbol{\theta}^{\{j\}}, \mathbf{a}^{\{j\}}) &= (1 - \alpha^{\{j\}}) \cdot Q^{\{j\}}(\boldsymbol{\theta}^{\{j\}}, \mathbf{a}^{\{j\}}) \\ +\alpha^{\{j\}}(C^{\{j\}}_{\text{nom}} + C_{\text{pert}}(\mathbf{a}^{\{j\}}) &+ \gamma^{\{j\}} \min_{\mathbf{a}\in\mathcal{A}_{\boldsymbol{\theta}'^{\{j\}}}} Q^{\{j\}}(\boldsymbol{\theta}'^{\{j\}}, \mathbf{a})) \end{aligned} \tag{7.1}$$

where $\boldsymbol{\theta}'^{\{j\}} = \boldsymbol{\theta}^{\{j\}} + \mathbf{a}^{\{j\}}$ and $\alpha^{\{j\}} = \alpha^{(k)}, \gamma^{\{j\}} = \gamma^{(k)}$, for any $j = k \cdot N_u, \ldots, (k+1) \cdot N_u - 1$. As for the value of $(\boldsymbol{\theta}^{\{j\}}, \mathbf{a}^{\{j\}}, C^{\{j\}}_{\text{nom}})$, it depends on whether the j-th update is obtained using an observed sample (Line 7 of Algorithm 1), experience replay (Line 15) or the model (Line 28):

$$(\boldsymbol{\theta}^{\{j\}}, \mathbf{a}^{\{j\}}, C^{\{j\}}_{\text{nom}}) = \begin{cases} (\boldsymbol{\theta}^{(k)}, \mathbf{a}^{(k)}, C_{\text{nom}}(\boldsymbol{\theta}^{(k)}, \omega)) & \text{if } j = k \cdot N_u \\ & \text{(use observed nominal cost)} \\ (\boldsymbol{\theta}^{\text{rd}}, \mathbf{a}^{\text{rd}}, C^{\text{rd}}_{\text{nom}}) \in \mathcal{M}^{(k)} & \text{if } j = k \cdot N_u + j', j' = 1, \ldots, N_{\text{memory}} \\ & \text{(use nominal cost from memory)} \\ (\boldsymbol{\theta}^{\text{rd}}, \mathbf{a}^{\text{rd}}, \hat{C}^{(k)}_{\text{nom}}(\boldsymbol{\theta}^{\text{rd}})) & \text{if } j = k \cdot N_u + N_{\text{memory}} + j', \\ \text{for randomly chosen} & j' = 1, \ldots, N_{\text{model}} \\ \boldsymbol{\theta}^{\text{rd}} \in \mathcal{S}, \mathbf{a}^{\text{rd}} \in \mathcal{A}_{\boldsymbol{\theta}^{\text{rd}}} & \text{(use estimation of nominal cost from the model)} \end{cases} \tag{7.1bis}$$

We also define function $\hat{C}^{\{j\}}_{\text{nom}}(\cdot) : \mathcal{S} \to \mathbb{R}$ as

$$\hat{C}^{\{0\}}_{\text{nom}}(\boldsymbol{\theta}) = 0, \forall \boldsymbol{\theta} \in \mathcal{S}$$
$$\hat{C}^{\{j\}}_{\text{nom}}(\boldsymbol{\theta}) \triangleq \begin{cases} C^{\{j\}}_{\text{nom}} & \text{if } \boldsymbol{\theta} = \boldsymbol{\theta}^{\{j\}} \text{(see (7.1bis))} \\ \hat{C}^{\{j-1\}}_{\text{nom}}(\boldsymbol{\theta}) & \text{otherwise} \end{cases}, \text{for } j = 1, 2, \dots \tag{7.1tris}$$

Definition (7.1tris) allows rewriting update (7.1) as follows:

$$Q^{\{j+1\}}(\boldsymbol{\theta}^{\{j\}}, \mathbf{a}^{\{j\}}) = (1 - \alpha^{\{j\}}) \cdot Q^{\{j\}}(\boldsymbol{\theta}^{\{j\}}, \mathbf{a}^{\{j\}})$$
$$+\alpha^{\{j\}}(\hat{C}^{\{j\}}_{\text{nom}} + C_{\text{pert}}(\mathbf{a}^{\{j\}}) + \gamma^{\{j\}} \min_{\mathbf{a} \in \mathcal{A}_{\boldsymbol{\theta}'^{\{j\}}}} Q^{\{j\}}(\boldsymbol{\theta}'^{\{j\}}, \mathbf{a})) \tag{7.1quadris}$$

In practice, we have just replaced $C^{\{j\}}_{\text{nom}}$ with $\hat{C}^{\{j\}}_{\text{nom}}$. Thanks to this minor change, we get rid of sequence of numbers $\{C^{\{j\}}_{\text{nom}}\}_j$, replacing it with sequence of functions $\{\boldsymbol{\theta} \to \hat{C}^{\{j\}}_{\text{nom}}(\boldsymbol{\theta})\}_j$, which is important as the proof of Theorem 7.1.2.3 (in particular (7.12)) requires a sequence of functions. We call (7.1quadris) the functional form of the Q-table update.

Observe that at step j, the Q-table is updated only in correspondence to pairs $(\boldsymbol{\theta}^{\{j\}}, \mathbf{a}^{\{j\}})$, while it remains unchanged in all other values:

$$Q^{\{j+1\}}(\boldsymbol{\theta}, \mathbf{a}) = Q^{\{j\}}(\boldsymbol{\theta}, \mathbf{a}) \forall (\boldsymbol{\theta}, \mathbf{a}) \neq (\boldsymbol{\theta}^{\{j\}}, \mathbf{a}^{\{j\}})$$

Similarly, by construction of (7.1tris), for any $\boldsymbol{\theta} \in \mathcal{S}$, the value $\hat{C}^{\{j\}}_{\text{nom}}(\boldsymbol{\theta})$ only changes at step j for which $\boldsymbol{\theta} = \boldsymbol{\theta}^{\{j\}}$. In all the other steps, the value is inherited from the previous steps. The following theorem characterizes function $\hat{C}^{\{j\}}_{\text{nom}}(\cdot)$ used in the functional form of the Q-table updates.

Theorem 7.1.1.1. *Function $\hat{C}^{\{j\}}_{nom}(\cdot)$ converges uniformly in expectation to the nominal cost, i.e.,*

$$\lim_{j \to \infty}^{u} \mathbb{E}\left[\hat{C}^{\{j\}}_{nom}(\cdot) | \mathcal{F}^{\{j\}}\right] = \mathbb{E}C_{nom}(\cdot). \tag{7.2}$$

where $\lim^u$ *has the same meaning as in Theorem 3.3.2.1 and* $\mathcal{F}^{\{j\}} = \{Q^{\{j\}}, Q^{\{j-1\}}, ...\}$ *stands for the past at step* j*.*

Proof. We first consider any $\boldsymbol{\theta} \in \mathcal{S}$ and prove pointwise convergence, i.e. that

$$\lim_{j \to \infty} \mathbb{E}\left[\hat{C}^{\{j\}}_{\text{nom}}(\boldsymbol{\theta}) | \mathcal{F}^{\{j\}}\right] = \mathbb{E}C_{\text{nom}}(\boldsymbol{\theta}) \tag{7.3}$$

To this aim, exploiting the construction of $\hat{C}^{\{j\}}_{\text{nom}}(\cdot)$, it suffices to show that

$$\lim_{z \to \infty} \mathbb{E}\left[\hat{C}^{\{j_z\}}_{\text{nom}}(\boldsymbol{\theta}) | \mathcal{F}^{\{j_z\}}\right] = \mathbb{E}C_{\text{nom}}(\boldsymbol{\theta}), \tag{7.4}$$

where $j_1, j_2, \dots$ is the subsequence of indices js such that $\boldsymbol{\theta}^{\{j_z\}} = \boldsymbol{\theta}$. We denote with $k\{j\}$ the timeslot in which version $Q^{\{j\}}(\cdot,\cdot)$ of the Q-table is calculated, i.e., $k\{j\} = k$ for $j = k \cdot N_u, \dots, (k+1) \cdot N_u - 1$. Indices $\{j_z\}_{z\in\mathbb{N}}$ can be divided in three subsequences of indices:

1. Indices $\{j_{z_w}\}_{w\in\mathbb{N}}$ such that $j_{z_w} = k\{j_{z_w}\} \cdot N_u$. In this case, thanks to (7.1bis),
$$\begin{aligned}\mathbb{E}\left[\hat{C}_{\text{nom}}^{\{j_{z_w}\}}(\boldsymbol{\theta})|\mathcal{F}^{\{j_z\}}\right] &= \mathbb{E}\left[C_{\text{nom}}(\boldsymbol{\theta}, \omega^{(k\{j_{z_w}\})})|\mathcal{F}^{\{j_{z_w}\}}\right] \\ &= \mathbb{E}C_{\text{nom}}(\boldsymbol{\theta})\end{aligned}$$

2. Indices $\{j_{z_w}\}_{w\in\mathbb{N}}$ such that $j_{z_w} = k\{j_{z_w}\} \cdot N_u + j', j' = 1, \dots, N_{\text{memory}}$. In this case, thanks to (7.1tris), $\hat{C}_{\text{nom}}^{\{j_{z_w}\}}(\boldsymbol{\theta})$ is a past observation of nominal cost when state $\boldsymbol{\theta}$ was visited. By construction, its expected value is $\mathbb{E}\left[\hat{C}_{\text{nom}}^{\{j_{z_w}\}}(\boldsymbol{\theta})|\mathcal{F}^{\{j_z\}}\right] = \mathbb{E}C_{\text{nom}}(\boldsymbol{\theta})$.

3. Indices $\{j_{z_w}\}_{w\in\mathbb{N}}$ such that $j_{z_w} = k\{j_{z_w}\} \cdot N_u + N_{\text{memory}} + j', j' = 1, \dots, N_{\text{model}}$. In this case, thanks to (7.1tris), $C_{\text{nom}}^{\{j_{z_w}\}}(\boldsymbol{\theta})$ is obtained via the model explained in Section 3.3.2. By exploiting Theorem 3.3.2.1, we have that $\lim_{w\to\infty} \mathbb{E}\left[C_{\text{nom}}^{\{j_{z_w}\}}(\boldsymbol{\theta})\right] = \mathbb{E}C_{\text{nom}}(\boldsymbol{\theta})$.

Since sequence $\mathbb{E}\left[\hat{C}_{\text{nom}}^{\{j_z\}}(\boldsymbol{\theta})|\mathcal{F}^{\{j_z\}}\right]$ is the union of the three subsequences above, each of which converges to $\mathbb{E}C_{\text{nom}}(\boldsymbol{\theta})$, we obtain the theorem.

Since pointwise convergence 7.3 holds for all $\boldsymbol{\theta} \in \mathcal{S}$ and $\mathcal{S}$ is finite, then [103, Proposition 1] convergence is also uniform. □

7.1.2 Convergence of the Q-table

In this section, we will prove that our Q-table $Q^{(k)}$, updated following (7.1bis), converges to the optimal Q-table Q^* with probability 1. As we combine Q-Learning with a model that approximates the expected nominal cost $\mathbb{E}_\omega\big[C_{\text{nom}}(\boldsymbol{\theta}, \omega)\big]$ (Section 3.3.2) and with other enhancements mentioned in Section 3.3.3, we cannot simply rely on the property of convergence of classical Q-learning (Theorem 2 of [106] or [107]). It is worth noting that many works claim that model-based RL in all its forms converges [170], however no explicit proof is provided for the specific form of combining Q-table updates with a model of the reward (cost in our case).

In what follows, we will prove that the process defined by (7.1bis) converges to the optimal Q-table Q^* with probability 1. We start with two general results Lemma 7.1.2.1 and Lemma 7.1.2.2, which we obtain by extending a previously known result.

Lemma 7.1.2.1. *Consider a random iterative process* $\delta^{\{j\}} : \mathcal{X} \to \mathbb{R}^n$ *defined as*

$$\delta^{\{j+1\}}(x) = (1 - \alpha^{\{j\}}) \cdot \delta^{\{j\}}(x) + \alpha^{\{j\}} \cdot F^{\{j\}}(x) \tag{7.5}$$

where $F^{\{j\}} : \mathcal{X} \to \mathbb{R}^n$ *is a random function, at each step* j*. Process* $\delta^{\{j\}}$ *converges with probability* 1 *to* 0*, i.e.,* $\mathbb{P}\left[\lim_{j\to\infty} \delta^{\{j\}}(x) = 0\right] = 1, \forall x \in \mathcal{X}$*, under the following assumptions:*

1. $\mathcal{X}$ *is finite*
2. $0 \leq \alpha^{\{j\}} \leq 1, \sum_k \alpha^{\{j\}} = \infty$ *and* $\sum_k (\alpha^{\{j\}})^2 < \infty$
3. $\exists 0 < \nu < 1$:
$$\|\mathbb{E}[F^{\{j\}}(x) - \eta^{\{j\}}(x) | \mathcal{F}^{\{j\}}]\|_\infty \leq \nu \|\delta^{\{j\}}\|_\infty,$$
where $\eta^{\{j\}}(\cdot)$ *is a sequence of functions such that* $\lim^u_{j\to\infty} \eta^{\{j\}}(\cdot) = 0$*.*
4. $\exists M > 0, \boldsymbol{var}[F^{\{j\}}(x) | \mathcal{F}^{\{j\}}] \leq M(1 + \|\delta^{\{j\}}\|^2_\infty)$

Here $\mathcal{F}^{\{j\}} = \{\delta^{\{j\}}, \delta^{\{j-1\}}, .., F^{\{j-1\}}, .., \alpha^{\{j-1\}}, ..\}$ *stands for the past at step* j*.*

Proof. The original version of this theorem is [171, Theorem 1]. The only difference is assumption 3, which in [171] is

$$\exists 0 < \nu < 1, \|\mathbb{E}[F^{\{j\}}(x) | \mathcal{F}^{\{j\}}]\|_\infty \leq \nu \|\delta^{\{j\}}\|_\infty.$$

We need to add the additional term $\mathbb{E}[\eta^{\{j\}}(x) | \mathcal{F}^{\{j\}}]$ to account for the error of our model in approximating the expected value of the cost. Let us define another random process in the following way:

$$\begin{aligned} F'^{\{j\}}(x) &= F^{\{j\}}(x) - \eta^{\{j\}}(x) \\ \delta'^{\{j+1\}}(x) &= (1 - \alpha^{\{j\}}) \cdot \delta'^{\{j\}}(x) + \alpha^{\{j\}} \cdot F'^{\{j\}}(x) \end{aligned} \tag{7.6}$$

Process $\delta'^{\{j\}}(x)$ respects the assumptions of the original [171, Theorem 1] and thus

$$\lim_{j\to\infty} \delta'^{\{j\}}(x) = 0, \forall x \in \mathcal{X}, \text{ with probability 1.} \tag{7.7}$$

Let us now define an additional random process

$$\delta^{\{j\}}_{\text{diff}}(x) = \delta'^{\{j\}}(x) - \delta^{\{j\}}(x) \tag{7.8}$$

and observe that

$$\begin{aligned} \delta^{\{j+1\}}_{\text{diff}}(x) &= \delta'^{\{j+1\}}(x) - \delta^{\{j+1\}}(x) \\ &= \left[(1 - \alpha^{\{j\}}) \cdot \delta'^{\{j\}}(x) + \alpha^{\{j\}} \cdot F'^{\{j\}}(x)\right] - \left[(1 - \alpha^{\{j\}}) \cdot \delta^{\{j\}}(x) + \alpha^{\{j\}} \cdot F^{\{j\}}(x)\right] \\ &= (1 - \alpha^{\{j\}}) \cdot (\delta'^{\{j\}}(x) - \delta^{\{j\}}(x)) + \alpha^{\{j\}} \cdot (-\eta^{\{j\}}(x)) \\ &= (1 - \alpha^{\{j\}}) \cdot \delta^{\{j\}}_{\text{diff}}(x) + \alpha^{\{j\}} \cdot (-\eta^{\{j\}}(x)) \end{aligned}$$

The iterative process above respects the assumptions of [171, Lemma 1], which states that

$$\lim_{j\to\infty} \delta^{\{j\}}_{\text{diff}}(x) = 0, \forall x \in \mathcal{X}, \text{ with probability 1.} \tag{7.9}$$

Thanks to (7.7) and (7.8), process $\delta^{\{j\}}(x)$ must converge to 0 with probability 1. □

Lemma 7.1.2.2. *Consider a random iterative process $\delta^{\{j\}} : \mathcal{X} \to \mathbb{R}^n$ such that, for each $x \in \mathcal{X}$, there is an infinite subsequence of indices $\mathcal{I}_x = \{j_1, j_2, ..., j_z, ...\}$ such that*

$$\delta^{\{j_z+1\}}(x) = (1 - \alpha^{\{j_z\}}) \cdot \delta^{\{j_z\}}(x) + \alpha^{\{j_z\}} \cdot F^{\{j_z\}}(x), \forall j_z \in \mathcal{I}_x$$

and

$$\delta^{\{j_z+1\}}(x) = \delta^{\{j_z\}}(x), \forall j_z \in \mathbb{N} \setminus \mathcal{I}_x.^1$$

Assume that the same assumptions of Lemma 7.1.2.1 hold:

1. *$\mathcal{X}$ is finite*
2. *$0 \leq \alpha^{\{j\}} \leq 1, \sum_k \alpha^{\{j\}} = \infty$ and $\sum_k (\alpha^{\{j\}})^2 < \infty$*
3. *$\exists 0 < \nu < 1$:*
$$\|\mathbb{E}[F^{\{j\}}(x) - \eta^{\{j\}}(x) | \mathcal{F}^{\{j\}}]\|_\infty \leq \nu \|\delta^{\{j\}}\|_\infty,$$
where $\eta^{\{j\}}(\cdot)$ is a sequence of functions such that $\lim^u_{j\to\infty} \eta^{\{j\}}(\cdot) = 0$.
4. *$\exists M > 0, \boldsymbol{var}[F^{\{j\}}(x) | \mathcal{F}^{\{j\}}] \leq M(1 + \|\delta^{\{j\}}\|^2_\infty)$*

Then, process $\delta^{\{j\}}$ converges with probability 1 to 0, i.e., $\mathbb{P}\left[\lim_{j\to\infty} \delta^{\{j\}}(x) = 0\right] = 1, \forall\, x \in \mathcal{X}$.

Proof. Let us fix any $x \in \mathcal{X}$. By applying Lemma 7.1.2.1 on the sequence $\delta^{\{j_1\}}(x), \delta^{\{j_2\}}(x), ...$ (which is subsequence of $\delta^{\{1\}}(x), \delta^{\{2\}}(x), ...$) we obtain that,

$$\lim_{z\to\infty} \delta^{\{j_z\}}(x) = 0, \forall x \in \mathcal{X}, \text{ with probability 1.}$$

Observe that by construction, for any $z \in \mathbb{N}$,

$$\delta^{\{j\}}(x) = \delta^{\{j_z+1\}}(x), \text{ for } j = j_z + 1, j_z + 2, ..., j_{z+1}.$$

This implies that,

$$\lim_{j\to\infty} \delta^{\{j\}}(x) = 0, \forall x \in \mathcal{X}, \text{ with probability 1.}$$

We have thus proved point-wise convergence. Thanks to [103, Proposition 1], since $\mathcal{X}$ is finite, this also implies uniform convergence. □

[1] Observe that the subsequence of indices $\mathcal{I}_x$ changes for every considered x.

Theorem 7.1.2.3. *If $Q^{\{j\}}$ is updated by update* (7.1quadris)*, then $Q^{\{j\}}$ converges to Q^* with probability* 1*.*

Proof. We will prove the theorem using Lemma 7.1.2.1. We define $\mathcal{X} = \mathcal{S} \times \mathcal{A}$ and

$$F^{\{j\}}(\boldsymbol{\theta}, \mathbf{a}) = \hat{C}^{\{j\}}_{\text{nom}}(\boldsymbol{\theta}) + C_{\text{pert}}(\mathbf{a}) + \gamma^{\{j\}} \min_{\mathbf{a}' \in \mathcal{A}_{\boldsymbol{\theta}+\mathbf{a}}} Q^{\{j\}}(\boldsymbol{\theta} + \mathbf{a}, \mathbf{a}') - Q^*(\boldsymbol{\theta}, \mathbf{a}) \tag{7.10}$$

Observe that, by fixing the past $\mathcal{F}^{\{j\}}$, i.e., all the observed rewards, actions, states and extractions from the memory and from the model, up to before update j, table $Q^{\{j\}}$ is univocally determined. The only stochastic term in (7.10) is thus $\hat{C}^{\{j\}}_{\text{nom}}(\boldsymbol{\theta})$, while all the others are deterministic. Let us define $\delta^{\{j\}}(\boldsymbol{\theta}, \mathbf{a})$ as follows:

$$\delta^{\{j\}}(\boldsymbol{\theta}, \mathbf{a}) = Q^{\{j\}}(\boldsymbol{\theta}, \mathbf{a}) - Q^*(\boldsymbol{\theta}, \mathbf{a}), \forall (\boldsymbol{\theta}, \mathbf{a}) \in \mathcal{X} \tag{7.11}$$

If $(\boldsymbol{\theta}, \mathbf{a}) = (\boldsymbol{\theta}^{\{j\}}, \mathbf{a}^{\{j\}})$, we obtain

$$\begin{aligned}
\delta^{\{j+1\}}(\boldsymbol{\theta}, \mathbf{a}) &= Q^{\{j+1\}}(\boldsymbol{\theta}, \mathbf{a}) - Q^*(\boldsymbol{\theta}, \mathbf{a}) \\
&\overset{(7.1\text{quadris})}{=} (1 - \alpha^{\{j\}}) \cdot Q^{\{j\}}(\boldsymbol{\theta}, \mathbf{a}) + \alpha^{\{j\}} \left(\hat{C}^{\{j\}}_{\text{nom}} + C_{\text{pert}}(\mathbf{a}) + \gamma^{\{j\}} \min_{\mathbf{a}' \in \mathcal{A}_{\boldsymbol{\theta}+\mathbf{a}}} Q^{\{j\}}(\boldsymbol{\theta} + \mathbf{a}, \mathbf{a}') \right) \\
&\quad - Q^*(\boldsymbol{\theta}, \mathbf{a}) + \alpha^{\{j\}} Q^*(\boldsymbol{\theta}, \mathbf{a}) - \alpha^{\{j\}} Q^*(\boldsymbol{\theta}, \mathbf{a}) \\
&= (1 - \alpha^{\{j\}}) \cdot \left(Q^{\{j\}}(\boldsymbol{\theta}, \mathbf{a}) - Q^*(\boldsymbol{\theta}, \mathbf{a}) \right) + \alpha^{\{j\}} \cdot \left(\hat{C}^{\{j\}}_{\text{nom}}(\boldsymbol{\theta}) + C_{\text{pert}}(\mathbf{a}) \right. \\
&\quad \left. + \gamma^{\{j\}} \min_{\mathbf{a}' \in \mathcal{A}_{\boldsymbol{\theta}+\mathbf{a}}} Q^{\{j\}}(\boldsymbol{\theta} + \mathbf{a}, \mathbf{a}') - Q^*(\boldsymbol{\theta}, \mathbf{a}) \right) \\
&= (1 - \alpha^{\{j\}}) \cdot \delta^{\{j\}}(\boldsymbol{\theta}, \mathbf{a}) + \alpha^{\{j\}} \cdot F^{\{j\}}(\boldsymbol{\theta}, \mathbf{a})
\end{aligned} \tag{7.12}$$

Instead, for $(\boldsymbol{\theta}, \mathbf{a}) \neq (\boldsymbol{\theta}^{\{j\}}, \mathbf{a}^{\{j\}})$,

$$\begin{aligned}
\delta^{\{j+1\}}(\boldsymbol{\theta}, \mathbf{a}) &= Q^{\{j+1\}}(\boldsymbol{\theta}, \mathbf{a}) - Q^*(\boldsymbol{\theta}, \mathbf{a}) \\
&\overset{(7.1\text{tris})}{=} Q^{\{j\}}(\boldsymbol{\theta}, \mathbf{a}) - Q^*(\boldsymbol{\theta}, \mathbf{a}) \\
&= \delta^{\{j\}}(\boldsymbol{\theta}, \mathbf{a})
\end{aligned} \tag{7.12bis}$$

Observe that for given past $\mathcal{F}^{\{j\}}$, $\delta^{\{j\}}(\boldsymbol{\theta}, \mathbf{a})$ is deterministic. Since the exploration never ends (ϵ is always greater than 0), state-action pair $(\boldsymbol{\theta}, \mathbf{a})$ is visited infinitely many times. Therefore, there is an infinite subsequences of indices for which (7.12) holds instead of (7.12bis). We are thus in the case of Lemma 7.1.2.2.

The first condition of Lemma 7.1.2.2 holds by definition of the state and action spaces. Moreover, the learning rate scheduling (Section 3.3.3) obeys the second condition of Lemma 7.1.2.2. The last condition holds because we define the cost function to be

bounded.[2] This means that we only have to show that the third condition of Lemma 7.1.2.2 holds to prove convergence of $Q^{\{j\}}$ to Q^*.

The optimal Q-table is a fixed point of a contraction operator $\mathbf{H}$ (see Section 1 of [172]), defined for function $m : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ as:

$$(\mathbf{H}m)(\boldsymbol{\theta}, \mathbf{a}) = \mathbb{E}C_{\text{nom}}(\boldsymbol{\theta}) + C_{\text{pert}}(\mathbf{a}) + \nu \cdot \min_{\mathbf{a}' \in \mathcal{A}_{\boldsymbol{\theta}+\mathbf{a}}} m(\boldsymbol{\theta} + \mathbf{a}, \mathbf{a}') \tag{7.13}$$

This operator is a contraction in the sup-norm (Section 1 of [107]), i.e.,

$$\begin{aligned} \|\mathbf{H}m_1 - \mathbf{H}m_2\|_\infty &\leq \nu \cdot \|m_1 - m_2\|_\infty \\ &= \nu \cdot \sup_{(\boldsymbol{\theta}, \mathbf{a}) \in \mathcal{S} \times \mathcal{A}} |m_1(\boldsymbol{\theta}, \mathbf{a}) - m_2(\boldsymbol{\theta}, \mathbf{a})| \end{aligned} \tag{7.14}$$

We now prove that the third condition of Lemma 7.1.2.2 holds for our $F^{\{j\}}(\boldsymbol{\theta}, \mathbf{a}) - \eta^{\{j\}}(\boldsymbol{\theta})$, where $F^{\{j\}}(.,.)$ is defined as in (7.10) and $\eta^{\{j\}}(\boldsymbol{\theta}) : \mathcal{S} \to \mathbb{R}$ is defined as $\eta^{\{j\}}(\boldsymbol{\theta}) = \hat{C}^{\{j\}}_{\text{nom}}(\boldsymbol{\theta}) - \mathbb{E}C_{\text{nom}}(\boldsymbol{\theta})$. Theorem 7.1.1.1 shows that $\lim^u_{j\to\infty} \eta^{\{j\}}(\cdot) = 0$. Therefore, via (7.10):

$\forall \boldsymbol{\theta} \in \mathcal{S}, \mathbf{a} \in \mathcal{A}_{\boldsymbol{\theta}}$,

$$\begin{aligned} & F^{\{j\}}(\boldsymbol{\theta}, \mathbf{a}) - \eta^{\{j\}}(\boldsymbol{\theta}) \\ &= \hat{C}^{\{j\}}_{\text{nom}}(\boldsymbol{\theta}) + C_{\text{pert}}(\mathbf{a}) + \gamma^{\{j\}} \cdot \min_{\mathbf{a}' \in \mathcal{A}_{\boldsymbol{\theta}+\mathbf{a}}} Q^{\{j\}}(\boldsymbol{\theta} + \mathbf{a}, \mathbf{a}') - Q^*(\boldsymbol{\theta}, \mathbf{a}) \\ &- \left(\hat{C}^{\{j\}}_{\text{nom}}(\boldsymbol{\theta}) - \mathbb{E}C_{\text{nom}}(\boldsymbol{\theta}) \right) \\ &= \mathbb{E}C_{\text{nom}}(\boldsymbol{\theta}) + C_{\text{pert}}(\mathbf{a}) + \gamma^{\{j\}} \cdot \min_{\mathbf{a}' \in \mathcal{A}_{\boldsymbol{\theta}+\mathbf{a}}} Q^{\{j\}}(\boldsymbol{\theta} + \mathbf{a}, \mathbf{a}') - Q^*(\boldsymbol{\theta}, \mathbf{a}) \\ &\overset{(7.13)}{=} \mathbf{H}Q^{\{j\}}(\boldsymbol{\theta}, \mathbf{a}) - Q^*(\boldsymbol{\theta}, \mathbf{a}) \\ &\overset{(\text{Since } Q^* = \mathbf{H}Q^*)}{=} \mathbf{H}Q^{\{j\}}(\boldsymbol{\theta}, \mathbf{a}) - \mathbf{H}Q^*(\boldsymbol{\theta}, \mathbf{a}). \end{aligned}$$

In the norm-sup, using (7.14), we obtain:

$$\begin{aligned} \|F^{\{j\}}(\cdot,\cdot) - \eta^{\{j\}}(\boldsymbol{\theta})\|_\infty &\leq \gamma^{\{j\}} \|Q^{\{j\}}(\cdot,\cdot) - Q^*(\cdot,\cdot)\|_\infty \\ &= \gamma^{\{j\}} \|\delta^{\{j\}}(\cdot,\cdot)\|_\infty \end{aligned}$$

Applying expectation given past $\mathcal{F}^{\{j\}}$ (and considering that $\delta^{\{j\}}(\cdot,\cdot)$ is deterministic, as we wrote right after (7.12bis)), we obtain

$$\mathbb{E}\left[\|F^{\{j\}}(\cdot,\cdot) - \eta^{\{j\}}(\cdot)\|_\infty | \mathcal{F}^{\{j\}} \right] \leq \gamma^{\{j\}} \|\delta^{\{j\}}(\cdot,\cdot)\|_\infty$$

which proves that the third condition in Lemma 7.1.2.1 holds. Then, by Lemma 7.1.2.2, $\delta^{\{j\}}$ converges to 0 with probability 1, which implies, via (7.11), that $Q^{\{j\}}$ converges to Q^* with probability 1. □

[2] For instance, one could consider that the cost measured at each time-slot cannot exceed the upstream link capacity.

7.1.3 Absorbing state

In this section, we will prove that the sequence of actions and states induced by Q^* has an absorbing state that is the discretely optimal state $\hat{\boldsymbol{\theta}}^*$. In what follows, we will use the concept of sequences defined as follows:

Definition 7.1.3.1. *A sequence $s = \{\boldsymbol{\theta}^{(k)}, \mathbf{a}^{(k)}\}_k$ is a sequence of states and actions such that*

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \mathbf{a}^{(k)}$$

Let us denote with $C^{\gamma}_{\text{cum}}(s)$ the cumulative discounted reward (3.11) of any sequence s.

Definition 7.1.3.2. *Sequence $s' = \{\boldsymbol{\theta}'^{(k)}, \mathbf{a}'^{(k)}\}_k$ is optimal if, for any other sequence s'' having the same initial state as s^*, we have $C^{\gamma}_{cum}(s'') \geq C^{\gamma}_{cum}(s')$.*

Definition 7.1.3.3. *Q-table Q is optimal if, for any initial state, any sequence s'' induced by Q, is an optimal sequence.*

We will use the notation $Q^{(k)}$ to refer to the Q-table of Algorithm 1. We will refer to the following sequences:

- s: state and action sequence induced by sequence $Q^{(k)}$ of Q-tables obtained with our Algorithm 1. We call it "offline sequence".
- s_ϵ: sequence induced by the online policy: such a sequence follows $Q^{(k)}$ with probability $1-\epsilon$ and takes a random action with probability ϵ. This is the sequence that comes out of the actions chosen in lines 3 and 4 of Algorithm 1. We call it "online sequence".
- s^*: sequence induced by the optimal Q-table Q^*.

It is worth emphasizing that when applying Algorithm 1, we do not traverse sequence s, as we do not take actions induced by $Q^{(k)}$. Indeed, we explore from time to time. In this sense, s is a theoretical sequence, that we use as a reference in our proofs, but that we never follow in reality. What we really follow is s_ϵ.

Lemma 7.1.3.4. *Any sequence s^* induced by Q^* has an absorbing state, i.e.,*

$$\exists \boldsymbol{\theta}_{abs} \in \mathcal{S}, k' > 0 : \boldsymbol{\theta}^{(k)} = \boldsymbol{\theta}_{abs}, \quad k \geq k'$$

Proof. Suppose by contradiction that $s^* = \{\boldsymbol{\theta}'^{(k)}, \mathbf{a}'^{(k)}\}_k$ does not have an absorbing state. If that were the case, we could construct a modified version s'' of s^* as follows. We take the best of the allocations visited, i.e., $\boldsymbol{\theta}_{\text{best}} \in \arg\min_{k=0}^{\infty} \mathbb{E}C_{\text{nom}}(\boldsymbol{\theta}^{(k)})$. Suppose k_1 is the first time-slot in which such allocation is visited. Sequence $s'' = \{\boldsymbol{\theta}''^{(k)}, \mathbf{a}''^{(k)}\}_k$ is as follows:

$$\boldsymbol{\theta}''^{(k)} = \begin{cases} \boldsymbol{\theta}'^{(k)} & \text{if } k \leq k_1 \\ \boldsymbol{\theta}_{\text{best}} & \text{otherwise} \end{cases} \qquad \mathbf{a}''^{(k)} = \begin{cases} \mathbf{a}'^{(k)} & \text{if } k \leq k_1 \\ \mathbf{0} \text{ (null action)} & \text{otherwise} \end{cases} \tag{7.15}$$

The difference of cumulative discounted cost (3.11) induced by the two sequences s^* and s'' is

$$\begin{aligned}
&C^{\gamma}_{\text{cum}}(s^*) - C^{\gamma}_{\text{cum}}(s'') \\
&= \lim_{T\to\infty} \mathbb{E}\Bigg[\sum_{k=k_1+1}^{T} \gamma^{(k)} \Bigg(C_{\text{nom}}(\boldsymbol{\theta}'^{(k)}, \omega) + C_{\text{pert}}(\mathbf{a}'^{(k)}) - C_{\text{nom}}(\boldsymbol{\theta}''^{(k_1)}, \omega) \Bigg) \Bigg] \\
&= \lim_{T\to\infty} \Bigg[\sum_{k=k_1+1}^{T} \gamma^{(k)} \Bigg(\mathbb{E}C_{\text{nom}}(\boldsymbol{\theta}'^{(k)}, \omega) - \mathbb{E}C_{\text{nom}}(\boldsymbol{\theta}_{\text{best}}, \omega) + C_{\text{pert}}(\mathbf{a}'^{(k)}) \Bigg) \Bigg] \qquad (7.16)
\end{aligned}$$

For any k, by construction of $\boldsymbol{\theta}_{\text{best}}$, we have

$$\mathbb{E}C_{\text{nom}}(\boldsymbol{\theta}'^{(k)}, \omega) - \mathbb{E}C_{\text{nom}}(\boldsymbol{\theta}_{\text{best}}, \omega) \geq 0.$$

Moreover, at least one action $\mathbf{a}'^{(k)}$ is non-null, as we have assumed that s^* does not have any absorbing state. Therefore, (7.16) is positive, which is absurd as it violates Definitions 7.1.3.2 and 7.1.3.3. □

Lemma 7.1.3.5. *If discount factor γ is sufficiently close to* 1*, the absorbing state of sequence s^* is a discretely optimal allocation* (3.20)*.*

Proof. Let us define an undirected graph $\mathcal{G} = (\mathcal{S}, \mathcal{A})$ where each node $\boldsymbol{\theta} \in \mathcal{S}$ is a state and each edge $\mathbf{a} \in \mathcal{A}$ is an action. Such an edge connects state $\boldsymbol{\theta}$ with state $\boldsymbol{\theta} + \mathbf{a}$ and has weight $\mathbb{E}C_{\text{nom}}(\boldsymbol{\theta}, \omega) + C_{\text{pert}}(\mathbf{a})$. Let us denote with $s(\boldsymbol{\theta}, \boldsymbol{\theta'})$ shortest path on such a graph between nodes $\boldsymbol{\theta}$ and $\boldsymbol{\theta'}$, where the cost of the path is the sum of the cost on the arcs. If such a path is $\boldsymbol{\theta} = \boldsymbol{\theta}^{[0]} \xrightarrow{\mathbf{a}^{[0]}} \boldsymbol{\theta}^{[1]} \dots \xrightarrow{\mathbf{a}^{[n-1]}} \boldsymbol{\theta}^{[n]} = \boldsymbol{\theta'}$, the discounted cost accumulated over this path is:

$$C^{\gamma}_{\text{cum}}(s(\boldsymbol{\theta}, \boldsymbol{\theta'})) = \sum_{j=0}^{n-1} \gamma^j \cdot \Big(\mathbb{E}C_{\text{nom}}(\boldsymbol{\theta}^{[j]}, \omega) + C_{\text{pert}}(\mathbf{a}^{[j]}) \Big) \qquad (7.17)$$

Let us define:

$$M \triangleq \max_{\boldsymbol{\theta}, \boldsymbol{\theta'} \in \mathcal{S}} C^{\gamma}_{\text{cum}}(s(\boldsymbol{\theta}, \boldsymbol{\theta'})) \qquad (7.18)$$

Let us take a discretely optimal state $\hat{\boldsymbol{\theta}}^*$. Thanks to Lemma 7.1.3.4, we know that s^* goes to an absorbing state $\boldsymbol{\theta}_{\text{abs}}$ at a certain timeslot k' and does change state anymore. Let us suppose by contradiction that $\boldsymbol{\theta}_{\text{abs}}$ is not discretely optimal and define quantity

$$\delta C = \mathbb{E}C(\boldsymbol{\theta}_{\text{abs}}, \omega) - \mathbb{E}C(\hat{\boldsymbol{\theta}}^*, \omega). \qquad (7.19)$$

By construction, $\delta C > 0$, otherwise $\boldsymbol{\theta}_{\text{abs}}$ would be the discretely optimal.

Let us construct another sequence s'' such that it is the same as s^* up to time slot k'. Then, while s^* stays in $\boldsymbol{\theta}_{\text{abs}}$, s'' takes non-null actions and follows the shortest path $s(\boldsymbol{\theta}_{\text{abs}}, \hat{\boldsymbol{\theta}}^*)$ and then it stays in $\hat{\boldsymbol{\theta}}^*$ and does not change state anymore. Suppose that n is the length of such a shortest path. Let us compute the difference between the cumulative discounted cost of $s^* = \{\boldsymbol{\theta}'^{(k)}, \mathbf{a}'^{(k)}\}$ and $s'' = \{\boldsymbol{\theta}''^{(k)}, \mathbf{a}''^{(k)}\}$:

$$C^{\gamma}_{\text{cum}}(s'') - C^{\gamma}_{\text{cum}}(s^*) \tag{7.20}$$

$$= \mathbb{E}\Bigg[\sum_{k=0}^{\infty} \gamma^{(k)} \cdot \Big(C_{\text{nom}}(\boldsymbol{\theta}''^{(k)}, \omega) + C_{\text{pert}}(\mathbf{a}''^{(k)})\Big) - \gamma^{(k)} \cdot \Big(C_{\text{nom}}(\boldsymbol{\theta}'^{(k)}, \omega) + C_{\text{pert}}(\mathbf{a}'^{(k)})\Big)\Bigg] \tag{7.21}$$

$$= \mathbb{E}\Bigg[\sum_{k=0}^{k'+n-1} \gamma^{(k)} \cdot \Big(C_{\text{nom}}(\boldsymbol{\theta}''^{(k)}, \omega) + C_{\text{pert}}(\mathbf{a}''^{(k)}) - C_{\text{nom}}(\boldsymbol{\theta}'^{(k)}, \omega) - C_{\text{pert}}(\mathbf{a}'^{(k)})\Big) \tag{7.22}$$

$$+ \sum_{k=k'+n}^{\infty} \gamma^{(k)} \cdot \Big(C_{\text{nom}}(\boldsymbol{\theta}''^{(k)}, \omega) + C_{\text{pert}}(\mathbf{a}''^{(k)}) - C_{\text{nom}}(\boldsymbol{\theta}'^{(k)}, \omega) - C_{\text{pert}}(\mathbf{a}'^{(k)})\Big)\Bigg]$$

$$= \gamma^{(k')} \cdot C^{\gamma}_{\text{cum}}(s(\boldsymbol{\theta}_{\text{abs}}, \hat{\boldsymbol{\theta}}^*)) + \sum_{k=k'+n}^{\infty} \gamma^{(k)} \cdot \mathbb{E}C_{\text{nom}}(\hat{\boldsymbol{\theta}}^*, \omega) - \sum_{k=k'}^{\infty} \gamma^{(k)} \cdot \mathbb{E}C_{\text{nom}}(\boldsymbol{\theta}_{\text{abs}}, \omega) \tag{7.23}$$

$$= \gamma^{(k')} \cdot C^{\gamma}_{\text{cum}}(s(\boldsymbol{\theta}_{\text{abs}}, \hat{\boldsymbol{\theta}}^*)) - \sum_{k=k'}^{k'+n-1} \gamma^{(k)} \cdot \mathbb{E}C_{\text{nom}}(\boldsymbol{\theta}_{\text{abs}}, \omega)$$

$$+ \sum_{k=k'+n}^{\infty} \gamma^{(k)} \cdot \Big(\mathbb{E}C_{\text{nom}}(\hat{\boldsymbol{\theta}}^*, \omega) - \mathbb{E}C_{\text{nom}}(\boldsymbol{\theta}_{\text{abs}}, \omega)\Big) \tag{7.24}$$

Via (7.19) and elementary calculus (for the summation of truncated geometric series), we obtain:

$$\sum_{k=k'+n}^{\infty} \gamma^{(k)} \cdot \Big(\mathbb{E}C_{\text{nom}}(\hat{\boldsymbol{\theta}}^*, \omega) - \mathbb{E}C_{\text{nom}}(\boldsymbol{\theta}_{\text{abs}}, \omega)\Big) = -\delta C \cdot \underbrace{\sum_{k=k'+n}^{\infty} \gamma^{(k)}}_{\text{trunc. geom. series}} = -\delta C \cdot \frac{\gamma^{k'+n}}{1-\gamma}. \tag{7.25}$$

By replacing (7.18) and (7.25) into (7.20), we get:

$$C^{\gamma}_{\text{cum}}(s'') - C^{\gamma}_{\text{cum}}(s^*) \leq \underbrace{\gamma^{(k')} \cdot M}_{a} \underbrace{- \sum_{k=k'}^{k'+n-1} \gamma^{(k)} \cdot \mathbb{E}C_{\text{nom}}(\boldsymbol{\theta}_{\text{abs}}, \omega)}_{b} \underbrace{- \delta C \cdot \frac{\gamma^{k'+n}}{1-\gamma}}_{c} \tag{7.26}$$

For $\gamma \to 1$, terms a and b tends to a constant, while term c tends to infinity. Therefore, $\lim_{\gamma \to 1} \left(C^{\gamma}_{\text{cum}}(s'') - C^{\gamma}_{\text{cum}}(s^*)\right) = -\infty$. This means that if γ is sufficiently close to 1, then $C^{\gamma}_{\text{cum}}(s'') < C^{\gamma}_{\text{cum}}(s^*)$, which is absurd as it violates Definitions 7.1.3.2 and 7.1.3.3. □

7.1.4 Convergence of the offline sequence

In this section, we will prove that sequence s of actions induced by $Q^{(k)}$ converges to the sequence of actions induced by Q^*.

Definition 7.1.4.1. *Let s_1 and s_2 be two sequences of actions and states. We denote the difference of average expected cost induced by sequences s_1 and s_2 by:*

$$D(s_1, s_2) \triangleq \lim_{T \to \infty} \frac{1}{T} \mathbb{E}\left[\sum_{k=0}^{T} \left(C_1^{(k)} - C_2^{(k)}\right)\right] \tag{7.27}$$

where $C^{(k)}$ is the instantaneous cost defined by (3.3)*.*

Definition 7.1.4.2. *We say that a sequence s_1 converges to a sequence s_2 if and only if*

$$D(s_1, s_2) = 0.$$

Proposition 7.1.4.3. *Let $Q_1^{(k)}$ and $Q_2^{(k)}$ be two sequences of Q-tables such that $\arg\min_{\mathbf{a}} Q_1^{(k)}(\boldsymbol{\theta}, \mathbf{a}) = \arg\min_{\mathbf{a}} Q_2^{(k)}(\boldsymbol{\theta}, \mathbf{a}), \forall k > k', \forall \boldsymbol{\theta} \in \mathcal{S}$. Then, if a sequence s_1 is induced by $Q_1^{(k)}$, it must be induced by $Q_2^{(k)}$ starting from k'.*

The following lemma proves that our offline sequence s approaches the optimal sequence of states and actions.

Lemma 7.1.4.4. *If s is a sequence induced by Q-tables $Q^{(k)}$, obtained with our Algorithm 1, then*

$$\mathbb{P}\Big(\exists K > 0, s \text{ is induced by } Q^* \text{ starting from } K\Big) = 1$$

Proof. In Theorem 7.1.2.3, we proved that $Q^{(k)}$ converges with probability 1 to Q^*. Then, by definition, we have

$$\mathbb{P}\left(\lim_{k \to \infty} |Q^{(k)}(\boldsymbol{\theta}, \mathbf{a}) - Q^*(\boldsymbol{\theta}, \mathbf{a})| < e\right) = 1, \forall e > 0, \forall (\boldsymbol{\theta}, \mathbf{a}) \tag{7.27bis}$$

Let us denote by $\epsilon_{\min}$ the minimum difference between two distinct Q-values of Q^*:

$$\epsilon_{\min} = \min \left\{ |Q^*(\boldsymbol{\theta}, \mathbf{a}) - Q^*(\boldsymbol{\theta}', \mathbf{a}')| \;\middle|\; \begin{array}{l} \boldsymbol{\theta}, \boldsymbol{\theta}' \in \mathcal{S}, \\ \mathbf{a} \in \mathcal{A}_{\boldsymbol{\theta}}, \mathbf{a}' \in \mathcal{A}_{\boldsymbol{\theta}'}, \\ Q^*(\boldsymbol{\theta}, \mathbf{a}) \neq Q^*(\boldsymbol{\theta}', \mathbf{a}') \end{array} \right\}$$

Formula (7.27bis) implies that $\mathbb{P}\Big(\lim_{k\to\infty}|Q^{(k)}(\boldsymbol{\theta},\mathbf{a})-Q^*(\boldsymbol{\theta},\mathbf{a})|<\frac{\epsilon_{\min}}{2}\Big)=1,\forall(\boldsymbol{\theta},\mathbf{a})$.

This implies that,

$$\mathbb{P}\Big(\exists K_{\boldsymbol{\theta},\mathbf{a}}>0,\forall k>K_{\boldsymbol{\theta},\mathbf{a}},|Q^{(k)}(\boldsymbol{\theta},\mathbf{a})-Q^*(\boldsymbol{\theta},\mathbf{a})|<\epsilon_{\min}\Big)=1,\forall(\boldsymbol{\theta},\mathbf{a}) \tag{7.27tris}$$

Let us call $E_{\boldsymbol{\theta},\mathbf{a}}$ the following event:

$$\exists K_{\boldsymbol{\theta},\mathbf{a}}>0,\forall k>K_{\boldsymbol{\theta},\mathbf{a}},|Q^{(k)}(\boldsymbol{\theta},\mathbf{a})-Q^*(\boldsymbol{\theta},\mathbf{a})|<\epsilon_{\min}$$

Formula (7.27tris) implies that $\mathbb{P}(E_{\boldsymbol{\theta},\mathbf{a}})=1,\forall(\boldsymbol{\theta},\mathbf{a})$. Hence $\mathbb{P}(\bar{E}_{\boldsymbol{\theta},\mathbf{a}})=0,\forall(\boldsymbol{\theta},\mathbf{a})$. By taking $K=\max_{\boldsymbol{\theta},\mathbf{a}}K_{\boldsymbol{\theta},\mathbf{a}}$ (that exists since $\mathcal{S}$ and $\mathcal{A}$ are finite), we can write:

$$\mathbb{P}\left(\begin{array}{c}\exists K>0,\forall k>K,\forall\boldsymbol{\theta}\in\mathcal{S},\mathbf{a}\in\mathcal{A}_{\boldsymbol{\theta}}\\ \Rightarrow|Q^{(k)}(\boldsymbol{\theta},\mathbf{a})-Q^*(\boldsymbol{\theta},\mathbf{a})|<\epsilon_{\min}\end{array}\right)=1$$

and thus

$$\mathbb{P}\left(\begin{array}{c}\exists K>0,\forall k>K,\forall\boldsymbol{\theta}\in\mathcal{S}\\ \Rightarrow\arg\min_{\mathbf{a}}Q^{(k)}(\boldsymbol{\theta},\mathbf{a})=\arg\min_{\mathbf{a}}Q^*(\boldsymbol{\theta},\mathbf{a})\end{array}\right)=1$$

Thanks to Proposition 7.1.4.3, we will have that s is induced by Q^* starting from K, hence the result. □

Corollary 7.1.4.5. *Sequence s converges to sequence s^*.*

Proof. Thanks to Lemma 7.1.3.5, we can claim that $\exists K^*>0$, starting from which s^* has arrived to a discretely optimal state $\hat{\boldsymbol{\theta}}^*$. Lemma 7.1.4.4 allows us to write:

$$\mathbb{P}\Big(\exists K>0,\forall k>K,s\text{ takes actions induced by }Q^*\Big)=1$$

Let $K_{\max}=\max(K^*,K)$ and let $\mathcal{E}$ be the following event:

$$\exists K_s^*\geq K_{\max},\text{ starting from which }s\text{ has arrived to }\hat{\boldsymbol{\theta}}^*.$$

Thanks to Lemma 7.1.3.5 and Lemma 7.1.4.4, we can write:

$$\mathbb{P}(\mathcal{E})=1$$

Let us now compute $D(s,s^*)$ by applying the law of total expectations:

$$D(s,s^*)=\lim_{T\to\infty}\frac{1}{T}\mathbb{E}\left[\sum_{k=0}^{T}\Big(C^{(k)}-C^{*(k)}\Big)\right]$$

$$=\lim_{T\to\infty}\left(\mathbb{P}(\mathcal{E})\cdot\frac{1}{T}\mathbb{E}\left[\sum_{k=0}^{T}\Big(C^{(k)}-C^{*(k)}\Big)|\mathcal{E}\right]+(1-\mathbb{P}(\mathcal{E}))\cdot\frac{1}{T}\mathbb{E}\left[\sum_{k}^{T}\Big(C^{(k)}-C^{*(k)}\Big)|\bar{\mathcal{E}}\right]\right)$$

$$=\lim_{T\to\infty}\frac{1}{T}\sum_{k=0}^{T}\Big[\mathbb{E}[C^{(k)}|\mathcal{E}]-\mathbb{E}[C^{*(k)}|\mathcal{E}]\Big]$$

After $K_c = \max(K_s^*, K^*)$ both s and s^* will be in the same state $\hat{\boldsymbol{\theta}}^*$, if event $\mathcal{E}$ is verified. Therefore, $\mathbb{E}[C^{(k)}|\mathcal{E}] = \mathbb{E}[C^{*(k)}|\mathcal{E}], \forall k \geq K_c$ and thus

$$D(s, s^*) = \lim_{T\to\infty} \frac{1}{T} \sum_{k=0}^{K_c} \Big[\mathbb{E}[C^{(k)}|\mathcal{E}] - \mathbb{E}[C^{*(k)}|\mathcal{E}]\Big] = 0$$

□

7.1.5 Convergence of the online sequence

In this section, we will prove that the sequence s_ϵ of states and actions visited by Algorithm 1 online, converges to $\hat{\boldsymbol{\theta}}^*$, which represents the main result of our work.

To do so, we need to show that

$$\exists\ \hat{\boldsymbol{\theta}}^*,\ \mathbb{P}\bigg(\lim_{k\to\infty}\ ||\ \boldsymbol{\theta}^{(k)} - \hat{\boldsymbol{\theta}}^*||\ > 0\ \bigg) = 0.$$

We will prove a stronger property:

$$\exists \text{ discretely optimal state } \hat{\boldsymbol{\theta}}^* \in \mathcal{S}, \text{ such that } \lim_{k\to\infty} \mathbb{P}(\boldsymbol{\theta}^{(k)} = \hat{\boldsymbol{\theta}}^*) = 1 \tag{7.28}$$

Let us denote with $\mathcal{B}(k, d)$ the event that from time-slot $k - d$ to $k - 1$ there have been no random actions taken. The probability of this event is:

$$\mathbb{P}(\mathcal{B}(k, d)) = \prod_{k'=k-d}^{k} (1 - \epsilon^{(k')}) \tag{7.29}$$

where $\epsilon^{(k')}$ follows (3.19). Since $\lim_{k\to\infty} \epsilon^{(k)} = 0$, then $\forall d > 0, \lim_{k\to\infty} \mathbb{P}(\mathcal{B}(k, d)) = 1$. If $\mathcal{B}(k, d)$ is verified, the actions taken by our algorithm are those induced by $Q^{(k'')}, \forall k'' = k - d, .., k$. Therefore, Lemma 7.1.4.4 applies and if we take $d > K$ (where K is the one indicated by Lemma 7.1.4.4), we know that if $\mathcal{B}(k, d)$ is verified, then the actions taken by our algorithm in time-slots $k'' = k - d + K, .., k$ are those suggested by Q^*. Hence, Lemma 7.1.3.4 applies and if we take $d > K + k'$ (where K is the one indicated by Lemma 7.1.3.4), we know that the state in which our algorithm brings the system in slots $k'' = k - d + K + k', .., k$, is an absorbing state.

Thanks to Lemma 7.1.3.5, we know that this absorbing state is $\hat{\boldsymbol{\theta}}^*$, if γ is sufficiently large. Therefore, if $\mathcal{B}(k, d)$ is verified, then

$$\boldsymbol{\theta}^{(k'')} = \hat{\boldsymbol{\theta}}^*, \forall k'' = k - d + K + k', .., k.$$

In other words, if $\mathcal{B}(k, d)$ is verified for a sufficiently large γ, then $\exists$ discretely optimal state $\hat{\boldsymbol{\theta}}^*$ such that $\boldsymbol{\theta}^{(k)} = \hat{\boldsymbol{\theta}}^*$. Therefore, $\mathbb{P}(\boldsymbol{\theta}^{(k)} = \hat{\boldsymbol{\theta}}^*) \geq \mathbb{P}(\mathcal{B}(k, d))$. Since the second term tends to 1 when $k \to \infty$ (7.29), then (7.28) is verified.

7.2 Proof of the Corollary 3.4.2.2

Proof. We have seen that our allocations have an absorbing state $\hat{\boldsymbol{\theta}}^*$ that is discretely optimal, i.e., there exists $k' > 0$ such that $\boldsymbol{\theta}^{(k)} = \hat{\boldsymbol{\theta}}^*$ for $k \geq k'$ with probability 1. We can compute

$$\begin{aligned}
&\lim_{Z\to\infty} \frac{1}{Z}\mathbb{E}\left[C_{\text{cum}}(Z) - C^*_{\text{cum}}(Z)\right] \\
&= \lim_{Z\to\infty} \frac{1}{Z}\mathbb{E}\left[C_{\text{cum}}(k') - C^*_{\text{cum}}(k')\right] + \lim_{Z\to\infty} \frac{1}{Z}\mathbb{E}\left[C_{\text{cum}}(Z) - C_{\text{cum}}(k') - (C^*_{\text{cum}}(Z) - C^*_{\text{cum}}(k'))\right] \\
&= \lim_{Z\to\infty} \frac{1}{Z} \sum_{k=k'+1}^{Z} \left(\mathbb{E}\left[(C_{\text{nom}}(\boldsymbol{\theta}^{(k)}, \omega) + C_{\text{pert}}(\mathbf{a}^{(k)}) - C_{\text{nom}}(\boldsymbol{\theta}^*, \omega)\right]\right) \\
&= \lim_{Z\to\infty} \frac{Z - (k'+1)}{Z} \cdot \left(\mathbb{E}_\omega\left[C_{\text{nom}}(\hat{\boldsymbol{\theta}}^*, \omega)\right] - \mathbb{E}_\omega\left[C_{\text{nom}}(\boldsymbol{\theta}^*, \omega)\right]\right) \\
&= \mathbb{E}_\omega\left[C_{\text{nom}}(\hat{\boldsymbol{\theta}}^*, \omega)\right] - \mathbb{E}_\omega\left[C_{\text{nom}}(\boldsymbol{\theta}^*, \omega)\right] = G_\Delta
\end{aligned}$$

□

7.3 Proof of Proposition 3.4.2.3

Suppose an oracle that knows exactly (i) the probability f_p that a request is for SP p, (ii) the cacheability ζ_p and (iii) the popularity of each object (c, p) within the catalog of each SP. Such an oracle can compute the rate of requests $\lambda_{c,p} = \lambda \cdot f_p \cdot \zeta_p \cdot \rho_{c,p}$ for each object. The expected value of the nominal cost when the set of cached objects is $\mathcal{K}$ is

$$\mathbb{E}C_{\text{nom}}(\mathcal{K}) = \lambda - \sum_{(c,p)\in\mathcal{K}} \lambda_{c,p}$$

The following property will be useful later.

Lemma 7.3.0.1. *The set function $\mathcal{K} \to \mathbb{E}C_{nom}(\mathcal{K})$ is monotonically increasing, i.e., if $\mathcal{K} \subseteq \mathcal{K}'$, then $\mathbb{E}C_{nom}(\mathcal{K}) \leq \mathbb{E}C_{nom}(\mathcal{K}')$.*

To minimize the nominal cost, we resort to a greedy algorithm for the Simple Allocation Problem [173]: the oracle puts into the cache the objects with the highest $\lambda_{c,p}$, up to filling all K cache-slots. Let us denote with $\mathcal{K}^*$ the set of cached objects in this way. The optimal allocation $\boldsymbol{\theta}^*$ can be obtained by simply counting the number of objects of each SP p that we find in $\mathcal{K}^*$. In particular, θ^*_p is equal to the number of objects of SP p present in $\mathcal{K}^*$.

With no loss of generality, suppose that within the catalog of each SP p the objects are indexed as $c = 1, 2, \ldots, N_p$ and sorted from the most popular to the least, so that $\lambda_{c,p} \geq \lambda_{c+1,p}$. If the discretization step is Δ, objects cannot be selected one by one, but

they can only be cached in batches of Δ elements. We thus divide the catalog of each SP in batches of Δ objects. For instance, batch $\mathcal{B}_{i,p}$ is

$$\mathcal{B}_{i,p} = \{\text{object}(c,p) | c = (i-1)\cdot\Delta + 1, \ldots, i\cdot\Delta\},\ i = 1, 2, ..$$

The rate of such a batch is defined as the traffic we can omit downloading from a distant location if we store this batch into the cache, i.e.:

$$\lambda(\mathcal{B}_{i,p}) \triangleq \sum_{(c,p)\in\mathcal{B}_{i,p}} \lambda_{c,p}.$$

In order to minimize the nominal cost with the discretized model, the oracle can add to the cache the batches with the highest rate, up to filling the K cache-slots. We denote by $\hat{\mathcal{K}}^*$ the set of cached objects obtained in this way. The discretely optimal allocation $\hat{\boldsymbol{\theta}}^* = (\hat{\theta}_1^*, \ldots, \hat{\theta}_P^*)$ can be obtained by simple counting: $\hat{\theta}_p^*$ is equal to the number of objects of p that are present in $\hat{\mathcal{K}}^*$.

The construction of $\hat{\mathcal{K}}^*$ and $\hat{\boldsymbol{\theta}}^*$ is summarized in Algorithm 6, which extends the greedy algorithm. Note that to construct $\mathcal{K}^*$ and $\boldsymbol{\theta}^*$ one can use the same algorithm, setting $\Delta = 1$.

Algorithm 6: Compute $\hat{\boldsymbol{\theta}}^*$

Data: Δ, $\lambda_{c,p} \forall (c,p)$.
Result: $\hat{\mathcal{K}}^*$, $\hat{\boldsymbol{\theta}}^*$

$\hat{\mathcal{K}}^* \leftarrow \emptyset$;
$\hat{\boldsymbol{\theta}}^* \leftarrow \mathbf{0} = (0, \ldots, 0)$;
$i_p = 1$ for $p = 1, \ldots, P$; // We use this pointer to save the last added batch of each SP
while $|\hat{\mathcal{K}}^*| + \Delta < K$; // We can still add a batch of Δ objects in the cache
do
  $p^{\text{best}} \in \arg\max_{p=1}^{P} \lambda(\mathcal{B}_{i_p,p})$; // Select the SP with the largest batch rate
  $\hat{\mathcal{K}}^* \leftarrow \hat{\mathcal{K}}^* \cup \mathcal{B}_{i_{p^{\text{best}}},p^{\text{best}}}$; // Add batch of SP p^{best} in the cache
  $\hat{\theta}^*_{p^{\text{best}}} \leftarrow \hat{\theta}^*_{p^{\text{best}}} + \Delta$; // Give it the corresponding cache-slots
end

The optimality gap can be expressed in terms of the sets $\mathcal{K}^*$ and $\hat{\mathcal{K}}^*$:

$$G_\Delta = \mathbb{E}C_{\text{nom}}(\hat{\mathcal{K}}^*) - \mathbb{E}C_{\text{nom}}(\mathcal{K}^*) \tag{7.30}$$

In order to bound the previous quantity, we construct two sets $\mathcal{K}_-$ and $\mathcal{K}_+$ around $\mathcal{K}^*$ and $\hat{\mathcal{K}}^*$.

Lemma 7.3.0.2. *There exists two sets $\mathcal{K}_-$ and $\mathcal{K}_+$ such that*

$$\mathcal{K}_- \subseteq \mathcal{K}^* \subseteq \mathcal{K}_+ \tag{7.31}$$

$$\mathcal{K}_- \subseteq \hat{\mathcal{K}}^* \subseteq \mathcal{K}_+ \tag{7.32}$$

and

$$\mathbb{E}C_{nom}(\mathcal{K}_-) - \mathbb{E}C_{nom}(\mathcal{K}_+) \leq \sum_{p=1}^{P} \sum_{c=1}^{\Delta} \lambda_{c,p}$$

Proof. We know that the θ_p^* most popular objects of SP p are stored in $\mathcal{K}^*$. These include the $\left\lfloor \frac{\theta_p^*}{\Delta} \right\rfloor$ batches of SP p with the highest rate. Let us construct a set composed of these batches:

$$\mathcal{K}_- = \bigcup_{p=1}^{P} \left\{ \bigcup_{i=1}^{\left\lfloor \frac{\theta_p^*}{\Delta} \right\rfloor} \mathcal{B}_{i,p} \right\} \tag{7.33}$$

By construction, $\mathcal{K}_- \subseteq \mathcal{K}^*$. By construction of $\hat{\mathcal{K}}^*$, the aforementioned batches are also contained into $\hat{\mathcal{K}}^*$. Therefore, $\mathcal{K}_- \subseteq \hat{\mathcal{K}}^*$.

We now construct set $\mathcal{K}_+$ adding to $\mathcal{K}_-$ one additional batch per each SP:

$$\mathcal{K}_+ = \bigcup_{p=1}^{P} \left\{ \bigcup_{i=1}^{\left\lceil \frac{\theta_p^*}{\Delta} \right\rceil} \mathcal{B}_{i,p} \right\}$$

By construction $\mathcal{K}_+ \supseteq \mathcal{K}^*$ and $\mathcal{K}_+ \supseteq \hat{\mathcal{K}}^*$. Summarizing what we have obtained so far:

$$\mathcal{K}_- \subseteq \mathcal{K}^* \subseteq \mathcal{K}_+$$

$$\mathcal{K}_- \subseteq \hat{\mathcal{K}}^* \subseteq \mathcal{K}_+$$

By construction, we have

$$\mathbb{E}C_{\text{nom}}(\mathcal{K}_+) - \mathbb{E}C_{\text{nom}}(\mathcal{K}_-) = \sum_{p=1}^{P} \left(\sum_{i=\left\lfloor \frac{\theta_p^*}{\Delta} \right\rfloor}^{\left\lceil \frac{\theta_p^*}{\Delta} \right\rceil} \lambda(\mathcal{B}_{i,p}) \right) \leq \sum_{p=1}^{P} \lambda(\mathcal{B}_{i',p})$$

where $i' = \left\lfloor \frac{\theta_p^*}{\Delta} \right\rfloor$. Since objects are sorted, within each SP p, from the highest to the lowest rate, the batches have decreasing rate, and thus $\lambda(\mathcal{B}_{i',p}) \leq \lambda(\mathcal{B}_{1,p}) = \sum_{c=1}^{\Delta} \lambda_{c,p}$. □

Thanks to Lemma 7.3.0.1, equations (7.31)-(7.32) imply that

$$\mathbb{E}C_{\text{nom}}(\mathcal{K}_-) \leq \mathbb{E}C_{\text{nom}}(\mathcal{K}^*) \leq \mathbb{E}C_{\text{nom}}(\mathcal{K}_+)$$
$$\mathbb{E}C_{\text{nom}}(\mathcal{K}_-) \leq \mathbb{E}C_{\text{nom}}(\hat{\mathcal{K}}^*) \leq \mathbb{E}C_{\text{nom}}(\mathcal{K}_+)$$

which, in turn, imply that $\mathbb{E}C_{\text{nom}}(\hat{\mathcal{K}}^*) - \mathbb{E}C_{\text{nom}}(\mathcal{K}^*) \leq \mathbb{E}C_{\text{nom}}(\mathcal{K}_-) - \mathbb{E}C_{\text{nom}}(\mathcal{K}_+)$. Using (7.30) and (7.33), we obtain the proposition.

Bibliography

[1] (2022) Netflix vs disney: Who's winning the streaming war? [Online]. Available: https://www.visualcapitalist.com/cp/netflix-versus-disney-subscribers/

[2] (2022) Public IT cloud services market revenue worldwide from 2016 to 2022. [Online]. Available: https://www.statista.com/statistics/370305/global-public-it-cloud-services-spending-by-segment/

[3] J. Qadir, B. Sainz-De-Abajo, A. Khan, B. García-Zapirain, I. De La Torre-Díez, and H. Mahmood, "Towards mobile edge computing: Taxonomy, challenges, applications and future realms," *IEEE Access*, 2020.

[4] R. F. El-Gazzar, "A literature review on cloud computing adoption issues in enterprises," in *International Conference on Transfer and Diffusion of IT*, 2014.

[5] (2017) Augmented and virtual reality: The first wave of 5g killer applications. [Online]. Available: https://www.qualcomm.com/media/documents/files/augmented-and-virtual-reality-the-first-wave-of5g-killer-apps.pdf

[6] P. G. Gopinath and S. K. Vasudevan, "An in-depth analysis and study of load balancing techniques in the cloud computing environment," *Procedia Computer Science*, 2015.

[7] A. Bisong, M. Rahman *et al.*, "An overview of the security concerns in enterprise cloud computing," *arXiv preprint arXiv:1101.5613*, 2011.

[8] C. Feng, Y. Wang, Q. Chen, Y. Ding, G. Strbac, and C. Kang, "Smart grid encounters edge computing: Opportunities and applications," *Advances in Applied Energy*, 2021.

[9] (2023) Autopilot and full self-driving capability. [Online]. Available: https://www.tesla.com/en_gb/support/autopilot

[10] (2023) Infotainment upgrade. [Online]. Available: https://www.tesla.com/en_gb/support/infotainment

[11] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, and W. Shi, "Edge computing for autonomous driving: Opportunities and challenges," *Proceedings of the IEEE*, 2019.

[12] F. Ahsan, N. H. Dana, S. K. Sarker, L. Li, S. Muyeen, M. F. Ali, Z. Tasneem, M. M. Hasan, S. H. Abhi, M. R. Islam *et al.*, "Data-driven next-generation smart grid towards sustainable energy evolution: techniques and technology review," *Protection and Control of Modern Power Systems*, 2023.

[13] Y. Zhang, T. Huang, and E. F. Bompard, "Big data analytics in smart grids: a review," *Energy informatics*, 2018.

[14] Z. Wang *et al.*, "Digital transformation in healthcare: Technology acceptance and its applications," *International Journal of Environmental Research and Public Health*, 2023.

[15] H. Dubey, M. R. Mehl, and K. Mankodiya, "Bigear: Inferring the ambient and emotional correlates from smartphone-based acoustic big data," in *IEEE International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)*, 2016.

[16] H. Dubey, R. Kumaresan, and K. Mankodiya, "Harmonic sum-based method for heart rate estimation using ppg signals affected with motion artifacts," *Journal of Ambient Intelligence and Humanized Computing*, 2018.

[17] L. Mahler, H. Dubey, J. C. Goldberg, and K. Mankodiya, "Use of smartwatch technology for people with dysarthria," in *Motor Speech Conference, Madonna Rehabilitation Hospital*, 2016.

[18] (2019) Study on communication services for critical medical applications. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/22_series/22.826

[19] T. Braud, F. H. Bijarbooneh, D. Chatzopoulos, and P. Hui, "Future networking challenges: The case of mobile augmented reality," in *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2017.

[20] N. Mohan, L. Corneo, A. Zavodovski, S. Bayhan, W. Wong, and J. Kangasharju, "Pruning edge research with latency shears," in *Proceedings of ACM Workshop on Hot Topics in Networks*, 2020.

[21] (2023) Aws for the edge. [Online]. Available: https://aws.amazon.com/edge/

[22] (2023) Lambda@edge. [Online]. Available: https://aws.amazon.com/lambda/edge/

[23] (2023) Hulu case study on aws. [Online]. Available: https://aws.amazon.com/solutions/case-studies/hulu/

[24] (2023) The volkswagen group on aws. [Online]. Available: https://aws.amazon.com/solutions/case-studies/innovators/volkswagen-group/

[25] A. Ali-Eldin, B. Wang, and P. Shenoy, "The hidden cost of the edge: a performance comparison of edge and cloud latencies," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2021.

[26] (2018) Netflix titus. [Online]. Available: https://netflix.github.io/titus/

[27] A. Araldo *et al.*, "Resource allocation for edge computing with multiple tenant configurations," *ACM/SIGAPP SAC*, 2020.

[28] ——, "Caching encrypted content via stochastic cache partitioning," *IEEE/ACM ToN*, 2018.

[29] F. Bronzino *et al.*, "Inferring streaming video quality from encrypted traffic: Practical models and deployment experience," *ACM Sigmetrics*, 2019.

[30] (2023) Netflix open connect. [Online]. Available: https://openconnect.netflix.com/

[31] (2023) Google Global Cache. [Online]. Available: https://support.google.com/interconnect/answer/9058809?hl=en

[32] S. K. Singh, R. Singh, and B. Kumbhani, "The evolution of radio access network towards open-ran: Challenges and opportunities," in *IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, 2020.

[33] (2019) Orange présente son nouveau plan stratégique, engage 2025. [Online]. Available: https://newsroom.orange.com/orange-presente-son-nouveau-plan-strategique-engage-2025/

[34] (2022) Types of instances in amazon EC2. [Online]. Available: https://aws.amazon.com/fr/ec2/instance-types/

[35] (2022) Amazon cloud front. [Online]. Available: https://aws.amazon.com/fr/cloudfront/

[36] (2023) Azure stack. [Online]. Available: https://azure.microsoft.com/fr-fr/products/azure-stack/edge

[37] (2023) Google distributed cloud edge. [Online]. Available: https://cloud.google.com/distributed-cloud-edge

[38] (2023) Edge compute solutions. [Online]. Available: https://www.akamai.com/en/solutions/edge

[39] (2023) Bienvenue dans l'ère de l'edge computing. [Online]. Available: https://cloud.orange-business.com/paroles-dexperts/bienvenue-dans-lere-de-ledge-computing/

[40] (2023) PowerEdge R760xs Rack Server. [Online]. Available: https://www.dell.com/fr-fr/dt/solutions/edge-computing/

[41] (2023) Possibilities at the Edge: Putting intelligence where your data is. [Online]. Available: https://www.hp.com/us-en/workstations/learning-hub/data-analytics-network-edge/

[42] (2023) Intel edge computing. [Online]. Available: https://www.intel.fr/content/www/fr/fr/edge-computing/overview.html

[43] (2023) Nvidia for edge computing. [Online]. Available: https://www.nvidia.com/fr-fr/autonomous-machines/

[44] T. K. Authors. Kubernetes k3. [Online]. Available: https://k3s.io/

[45] (2023) Augmented Reality Market Size to Surpass USD 1109.71 Billion by 2030. [Online]. Available: https://www.globenewswire.com/en/news-release/Augmented-Reality-Market-Size/

[46] N. Hassan, K.-L. A. Yau, and C. Wu, "Edge computing in 5g: A review," *IEEE Access*, 2019.

[47] A. Al-Ansi, A. M. Al-Ansi, A. Muthanna, I. A. Elgendy, and A. Koucheryavy, "Survey on intelligence edge computing in 6g: Characteristics, challenges, potential use cases, and market drivers," *Future Internet*, 2021.

[48] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, 2016.

[49] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE communications surveys & tutorials*, 2017.

[50] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE pervasive Computing*, 2009.

[51] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, 2012.

[52] W. Chu *et al.*, "Joint cache resource allocation and request routing for in-network caching services," *Computer Networks*, 2018.

[53] S. Yang *et al.*, "Online orchestration of collaborative caching for multi-bitrate videos in edge computing," *IEEE TPDS*, 2022.

[54] P. Blasco and D. Gündüz, "Multi-armed bandit optimization of cache content in wireless infostation networks," in *IEEE International Symposium on Information Theory*, 2014.

[55] S. Hoteita *et al.*, "On fair network cache allocation to content providers," *Computer Networks*, 2016.

[56] G. Zheng and V. Friderikos, "Fair cache sharing management for multi-tenant based mobile edge networks," *MobiArch*, 2020.

[57] M. Ahmadi *et al.*, "Cache subsidies for an optimal memory for bandwidth tradeoff in the access network," *JSAC*, 2020.

[58] F. Tütüncüoğlu and G. Dán, "Optimal pricing for service caching and task offloading in edge computing," in *IEEE/IFIP WONS*, 2022.

[59] S. Jošilo and G. Dán, "Joint wireless and edge computing resource management with dynamic network slice selection," *IEEE/ACM ToN*, 2022.

[60] ——, "Wireless and computing resource allocation for selfish computation offloading in edge computing," in *IEEE INFOCOM*, 2019.

[61] J. Du *et al.*, "SDN-based resource allocation in edge and cloud computing systems: An evolutionary stackelberg differential game approach," *IEEE/ACM ToN*, 2022.

[62] T. Bahreini *et al.*, "Mechanisms for resource allocation and pricing in mobile edge computing systems," *IEEE TPDS*, 2021.

[63] F. Fossati *et al.*, "Multi-resource allocation for network slicing," *IEEE/ACM ToN*, 2020.

[64] M. Elkael *et al.*, "Monkey business: Reinforcement learning meets neighborhood search for virtual network embedding," *Computer Networks*, 2022.

[65] W. Huang *et al.*, "Dimensioning resources of Network Slices for energy-performance trade-off," in *IEEE SCC*, 2022.

[66] D. T. Nguyen, L. B. Le, and V. Bhargava, "Price-based resource allocation for edge computing: A market equilibrium approach," *IEEE Transactions on Cloud Computing*, 2021.

[67] S. Zhang, Y. Liang, J. Ge, M. Xiao, and J. Wu, "Provably efficient resource allocation for edge service entities using hermes," *IEEE/ACM ToN*, 2020.

[68] W. Xiaofei *et al.*, "In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning," *IEEE Network*, 2019.

[69] H. Mao *et al.*, "Resource management with deep RL," *HotNets*, 2016.

[70] Z. Fang *et al.*, "Qos-aware scheduling of heterogeneous servers for inference in deep neural networks," *CIKM*, 2017.

[71] J. Rao *et al.*, "VCONF: a RL approach to VMs auto-configuration," *ACM ICAC*, 2009.

[72] F. Mason, G. Nencioni, and A. Zanella, "Using distributed reinforcement learning for resource orchestration in a network slicing scenario," *IEEE/ACM ToN*, 2022.

[73] J. Yuang *et al.*, "Fast reinforcement learning algorithms for resource allocation in data centers," *IFIP*, 2020.

[74] Y.-H. Hung, C.-Y. Wang, and R.-H. Hwang, "Combinatorial clock auction for live video streaming in mobile edge computing," in *Proc. of IEEE INFOCOM WKSHPS*, 2018.

[75] L. Yang, H. Zhang, X. Li, H. Ji, and V. C. Leung, "A distributed computation offloading strategy in small-cell networks integrated with mobile edge computing," *IEEE/ACM Transactions on Networking*, 2018.

[76] S. Chen, L. Wang, and F. Liu, "Optimal admission control mechanism design for time-sensitive services in edge computing," in *Proc. of IEEE INFOCOM*, 2022.

[77] Z. Xiong, S. Feng, D. Niyato, P. Wang, and Z. Han, "Optimal pricing-based edge computing resource management in mobile blockchain," in *Proc. of IEEE ICC*, 2018.

[78] Y. Chen, Z. Li, B. Yang, K. Nai, and K. Li, "A stackelberg game approach to multiple resources allocation and pricing in mobile edge computing," *Future Generation Computer Systems*, 2020.

[79] G. Mitsis, E. E. Tsiropoulou, and S. Papavassiliou, "Price and risk awareness for data offloading decision-making in edge computing systems," *IEEE Systems Journal*, 2022.

[80] R. Roostaei, Z. Dabiri, and Z. Movahedi, "A game-theoretic joint optimal pricing and resource allocation for mobile edge computing in noma-based 5g networks and beyond," *Computer Networks*, 2021.

[81] T. Zhang, "Data offloading in mobile edge computing: A coalition and pricing based approach," *IEEE Access*, 2017.

[82] J. Yan, S. Bi, L. Duan, and Y.-J. A. Zhang, "Pricing-driven service caching and task offloading in mobile edge computing," *IEEE Transactions on Wireless Communications*, 2021.

[83] L. Li, M. Siew, and T. Q. Quek, "Learning-based pricing for privacy-preserving job offloading in mobile edge computing," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2019.

[84] Z. Tang, F. Zhang, X. Zhou, W. Jia, and W. Zhao, "Pricing model for dynamic resource overbooking in edge computing," *IEEE Transactions on Cloud Computing*, 2022.

[85] S. Chen, L. Li, Z. Chen, and S. Li, "Dynamic pricing for smart mobile edge computing: a reinforcement learning approach," *IEEE Wireless Communications Letters*, 2021.

[86] F. Lyu, X. Cai, F. Wu, H. Lu, S. Duan, and J. Ren, "Dynamic pricing scheme for edge computing services: A two-layer reinforcement learning approach," in *IEEE/ACM IWQoS*, 2022.

[87] P. Wang, B. Di, L. Song, and N. R. Jennings, "Multi-layer computation offloading in distributed heterogeneous mobile edge computing networks," *IEEE Trans. on Cognitive Comm. and Netw.*, 2022.

[88] X. Wang, J. Ye, and J. C. Lui, "Decentralized scheduling and dynamic pricing for edge computing: A mean field game approach," *IEEE/ACM Transactions on Networking*, 2022.

[89] S. Wang *et al.*, "Adaptive federated learning in resource constrained edge computing systems," *IEEE JSAC*, 2019.

[90] B. Li, Q. He, F. Chen, H. Jin, Y. Xiang, and Y. Yang, "Auditing cache data integrity in the edge computing environment," *IEEE TPDS*, 2021.

[91] Cisco, "White paper," *Cisco Visual Networking Index: Forecast and Trends*, 2017–2022.

[92] T. V. Doan, L. Pajevic, V. Bajpai, and J. Ott, "Tracing the Path to YouTube: A Quantification of Path Lengths and Latencies Toward Content Caches," *IEEE Communications Magazine*, 2019.

[93] (2020) Internet IP transit provider. [Online]. Available: https://www.thousandeyes.com/learning/techtorials/transit-provider

[94] G. Çakmak and H. Suomi, "A comparison of ISP and MNO interconnection models," in *ICT*, 2014.

[95] B. Ager, N. Chatzis, A. Feldmann, N. Sarrar, S. Uhlig, and W. Willinger, "Anatomy of a large european IXP," in *ACM SIGCOMM*, 2012.

[96] V. Giotsas *et al.*, "O Peer, Where Art Thou? Uncovering Remote Peering Interconnections at IXPs," *IEEE/ACM ToN*, 2021.

[97] C.-F. Liu *et al.*, "Latency and reliability-aware task offloading and resource allocation for mobile edge computing," *IEEE Globecom*, 2017.

[98] X. Lyu *et al.*, "Optimal schedule of mobile edge computing for internet of things using partial information," *IEEE JSAC*, 2017.

[99] S. Arnautov *et al.*, "SCONE: Secure linux containers with intel SGX," *OSDI*, 2016.

[100] A. Krause *et al.*, "Submodular function maximization," *Tractability*, 2011.

[101] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," *The MIT Press*, 2018.

[102] H. Pishro-Nik, *Introduction to probability, statistics, and random processes.* Kappa Research, LLC, 2014.

[103] M. Walker, "Convergence of sequences of functions: Some additional notes lecture notes," *Economics 519: Mathematics for Economists, University of Arizona*, 2017.

[104] W. Fedus *et al.*, "Revisiting fundamentals of experience replay," *ICML*, 2020.

[105] S. Natarajan. (2020) Stretched exponential decay function for epsilon greedy algorithm. [Online]. Available: https://medium.com/analytics-vidhya/stretched-exponential-decay-function-for-epsilon-greedy-algorithm

[106] M. T. Regehr and A. Ayoub, "An elementary proof that Q-learning converges almost surely," *CMPUT 653 course, University of Alberta*, 2021.

[107] F. S. Melo, "Convergence of q-learning: A simple proof," *Institute Of Systems and Robotics, Tech. Rep*, 2001.

[108] O. Granichin *et al.*, "Simultaneous perturbation stochastic approximation for tracking under unknown but bounded disturbances," *IEEE Transactions on Automatic Control*, 2015.

[109] B. Shahriari *et al.*, "Taking the human out of the loop : A review of bayesian optimization," *Proceedings of the IEEE*, 2016.

[110] L. F. Yang *et al.*, "Learning to control in metric space with optimal regret," in *IEEE Annual Allerton Conference on Communication*, 2019.

[111] R. Ortner, "Online regret bounds for markov decision processes with deterministic transitions," *Theoretical Computer Science*, 2010.

[112] O. Dekel, J. Ding, T. Koren, and Y. Peres, "Bandits with switching costs: T 2/3 regret," *ACM Symposium on Theory of computing*, 2014.

[113] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, "The nonstochastic multi-armed bandit problem," *Society for Industrial and Applied Mathematics*, 2003.

[114] R. Warlop *et al.*, "Fighting boredom in recommender systems with linear reinforcement learning," *Advances in NIPS*, 2018.

[115] P. Alatur *et al.*, "Inferring streaming video quality from encrypted traffic: Practical models and deployment experience," *ACM Sigmetrics*, 2019.

[116] O. Dekel and E. Hazan, "Better rates for any adversarial deterministic MDP," *ICML*, 2013.

[117] G. Goel *et al.*, "Beyond Online Balanced Descent: An Optimal Algorithm for Smoothed Online Optimization," *Advances in NIPS*, 2019.

[118] M. Lin *et al.*, "Online optimization with switching cost," *Performance Evaluation Review*, 2012.

[119] W. W. Cohen and H. Hirsh, "To discount or not to discount in reinforcement learning: A case study comparing r learning and q learning," in *Machine Learning Proceedings*, 1994.

[120] R. K. Jain *et al.*, "A quantitative measure of fairness and discrimination," *Eastern Research Laboratory, Digital Equipment Corporation*, 1998.

[121] Y. Siriwardhana *et al.*, "A survey on mobile augmented reality with 5G mobile edge computing: Architectures, applications, and technical aspects," *IEEE Communications Surveys & Tutorials*, 2021.

[122] (2017) Google glass. [Online]. Available: https://www.google.com/glass/start/

[123] (2016) Hololens. [Online]. Available: https://fr.wikipedia.org/wiki/Microsoft_HoloLens

[124] (2019) Hololens 2. [Online]. Available: https://www.microsoft.com/en-us/hololens

[125] (2019) Oculus quest. [Online]. Available: https://store.facebook.com/fr/quest

[126] (2020) Meta quest 2. [Online]. Available: https://store.facebook.com/fr/quest/products/quest-2

[127] M. Erol-Kantarci *et al.*, "Caching and computing at the edge for mobile (AR/VR) in 5G," *Ad Hoc Networks*, 2018.

[128] A. B. Ameur *et al.*, "On the deployability of augmented reality using embedded edge devices," in *IEEE CCNC*, 2021.

[129] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[130] T. K. Authors. Kubernetes k8. [Online]. Available: https://kubernetes.io/docs/concepts/workloads/pods/

[131] L. Kleinrock, *Queuing Systems*. Wiley-Interscience, 1975, vol. 1.

[132] S. Fujishige, *Submodular functions and optimization*. Elsevier, 2005.

[133] Q. Yu *et al.*, "Submodular maximization with multi-knapsack constraints and its applications in scientific literature recommendations," in *IEEE GlobalSIP*, 2016.

[134] (2022) Pokemon GO requirements. [Online]. Available: https://support.pokemon.com/hc/en-us/articles/-Pokemon-GO-Plus-system-requirements-and-compatibility

[135] N. Mohan, L. Corneo, A. Zavodovski, S. Bayhan, W. Wong, and J. Kangasharju, "Pruning edge research with latency shears," in *ACM HotNets*, 2020.

[136] AT&T. Edge solutions. [Online]. Available: https://www.business.att.com/categories/att-edge-solutions.html

[137] R. Xie, Q. Tang, S. Qiao, H. Zhu, F. R. Yu, and T. Huang, "When serverless computing meets edge computing: Architecture, challenges, and open issues," *IEEE Wireless Communications*, 2021.

[138] W.-T. Tsai and G. Qi, "DICB: Dynamic intelligent customizable benign pricing strategy for cloud computing," in *Proc. of IEEE International Conference on Cloud Computing*, 2012.

[139] X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet, and D. Pendarakis, "Efficient resource provisioning in compute clouds via vm multiplexing," in *Proc. of International Conference on Autonomic Computing*, 2010.

[140] (2023) What is serverless? [Online]. Available: https://www.oracle.com/cloud/cloud-native/functions/what-is-serverless/

[141] K. Choi, R. Soma, and M. Pedram, "Off-chip latency-driven dynamic voltage and frequency scaling for an mpeg decoding," in *Annual Design Automation Conference*, 2004.

[142] F. Tütüncüoğlu and G. Dán, "Optimal service caching and pricing in edge computing: a Bayesian Gaussian process bandit approach," *IEEE Transactions on Mobile Computing*, 2022.

[143] ——, "Optimal pricing for service caching and task offloading in edge computing," in *Proc. of IFIP/IEEE WONS*, 2022.

[144] F. Guillemin and V. Q. Rodriguez, "Evaluating the impact of tower companies on the telecommunications market," in *IEEE Conference on Innovation in Clouds, Internet and Networks and Workshops*, 2021.

[145] M. Bilal, M. Canini, R. Fonseca, and R. Rodrigues, "With great freedom comes great opportunity: Rethinking resource allocation for serverless functions," in *ACM European Conference on Computer Systems*, 2023.

[146] H. Qiu and T. Li, "Auction method to prevent bid-rigging strategies in mobile blockchain edge computing resource allocation," *Future Generation Computer Systems*, 2022.

[147] Z. Zhang, Z. Li, and C. Wu, "Optimal posted prices for online cloud resource allocation," in *ACM POMACS*, 2017.

[148] I. Chatzigeorgiou, "Bounds on the lambert function and their application to the outage analysis of user cooperation," *IEEE Communications Letters*, 2013.

[149] F. Doshi-Velez and G. Konidaris, "Hidden parameter markov decision processes: A semiparametric regression approach for discovering latent task parametrizations," in *Proc. of International Joint Conferences on Artificial Intelligence (IJCAI)*, 2016.

[150] T. W. Killian *et al.*, "Robust and efficient transfer learning with hidden parameter markov decision processes," *Proc. of NeurIPS*, 2017.

[151] V. G. Kulkarni, *Modeling and analysis of stochastic systems, third edition.* CRC Press, 2016.

[152] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," in *Proc. of ICML*, 2015.

[153] D. J. C. MacKay, "A Practical Bayesian Framework for Backpropagation Networks," *Neural Computation*, 1992.

[154] J. Hernandez-Lobato, Y. Li, M. Rowland, T. Bui, D. Hernandez-Lobato, and R. Turner, "Black-box alpha divergence minimization," in *Proc. of ICML*, 2016.

[155] S. Jošilo and G. Dán, "Selfish decentralized computation offloading for mobile cloud computing in dense wireless networks," *IEEE Transactions on Mobile Computing*, 2019.

[156] ——, "A game theoretic analysis of selfish mobile computation offloading," in *Proc. of IEEE INFOCOM*, 2017.

[157] J. Haj-Yahya, L. Orosa, J. S. Kim, J. G. Luna, A. G. Yağlıkçı, M. Alser, I. Puddu, and O. Mutlu, "Ichannels: Exploiting current management mechanisms to create covert channels in modern processors," in *ACM/IEEE ISCA*, 2021.

[158] M. Botacin, F. B. Moreira, P. O. Navaux, A. Grégio, and M. A. Alves, "Terminator: A secure coprocessor to accelerate real-time antiviruses using inspection breakpoints," *ACM TOPS*, 2022.

[159] J. Kwak, Y. Kim, J. Lee, and S. Chong, "Dream: Dynamic resource and task allocation for energy minimization in mobile cloud systems," *IEEE Journal on Selected Areas in Communications*, 2015.

[160] 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*, 2019.

[161] Why do we need 5G? [Online]. Available: https://www.etsi.org/technologies/5g

[162] Y. Li, A. Zhou, X. Ma, and S. Wang, "Profit-aware edge server placement," *IEEE Intenet of Things Journal*, 2022.

[163] M. K. Kasi, S. Abu Ghazalah, R. N. Akram, and D. Sauveron, "Secure mobile edge server placement using multi-agent reinforcement learning," *Electronics*, 2021.

[164] S. Wang, Y. Guo, N. Zhang, P. Yang, A. Zhou, and X. Shen, "Delay-aware microservice coordination in mobile edge computing: A reinforcement learning approach," *IEEE TMC*, 2019.

[165] A. Hill et al., "Stable baselines," 2018. [Online]. Available: https://github.com/hill-a/stable-baselines

[166] S. Depeweg, J. M. Hernández-Lobato, F. Doshi-Velez, and S. Udluft, "Learning and policy search in stochastic dynamical systems with Bayesian Neural Networks," in *Proc. of ICLR*, 2017.

[167] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," in *Proc. of ICML*, 2010.

[168] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. of ICML*, 2018.

[169] J. N. Tsitsiklis, "Asynchronous stochastic approximation and Q-learning," *Machine learning*, 1994.

[170] M. L. Littman and C. Szepesvári, "A generalized reinforcement-learning model: Convergence and applications," in *ICML*, 1996.

[171] T. S. Jaakkola *et al.*, "On the convergence of stochastic iterative dynamic programming algorithms," *Neural Computing*, 1994.

[172] M. G. Bellemare, G. Ostrovski, A. Guez, P. S. Thomas, and R. Munos, "Increasing the action gap: New operators for reinforcement learning," in *AAAI*, 2016.

[173] D. S. Hochbaum, "Lower and Upper Bounds for the Allocation Problem and Other Nonlinear Optimization Problems," *Mathematics of Operation Research*, 1994.

DOCTORAL SCHOOL

Titre : L'Intelligence artificielle pour l'allocation des ressources dans le Multi-tenant Edge Computing

Mots clés : Edge computing, Optimisation orientée données, Réseaux nouvelle génération, Intelligence artificielle, Allocation des ressources, Multi-tenancy

Résumé : Dans cette thèse, nous considérons le Edge Computing (EC) comme un environnement multi-tenant où les Opérateurs Réseau (NOs) possèdent des ressources en périphérie déployées dans les stations de base, les bureaux centraux et/ou les boîtiers intelligents, les virtualisent, et permettent aux Fournisseurs de Services tiers (SPs) - ou tenants - de distribuer une partie de leurs applications en périphérie afin de répondre aux demandes des utilisateurs. Les SPs aux besoins hétérogènes coexistent en périphérie, allant des Communications Ultra-Fiables à Latence Ultra-Basse (URLLC) pour le contrôle des véhicules ou des robots, à la Communication de Type Machine Massive (mMTC) pour l'Internet des Objets (IoT) nécessitant un grand nombre de dispositifs connectés, en passant par les services multimédias tels que la diffusion vidéo et la Réalité Augmentée/Virtuelle (AR/VR), dont la qualité d'expérience dépend fortement des ressources disponibles. Les SPs orchestrent indépendamment leur ensemble de microservices, exécutés dans des conteneurs, qui peuvent être facilement répliqués, migrés ou arrêtés. Chaque SP peut s'adapter aux ressources allouées par le NO, en décidant s'il doit exécuter des microservices sur les appareils, les nœuds en périphérie ou dans le cloud. L'objectif de cette thèse est de promouvoir l'émergence de déploiements réels du "véritable" EC dans de vrais réseaux, en montrant l'utilité que les NOs peuvent tirer de l'EC. Nous croyons que cela peut contribuer à encourager l'engagement concret et les investissements des NOs dans l'EC. À cette fin, nous proposons de concevoir de nouvelles stratégies basées sur les données qui allouent efficacement les ressources entre les SPs hétérogènes, en périphérie, appartenant au NO, afin d'optimiser ses objectifs pertinents, tels que la réduction des coûts, la maximisation des revenus et l'amélioration de la Qualité de Service (QoS) perçue par les utilisateurs finaux, en termes de latence, de fiabilité et de débit, tout en répondant aux exigences des SPs. Cette thèse présente une perspective sur la manière dont les NOs, les seuls propriétaires de ressources en périphérie, peuvent extraire de la valeur grâce à la mise en œuvre de l'EC dans un environnement multi-tenant. En promouvant cette vision de l'EC et en la soutenant par des résultats quantitatifs et une analyse approfondie, cette thèse fournit principalement aux NOs des conclusions susceptibles d'influencer les stratégies de décision concernant le déploiement futur de l'EC. Cela pourrait favoriser l'émergence de nouvelles applications à faible latence et à forte intensité de données, telles que la réalité augmentée haute résolution, qui ne sont pas envisageables dans le cadre actuel du Cloud Computing (CC). Une autre contribution de la thèse est qu'elle propose des solutions basées sur des méthodes novatrices exploitant la puissance de l'optimisation basée sur les données. En effet, nous adaptons des techniques de pointe issues de l'Apprentissage par Renforcement (RL) et de la prise de décision séquentielle au problème pratique de l'allocation des ressources en EC. Ce faisant, nous parvenons à réduire le temps d'apprentissage des stratégies adoptées à des échelles compatibles avec la dynamique de l'EC, grâce à la conception soignée de modèles d'estimation intégrés au processus d'apprentissage. Nos stratégies sont conçues de manière à ne pas violer les garanties de confidentialité essentielles pour que les SPs acceptent d'exécuter leurs calculs en périphérie, grâce à l'environnement multi-tenant.

Title : Artificial Intelligence for Resource Allocation in Multi-Tenant Edge Computing

Keywords : Edge computing, Data-driven optimization, Next generation networks, Artificial intelligence, Resource allocation, Multi-tenancy

We consider in this thesis Edge Computing (EC) as a multi-tenant environment where Network Operators (NOs) own edge resources deployed in base stations, central offices and/or smart boxes, virtualize them and let third party Service Providers (SPs) - or tenants - distribute part of their applications in the edge in order to serve the requests sent by the users. SPs with heterogeneous requirements coexist in the edge, ranging from Ultra-Reliable Low Latency Communications (URLLC) for controlling cars or robots, to massive Machine Type Communication (mMTC) for Internet of Things (IoT) requiring a massive number of connected devices, to media services, such as video streaming and Augmented/Virtual Reality (AR/VR), whose quality of experience is strongly dependant on the available resources. SPs independently orchestrate their set of microservices, running on containers, which can be easily replicated, migrated or stopped. Each SP can adapt to the resources allocated by the NO, deciding whether to run microservices in the devices, in the edge nodes or in the cloud. We aim in this thesis to advance the emergence of real deployments of the “true” EC in real networks, by showing the utility that NOs can collect thanks to EC. We believe that this can contribute to encourage concrete engagement and investments engagement of NOs in EC. For this, we point to design novel data-driven strategies that efficiently allocate resources between heterogeneous SPs, at the edge owned by the NO, in order to optimize its relevant objectives, e.g., cost reduction, revenue maximization and better Quality of Service (QoS) perceived by end users, in terms of latency, reliability and throughput, while satisfying the SPs requirements. This thesis presents a perspective on how NOs, the sole owners of resources at the far edge (e.g., at base stations), can extract value through the implementation of EC within a multi-tenant environment. By promoting this vision of EC and by supporting it via quantitative results and analysis, this thesis provides, mainly to NOs, findings that can influence decision strategies about the future deployment of EC. This might foster the emergence of novel low-latency and data-intensive applications, such as high resolution augmented reality, which are not feasible in the current Cloud Computing (CC) setting. Another contribution of the thesis it that it provides solutions based on novel methods that harness the power of data-driven optimization.We indeed adapt cutting-edge techniques from Reinforcement Learning (RL) and sequential decision making to the practical problem of resource allocation in EC. In doing so, we succeed in reducing the learning time of the adopted strategies up to scales that are compatible with the EC dynamics, via careful design of estimation models embedded in the learning process. Our strategies are conceived in order not to violate the confidentiality guarantees that are essential for SPs to accept running their computation at the EC, thanks to the multi-tenant setting.

Institut Polytechnique de Paris
91120 Palaiseau, France