Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

PhD Thesis

Optimization Models and Learning Algorithms for Slice Reservation in Virtualized Communication Networks

Author:
Jean-Baptiste Monteil

Supervisors:
Prof. Ivana Dusparic
Prof. George Iosifidis
Prof. Luiz DaSilva

2023

Declaration

I declare that this thesis has not been submitted as an exercise for a degree at this or any other university and is entirely my own work.

I agree to deposit this thesis in the University's open access institutional repository or allow the Library to do so on my behalf, subject to Irish Copyright Legislation and Trinity College Library conditions of use and acknowledgement.

Signed:

Jean-Baptiste Monteil, 9th June 2023

Abstract

The surge of mobile users with increasing needs of data and services has brought a tremendous momentum to telecommunications. This revolution paves the way to the development of next generation mobile networks, able to accommodate many businesses with specific needs on the same network infrastructure. The network slicing paradigm, enabled by technologies such as Network Function Virtualization (NFV) and software-defined networking (SDN), allows the Network Operator (NO) to host multiple verticals or Service Providers (SPs) on the same substrate network. The network slices are dedicated logical networks that are tailored to the specific needs of each service. They are essentially composed of virtualized resources embedded on the Radio-access networks (RAN) and core elements of the network.

Virtualization has been identified as a key pillar of 6G. In virtualized systems, the SPs ask for bundles of resources that the NO allocate accordingly to its admission control policy. The interaction between the SP and the NO has been modeled through dynamic markets, where the SP can make in-advance reservation and bid for resources on the spot market. These markets have already been applied in the realm of cloud computing and have been recently transferred in the network slicing domain.

The problem has been studied extensively from the perspective of the NO with solutions about how to find an optimal admission policy, how to solve the allocation problem which is typically NP-hard (e.g. virtualized network function (VNF) embedding), how to charge for the slices with the aim to maximize the network utilization, or the NO revenue. However, there has been little focus on the equally important problem of how the SPs should request the slices. Indeed, the efficiency of these markets depend equally on the SPs being able to optimize their reservation decisions.

The thesis focuses on the viewpoint of an individual SP which aims to derive its reservation policy. The developed solutions must provide decisions robust to arbitrary changes of the service demand or the prices of the requested resources. Thus, the thesis proposes decision-making algorithms relying on modern learning tools, namely machine learning techniques and Online Convex Optimization (OCO).

First, the research focuses on a RAN reservation model, where the SP can contract bandwidth capacity from the Mobile Network Operator (MNO). The proposed solutions provide a near optimal reservation but the model does not account yet for the prices of the contracted resources. Then, the SP can reserve different kinds of network resources, thus enabling the creation of an end-to-end (E2E) network slice. The derived optimization problem also accounts for the varying demand of the SP and the evolving prices of the resources. Finally, the model is enhanced to allow the incorporation of key predictions on the demand and prices which improve the quality of the reservation decisions.

The first solutions are a Deep Neural Network (DNN) and a Long Short-Term Memory (LSTM), which use as input the feedback from the MNO from the q previous slots and output the reservations for the following h slots. The solutions outperform the Auto-Regressive Integrated Moving Average (ARIMA) baseline with idealized knowledge of the environment, by drastically reducing the over-reservation of RAN resources. Then, the new objective is to maximize the long-term utility of the slice from the reservations, while not exceeding the allocated budget for the purchase of resources. The problem has convex objective and constraint functions and thus falls under the scope of OCO. Finally, based on the feedback from the NO, the SP can derive accurate predictions of the future. The prediction module and the reservation solution are both online learning solutions which combined provide performance guarantees, while they outperform the Follow-the-Regularized-Leader (FTRL) baseline.

The proposed solutions provide a near optimal online reservation policy to the SP. The DNN and LSTM solutions present almost zero over-reservation and under-reservation under all traffic conditions (from congested to low traffic). The OCO-based solution converges to the performance of the optimal online reservation policy under stationary and non-stationary settings. The prediction-assisted reservation solution provides better guarantees and outperforms the well-known FTRL algorithm against real-world data.

Acknowledgements

More than a job or the continuation of the student life, this PhD has been a journey. Not a journey in the metaphoric sense, but a colourful journey made of places at the edge of the world and of a crowd of people of all background. I would like to take advantage of this little chapter to give credit to all the people I have met through this journey and that made the end goal achievable.

Before all, I would like to thank my parents. To all the people who gave me shelter during my moves: my sister Clemence, Nanou and Laurent a few times, Sylvie and Serge, Florence, Thomas and his crew, Julien. I would like also to thank my brother Julien and his wife Kristen, my sister Marie and her husband Florian for choosing me as the godfather of Georges and Jackson, respectively. Also let's not forget James and Mathilde, the two other grandchildren of my proud parents. I truly dedicate this thesis to my family, which I am very proud of.

First and foremost, I would like to thank my supervisors Luiz, George and Ivana, who, at various stages of my PhD have brought to me steady help and support: I could not have achieved the work without their immense contribution. To Jernej and Pieter, who contributed to my first paper. To Anthony Quinn, for his 5C3 course and getting me into teaching. I would like to thank Boris for being a great landlord.

Thanks to my colleagues of the lab Sandip, Sangita, Alberto, Mohit, Jose, Hriday, Tunji and Paul, you are great people and I wish you all the best. Thank you Andre, you were a good companion in Croatia, Galway and the Dublin nights. Thank you people of the choir, Lisbon and Saint Patrick's cathedral were amazing. Thank you to my colleagues of calisthenics from Trois-Bassins: Mike, Tony, Quentin, Shaim, Maxime, Kevin, Guillaume, Saul, Fred, Edouard, Flavien, and the others. Thank you to Nadejda, Johannes, Anais, Jack, Arman, Pascal. Thank you to Arijeet, Eashan and Danilo, my gym partners in Ireland.

Contents

List of Figures

List of Tables

List of Acronyms

IoT	Internet of Things
SP	Service Provider
NO	Network Operator
RAN	Radio-access networks
E2E	end-to-end
BS	Base Station
OCO	Online Convex Optimization
MNO	Mobile Network Operator
MVNO	Virtual Mobile Network Operator
OTT	over-the-top
InP	Infrastructure Provider
NFV	Network Function Virtualization
eMBB	Enhanced Mobile Broadband
URLLC	Ultra-Reliable Low-Latency Communications
mMTC	Massive Machine Type Communications
ITU	International Telecommunication Union
VNF	virtualized network function
SDN	software-defined networking
VM	Virtual Machine
5G-PPP	5th Generation Public Private Partnership

MANO	management and orchestration
ETSI	European Telecommunications Institute
SCEF	Service Capability Exposure Function
SLA	Service Level Agreement
QoS	Quality of Service
MEC	Multi-Edge Cloud
ML	Machine Learning
MSE	Mean Square Error
DNN	Deep Neural Network
LSTM	Long Short-Term Memory
OLR	Online Learning for Reservations
OLR-MTS	Online Learning for Mixed Time Scale Reservations
OLR-SO	Online Learning Reservations for Slice Orchestration
OLR-MTS-SO	Online Learning for MTS with Slice Orchestration
FTRL	Follow-the-Regularized-Leader
ARIMA	Auto-Regressive Integrated Moving Average
OOLR	Optimistic Online Learning for Reservation

Notations

We use bold typeface for vectors, $\boldsymbol{a}$, and vector transpose is denoted $\boldsymbol{a}^\top$. We also use bold typeface for a function with multiple images, e.g., $\boldsymbol{g} : \mathbb{R}^n \to \mathbb{R}^m$. A sequence of vectors is denoted with braces, e.g., $\{\boldsymbol{a}_t\}$, and we use sub/superscripts to define a sequence of certain length, e.g., $\{\boldsymbol{a}_t\}_{t=1}^T$ is the sequence $\boldsymbol{a}_1, \boldsymbol{a}_2, \dots, \boldsymbol{a}_T$. Sets are denoted with calligraphic capital letters, e.g., $\mathcal{M}$. Constants are denoted with normal capital letters, e.g., G. We use the symbol $\preceq$ to represent the inequality between vectors, e.g., $\boldsymbol{x} \preceq \boldsymbol{y}$ means $\forall x \in \boldsymbol{x}, y \in \boldsymbol{y}, x \leq y$. The projection onto the non-negative orthant is denoted $[\cdot]_+$, and $\| \cdot \|$ is the ℓ_2 norm. The key notation symbols are summarized in a table for each chapter.

1 Introduction

In this introductory chapter, we describe the motivation and the framework of our research and then we state our contribution. We begin the discussion by presenting the evolution of today's networks towards 6G. We then present how the network slicing framework with the support of virtualization technologies can realize the accommodation of new services on the same infrastructure. We emphasize the increasing role these new services can play in the creation of the network slices. We then highlight the research questions that must be addressed to fully unleash the potential of these new services in network slicing. We finally state our contribution and list the different publications associated with this research.

1.1 Motivation

Telecommunications have gained tremendous momentum with the proliferation of mobile end users and the development of a plethora of businesses in need of telecommunication support. The traffic demand has skyrocketed over the past few years primarily due to the widespread of smartphone users, which consume mainly video streaming, internet access services and applications. The number of smartphone subscriptions has evolved from 2.6 billions in 2016 to 6.3 billions at the end of 2021, with the average traffic consumption per smartphone making a giant leap from 800 megabytes/month to 11.4 gigabytes/month over the same period [2]. This is due to the growing demand for services that necessitate higher data transmission and processing. Especially during the lockdown, the digital world has grown at the expense of the physical world, shaping a new business ecosystem and changing the mentalities with the rise of remote working and the success of videoconferencing applications such as Skype or Zoom. The 5G networks have addressed the ever-increasing demand by expanding the bandwidth to millimeter waves enabled by the installation of small cells. The massive MIMO antennas with the support of the beamforming technology also play a major role in the increase of wireless communication transmitted loads. While these technologies have augmented the potential of accommodation of the traffic load, they come with capital expenditure and operational expenditure (CAPEX/OPEX) costs. Additionally, the today's networks must deal with the development of new use cases such as the Internet of Things (IoT) with the increase of connected devices, the autonomous vehicles, e-health and smart factories which require more reliability, less latency or more capacity. Therefore, arises the need of new concepts that limit the CAPEX/OPEX costs and accommodate diverse use cases with specific requirements on the same network architecture.

The survey [3] has detailed the evolution of the *network sharing* paradigm towards the direction of allocating multiple tenants (Mobile Network Operators (MNOs), Virtual Mobile Network Operators (MVNOs), Service Providers (SPs)) with diverse needs on the same network infrastructure. In the early 3GPP release 99, network sharing was introduced with the two pillars of passive network sharing and network roaming. Passive Radio-access networks (RAN) sharing would concern the sharing of site locations or radio equipment, with each MNO having access to an individual carrier layer. Network roaming would allow mobile subscribers of a MNO to use the network of other MNOs. For example, in France, the operator Free relies on the France Telecom infrastructure, former name of Orange and owner of the French backbone network. One important motivation for the first collaboration between MNOs has been the coverage of large geographical areas - countries or even larger territories like EU. Then, the active RAN sharing appeared with the possibility for MNOs to pool spectrum resources which implies sharing resources dynamically. In [4], two types of active RAN sharing have been described, namely the multi-core operator network where the MNOs share the Base Station (BS) and have separate core networks and the gateway core network in which the MNOs share one additional core network element which reduces even more the costs. Smart resource sharing solutions become crucial in the context of network sharing, so as to avoid privacy infringement of the subscribers or the MNOs and potential conflict in the usage of resources that would negatively impact the service experience of the subscribers. Although network sharing has immensely contributed to the reduction of

CAPEX/OPEX costs, there is still a need for new concepts to design a thriving network that would admit new players requiring network resources.

Enabling the admission of new players like MVNOs, over-the-top (OTT) SPs or vertical industries poses real network resource management challenges. The issue on how to allocate resources to these different players with very specific needs naturally arises. MVNOs provide wireless communications services but do not own the wireless network infrastructure which belongs to the Infrastructure Provider (InP). To illustrate the difference between the MVNO and the InP in Ireland, Virgin Mobile is a MNO with no network infrastructure hence an MVNO and Eir Mobile is a MNO owner of the network infrastructure hence an InP. OTT SPs include all the services which require connection to the Internet for their service delivery, where OTT indicates these services are built on top of the network infrastructure at the application layer. In this thesis, we refer to them simply as SPs. The most popular services are the platforms of video streaming like Netflix or Amazon Prime Video, and the numerous smartphone applications like Uber or Instagram. Vertical industries come from different sectors and designate the businesses adopting a digital transformation that necessitates the use of network resources for one peculiar service. These verticals come from the transport and logistics sector with the use case of the automotive industry, the public sector with the government and non-government agencies, the energy and utility sector with the smart grids, the finance and insurance sector, the healthcare system, the education, the manufacturing and construction with the smart factories. All of these services have specific requirements; e.g. the IoT must deal with the massive arrival of small data packets coming from connected devices like low-power sensors or CCTV cameras which require high transmission, storage, and processing capacity. Hence, IoT services need solutions to prioritize meaningful information and optimize the capacity usage [5]. Another example is the viability of autonomous vehicles where the communications between vehicles must be ultra-reliable and present low latency. Privacy of patient information is key in the context of e-health and safety is critical in the case of government agencies. Therefore, 5G networks need a new concept to be able to welcome such a variety of services on the same network infrastructure, which is *network slicing*.

1.2 The network slicing framework

1.2.1 Definition

We begin with the definition given in [6]: *"Network slicing is centered on the concept of deploying multiple dedicated logical mobile networks with varying levels of mutual isolation on top of the same infrastructure. A network slice is a collection of mobile network functions (or groups of functions) and a specific set of radio access technologies (RATs) (or specific RAT configurations) necessary to operate an end-to-end (self-contained) logical mobile network."* In the network slicing framework, the slice manager, often located at the InP [3], allocates portions of network capacity to the different SPs. These portions must include different types of network resources (e.g. RAN resources, core network resources) so as to provide the SP with an end-to-end network slice dedicated to its specific service. The network slice is auto-sufficient, i.e. does not need other allocated resources than those already present and thus corresponds to *"an end-to-end self-contained logical mobile network"* tailored to the service. The International Telecommunication Union (ITU) [7] has designed a classification of the existing use cases and services which yields the three main families, namely the Enhanced Mobile Broadband (eMBB), the Ultra-Reliable Low-Latency Communications (URLLC) and the Massive Machine Type Communications (mMTC). The eMBB family addresses the connectivity of high-speed mobile users or dense urban areas. The URLLC encompass autonomous vehicles or critical mission services. The mMTC concern the IoT with the massive amount of connected devices, e.g. the use case of mobile video surveillance through CCTV cameras. The use cases have different requirements; e.g. the autonomous vehicles in URLLC services will require reliability, low latency and high mobility, the mobile video surveillance in mMTC will require power efficiency of the cameras and connection density for good coverage, the connectivity of urban areas in eMBB will need spectrum efficiency, high user data rate and traffic density. Therefore, there is a need to tailor the slices to the different use cases which have specific requirements even within the same family. The difficulty arises when the service-tailored slices must co-exist on the same network infrastructure in an isolated way – the isolation of the slices is important as to avoid the collapse of the whole system when one slice undergoes a cyber-attack or a service failure.

1.2.2 The enabling technologies of network slicing

The management and orchestration (MANO) of the network resources to achieve the accommodation of isolated slices on the same network is a real challenge. Moreover, the demand for each service is constantly evolving hence, to avoid under- or over-provisioning, the slice manager must be able to reconfigure the slices dynamically which suppose an immediate and flexible access to the resources. The abstraction of the network resources into network functions with specific goal, which are both easy to instantiate and reusable, is a key step towards enabling network slicing. This necessitates the development of virtualization techniques to map the built-up network functions to the network resources at the infrastructure layer. The 5^{th} Generation Public Private Partnership (5G-PPP) [8] envisions an architecture with three main layers which are the infrastructure, the network function, and the service layer. The virtualization of the resources into network functions and the incoming requests from the SPs at the service layer are the major challenges at stake.

The software-defined networking (SDN) technology has been envisioned to support the huge, heterogeneous and complex 5G networks [9]. Specifically, SDN separates the data plane (data processing) from the control plane (data routing). The SDN controller located at the control plane configures dynamically via protocols the data-forwarding elements (switches and routers) located at the data plane, based on the network policy defined at the management plane [10]. This architecture enables the dynamic configuration of network elements to accommodate diverse network applications. The design of southbound and northbound interfaces between the control plane and the two others allow the SDN controller to perform network management.

While SDN separates forward from control, Network Function Virtualization (NFV) separates function from hardware [11]. The NFV paradigm proposes to abstract the network functions into software which can be run on dedicated hardware. The virtualized network functions (VNFs) execute specific service tasks using the underlying infrastructure, e.g. Virtual Machines (VMs). The possibility to dynamically perform VNF placement and migration enables flexible resource allocation and is a huge step towards the realization of network slicing. Still, the VNF placement is an NP-hard problem and necessitates the design of advanced heuristic solutions, especially in large 5G networks. Service chaining allows to chain the VNFs with the aim to perform a complete network service. By separating control and data planes, service chaining can also steer the traffic trough the placed functions of the chain.

Cloud computing with the pooling of VM resources, SDN and NFV enable the virtualization of core network elements, ensuring the creation of isolated and dedicated VNFs which are flexibly allocated on the network. On the other hand, the RAN virtualization is more challenging. Two directions exist: either dedicate a spectrum chunk to each slice, or dynamically share the spectrum between the slices [12]. The first approach enforces the slice isolation but waste already scarce radio resources due to under-utilization. The second approach has more potential in terms of efficient resource utilization but renders difficult the slice isolation.

1.2.3 The requests from the Service Providers

The European Telecommunications Institute (ETSI)-NFV [13] has envisioned a centralized MANO entity for the management of the VNFs and the underlying network resources. Then, [14] proposes a distributed approach which improves the scalability and reliability of the management when the number of tenants/slices increases and provides more autonomy to the SPs in the management of their slice. Specifically, the MANO-NFV is in charge of the life cycle control of the different VNFs, i.e. the creation, reconfiguration and termination of the VNF instances. This MANO framework enables dynamic service provision for the different slices.

There exists an important sequence of decisions in the slice life-cycle as described in Fig. 1.1. The first phase concerns the design of the slice, opening with the request the SP can issue to the InP. Different forms of SP requests have been envisioned. First, the 3GPP release 13 [15] has introduced the Service Capability Exposure Function (SCEF) which handles the interaction between the services and the Network Operator (NO). The important phases of this interaction are the negotiation of the Service Level Agreements (SLAs) in which the SP specifies its Quality of Service (QoS) requirements, the authentication and secure access to the network and the charging based on delivered service. Although straightforward from the SP viewpoint, this high-level form of request is very conceptual which renders difficult the mapping of the desired capabilities to the actual network resources. Another type of request is the thorough specification of the set of desired network functions for the service. This approach is unpractical for some SPs and constrains the MANO task of the NO, which needs some flexibility to design its resource allocation policy. Finally, the SP request can define the desired portion of network capacity for multiple kinds of resources, i.e. radio-access resources, link capacity, storage and processing, etc. The SP specifies its overall needs and let the NO flexibly allocate network functions and components to match the requested capacity for each resource.

The NO is in charge of the creation of the slice based on the request and can operate admission control to accept or reject the request, depending on the expected profitability of the corresponding slice. Once the admission control step is over and all the accepted slices have been created, the MANO task of the network resources begins first with the activation of the accepted slices, then with the management of the traffic load within each slice. The third phase consists of the operation of the slice: while monitoring its performance the NO can deem relevant to re-configure each slice by withdrawing allocated resources being unused or allocating even more resources due to the high demand profile of the slice. The SP can also play a decision role in the re-configuration of the slice, by making intermediate requests for additional resources if need be. The final phase is the deactivation of the slice when the corresponding network service is no longer active.

An arising issue is to define the extent of the SP's contribution to the decisions of the slice life-cycle. The parts of the figure highlighted in red show the opportunities for the SP to make relevant requests with regard to the slice creation or the slice reconfiguration. With the anticipated growth of the number of accommodated slices in the 5G networks and their diversity in terms of requirements, the master NO has the opportunity to host more slices by giving greater autonomy to the SPs in the slice request and the slice management processes.

1.2.4 Thesis aims and objectives

The resource allocation problem from the perspective of the NO has been extensively studied. Researchers have found that the VNF placement is NP-hard [16]. Publications contain

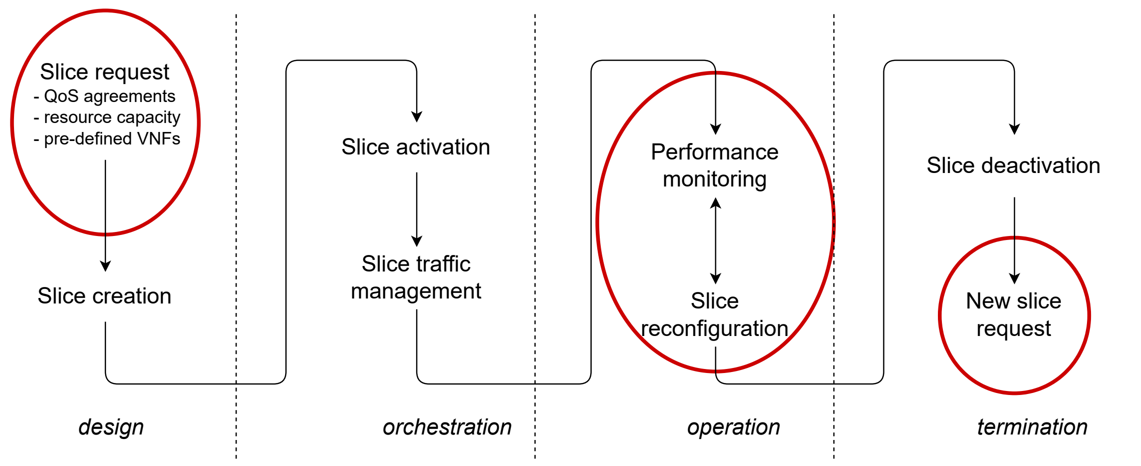


Figure 1.1: The slice life-cycle

sophisticated solutions on how to design the admission control policy of the NO [17–19], how to manage the resource allocation [20–22], how to charge for the leased resources [23]. However, fewer works envision the resource reservation problem faced by the SP. In this context, the NO makes network resources available in a marketplace and the SPs can request a certain capacity amount for all their needed resources [24]. The diversity of network resources – comprising processing, storage and bandwidth – compels the SP to design a smart resource reservation policy which can maximize the utility of its service and minimize the price of such reservation.

The nature of the SP request becomes essential to determine the SP reservation policy. Some works have dealt with a high-level type of request [25–28], where not much than the slice duration, the desired data rate or overall slice capacity are specified. This would give to the SP a very minor role in the MANO task of network resources and would not alleviate the task of the NO. A very detailed reservation request of the SP through an exhaustive template or about the complete set of needed VNFs has been considered in [24, 29–31]. However, the required level of detail can make the request unpractical to the SP. There remains an open field where the SP specifies its resource-wise overall need of network capacity. We anticipate that this scenario is the best option as it gives to the NO sufficient insight about the needs of the SP and sufficient flexibility in the allocation process. This yields the following research question:

RQ1: How can the SP request capacity for the different kinds of resource necessary to the *operation of its slice/its service?*

The RAN functions placement and load balancing constitute an NP-hard problem [32]. Thus, the design of smart requests from the SPs in need of RAN resources becomes imperative. The options of RAN virtualization is either to opt for a share-based approach of the spectrum between the different slices [33, 34], or to opt for a reservation-based approach with dedicated RAN to each slice [24, 35]. In the shared-based approach, the RAN virtualization enables the transfer of sub-carriers from one base station to another, depending on the profile of the traffic demand [33]. It provides flexible management and dynamic sharing but does not ensure isolation of the spectrum resources which threatens the reliability of the slices. The reservation-based approach provides the SP with the same static RAN functions which yields to over-/under-reservation depending on the SP demand [12]. Only temporary lease of those functions –through a dynamic marketplace [24]– can counterbalance this non-optimal usage of already scarce RAN resources. Thus, the following research question arises:

RQ2: How can the SPs dynamically request dedicated spectrum resources?

Those requests must be made in advance to let the NO prepare the dedicated radio hardware necessary to the accommodation of desired spectrum/RAN functions.

In cloud computing, the main objective of the cloud provider is to achieve high utilization of its cloud VM instances to increase their profitability, as proven in [36]. Similarly, in 5G networks, the NO aims at accommodating more services/tenants, i.e. slices, on its network, which will lead to a thriving network slicing business [17, 27]. The resource allocation policy becomes essential to apply a parsimonious and dynamic usage of the resources to host the maximum number of slices possible. On the other hand, the SP aims at maximizing its service performance while paying a decent fare for the contracted resources to ensure the long-term profitability of the delivered service. Both entities pursue compatible goals which are in accordance with a meaningful and valuable usage of the network resources. Some works have designed the buying and selling of network resources through auction games [37]. Another line of search has been to investigate pricing mechanisms [23]. The following research question arises:

RQ3: How to manage the economic interaction between the NO and the SP in the leasing process of network resources?

We expect the *network slicing market* to operate the leasing of the network resources, with the NO acting as the broker and the SP acting as the bidder. The NO can leverage the pricing of resources to promote scarcity or profusion of the resources, with the goal to always stay in-between so to avoid resource provisioning issues when the resources are too scarce or profit loss when the resources are barely used.

1.3 Thesis contribution

The main contribution of the thesis is the design of a family of algorithms that enable the SP to design its online reservation policy of network resources. The solutions can be found online at our public GitHub repository [1]. The algorithms are solutions to a model which progressively grows in complexity, accounting for more variables and parameters of the network slicing market. All the algorithms are evaluated against strong baselines and under different network conditions. In detail, our contributions are:

1. C1: this contribution addresses RQ2. The SP can reserve in-advance bandwidth capacity at the BS. The contribution is the design of the Deep Neural Network (DNN) and Long Short-Term Memory (LSTM) solutions. Specifically, the solutions output the reservations for the next h steps, taking account of the MNO's feedback about different traffic traces. The solutions outperform auto-regressive methods, represented by the sophisticated Auto-Regressive Integrated Moving Average (ARIMA) baseline, under all kinds of traffic conditions. The solutions drastically reduce over-reservation, and clearly outperform the baseline for multi-steps ahead reservations. The MNO and the SP traffic loads are generated from real world data, encompassing the recorded traffic volumes at various BSs in the city of Shanghai in August 2014.
2. C2: this contribution addresses RQ1 and RQ3. The SP can reserve in-advance and on the spot multiple kinds of network resources. The contribution is the design of four solutions to derive the online reservation policy of the SP. At each period,

[1]https://github.com/jeanba19

the SP decides its reservation plan using a primal-dual solution to solve a constrained convex problem. Then, the Online Learning Reservations for Slice Orchestration (OLR-SO) algorithm complements the basic Online Learning for Reservations (OLR) solution, allowing the SP to detail its request for all kind of resources. Both solutions present strong performance guarantees. Finally, the Online Learning for Mixed Time Scale Reservations (OLR-MTS) and Online Learning for MTS with Slice Orchestration (OLR-MTS-SO) algorithms allow the SP to contract additional on the spot resources within the period. The four solutions are tested under stationary and non-stationary settings, verifying the theoretical bounds. The parameter sensitivity analysis demonstrates the robustness of the solutions and allows the SP to find its optimal budget choice for the purchase of resources.

3. C3: this contribution addresses RQ1 and RQ3. The SP can reserve in-advance and on the spot multiple kinds of network resources. The contribution is the design of the Optimistic Online Learning for Reservation (OOLR) solution, which allows the SP to incorporate key predictions in its reservation decision model. Then, the association of the decision algorithm with an online learning for time series prediction algorithm improves the performance over the Follow-the-Regularized-Leader (FTRL) baseline.

1.4 Thesis outline

The remainder of this thesis is organized as follows:

Chapter 2 – A Review of Related Work. This chapter reviews the existing work in network slicing relevant to the SP reservation problem, which encompasses reservation-based allocation problems and slicing market models. It provides insight from the recent literature on the Online Convex Optimization (OCO) framework we use to derive solutions for the SP reservation problem.

Chapter 3 – The reservation of radio resources: a cost-reduction strategy for the SP. This chapter presents the first contribution C1, which answers RQ2. It showcases the deep learning solutions proposed to solve the problem of spectrum reservation in the context of virtualized RAN. The solutions allow the SP to request spectrum resources far in advance which enables the NO to anticipate the resource allocation policy so to satisfy the request of the SP and the needs of the other slices/SPs. It focuses on the performance of the SP decisions in terms of the resulting over- and under-reservation. It explores the impact of different traffic conditions on our solutions' performance. For comparison, it considers an ARIMA baseline model.

Chapter 4 – No-regret slice reservation solutions in constrained OCO. This chapter presents the second contribution C2, which addresses RQ1 and RQ3. It introduces the *network slicing market* which models the economic interaction between the SP and the NO in the leasing of network resources. Then, it formulates the SP reservation problem as convex, where the objective is to maximize the service performance while respecting the budget constraint on the cost of reservation. It develops an online learning solution to decide the online reservation policy of the SP. The derived solution provides guarantees in terms of regret and constraint violation, which are used in OCO to assess the convergence to the optimal solution. In the first extension, the reservation encompasses multiple kinds of resources, which allow the SP to make a complete slice request. Again, guarantees of regret and constraint violation are given in this slice orchestration case. In the second extension, the SP can make requests during the reconfiguration phase of the slice. The performance analysis shows improved results in the case the SP participates to the reconfiguration process. Finally, it conducts a battery of tests to assess the sensitivity of the solution to the different parameters of the model, e.g. the allocated budget.

Chapter 5 – Optimistic online reservation of virtualized resources. This chapter presents the contribution C3, which addresses RQ1 and RQ3. It introduces a new convex problem for the SP reservation problem, where the objective is to maximize a weighted sum of the service performance and the reservation expenses. This formulation lies under the frame of unconstrained OCO. The proposed solution that decides the online reservation policy of the SP is able to incorporate key predictions of the future to enhance the performance, while providing performance guarantees against arbitrarily bad predictions. The decision model is then associated with a prediction model and the combined solution shows improved performance over the FTRL baseline algorithm without prediction assistance.

Chapter 6 – Conclusions and future directions. This chapter details the conclusions of this thesis and discusses gaps in network slicing still open to new contributions. It then elaborates on the future directions of this research.

1.5 Dissemination

In this section, we draw up the steps of the work dissemination throughout the course of the PhD. We separate journal and conference publications and we associate each publication to the research question(s) it addresses.

Journal

- **RQ1 and RQ3/C2:** J-B Monteil, G. Iosifidis, and L. A. Da Silva, "Learning-based Reservation of Virtualized Network Resources,“ *IEEE Transactions on Network and Service Management*, January 2022.

Conferences

- **RQ1 and RQ3/C2:** J-B Monteil, G. Iosifidis, and L. A. Da Silva, "No-Regret Slice Reservation Algorithms,“ *ICC 2021-IEEE International Conference on Communications*, June 2021.
- **RQ2/C1:** J-B Monteil, J. Hribar, P. Barnard, Y. Li, and L. A. Da Silva, "Resource reservation within sliced 5G networks: A cost-reduction strategy for service providers,“ *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, June 2020.

Under Review

- **RQ1 and RQ3/C3:** J-B Monteil, I. Dusparic, and G. Iosifidis, "Reservation of Virtualized Resources with Optimistic Online Learning,“ accepted at ICC 2023.

2 Background and Related Work

In the previous chapter, we provided an overview of the slicing concepts, elaborating on the advantages of this new architecture but also on its resource orchestration requirements, while naming its key enabling technologies: SDN and NFV. We envisioned different ways to reserve a slice, such as selecting from a template or having the verticals to ask directly for the necessary computing, storage, and networking resources. The purpose of this chapter is to review the existing work which considers the reservation-based resource management problems in network slicing or focuses on the economic relationship between the NO owner of the network and the SP tenant. It also details the literature on OCO, which is the mathematical framework of our proposed solutions.

2.1 Dynamic resource provisioning, allocation and reconfiguration in network slicing

In this section, we review the works which consider dynamic resource provisioning/ reservation as the central problem to the creation of network slices. We also mention papers with resource allocation and slice reconfiguration solutions. With a thorough analysis of virtual network embedding and the arising challenges in network slicing, [38] insists on the need of online optimization solutions and network state monitoring and prediction methods to enable dynamic network slicing. Indeed, we present studies in the sequel which precisely address network slicing issues with the tools of optimization and machine learning.

2.1.1 Convex optimization problems

In [39], the authors consider a reservation model for cloud resources, where an entity requests resources in each slot without knowing the actual demand. There are multiple cloud resource instances, the cost per unit for each of them is fixed and given, while the demand is drawn from a general distribution. The goal is to minimize the total cost while preserving the violation error below a threshold. The authors formulate this as an OCO problem with budget and time-varying constraints (probability of not satisfying the request), and propose a primal-dual type of algorithm (extending OCO) to achieve asymptotically sublinear regret. In [40], the context is a NFV provider buying VM instances to a cloud operator and delivering service chains composed of VNFs to customers. The authors design a reservation model of long-term and short-term VM instances, then used for VNF deployment, and create a complete online solution - composed of VNF demand prediction and reservation decisions - with performance guarantees. Although the formulated loss is non-convex, the authors are able to cast the problem into OCO.

The authors in [20] studied the problem of allocating routing and processing resources to a set of slices with the goal to maximize a system-wide utility function (α-fair type of function so to achieve fair allocation across slices). They solved the formulated static convex optimization problem with a distributed approach, where the slice owner decides its bandwidth demand profile, the SDN controller decides the routing allocation profile, and the cloud controller decides the processing demand profile. By employing the alternating direction method of multipliers (ADMM), the authors allowed the three entities to coordinate their decisions for the sake of system operation. [41] takes a more holistic modeling approach and proposed a static convex optimization framework for embedding VNF chains (perceived as slices) in a shared network. The key contribution in this work is the very detailed modeling that allows for accounting for reliability, delay, and other requirements of each slice (each chain), which are all included in the formulated problem. Although both [41] and [20] are static convex problems, their polynomial complexity make them viable frameworks for resource allocation in dynamic settings.

In [42], the slice reconfiguration happens in a hybrid timescale fashion, with a main scheduler deciding when to apply a fast or thorough reconfiguration. The authors model the slice reconfiguration as an optimization problem, where the objective is to maximize the profit of the network slice customer while minimizing the reconfiguration cost. They approximate the latter cost function by using the L_1 norm, which casts the problem into a convex one. The main scheduler algorithm complexity lies in polynomial time, which renders practical their reconfiguration solution in dynamic settings.

2.1.2 General optimization problems

In [43], a slice reservation mechanism is presented, where a broker allocates sliced resources to a set of requesters. The resources include base stations; network links; and computing units. Each request is described by several parameters that include how much of each type of resource is necessary and for how long time. The decisions of the system are how much rate to allocate for each request, and whether to use path p for that request or not. These decisions appear in the objective function that maximizes the rewards from the successful reservations discounted by the expected resource violation penalties. Essentially, this model differs from the one in [39] in terms of: using multiple types of network resources from RAN to cloud; handling both the reward and the penalty of under-provisioning; and solving the problem exactly using an offline method. The work [44] evaluates the efficiency of sharing between multiple slices from a common overbooked slice across different network slicing reference scenarios. The NO guarantees a portion δ of the total slice traffic is being served, while endorses a penalty for violated portion π which happens at peak demand due to the overbooking strategy.

The authors in [17] built their admission control solution of the slices upon a forecasting module of the required resources by tenants for the future time window. The slice controller uses the traffic information to decide whether to admit new slice requests by solving a geometrical knapsack problem. After a thorough complexity analysis, the authors find the problem NP-hard and provide a heuristic solution with a satisfying performance ratio. The slice scheduler minimizes the consumed resources within the slice while complying with SLAs requirements. The forecasting module can adapt its predictions by changing the forecasting error probability, according to the feedback sent by the slice scheduler. In [45], the authors essentially extend their previous work [17] by adding a reinforcement learning solution to manage the feedback of the slice scheduler towards the forecasting module. In [26], the tenants issue one slice request with the amount of resource and the time duration specified. The Network Slice Broker decides whether to admit new requests, ensuring that SLAs are being met. The authors formulate a Multi-Armed Bandit problem, taking account of the limited resource budget and the lock-up periods of tenants. Then they design three heuristic online solutions with regret bounds, and finally give a proof-of-concept.

The focus of [46] is on wireless spectrum that, as the authors explain, has the particularity of being reusable, subject to interference constraints being satisfied, but also is more unpredictable due to channel variations over time. With advance reservation, the MVNO reserves sub-channels for a long time period, using estimations of the user traffic and the on-demand price over the period. Once the MVNO has observed the users' requests and the current on-demand price set by the MNO, it decides whether to reserve (or not) on-demand additional resources. There is also a third stage where the MVNO allocates the channels to its users. The authors essentially follow a reverse course and find analytical solutions for the slot and session problems, which then are plugged into the period problem. The latter becomes a stochastic convex problem, which they solve with stochastic gradient descent. In [47], the authors propose a hybrid solution for resource provisioning and resource allocation. The tenant reserves resources in the long timescale and allocates those to the slice in the short timescale. By operating a series of transformation, the authors are able to cast the two stage stochastic problem into a deterministic problem solvable with the branch-and-bound technique.

2.1.3 Machine learning solutions

The authors in [48] present a metric for the resource utilization per slice, and discuss how it can be predicted using time-series analysis. The metric relies heavily on the usage pattern of the physical resource blocks. The ultimate goal is that such metrics can be used by the NOs to slice efficiently their networks. The paper [49] formulates the problem of allocating dynamically slice resources (radio, computing, storage resources) in order to maximize the reward of the operator. There are different classes of slices (templates); each class c requires r_c^{re}, w_c^{re}, and δ_c^{re} units of radio, computing and storage resources; the operator receives reward r_c each time it admits a slice of class c; and the system has hard resource capacity constraints. There is uncertainty about the demands, and the slicing admission is modeled as a Reinforcement Learning problem. In [50], the authors employed deep learning to forecast capacity in a sliced network. The AI solution is a mix of 3-D Convolutional Neural Network and Multi Layer Perceptron architectures. It takes as input traffic snapshots at various BSs for one specific slice and outputs the desired capacity at the various data centers for the very same slice. The authors design a loss function able to leverage between SLA violations due to under-provisioning (under-estimation of the optimal capacity values at the data-centers) and unnecessary costs due to over-provisioning (over-estimation). Using a data-driven approach including c-RAN, Multi-Edge Cloud (MEC) and core networks, the authors outperform other state-of-the-art deep learning solutions [51], [52]. In [53], the focus is on zero-touch slicing mechanisms, i.e., fully-automated slicing solutions, from the perspective of the NO (as in [43]). The goal of any NO is to allocate resources in slices so as to avoid on the one hand unnecessary assignments (yielding unused resources) and on the other hand minimize the SLA violations (under-allocations in some slices). Yet, one needs also to account for the operational costs induced when a slice is initialized and/or reconfigured. In substance, this paper considers minimum cost slicing by accounting for: resource over-provisioning; non-serviced demands; resource instantiation/reconfiguration costs. The provided solution outperforms the previous solution of the same authors [50].

In [54], the InP can accept two kinds of slice, with strict latency constraints or non-strict latency constraints. The former has a higher revenue/penalty than the latter, which means it can lead to higher profit if the slice is setup properly, or to higher penalty, if the slice cannot be accommodated due to a lack of resources, at the radio, transport or cloud controller. The authors design a Reinforcement Learning solution, where an Artificial Neural Network model is trained to minimize the loss due to slice rejection and the loss caused by service degradation (when available resources are not sufficient to accommodate the slice). [55] formulates an optimization problem which aims to minimize the slice embedding and reconfiguration costs, subject to node computational capacity and link bandwidth capacity constraints. The cost functions are drawn from [42]. The authors prove the formulated problem is NP-hard and proposes a Reinforcement Learning approach to take the reconfiguration

Table 2.1: Papers classification

Problem	Optimization-based	Machine Learning-based
Resource provisioning	[39], [40], [46], [47]	[48], [50], [51], [52], [53]
Resource allocation	[20], [41], [43], [44], [47]	[49], [50], [53]
Slice admission	[17], [45], [26]	[49], [54]
Slice reconfiguration	[42]	[55]

actions.

2.1.4 Section summary

We classify the cited works in Table 2.1, according to the problems they address. The problems presented in this section are namely resource provisioning/reservation, resource allocation, slice admission and slice reconfiguration. Some works have addressed the aforementioned problems using machine learning tools. We compare them with the contribution C1 of our thesis in Table 2.2. Other works have formulated optimization problems and solved them with relevant methods. We list their details in the Table 2.3, as well as our contribution C2. Most related papers dealing with resource reservation and allocation in network slicing have focused on the NO perspective. Fewer works [39], [40], [46] and [47] have taken the SP viewpoint. Although we also use the OCO framework, our approach differs from [39] as we consider the prices of the resource instances to be unknown and varying. In [46], there are assumptions about the MVNO knowing the distribution of the on-demand prices and the users' needs, based on which it optimizes its strategy. Unlike this work, we make no assumptions on this, as in practice this information will not be available; what is more, probably these parameters will not even follow a stationary distribution and might be even selected in an adversarial fashion (e.g., by the NO which will try to increase its revenue). While [40] focuses on the reservation of VM instances in a cloud computing environment, our reservation scheme encompasses different kinds of network resources across the infrastructure from RAN to the cloud. [47] considers fixed reservation costs and the solution has non-polynomial time complexity. We make no assumptions on the prices of resources and our derived solutions present low time complexity.

Table 2.2: Machine Learning papers

References	viewpoint	learning method	baseline(s)	knowledge
[48]	NO	LSTM	ARIMA	RAN congestion level
[49]	NO	Q-learning	[27, 56, 57]	slice requests
[53]	NO	3D-CNN	[50–52]	slice traffic demand
[54]	NO	reinforcement	[43, 58]	slice requests
Thesis	SP	DNN, LSTM	ARIMA	SP traffic trace

Table 2.3: Optimization papers

References	viewpoint	formulation	solution	knowledge
[39]	SP	OCO	Virtual queue	fixed, known costs
[40]	SP	OCO	OGD	–
[20]	NO	static convex	ADMM	–
[41]	NO	static convex	expanded graph	known costs and requests
[42]	NO	convex after L_1 approximation	heuristic	fixed, known costs known link and data center capacity
[43]	NO	quadratic	Benders decomposition and heuristic	known base station capacity and Gaussian requests
[44]	NO	integer linear programming	configured version of [59]	–
[45]	NO	general problem	heuristic	–
[46]	SP	nested problems	analytical solution and SGD	known reservation price, uniform on-demand price
[47]	SP	nested problems	branch and bound	known, fixed costs
Thesis	SP	OCO	primal-dual	–

2.2 Market models in network slicing

To render effective the dynamic resource provisioning framework, the design of business models or auction mechanisms which characterize the interaction between the SP and the NO in a dynamic scenario becomes essential. In this section, we review the recent works which proposed to model the economic relationship between the different third-parties of network slicing through pricing mechanisms, auctions, business models, requests and admissions, or marketplaces.

2.2.1 Pricing mechanisms and auction-based slicing solutions

Some works [23, 42, 60, 61] have considered resource pricing mechanisms to optimize the profit of the different participants (the owner and the customers of network resources). In [60], the authors proposed to maximize the net social welfare of the InP and the virtual network subscribers by dynamically adjusting the prices of processing and bandwidth resources. In [23], the Slice Customers and the Slice Provider seek out to maximize their net profit. The authors also formulate the net social welfare problem, which aims to maximize the sum of each profit. The authors prove there exist a trade-off between the net social welfare and the Slice Provider profit. They propose a solution in two phases to adjust the pricing policy which achieves near optimal solution for the Slice Provider and the net social welfare problems with reduced time complexity. The authors apply the same pricing mechanism in [42] while focusing rather on slice reconfiguration than slice dimensioning. In [61], the competition occurs between the MVNOs which try to acquire slices from the network owner. The authors use a congestion game model and find the Nash equilibrium not equal to the social optimum. They also provide a pricing policy that the network owner can leverage to optimize its profit. The pricing policy is adaptive to the demand, i.e. if the number of mobile users increases then the price of radio resources increases proportionally. In contrast to these works, we do not make any assumptions on the pricing mechanisms as this policy can only be decided by the owner of the network, i.e. the NO. Moreover, we foresee that the pricing policy may depend on multiple latent factors thus causing non-stationary or even adversarial variations of the resource prices. Consequently, we will deem the latter as unknown and time-varying variables in the SP provisioning problem.

Other works [62–65] proposed auctions to model the interaction between the owner and the customers. [62] provides an overview of auction theory and auction theory-based slicing solutions. The authors first define network virtualization in the context of wireless networks. Then, they detail the auction theory framework, before showing how it can be applied to model the interaction between the InP and the SPs bidding for virtualized resources. In [63], the SP receives bids from the users and is in charge of the VNF deployment to accommodate the accepted bids. The authors formulate the social welfare problem to jointly maximize the utility of the SP and the users participants of the auction. The utility of a user is the bidding price minus the payment price if accepted and zero otherwise. The utility of the SP is the revenue from the payment of accepted users minus the resource operation cost. The formulation also takes account of the various link capacity constraints (inter-zone, upload and download). Then, the SP calculates the payment price for all VNF deployment strategies and select the one which maximizes the winning bid utility. Note that the SP only accepts bids which show non-negative utility and do not exceed capacity constraints. In [64], a slice is composed of multiple VNFs and the data centers provide available network resources of different types (storage, bandwidth, etc.) to host those VNFs. Halabian defines the demand vector which specifies the amount of resources necessary to provide a unit of wireless service for one unit of time and then defines the slice as the demand vectors for each VNF that composes the slice. Halabian then introduces the slice thickness which precises the number of wireless service units that can be executed in one unit of time. The author formulates the system utility problem and designs an auction game between the slices and the data centers, where the slices bid for the network resources and seek to maximize their own utility. The author proves the Nash equilibrium is not distinct from the optimal solution of the system utility problem. Finally, the author studies the case where slices are non-collaborative, i.e. they lie about their demand vectors. [65] proposed an auction for resource allocation in slices. The resource model is quite simplistic, considering network chunks that are combined linearly to support different slices. Unlike these works, we consider a dynamic pricing scheme, that does not require to organize and run any type of auction, hence it is more practical/applicable, and follows similar solutions that have been successfully applied in, e.g., cloud computing ecosystems.

2.2.2 The network slicing market

Recent works have proposed new business models to enable the interaction between the NO and the SPs [27, 66–68]. [66] defines a market to enable the creation of end-to-end (E2E) network slices in a dynamic manner. A blend of resources across multiple domains are available in the marketplace and the different network entities take actions so to orchestrate the slice creation and lifecycle. The goal in [27] is to maximize the InP revenue by performing slice requests admission and to ensure the SLAs are being satisfied for admitted slices. In its request, the tenant must specify the slice duration, the traffic type of the slice either elastic or inelastic, and the slice size. Only spectrum resource is considered for the slice. The InP then receives constant payoffs ρ_i and ρ_e for admitted inelastic and elastic slices. The authors model the dynamic problem as a Markov Decision Process and provide two reinforcement learning solutions. In [67], the same authors improve the Q-learning solution by adding a neural network entity which trains to reduce the error on the estimated Q-values. [68] leverages the blockchain technology to enable dynamic and secured trading exchanges on the resource ownership from the infrastructure to the tenants, with the introduction of intermediate brokers in charge of the transactions. Specifically, each intermediate broker manages a consortium where admitted tenants can trade resources. The designed slicing brokerage solution can include multiple types of network resources, which unleashes E2E network slicing. For the system to remain secure, the tenants must be admitted by the InP itself. Other papers have investigated blockchain as a mean to build a marketplace for network slicing [69, 70]. [69] investigated how to apply the blockchain technology to the network slice broker concept presented in [3]. [70] implemented a small network of VMs which act as peers to validate transactions.

Another line of work is to draw ideas from successful cloud computing marketplaces [71–74] that offer both in-advance reservation and on-the-spot bidding opportunities for computing and storage resources. Prior works that focus on such hybrid cloud market models have studied spot pricing models and devised intelligent bidding strategies for the buyers [36, 75–78]. In [75], the goal is to minimize the average cost of job completion while meeting the deadline for the job to be completed. The authors distinguish two types of jobs: the serial job where the tasks must be completed in a specific chain, and the parallel job where any task can be completed, thus allowing to use multiple VMs in parallel. For a parallel job, they derive an analytical solution for the dynamic VM bidding strategy which is optimal. In [76], the authors proposed to analyze the volatility of the spot markets by using inequality indices and moving averages as statistics. Then, they provide a formula to choose the smallest bid price with probability α of acceptance. In [36] the users place bids to reserve cloud resources for executing certain long tasks, aiming to minimize their costs while ensuring task completion over successive bidding periods. The main idea is to employ a hidden Markov model for tracking the evolution of spot prices; however, the analysis relies on the user needs complying to certain statistical assumptions. Similarly, in [77] an interesting bidding approach is considered where the users try to infer the pricing strategy of the cloud provider and bid accordingly in a spot market. Against the tide, the authors in [78] showcase that bidding strategies need not to be sophisticated to provide satisfactory results. This is mainly due to the wide number of spot markets available which guarantee to the bidder a constant access to cloud resources at a low and stable price. The strategy of the bidder should consist of selecting a high bid price and seeking a new spot market with lower price when an instance is revoked.

2.2.3 Section summary

Some works have designed pricing mechanisms to control the resource usage. This is a good leverage to avoid both under-utilization of resources leading to non-profitable 5G sliced networks, or over-utilization which yields violations of SLAs and impacts negatively the QoS of the slices. In our work, we do not model such pricing mechanism as its decision-making only belongs to the NO, owner of the network. On the contrary, we aim to design a reservation policy from the SP viewpoint which is robust to any type of underlying pricing mechanism.

Other works have designed auction games to model the competition between the SPs for the acquisition of resources. Papers notably look at the existence of Nash equilibria and their price of anarchy. Though relevant, we do not use this game theory modeling, as we consider the SPs can make reservation requests while ignoring the prices of resources, which change dynamically according to non-stationary or even adversarial patterns.

Finally, we expect the *network slicing market* to follow the design of cloud marketplaces, where the customers can request resources in-advance or on-the-fly through spot opportunities. We aim to design dynamic reservation policies incorporating both types of requests, by considering a two time-scale approach.

2.3 Background on OCO

In online learning [79], a learner has to take a decision $\boldsymbol{x}_t$ at each round t. At the time of the decision, the learner does not know the incurred loss $f_t(\boldsymbol{x}_t)$, only revealed by nature after the decision. The learner repeats this process of decision and observation and tries by its successive decisions to minimize the overall incurred loss, which is $\sum_{t=1}^{T} f_t(\boldsymbol{x}_t)$, where T is the duration of the learning time, also called the horizon. In OCO [80], the decision

$\boldsymbol{x}_t$ must belong to a bounded convex set $\mathcal{X} \subseteq \mathbb{R}^n$, and the loss function $f_t : \mathcal{X} \rightarrow \mathbb{R}$ must be convex and bounded over $\mathcal{X}$. The learner aims to minimize the overall loss $\sum_{t=1}^{T} f_t(\boldsymbol{x}_t)$, without knowing the losses $\{f_t\}_{t=1}^{T}$ which can possibly be chosen by an adversary. Two questions arise: how the learner can design an effective algorithm which takes relevant decisions and how to assess the algorithm's performance? In this section, we propose to the reader a summary of the different algorithms available in the OCO literature as well as their performance against the regret and constraint violation metrics that we define in the sequel.

2.3.1 Static regret

Zinkevich [81] introduced in his founding paper of OCO the regret metric to measure the performance of an algorithm $\mathcal{A}$ which helps the learner to make the decisions $\boldsymbol{x}_t$ at each round. The static regret definition is:

$$R_T^s(\mathcal{A}) = \sum_{t=1}^{T} f_t(\boldsymbol{x}_t) - \min_{\boldsymbol{x} \in \mathcal{X}} \sum_{t=1}^{T} f_t(\boldsymbol{x}).$$

Concretely, the regret measures the difference between the incurred cumulative loss of $\mathcal{A}$ against the cumulative loss of the optimal static solution with full hindsight, i.e. $\boldsymbol{x}^\star = \arg\min_{\boldsymbol{x} \in \mathcal{X}} \sum_{t=1}^{T} f_t(\boldsymbol{x})$. The goal is to achieve a sublinear regret, which means that the algorithm $\mathcal{A}$ performs no worse on average than the optimal static solution. In other words, one aims to achieve $R_T^s(\mathcal{A}) = o(T)$, i.e. $\lim_{T \rightarrow \infty} R_T^s(\mathcal{A})/T = 0$. This ensures the learning algorithm $\mathcal{A}$ asymptotically converges to the optimal solution. Designing such algorithm requires a few assumptions which are imperative in OCO:

- The feasible set $\mathcal{X}$ is bounded, closed, nonempty and convex.
- The functions $\{f_t\}_{t=1}^{T}$ are convex and differentiable.
- The gradient of each f_t is bounded, i.e. there exists some constant G such $\forall \boldsymbol{x} \in \mathcal{X}, ||\nabla f_t(\boldsymbol{x})|| \leq G$. This is equivalent to f_t being G-Lipschitz continuous.

For instance, the online gradient decent algorithm [80], also called *Greedy Projection* [81], which consists of the update:

$$\boldsymbol{x}_{t+1} = \pi_{\mathcal{X}}(\boldsymbol{x}_t - \eta_t \nabla f_t(\boldsymbol{x}_t)),$$

where $\pi_{\mathcal{X}}$ is the projection operator onto the convex set $\mathcal{X}$ and η_t is the step-size at round t, shows a $\mathcal{O}(\sqrt{T})$ regret, hence a sublinear regret. The regret is further reduced to $\mathcal{O}(\log T)$ in the work of Hazan et al [82] for strongly convex and exponentially concave functions.

2.3.2 Constrained OCO

A convex constrained problem is a problem where the goal is to minimize the objective function subject to some equality and/or inequality constraints [83]:

$$\begin{aligned} & & \min_{\boldsymbol{x}\in\mathcal{X}} \quad & f(\boldsymbol{x}) & (2.1)\\ & \text{s.t.} & \forall i = 1\dots m, \quad & g_i(\boldsymbol{x}) \leq 0 & (2.2)\\ & & \forall j = 1\dots n, \quad & h_j(\boldsymbol{x}) = 0 & (2.3) \end{aligned}$$

The functions f, g_i and h_j are all convex. Our thesis restricts the general convex constrained problem to the case of an objective to minimize with inequality constraints. Even more, the inequality constraints can be occasionally violated as long as the long-term inequality $\sum_{t=1}^{T} \boldsymbol{g}(\boldsymbol{x}_t) \preceq \mathbf{0}$ is respected. This case falls under the scope of constrained OCO, which is the framework of the main solutions presented in this thesis. In this subsection, we propose to detail the related literature within constrained OCO.

Mahdavi et al [84] proposes to solve the following problem over $\boldsymbol{x}_1, \dots, \boldsymbol{x}_T$:

$$\begin{aligned} \min_{\boldsymbol{x}_1\dots\boldsymbol{x}_T\in\mathcal{X}} \quad & \sum_{t=1}^{T} f_t(\boldsymbol{x}_t) \quad - \min_{\boldsymbol{x}\in\mathcal{K}} \sum_{t=1}^{T} f_t(\boldsymbol{x}), & (2.4)\\ \text{s.t.} \quad & \sum_{t=1}^{T} g_i(\boldsymbol{x}_t) \leq 0, \quad i = 1\dots m, & (2.5) \end{aligned}$$

where $\mathcal{K} = \{\boldsymbol{x} \in \mathcal{X} : g_i(\boldsymbol{x}) \leq 0, i = 1\dots m\}$. We make three important remarks on this problem:

- the regret is static;
- the constraints g_i do not vary over time;
- the benchmark must abide to a more stringent constraint ($\boldsymbol{x} \in \mathcal{K}$), while the solution of [84] only needs to respect the long-term constraint $\sum_{t=1}^{T} g_i(\boldsymbol{x}_t) \leq 0$, which means occasional violation is permitted.

The provided solution based on a Lagrangian method (with primal/dual update based on gradient descent/ascent) achieves a $\mathcal{O}(T^{1/2})$ regret and $\mathcal{O}(T^{3/4})$ constraint violation, i.e.:

$$\sum_{t=1}^{T} f_t(\boldsymbol{x}_t) - \min_{\boldsymbol{x} \in \mathcal{K}} \sum_{t=1}^{T} f_t(\boldsymbol{x}) = \mathcal{O}(T^{1/2}) \tag{2.6}$$

$$\sum_{t=1}^{T} g_i(\boldsymbol{x}_t) = \mathcal{O}(T^{3/4}), \quad i = 1 \dots m. \tag{2.7}$$

In the special case of linear constraints g_i, the authors provide a mirror-based method that achieves $\mathcal{O}(T^{2/3})$ regret and $\mathcal{O}(T^{2/3})$ constraint violation. Jennaton et al [85] addresses the same problem and provides a solution that achieves $\mathcal{O}(T^{\max\{\beta, 1-\beta\}})$ regret and $\mathcal{O}(T^{1-\beta/2})$ constraint violation, where β is a tunable parameter which allows the authors to trade-off between regret and violation. They achieve such improvement by introducing more elaborate step-sizes and regularizer in the Lagrangian.

One way to enrich the previous scope is to consider time-varying constraints. Victor et al [86] try to address the following problem:

$$\min_{\boldsymbol{x}_1 \dots \boldsymbol{x}_T \in \mathcal{X}} \sum_{t=1}^{T} f_t(\boldsymbol{x}_t) \quad - \min_{\boldsymbol{x} \in \mathcal{X}_{max}} \sum_{t=1}^{T} f_t(\boldsymbol{x}), \tag{2.8}$$

$$\text{s.t.} \quad \sum_{t=1}^{T} \boldsymbol{g}_t(\boldsymbol{x}_t) \preceq \boldsymbol{0}, \tag{2.9}$$

where $\mathcal{X}_{max} = \{\boldsymbol{x} \in \mathcal{X} : \sum_{t=1}^{T} \boldsymbol{g}_t(\boldsymbol{x}) \preceq \boldsymbol{0}\}$. This benchmark is stronger than $\mathcal{K}$ as now the oracle has the possibility to violate the constraint as long as the long-term constraint $\sum_{t=1}^{T} \boldsymbol{g}_t(\boldsymbol{x}) \preceq \boldsymbol{0}$ is respected. [87] has proved that it is not possible to get sublinear regret and constraint violation against such benchmark. Hence, most papers [88–90] have addressed the weaker benchmark:

$$\min_{\boldsymbol{x}\in\mathcal{X}_{min}} \sum_{t=1}^{T} f_t(\boldsymbol{x}), \quad \mathcal{X}_{min} = \{\boldsymbol{x} \in \mathcal{X} : \boldsymbol{g}_t(\boldsymbol{x}) \preceq \mathbf{0}, \quad \forall t = 1 \dots T\}. \tag{2.10}$$

In [86], the authors prove that obtaining sublinear regret and constraint violation is possible against a benchmark that lies in a set X such that $\mathcal{X}_{min} \subseteq X \subseteq \mathcal{X}_{max}$. Nevertheless, the authors consider varying constraints with only linear perturbations.

Liakopoulos et al [91] creates the K-benchmark defined as:

$$\min_{\boldsymbol{x}\in\mathcal{X}_{K}} \sum_{t=1}^{T} f_t(\boldsymbol{x}), \quad \mathcal{X}_{K} = \{\boldsymbol{x} \in \mathcal{X} : \sum_{\tau=t}^{t+K-1} \boldsymbol{g}_\tau(\boldsymbol{x}) \preceq \mathbf{0}, 1 \leq t \leq T-K+1\}. \tag{2.11}$$

The authors provide a solution with a $\mathcal{O}(T^{1-\epsilon/2})$ regret and $\mathcal{O}(T^{1-\epsilon/4})$ constraint violation against a $T^{1-\epsilon}$-benchmark, where $\epsilon > 0$ is a tunable parameter. Instead of a primal-dual approach, the authors design a virtual queue which buffers the violations. By bounding the Lyapunov drift of the virtual queue plus the loss and smoothness terms, the authors are able to provide an upper bound on the regret and constraint violations. Following the same kind of methodology (Lyapunov drift bound), [39] provides the same guarantees.

2.3.3 Dynamic regret

Another line of research assesses the learning policy using the dynamic regret metric, defined as:

$$R_T^d(\mathcal{A}) = \sum_{t=1}^{T} f_t(\boldsymbol{x}_t) - \sum_{t=1}^{T} f_t(\boldsymbol{x}_t^*),$$

where $\boldsymbol{x}_t^*$ is one minimizer of the dynamic benchmark, i.e. the optimal solution of the following convex problem:

$$\min_{\boldsymbol{x}\in\mathcal{X}} \quad f_t(\boldsymbol{x}), \tag{2.12}$$

$$\text{s.t.} \quad \boldsymbol{g}_t(\boldsymbol{x}) \preceq \mathbf{0}. \tag{2.13}$$

It is quite evident to see that the dynamic benchmark is stronger than the static benchmark:

$$\begin{aligned}
\forall t, \quad f_t(\boldsymbol{x}_t^*) &\leq f_t(\boldsymbol{x}^*) && (2.14)\\
\Leftrightarrow \quad \sum_{t=1}^{T} f_t(\boldsymbol{x}_t^*) &\leq \sum_{t=1}^{T} f_t(\boldsymbol{x}^*) && (2.15)\\
\Leftrightarrow \quad -\sum_{t=1}^{T} f_t(\boldsymbol{x}^*) &\leq -\sum_{t=1}^{T} f_t(\boldsymbol{x}_t^*) && (2.16)\\
\Leftrightarrow \quad R_T^s(\mathcal{A}) &\leq R_T^d(\mathcal{A}). && (2.17)
\end{aligned}$$

Hence, if an algorithm $\mathcal{A}$ achieves sublinear dynamic regret, it will also achieve sublinear static regret.

Zinkevich [81] defined the dynamic regret in his founding paper and also provided an upper bound within the context of unconstrained OCO by using the online gradient descent method. Specifically, Zinkevich defines the path length of the dynamic sequence $\{\boldsymbol{x}_t^*\}_{t=1}^T$ as:

$$L_T = \sum_{t=1}^{T-1} ||\boldsymbol{x}_t^* - \boldsymbol{x}_{t+1}^*||_2,$$

and shows that the online gradient descent can achieve a $\mathcal{O}(L_T\sqrt{T})$ regret. [92] improves the bound to $\mathcal{O}(L_T)$ for the case of strongly convex and smooth functions. Although it is not possible to achieve a sublinear dynamic regret bound, one can still express the bound in terms of the path length. [93] introduces the squared path length:

$$S_T = \sum_{t=1}^{T-1} ||\boldsymbol{x}_t^* - \boldsymbol{x}_{t+1}^*||_2^2,$$

and reduces the dynamic regret bound to $\mathcal{O}(\min(L_T, S_T))$ in the case of strongly convex and smooth functions, which is quite an improvement as S_T can be much smaller than L_T if the local variations of the dynamic sequence are small. [94] provides a $\mathcal{O}(L_T)$ bound for convex and smooth functions, given that the $\boldsymbol{x}_t^*$'s are interior points of $\mathcal{X}$. Other regularities than the path length or the squared path length have been introduced, like the functional variation in [95]:

$$\mathcal{F}(f_1, \dots, f_T) = \sum_{t=2}^{T} \max_{\boldsymbol{x}\in\mathcal{X}} |f_t(\boldsymbol{x}) - f_{t-1}(\boldsymbol{x})|,$$

or the gradient variation in [96]:

$$\mathcal{G}(f_1, \dots, f_T) = \sum_{t=2}^{T} \max_{\boldsymbol{x} \in \mathcal{X}} ||\nabla f_t(\boldsymbol{x}) - \nabla f_{t-1}(\boldsymbol{x})||^2,$$

The range of regularities allows to choose the algorithm that provides a bound with regard to the smallest regularity for the problem at hand.

In the context of constrained OCO, Giannakis et al [1] consider the dynamic regret and fit metrics defined as:

$$R_T^d = \sum_{t=1}^{T} f_t(\boldsymbol{x}_t) - \sum_{t=1}^{T} f_t(\boldsymbol{x}_t^*), \tag{2.18}$$

$$Fit_T = \Big|\Big| [\sum_{t=1}^{T} \boldsymbol{g}_t(\boldsymbol{x}_t)]^+ \Big|\Big|, \tag{2.19}$$

where $\boldsymbol{g}_t$ is the vector of m time-varying constraint functions at round t, and $[.]^+$ nullifies the negative entries of the accumulated violation vector $\sum_{t=1}^{T} \boldsymbol{g}_t(\boldsymbol{x}_t)$. The authors express the dynamic regret and fit bounds in terms of the path length and of a new regularity, the accumulated variation of constraints defined as:

$$V(\{\boldsymbol{g}_t\}_{t=1}^{T}) = \sum_{t=1}^{T} \max_{\boldsymbol{x} \in \mathcal{X}} ||[\boldsymbol{g}_{t+1}(\boldsymbol{x}) - \boldsymbol{g}_t(\boldsymbol{x})]^+||.$$

They achieve a regret bound of $\mathcal{O}(\max\{L_T T^{1/3}, V(\{\boldsymbol{g}_t\}_{t=1}^{T}) T^{1/3}, T^{2/3}\})$ and a fit bound of $\mathcal{O}(T^{2/3})$, by setting the step-sizes to $T^{-1/3}$. Their solution is a modified online saddle point method, where the primal update consists of minimizing the Lagrangian, and the dual update is a gradient ascent. The validity of their proof relies on the Slater's condition and another condition on the slack constant, called strong Slater condition. We give the definitions of the weak and strong Slater conditions in chapter 4 and we detail the importance of such conditions in the Appendix. Liu et al [97] improve the results of [1] in terms of regret and constraint violation bounds and provides an algorithm that generalizes to the bandit feedback settings. In the latter, the learner does not know the full gradient feedback, but only a single point [98] or multiple points [99] of the loss and constraint functions. The authors in [97] provide a regret bound of $\mathcal{O}(\sqrt{TL_T})$ and a constraint violation bound of

$\mathcal{O}(T^{3/4}L_T^{1/4})$, under the assumption that both T and L_T are known so that the step-size η can be set to $\sqrt{L_T/T}$, and the path length is sublinear, i.e. $\lim_{T\to\infty} L_T/T = 0$. On the other hand, the proof does not need the Slater's condition to hold. Johansson et al [100] propose a distributed online optimization algorithm and provide two regret bounds with and without the Slater's condition. They achieve sublinear regret and fit in both cases, given that some regularity on the regularized Bregman projection of the dynamic sequence grows sublinearly. In [101], the authors provide both sublinear dynamic regret and constraint violation without the Slater's condition, given that the regularities are sublinear. They also preserve their solution's performance in the case the time horizon T is unknown by using a doubling trick. The provided bounds depend on the horizon T, the path length L_T and the accumulated variation of constraints $V(\{\boldsymbol{g}_t\}_{t=1}^T)$: $\mathcal{O}(\max\{\sqrt{TL_T}, V(\{\boldsymbol{g}_t\}_{t=1}^T)\})$ for the regret and $\mathcal{O}(\max\{\sqrt{T}, V(\{\boldsymbol{g}_t\}_{t=1}^T)\})$ for each constraint violation. Then, using the strong Slater's condition (with the additional condition on the slack constant), the constraint violation bound reduces to $\mathcal{O}(1)$. The solution is a primal-dual approach similar to the one of [1]. In [102], the authors provide tunable bound of $\mathcal{O}(\max\{T^{\delta}L_T, T^{1-\delta}\})$ for the regret, with $\delta \in (0,1)$. With the Slater's condition, the constraint violation bound reduces from $\mathcal{O}(T^{1-\delta/2})$ to $\mathcal{O}(\max\{T^{1-\delta}, T^{\delta}\})$. We provide a summary of these works in Table 2.4.

2.3.4 Section summary

OCO is a new mathematical framework introduced by Zinkevich in 2003 [81] which provides strong performance guarantees under basic assumptions on the feasible set $\mathcal{X}$ and the losses $\{f_t\}_{t=1}^T$. The regret metric allows the learner to assess the performance of its online policy. There are multiple types of benchmark, which prove to be more or less tight comparators for the regret: static benchmark, dynamic benchmark, optimal in hindsight policy.

Constrained OCO is a practical extension where convex constraint functions are added to the problem at hand. It constitutes a great framework to assess the performance of online optimization solutions. It meets the needs of many applications in the network slicing area, like resource provisioning [39, 40], or resource allocation [1], problems where the learner tries to maximize a utility function under capacity constraints of the network resources, which are inherent to 5G networks.

Table 2.4: Constrained OCO overview

References	Regret type	Benchmark	Regret, fit bounds
[84]	static	$\mathcal{X}_{min}$	$\mathcal{O}(T^{1/2}), \mathcal{O}(T^{3/4})$
[85]	static	$\mathcal{X}_{min}$	$\mathcal{O}(T^{\max\{\beta,1-\beta\}}), \mathcal{O}(T^{1-\beta/2})$
[86]	static	$\mathcal{X}_{min} \subseteq X \subseteq \mathcal{X}_{max}$	$\mathcal{O}(T^{\max\{\epsilon,1-\epsilon\}}), \mathcal{O}(T^{\epsilon})$
[39, 91]	static	$\mathcal{X}_K$	$\mathcal{O}(T^{1-\epsilon/2}), \mathcal{O}(T^{1-\epsilon/4})$
[1], thesis	dynamic	$\mathcal{X}_{min}$, strong Slater	$\mathcal{O}(\max\{L_T T^{1/3}, V(\{\boldsymbol{g}_t\}_{t=1}^T)T^{1/3}, T^{2/3}\})$, $\mathcal{O}(T^{2/3})$
[97]	dynamic	$\mathcal{X}_{min}$, no Slater	$\mathcal{O}(\sqrt{TL_T}), \mathcal{O}(T^{3/4}L_T^{1/4})$
[100]	dynamic	$\mathcal{X}_{min}$, no Slater Slater	$\mathcal{O}(\max\{T^{\kappa}L_T, T^{\max\{1-\kappa,\kappa\}}\}), \mathcal{O}(T^{1-\kappa/2})$ idem, $\mathcal{O}(T^{\max\{1-\kappa,\kappa\}})$
[101]	dynamic	$\mathcal{X}_{min}$, no Slater strong Slater	$\mathcal{O}(\max\{\sqrt{TL_T}, V(\{\boldsymbol{g}_t\}_{t=1}^T)\})$, $\mathcal{O}(\max\{\sqrt{T}, V(\{\boldsymbol{g}_t\}_{t=1}^T)\})$ idem, $\mathcal{O}(1)$
[102]	dynamic	$\mathcal{X}_{min}$, no Slater Slater	$\mathcal{O}(\max\{T^{\delta}L_T, T^{1-\delta}\})$, $\mathcal{O}(T^{1-\delta/2})$ idem, $\mathcal{O}(\max\{T^{1-\delta}, T^{\delta}\})$

Within dynamic constrained OCO, many solutions [1, 97, 100–104] have provided a dynamic regret bound expressed in terms of the horizon T and the regularities of the problem. These solutions provide sublinear regret and constraint violation guarantees only if the defined regularities are as well sublinear. The differences between these solutions are about the assumptions they make (some achieve similar results without the Slater condition [97, 100]) or about the values of the step-sizes (some can design the step-sizes with unknown time horizon for instance [101]).

Our contribution lies in the application of the dynamic constrained OCO framework to resource provisioning problems in network slicing. Our solutions are similar to some recent works [1, 101, 102] using a primal-dual optimization methodology and are designed to the specific resource reservation problem faced by the SP.

3 The reservation of radio resources: a cost-reduction strategy for the SP

This chapter presents the contribution C1 which addresses the research question RQ2. The paper associated with this contribution is "Resource Reservation within Sliced 5G Networks: A Cost-Reduction Strategy for SPs" and was presented at the IEEE International Conference on Communications (ICC) Workshops, in June 2020.

3.1 Introduction

As described in the Introductory chapter, the RAN virtualization offers two main approaches which are namely the sharing approach where the spectrum is dynamically shared between the SPs and the reservation approach where each SP can access to dedicated spectrum resources. We envision the in-between scenario where the SPs request dynamically the spectrum resources for each slicing window, which represents a unit time for RAN slicing [105]. The SPs can serve their evolving demand based on the shared spectrum and the dedicated requested spectrum chunk.

Consequently, the SP has access to two types of resources, representing portions of the overall MNO capacity: guaranteed resources, and unreserved resources available on a best-effort basis. Guaranteed resources are those that the SP booked in advance, while best-effort resources are portions of the spare capacity (after we account for the pre-reserved resources) shared between multiple SPs. The guaranteed resources will come at a higher cost, and therefore, to minimise its costs, the SP is inclined to reserve resources only when it expects that best-effort resources will not suffice

The contributions of the work presented in this chapter are listed as follows:

1. We envision a network model comprising a set of base stations and a reservation model for virtualized RAN resources, which implies bandwidth capacity at the base station in our specific case.
2. We define the optimal reservation policy in terms of the network capacity and the traffic loads encountered by the MNO. The SP does not have access to such information but can infer its reservation decisions based on the MNO's feedback it receives at each time-step, which comprises the amount of best-effort traffic, the delivered traffic and undelivered traffic of the SP.
3. We propose two supervised solutions able to output the reservation decisions of the SP for the next h time-steps. We feed into the neural network architecture of our solutions the historical data the SP possesses (past best-effort, delivered and undelivered traffic) and its past reservations.
4. We train the solutions offline and evaluate them in an online environment. We emulate different network traffic conditions to assess the performance of our solutions under all circumstances and we compare them to an ARIMA baseline, which possesses ideal information of the network capacity and the historical traffic loads encountered by the MNO.

To the best of our knowledge, this is the first time supervised learning solutions, namely a DNN and a LSTM are proposed to assist the SP in designing its online reservation policy. We refer the reader to the Table 2.2 for more details on the literature.

This chapter is structured as follows. We open with the network model description and the definition of the different traffic traces which determine the reservation rules. We then derive an expression for the optimal reservation policy and define the reservation policy of the SP. Then, we thoroughly describe the architecture of our supervised solutions and discuss the choice of the hyper-parameters. We also explain the two-phase learning approach, where the solutions are trained offline with access to historical data and then loaded in an online environment. Finally, we validate our solutions against real-world traffic data and under different traffic conditions. We compare the obtained results to the ARIMA baseline. We conclude the chapter with a discussion of the obtained results and the potential extensions

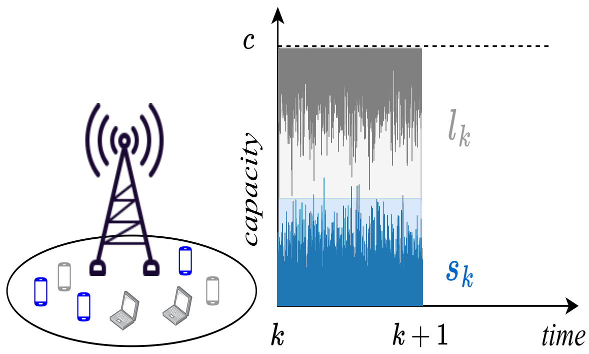


Figure 3.1: A SP can reserve bandwidth capacity (blue square) at the base station for its demand (blue signal) coming from User Equipments (UEs) using its service.

of the reservation model.

3.2 System Model

3.2.1 Network Model

We consider a single MNO with its physical network simultaneously serving the traffic demands of multiple SPs. We outline the model considering a single base station; the solution presented here can be replicated at multiple base stations. The base station has limited capacity, which we denote as c. We represent the total traffic demand faced by the base station as a time-series signal consisting of two components: the first component is the traffic demand related to the SP of interest; we denote this as s_k, with $k \in \mathbb{N}$ representing the k-th time-step . The second component is the aggregate traffic demand of all other SPs that are tenants of the network, denoted as l_k. Throughout this chapter, we will refer to s_k as simply the SP traffic and to l_k as the MNO traffic. Note that we measure both traffic and c in bytes/time-step. We refer to Fig. 3.1 for a visual description of the system model. Furthermore, we denote the past sequences of traffic for l_k and s_k, respectively, as the vectors $\boldsymbol{l}_K = [l_1, ..., l_K]$ and $\boldsymbol{s}_K = [s_1, ..., s_K]$, with K representing the current time-step.

3.2.2 Reservation Decisions

The SP can reserve resources in advance and the MNO will ensure that those resources will be available when needed. The SP can also rely on resources that will be available in the network on a best-effort basis. Note that when the SP relies on the best-effort approach, there might not be enough resources available to serve all of its traffic demands, especially during periods of peak traffic. At time-step k, the SP leases for the next h time steps a given amount of guaranteed resources, which we model as a portion of the overall capacity $\boldsymbol{r}_k = [r_{k+1}, ..., r_{k+h}]$. Therefore, the next reservation decision will occur at time-step $k+h$. A time-step can range from a few minutes to a few hours in a dynamic reservation system.

We assume both c and l_k are unknown to the SP, as the MNO may be reluctant to share this information in a real-world context. Instead, the SP must rely on other signals, including its delivered traffic, d_k, undelivered traffic, u_k and the portion of its traffic that was served on a best-effort basis, b_k, as summarised in Table 3.1. The delivered traffic includes the SP's reservation and its traffic served as best-effort. If the aggregated traffic from all SPs is less than the capacity of the network, then all of the SP traffic is delivered. If the aggregated traffic is greater than the capacity, the delivered traffic corresponds to the reservation r_k plus the resources that are available on a best-effort basis, equal to $c - l_k$.

Table 3.1: System model equations

if $s_k + l_k \leq c$	if $s_k + l_k > c$
$d_k = s_k$	$d_k = \min(\max(c - l_k + r_k, r_k), s_k)$
$u_k = 0$	$u_k = s_k - d_k$
$b_k = max(d_k - r_k, 0)$	$b_k = \max(d_k - r_k, 0)$

Additionally, in Table 3.2 we list the complete information which the SP possesses, including its own past traffic demands, $\boldsymbol{s}_K$, and past reservations $\boldsymbol{r}_K$, and how much of its traffic has been served in the past.

Table 3.2: Summary of SP knowledge

Known to SP:	$\boldsymbol{s}_K$, $\boldsymbol{r}_K$, $\boldsymbol{d}_K$, $\boldsymbol{u}_K$, $\boldsymbol{b}_K$
Unknown to SP:	c , $\boldsymbol{l}_K$

Guaranteed resources, i.e. those reserved in advance by the SP, come at a higher cost than resources that are made available to the SP on a best-effort basis. Thus, the SP is motivated to minimise its reservation of resources, if it believes that there would be sufficient capacity in the network to serve its traffic as best-effort. We also assume that, the farther in advance the SP makes a reservation, the cheaper that reservation will be, and therefore if the SP is to make a reservation it will try to do it as far in advance as possible.

In Fig. 3.2, we depict some possible scenarios of traffic load experienced by the MNO and reservations by the SP. For each case, we indicate a hypothetical reservation r_k and the resulting portion of the SP traffic that is treated as best-effort.

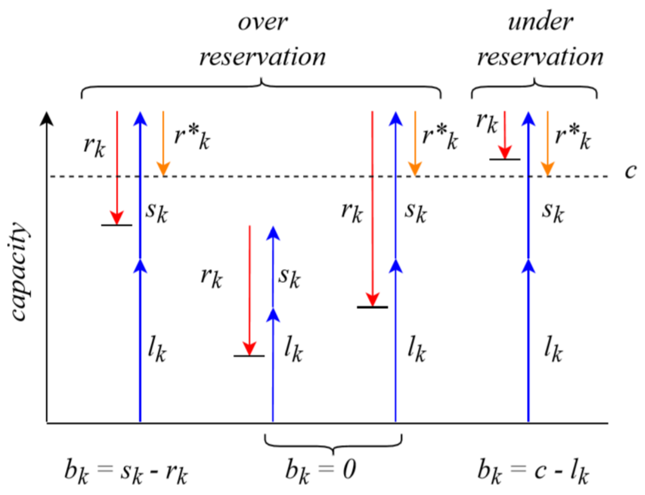


Figure 3.2: Example reservation policies for various edge cases of our model.

We summarise the key assumptions as follows:

Assumption 1. *When the SP reserves guaranteed resources, these will get allocated regardless of the network load.* [1]

Assumption 2. *"multi-steps-ahead reservation": At each time step, if needed, the SP can reserve resources for the h next time-steps.*

Assumption 3. *At each time k, the spare network capacity allocated on a best-effort basis to the SP of interest is what is left after all other network traffic is served, i.e.* $\max(\min(s_k - r_k, c - l_k), 0)$.

3.3 Problem Formulation

The objective of the SP is to avoid over or under reservation of MNO's resources, while getting its traffic served. When the reservation surpasses the optimal value, resulting in

[1]The MNO can exercise admission control, and only accept reservations that can be met with its available capacity.

over-reservation, the SP incurs higher cost than necessary, as a portion or all of its traffic could have been served with best-effort resources. In the case of *under-reservation*, the SP does not reserve enough resources and therefore a portion or all of its traffic is not delivered. The SP aims to make reservations that avoid both unnecessary guaranteed resources and undelivered traffic.

We can formulate the optimal reservation as follows:

Lemma 1. *$\forall k \in \mathbb{N}$, let $c - l_k$ be the MNO's capacity availability; then, the optimal reservation policy for the SP is $r_k^* = 0$ if $s_k + l_k \leq c$, and $r_k^* = \min(\min(s_k - (c - l_k), s_k), c)$ otherwise.*

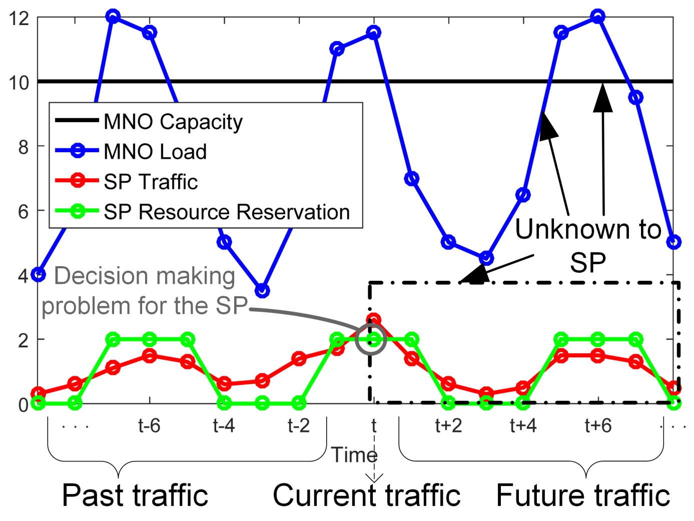


Figure 3.3: Graphical example of the SP's reservation problem we address in this chapter.

We illustrate the decision making problem faced by the SP in Fig. 3.3. The SP must reserve for the future time-steps, without the knowledge of the MNO load and the MNO capacity. We can now formulate the problem as follows:

Problem 1. *At time-step k, if needed, make a reservation for the next h time-steps so as to minimise the expected Mean Square Error (MSE) $\sum_{j=1}^{h} |r_{k+j} - r_{k+j}^*|^2$.*

As indicated previously, we assume the SP does not have access to the MNO traffic load l_k or the MNO capacity c, and therefore the SP must base its decision on its past history of reservations and how much of its traffic has been served, i.e., on $\boldsymbol{b}_K$, $\boldsymbol{d}_K$ and $\boldsymbol{u}_K$. We note that the biggest difficulty resides in the prediction of future availability of excess capacity, as both c and l_k are unknown, whereas the prediction of future SP demand can rely directly on $\boldsymbol{s}_K$. As $d_k + u_k = s_k$, we include only the signals s_k, r_k, u_k and b_k in our final solution, described in the sequel.

We adopt supervised machine learning in our solution, due to the following reasons:

- The SP faces a prediction problem with exogenous data from s_k, b_k and u_k. In comparison to unsupervised learning, supervised methods are generally more robust and successful at extracting patterns from exogenous data.
- We have an exact solution for the optimal reservation policy, r_k^*, for all k. By having access to a data set consisting of multiple weeks of recorded data, we can construct the necessary labels $\boldsymbol{r}_k^* = [r_{k+1}^*, ..., r_{k+h}^*]$ and signal traces that are required in training a supervised Machine Learning (ML) model based on our previously defined loss function in Problem 1.

3.4 Proposed solutions

In this section, we describe the two proposed solutions the SP should adopt to address the resource reservation problem we defined in Section 3.2. In the first subsection, we specify in detail how the SP can employ the DNN-based solution. In the second subsection, we explain how the SP can make use of an LSTM model. Finally, in the third subsection, we describe the framework we developed that would enable the SP to use our solution in real network deployments.

3.4.1 DNN model

Our first solution adopts a DNN architecture. The previous work of [106] has demonstrated how a sliding window technique can be used on time-series data to capture temporal correlations with DNN models: the moving window generates sequences that are fed as inputs to predict the next time-steps' reservation values.

In our model, we use q to denote the length of the past sequences, i.e., the length of $\boldsymbol{s}_{k,q} = [s_{k-q+1}, ..., s_k]$, for all $k \leq K$, and the k-th sequence sample we denote with $\boldsymbol{x}_{k,q} = [\boldsymbol{s}_{k,q}, \boldsymbol{r}_{k,q}, \boldsymbol{u}_{k,q}, \boldsymbol{b}_{k,q}]$. Using these samples, we construct a two-dimensional matrix

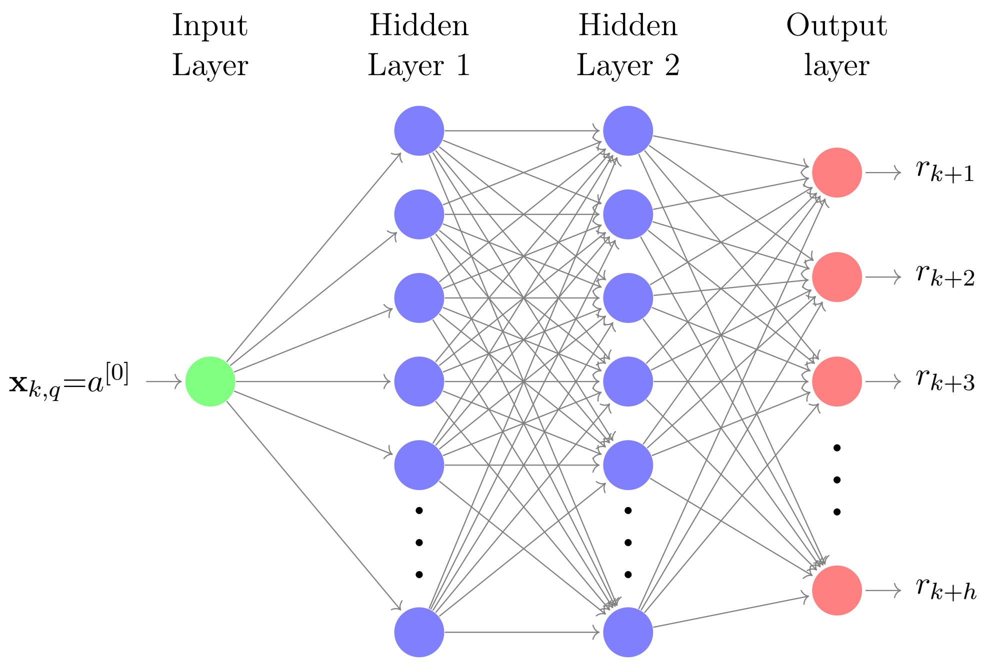


Figure 3.4: DNN structure.

with dimensions $(< T_{\boldsymbol{x}} >, 4q)$, where $< T_{\boldsymbol{x}} >$ represents the number of used samples, and employ the matrix as an input to the DNN.

We visualize the structure of our neural network with two hidden layers in Fig. 3.4. We can write its output as:

$$[r_{k+1}, ..., r_{k+h}] = f^{[3]}_{W,b} \circ f^{[2]}_{W,b} \circ f^{[1]}_{W,b}(\boldsymbol{x}_{k,q}), \tag{3.1}$$

where W is the matrix of weights, b the vector of biases, and f the activation function at each layer.

We have selected the gradient descent method to update the weights and biases of the architecture. In our DNN architecture, there are $4q$ input neurons, 50 neurons in the first hidden layer, and 10 neurons in the second hidden layer. The output layer has h neurons, corresponding to the predictions for the next h time steps. We use tanh activation functions for all layers, except for the last layer, which uses a sigmoid function.

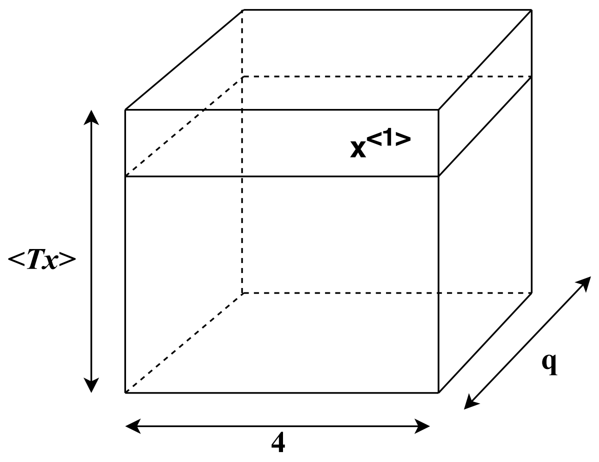


Figure 3.5: LSTM input tensor.

3.4.2 LSTM model

We designed the second solution using an LSTM model [107], which is a type of Recurrent Neural Network that has proven to be highly suitable for time series forecasting [108]. We use the same sequences that we generated previously for the DNN solution. For the LSTM model, we convert the $\boldsymbol{x}_{k,q}$ super vector into a 3D tensor, represented in Fig. 3.5, with shapes $(< T_{\boldsymbol{x}} >, 4, q)$ corresponding to the number of samples, number of input time-series and sequence length, respectively.

The first hidden layer of type LSTM outputs to a fully connected layer with ten neurons. The output layer has h neurons, corresponding to the predictions for the next h time-steps. The activation functions used are *tanh*, followed by *sigmoid* for the dense layers. Both our neural network models are trained using the Adam optimiser [109] and the loss function defined in Problem 1. Additionally, the learning rates we use are 0.01 for the DNN and 0.001 for the LSTM.

3.4.3 Development of the framework

In Fig. 3.6, we provide a high-level overview of the process the SP has to follow during the training phase and how it can then transition to use it in the online phase. In the *training phase*, we use the signals s_k, l_k and c, in accordance with Lemma 1, to calculate the optimal reservation labels at each time-step k of our data set. During the *training phase*, we use only the signals s_k, u_k, b_k, r_k as inputs to the model, recalling that these are assumed to

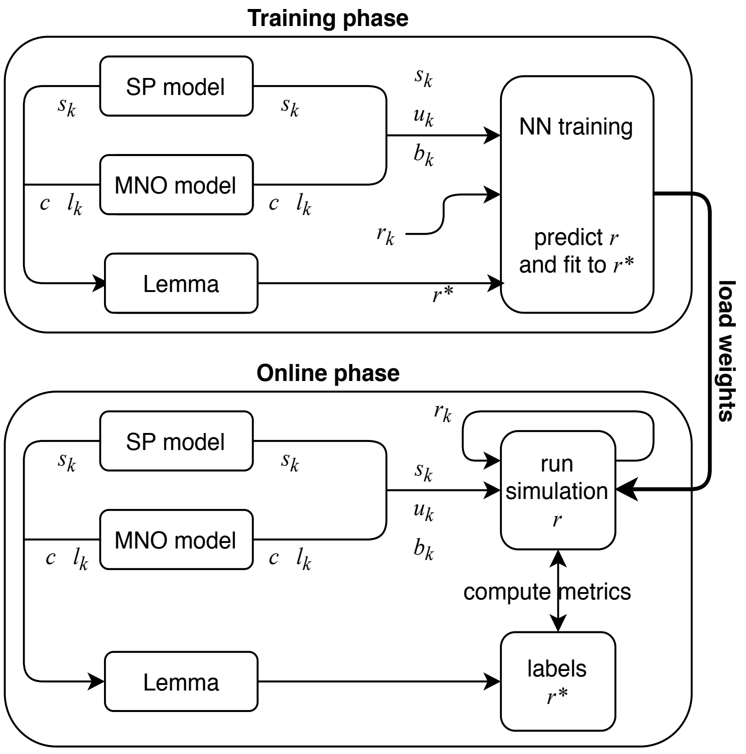


Figure 3.6: System model for reservation decision making.

be the only signals available to SP in a real-world context. We construct r_k as a random trace within the range $[0, max(s_k)]$ at each time-step k of our data set. Furthermore, we extract u_k and b_k using the equations we listed in Table 3.1. During the *online phase*, the SP obtains the real-time values for signals u_k and b_k at each time-step k, which the SP then combines with r_k and s_k, in order to predict the future reservations. The predicted outputs r_k are fed back as input into the model to perform the next predictions. We initialise the first sequence for r_k at $max(s_k)$ during this phase.

3.5 Validation

In this section, we validate our solution using real-world data collected in the city of Shanghai. To evaluate the effectiveness of our solution we designed a baseline approach using an ARIMA model. Then we compare our two proposed solutions with the baseline according to three metrics, namely MSE, over-reservation and under-reservation, and over three distinct use cases, low traffic, medium traffic and high traffic conditions.

3.5.1 The data set

The data set we use contains the aggregated traffic volumes seen across 20 base stations owned by a major MNO within the city of Shanghai, see Fig. 3.7. For each base station, traffic volumes have been recorded over a one-month period, spanning from Friday 1 August 2014 00:00 to Sunday 31 August 2014 23:50, with each recording averaged over a period of 10 minutes. Hence, there are 6 measurements per hour and a total of 4464 measurements for each base station over this period. We use 14 base stations to train and test the models, and 1 base station to evaluate the model. We split the data into training and test sets. The training set consists of 90% of the data points, while the remaining 10% we use for testing.

In order to train and evaluate our proposed solutions, we are required to extract signals from our data set corresponding to the MNO traffic l_k, the capacity of the base station c and the SP traffic s_k. We use the aggregated traffic volumes of the MNO of Shanghai as the MNO traffic l_k. We model the SP signal as a percentage of the MNO traffic plus a noise term. In this case, we have chosen a percentage of $\frac{15}{100}$, as we expect the MNO to host between 5 to 10 other SPs on its network. We also add Ornstein-Ulhenbeck noise instead of white noise, as we can see from Fig. 3.8, as it is more realistic to have correlated noise, as opposed to independent spurious noise.

By varying the base station capacity c, we are also able to simulate three use cases: low traffic, medium traffic, and high traffic. For example, by selecting c as the 70th percentile of the MNO traffic, we evaluate the case that 70% of the MNO traffic data points is less or equal than c. By selecting c as the 90th, 80th and 70th percentiles we are able to simulate low, medium, and high traffic, respectively.

3.5.2 Technical implementation

The code related to the architecture of our solutions is available to the community in our public GitHub[2]. Fig. 3.9 provides an overview of our system architecture that we aim to describe with more details in this subsection. We develop under Python 3.6 programming environment. We use the keras.models, keras.layers and keras.optimizers to implement the

[2]https://github.com/jeanba19/DNN-LSTM

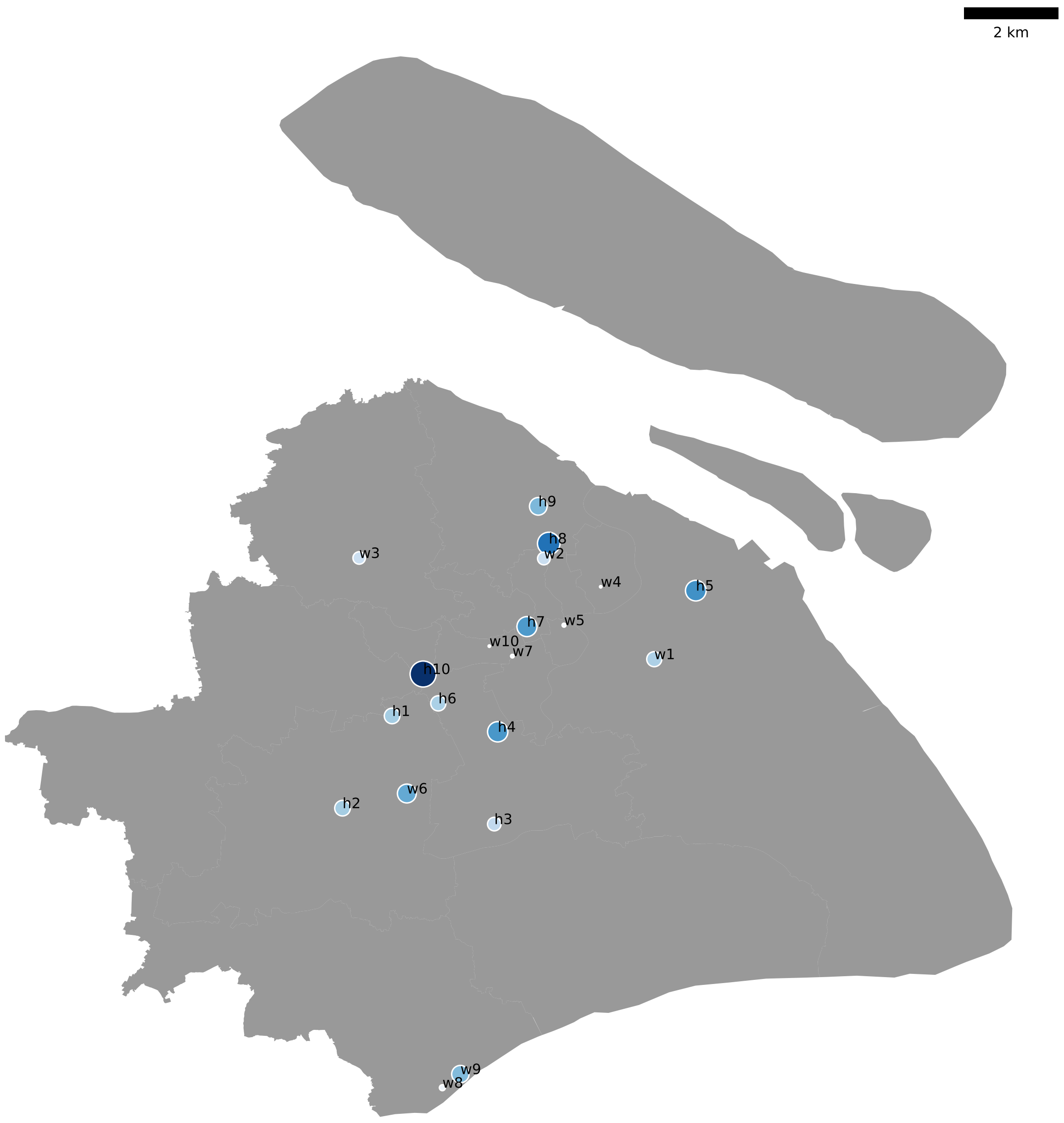


Figure 3.7: Average traffic density per base station over one month, August 2014, Shanghai city.

DNN and LSTM architectures with Adam optimizer. With a python script, we prepare time series of traffic volumes for each BS from the data available in a .csv file. We normalize the timestamp values by the overall average traffic volume. Some BSs present zero traffic, thus we discard these outliers from the training as our reservation solutions only apply if traffic exists (otherwise best reservation is constantly zero).

For both the training and evaluation stage, we generate the inputs $\{\boldsymbol{x}_{k,q}\}_k$ using a sliding window of size q. Thus, the input signals are finite sequences starting at different time steps that allow us to capture temporal relations in our inputs. For example, we have taken

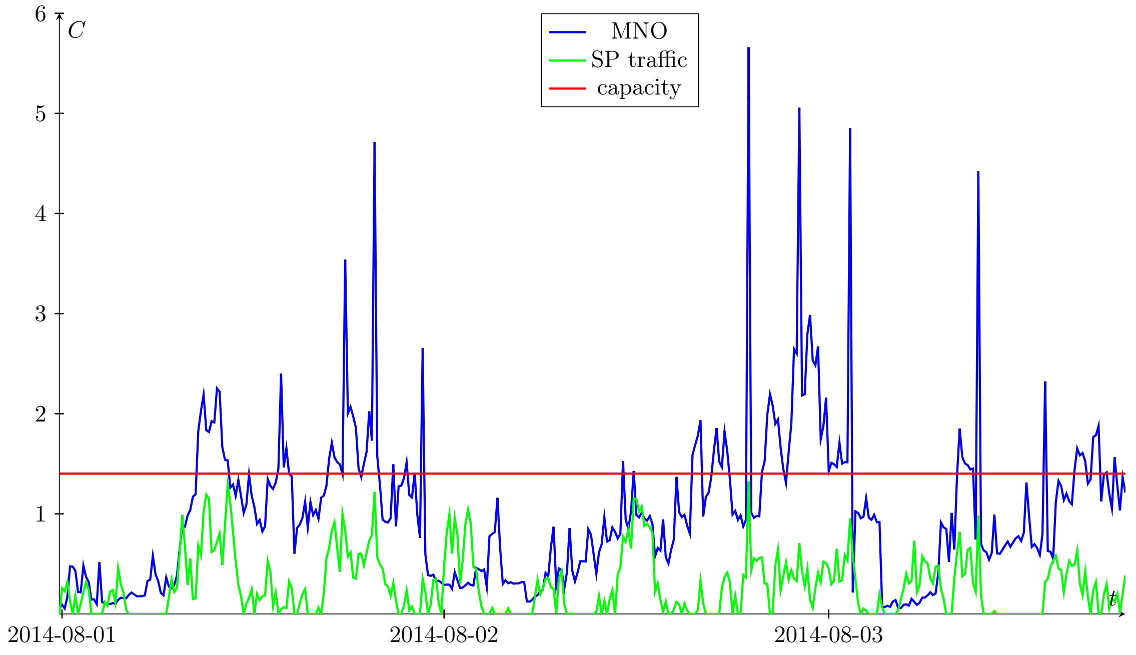


Figure 3.8: Example traffic traces and capacity model. The y-axis units are normalized values of the traffic volumes, i.e. the number bytes per time period in the original dataset after normalization.

different sequence lengths, covering 1 hour up to 10 hours of past traffic history. As the temporal data is measured every 10 minutes, 1 hour corresponds to $q = 6$ time steps, 10 hours to $q = 60$ time steps. The sequence lengths are $q = 6$, 18, 36 and 60.

We train the DNN and the LSTM models for all the sequence lengths and under all traffic conditions (high, regular, low). It makes a total of $4 \times 3 = 12$ trained models that we save under .h5 file format. We then test the models and select for each type of traffic the sequence length showing the best results in terms of MSE. For the training stage, we generate the multi-steps ahead labels according to Lemma 1. We replicate 10 times and scramble the training examples, coming from 14 BSs. The number of training epochs is 10.

We evaluate the solutions at one BS over a month (the length of the time series). We simulate the three different traffic conditions and evaluate the best DNN/LSTM model for each case. We generate the labels to measure the performance of our solutions through different metrics defined in subsection 3.5.4. These labels are hidden to the solutions – as it is no longer the training stage – and only used to generate performance results.

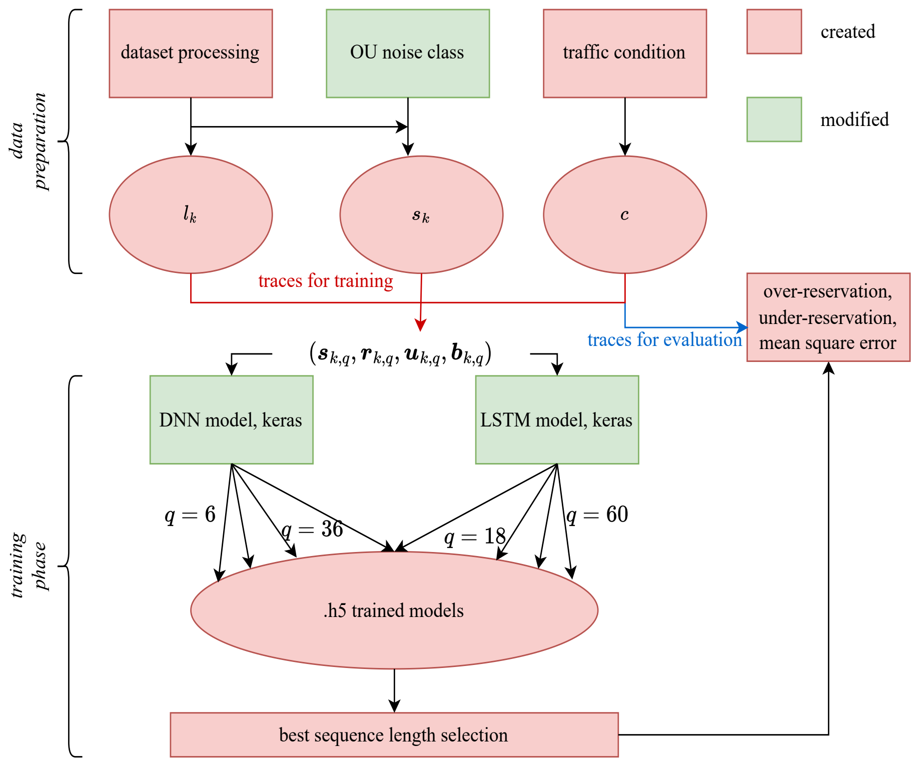


Figure 3.9: Architecture of our solutions

3.5.3 The baseline: ARIMA

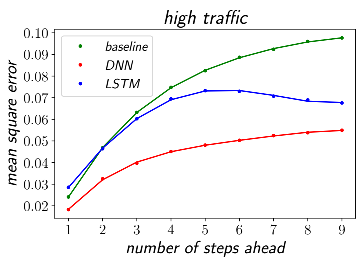
To evaluate the performance of the two proposed solutions, we have constructed a third model based on the classical ARIMA predictor, which we use as a baseline. We have chosen this particular model as it has previously been proven to yield great predictive ability for single-step-ahead predictions [110].

ARIMA does not require training. Instead, we employ two instantiations of the ARIMA model to predict estimates of the MNO traffic and the SP traffic, based on their past traces. The ARIMA model has access to the MNO traffic, meaning that the baseline possesses more information than our two solutions. To this extent, the baseline is idealized.

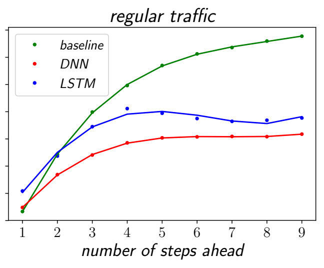
We denote the k-th sequence of SP traffic as $\boldsymbol{s}_{k,q} = [s_{k-q+1}, ..., s_k]$ and the k-th sequence of MNO traffic as $\boldsymbol{l}_{k,q} = [l_{k-q+1}, ..., l_k]$. For both cases, we predict $[s_{k+1}, ..., s_{k+h}]$ and $[l_{k+1}, ..., l_{k+h}]$ and from Lemma 1, obtain $[r_{k+1}, ..., r_{k+h}]$. On a separate development set, we tested different combinations for the choice of our ARIMA coefficients, comparing each based on their MSE. From this, we found the lowest MSE was given by the (3,0,0)-ARIMA model.

3.5.4 Simulation results

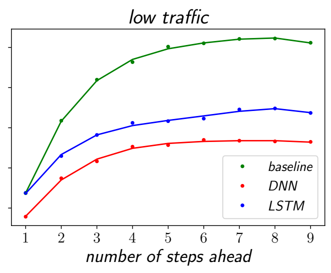
In this subsection, we compare the two proposed solutions against the baseline approach. For that, we use three metrics: the MSE, and the average values of under-reservation and over-reservation. We define them for all future steps $j = 1, \ldots, h$, as we are interested in multi-steps-ahead reservation.



(a) High traffic

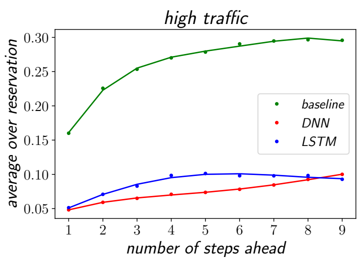


(b) Regular traffic

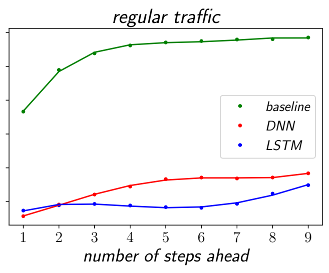


(c) Low traffic

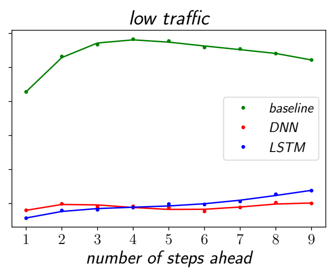
Figure 3.10: MSE comparison under different traffic conditions



(a) High traffic

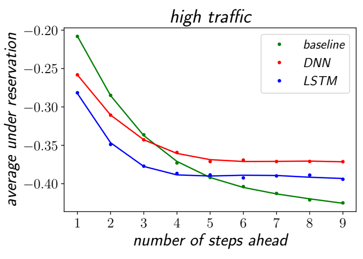


(b) Regular traffic

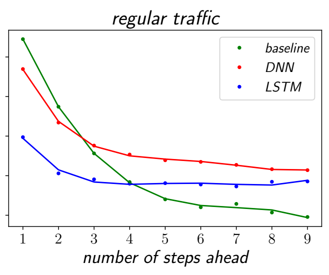


(c) Low traffic

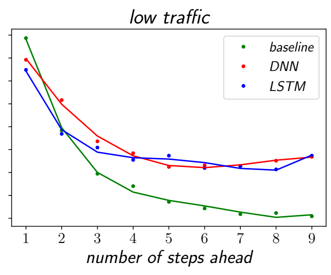
Figure 3.11: Over-reservation comparison under different traffic conditions



(a) High traffic



(b) Regular traffic



(c) Low traffic

Figure 3.12: Under-reservation comparison under different traffic conditions

$$\begin{aligned} MSE_j &= \frac{\sum_{k=1}^{<T_x>-h} |r_{k+j} - r^*_{k+j}|^2}{<T_x> -h} \\ Over_j &= \frac{\sum_{k=1}^{<T_x>-h} (r_{k+j} - r^*_{k+j}) \mathbb{1}\{r_{k+j} > r^*_{k+j}\}}{\sum_{k=1}^{<T_x>-h} \mathbb{1}\{r_{k+j} > r^*_{k+j}\}} \\ Under_j &= \frac{\sum_{k=1}^{<T_x>-h} (r_{k+j} - r^*_{k+j}) \mathbb{1}\{r_{k+j} < r^*_{k+j}\}}{\sum_{k=1}^{<T_x>-h} \mathbb{1}\{r_{k+j} < r^*_{k+j}\}} \end{aligned} \tag{3.2}$$

In Figs. 3.10, 3.11, and 3.12, we compare the DNN and LSTM performance against ARIMA's. After testing different sequence lengths, we have selected the time sequence that yielded the best results for each use case. We recall that our model simulates three types of traffic conditions, low, regular and high. We conduct a 9-step ahead prediction in each of our experiments. This corresponds to one hour and a half of advance reservation.

In Figs. 3.10(a), 3.10(b) and 3.10(c), we examine the quality of the reservation in terms of MSE. Under high traffic conditions, Fig. 3.10(a) shows that the three methods perform similarly for the case of single-step-ahead reservation. However, as the number of decision steps is increased, the learning-based solutions soon begin to outperform the baseline, with the DNN performing slightly better than the LSTM. For the regular traffic case, as shown in 3.10(b), the LSTM initially under-performs the other two models but then gradually outperforms the baseline as the number of steps is increased. Overall, the DNN appears to be the best method when considering the MSE metric.

In Figs. 3.11(a), 3.11(b) and 3.11(c), we consider the over-reservation metric and see that both supervised solutions outperform the baseline by a wide margin. This demonstrates a major gain in the use of the DNN and the LSTM models, indicative of the fact that these learning based solutions were able to successfully extract the information related to the MNO's availability $c - l_k$ from the traffic traces r_k, b_k and u_k.

In Figs. 3.12(a), 3.12(b) and 3.12(c), we focus on the under-reservation case. Generally, a method that demonstrates good performance on the over-reservation case, tends to under-perform in the under-reservation case. As the solutions we propose bring a major improvement in reducing over-reservations, it is also essential that they do not under-perform with respect to under-reservations, as this could lead to a consequent traffic outage and therefore greatly impact the QoS observed by tenants and end users. We can see that the baseline initially outperforms the supervised solutions for single-step-ahead reservations. However, at multi-steps-ahead reservations, the DNN and LSTM methods demonstrate superior performance. For high traffic, as shown in Fig. 3.12(a), the DNN surpasses the baseline at the 4th time-step and the LSTM at the 5th time-step.

We observe that the learning based solutions consistently outperform the baseline solution for multi-steps-ahead reservations. The traffic traces can cycle rapidly from normal demand to extreme peak values. For a linear model such as ARIMA, which only considers the last few hours in its prediction, a sudden change of this nature becomes extremely challenging to predict. In contrast, the DNN and LSTM are able to understand that these fast fluctuations are not noise but are in fact part of the daily/seasonal trends of the data.

3.6 Conclusion & Discussion

In this chapter, we have investigated an important problem within the networking domain, namely the problem of resource reservation within virtualized RAN from an SP's perspective. We have designed solutions that allow the SP to rely on best-effort resources and to reserve dedicated spectrum capacity in-advance thus enabling the isolation of such resources by the NO. We have set up a training phase where we could prepare two supervised solutions and then have evaluated them in an online environment with real-world conditions. We have shown that both DNN and LSTM were able to outperform a baseline with access to the MNO trace. Overall, the DNN has shown the best results. The two supervised solutions have drastically reduced the over-reservation, which motivates the title of this chapter: "a cost-reduction strategy for the SP".

While our current solutions demonstrate good performance against our baseline model, we aim to examine additional factors which may affect our system model. For example, we note that it is possible to introduce various additional leasing agreements into this problem space, such as the real-time and forward markets discussed in [111], wherein the SP could potentially reserve slices on-the-fly or in-advance, like in the cloud marketplaces [71–74]. This aspect appears quite promising, as we foresee the possibility to extend the reservation decision to both fine-grained and course-grained time scales. In addition, we note that such a policy would also require further work to be done in defining a more appropriate cost function related to the trade-off between the cost of future resources and the utility drawn from reserving those resources. In this chapter, we limit the reservation to the bandwidth capacity at the base station. Ideally, we aim to design a reservation policy dealing with multiple kinds of resources, comprising bandwidth at the base stations, backhaul link capacity, processing capacity at the nodes, etc. Only this complete reservation scheme can fully unleash the creation of a network slice. We propose to tackle the following extensions in the next chapter: consider multiple time-scales for the reservations, incorporate the cost of reservation into the optimization problem, extend the reservation to multiple types of resources.

4 No-regret slice reservation solutions in constrained OCO

In this chapter we present the contribution C2 that addresses the research questions RQ1 and RQ3. The work presented in this chapter has been published in the conference paper "No-Regret Slice Reservation Algorithms" presented at the IEEE International Conference on Communications (ICC), in June 2021, and the journal paper "Learning-based Reservation of Virtualized Network Resources" published in IEEE Transactions on Network and Service Management (TNSM), in January 2022.

4.1 Introduction

As discussed in the conclusion of the previous chapter (see 3.6), the scope of the SP reservation model is limited to one type of resource and to one time-scale. So we aim to design a complete reservation scheme whereby the SP can reserve multiple kinds of resources which orchestration will enable the operation of the slice. Our reservation model must account for the utility stemming from the orchestration of the contracted resources and the prices of such resources. This extends our previous model, which would consider only one type of resource and not account for the pricing of the multi-steps ahead reserved resources. We achieve such improvement with the slice orchestration policy presented in this chapter in section 4.5. Our reservation model also must allow the SP to re-configure the slice dynamically with the possible lease of spot resources available on-the-fly. This adds a new contribution of the SP to the slice life-cycle decisions (see Fig. 1.1), not considered in the previous model. The mixed time scale reservation model which we present later in this chapter (section 4.6) allows the SP to reserve additional spot resources, thus enabling the slice re-configuration.

While we extend our system model in the aforementioned directions, we aim to design solutions which do not necessitate an offline training phase and still deliver strong performance guarantees. Online optimization presents a set of useful algorithms that can adapt to the users' needs and pricing decisions of the NO, are robust and practical to different traces. More specifically, the OCO framework introduced by Zinkevich [81] provides strong guarantees of regret and constraint violation, given the feasible set, the objective and constraint functions are convex. We aim to use a primal-dual method [1] which relies on the gradient update at each slot. This type of solution presents low time-complexity and thus is applicable to short timescale reservations.

The contributions of the work presented in this chapter are listed as follows:

1. We introduce a general reservation model for virtualized network resources and formulate this process as a learning problem where a service provider learns to optimally request slices while being oblivious to user needs and network prices.
2. A new suite of online learning algorithms is proposed for slice reservations, that ensures sample-path asymptotically-optimal performance while respecting budget constraints. The algorithms are general enough to tackle scenarios where the SP learns the optimal slice composition or exploits additional pricing information.
3. We analyze the impact of key system parameters on the learning performance, and discuss the implications for the design of such network virtualization markets.
4. A battery of numerical tests is employed, using stationary and non-stationary parameter patterns, to assess the performance of the proposed algorithms. The results verify their robustness and efficacy and reveal the impact of the different system parameters.

To the best of our knowledge, this is the first online learning model for budgeted slice reservation. Our reservation model pursues the same goals as the models presented in [39], [40], [46] and [47], to the difference it considers multiple resource types, makes no assumptions on the pricing policy incumbent upon the NO and allows the dynamic reconfiguration of the slice. We refer the reader to the subsection 2.1.4 and to Table 2.3 for more details. We also develop solutions which present low time complexity and therefore are interesting for fine-grained reservations with small duration time.

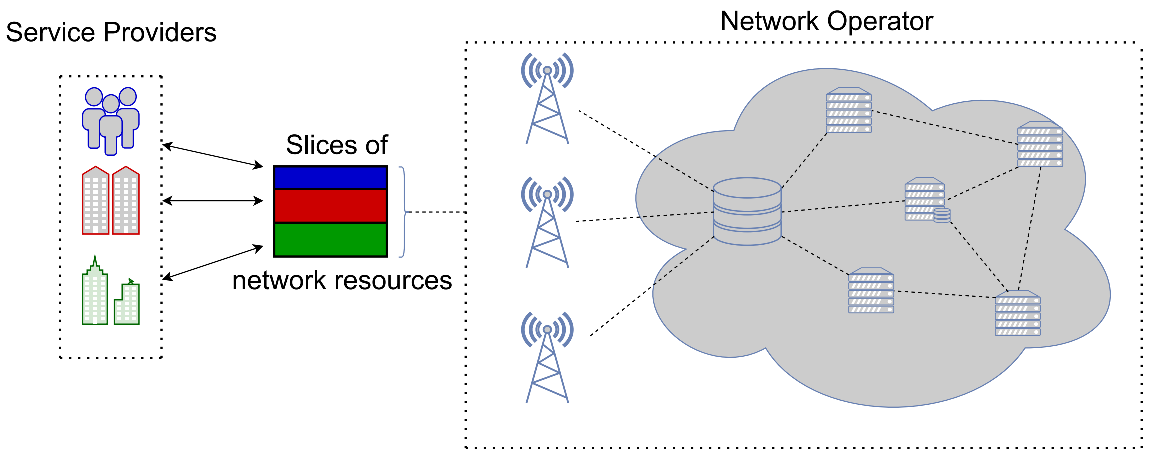


Figure 4.1: A NO leases different types of resources, e.g., wireless capacity, storage capacity and edge computing capacity, to different types of SPs that offer over-the-top services to their users.

This chapter is structured as follows. We open with the description of the new network model comprising multiple kinds of resources and the new market model with resource pricing. Consequently, we define the specific decisions the SP must take under such framework. Then we state the long-term optimization problem of the SP and motivates the design of an online learning solution to address the problem at hand. We also present the benchmark selected to evaluate the performance of our learning policy. The latter we detail in the following section providing performance guarantees. We then present two major extensions, for one of which we give performance guarantees. Finally, we conduct extensive numerical evaluation of our solutions and we analyze the sensitivity of the performance to the parameters of the model. We then conclude the chapter.

4.2 System Model

4.2.1 Network Model

We consider a mixed time-scale model with long periods that are divided into small slots. Namely, each period t includes K slots and we study the system for $t=1,\ldots,T$ periods or, equivalently, for $k=1,\ldots,KT$ slots.[1] This modification allows the SP to reserve at two different time scales, while in the previous chapter the reservation is limited to one time scale. A NO sells virtualized resources to SPs, see Fig. 4.1, and we denote with $\mathcal{H}$ the set

[1]For example, each period can be one day comprising 24 one-hour slots; or, an hour comprising 60 one-minute slots for more fine-grained models.

of $m = |\mathcal{H}|$ types of resources that comprise each slice. For instance $\mathcal{H}$ may include wireless spectrum, backhaul capacity, computing and storage resources $(m = 4)$. We introduce the *bundling* vector $\boldsymbol{\theta} \in \mathbb{R}^m$ which determines the amount of each resource required to build a slice unit. If it is $\boldsymbol{\theta} = (0.5, 0.8, 0.2, 0.1)$ in the above example, a slice unit needs 0.5 units of spectrum, 0.8 units of link capacity, and so on. These values can be normalized or expressed using the actual physical metrics. In Fig. 4.1, multiple SPs can lease from the NO slice units composed of different kinds of network resources. We consider initially $\boldsymbol{\theta}$ to be fixed, but we drop this assumption in Sec. 4.5.

4.2.2 Market Model

The integration of a market model is a major improvement over our previous model, which does not account for the prices of the contracted resources. The market operates using a hybrid model where reservations can be updated at the beginning of each period and the SP can lease (additional) resources at the beginning of each slot in a spot market. We denote with $p_t \in \mathbb{R}_+$ the t-period reservation price per slice unit, and with $q_k \in \mathbb{R}_+$ the spot price for slot k; both announced by the NO. Since $\boldsymbol{\theta}$ is given, these scalar prices suffice to model the slice cost. We also define the vector of spot prices for period t as $\boldsymbol{q}_t = \big(q_k, k \in \mathcal{K}_t\big)$ where $\mathcal{K}_t = \{(t-1)K + 1, \dots, tK\}$. Clearly, prices p_t and $\boldsymbol{q}_t$ vary with time and may change in an unpredictable fashion. For instance, the NO might increase or decrease the prices based on the requests it received in previous periods; or based on the available spot resources which are affected by its own needs. We make no assumptions about these quantities other than being uniformly bounded. Moreover, we assume the NO can impose upper limits on the slice size each SP can lease, and we define the set of feasible reservations $\mathit{\Gamma} = [0, D]$. Such limitations arise due to capacity constraints or when the NO reserves resources for its needs. Note there that if the slice has minimum requirements because it accommodates semi-elastic demand, we redefine the set of feasible reservations as $[d, D]$, where d amounts of resources are necessary to accommodate the inelastic demand.

4.2.3 Reservation Decisions

We focus on one SP and study its slice reservation policy. This consists of the t-period reservation of $x_t \in \mathbb{R}_+$ slice units and the per-slot reservations $y_k \in \mathbb{R}_+$ within t. Note that the actual reserved resources are $x_t\boldsymbol{\theta}$ and $y_k\boldsymbol{\theta}$ respectively, but we drop the bundling vector until Sec. 4.5. At the beginning of each period t the SP decides its t-period reservation plan $(x_t, \boldsymbol{y}_t)$, where $\boldsymbol{y}_t = \big(y_k, k \in \mathcal{K}_t\big)$. The SP's goal is to maximize its service while not exceeding its monetary budget B. The service performance is quantified with a concave *utility* function increasing on the resources and modulated by parameter $a_k \geq 0$ that captures the users' needs in slot k. For example, a_k might represent the total user demand, their willingness to pay, and so on. We also define $\boldsymbol{a}_t = \big(a_k, k \in \mathcal{K}_t\big)$. The concavity of the utility captures the diminishing returns which arise naturally in such systems. [2]

Following the standard practice, we use a concave utility function to model the benefit of the SP from using certain amount of network resources: see [20], [112], [37] and references therein. These papers provide the general form of α-fair utility function, namely:

$$f(\boldsymbol{z}) = \begin{cases} \frac{\boldsymbol{z}^{1-\alpha}}{1-\alpha} & \alpha \neq 1 \\ \log(\boldsymbol{z}) & \alpha = 1 \end{cases} \tag{4.1}$$

In [113], the utility from allocating bandwidth x to a certain network flow f is modeled as $a_f \log(x_f)$, where a_f is a problem (and flow)-specific parameter. We also refer to other resource reservation papers that model the problem as convex [41], [46], [39].

[2]E.g., the data rate is a logarithmic function of the spectrum; the additional revenue of the SPs from more slice resources are typically diminishing, etc.

4.3 Problem Formulation

4.3.1 Convex problem

Putting the above together, the ideal slice reservation policy of the SP for the entire operation of the system is described with the following convex program:

$$(\mathbb{P}): \quad \max_{\{x_t, \boldsymbol{y}_t\}_{t=1}^T} \sum_{t=1}^{T} \sum_{k \in \mathcal{K}_t} a_k \log(x_t + y_k + 1) \tag{4.2}$$

$$\text{s.t.} \quad \sum_{t=1}^{T} \left(x_t p_t + \boldsymbol{y}_t^\top \boldsymbol{q}_t \right) \leq BT, \tag{4.3}$$

$$y_k \in \Gamma, \quad \forall k = 1, \ldots, KT, \tag{4.4}$$

$$x_t \in \Gamma, \quad \forall t = 1, \ldots, T. \tag{4.5}$$

Objective (4.2) is the total utility that the SP achieves with its reservations, after a duration of T periods. Constraint (4.3) captures the total budget constraint of the SP; and (4.4), (4.5) confine the decision variables to a compact convex set that collects upper reservation bounds set by the NO. We assume that the NO always provides the SP with the whole request x_t or y_k, as long as the latter belongs to Γ.

Henceforth, in order to streamline presentation we define the vector functions related to each period t:

$$f_t(x_t, \boldsymbol{y}_t) = - \sum_{k \in \mathcal{K}_t} a_k \log(x_t + y_k + 1), \tag{4.6}$$

$$g_t(x_t, \boldsymbol{y}_t) = x_t p_t + \boldsymbol{y}_t^\top \boldsymbol{q}_t - B. \tag{4.7}$$

and we will be also using vector $\boldsymbol{z}_t = (x_t, \boldsymbol{y}_t) \in \mathcal{Z} \triangleq \Gamma^{K+1}$. Note that when the SP does not reserve resources in a period t, the objective is still well defined and we get $f_t(0, \mathbf{0}) = 0$.

($\mathbb{P}$) is a convex optimization problem but the SP cannot tackle it directly due to the following challenges:

• *(Ch1)*: The user needs $\{a_k\}_k$ are unknown, time-varying and possibly generated by a non-stationary random process.

Table 4.1: Key Notations

Symbol	Physical Meaning
a_k	User needs for the SP service in slot k
p_t	Advance-reservation price of NO for period t
$q_k, \boldsymbol{q}_t$	Spot price on slot k; and spot price vector for period t
m	Number of resource types composing each slice
$\boldsymbol{\theta}_t \in \mathbb{R}^m$	Slice composition vector for period t
B	Monetary budget for the reservations of the SP
K	Period length or number of slots per period
y_k	Reservation in spot market in slot k
x_t	Reservation in advance market in period t
$\mathit{\Gamma}$	$\mathit{\Gamma} = [0, D]$, feasible set for reservations
$\boldsymbol{y}_k$	Vector of reserved instances in spot market in slot k
$\boldsymbol{x}_t$	Vector of reserved instances in advance market in period t

• *(Ch2)*: The spot $\{q_k\}_k$ and reservation prices $\{p_t\}_t$ are unknown, time-varying and unpredictable as they depend on the NO pricing strategy and possibly on other SPs' demand.

• *(Ch3)*: Finally, parameters $\{q_k, a_k\}_k$ are revealed *after* the slice reservation at the respective slot k has been decided.

Due to (*Ch1-2*) ($\mathbb{P}$) cannot be solved at $t = 1$ since the evolution of the system parameters for the next T periods is unknown. Furthermore, it cannot be tackled with standard online optimization techniques as the function parameters are revealed after the reservation decisions are made in each period t (*Ch3*). This renders imperative the design of an online *learning* algorithm that adapts to system and market dynamics, offering guarantees for achieving a *satisfactory* performance.

4.3.2 Benchmark policy

The efficacy of a learning policy is mainly characterized by the benchmark to which we compare its performance; and by the convergence (or, learning rate) at which the policy's performance converges to this benchmark's performance. As it was proved in [87], constrained OCO problems like ($\mathbb{P}$) cannot learn efficiently to perform as benchmarks from set $\mathcal{X}_{max}$ (2.3.2). In light of this result, we settle for a weaker benchmark for our learning algorithm, where the benchmark is selected such that it satisfies each t-period constraint

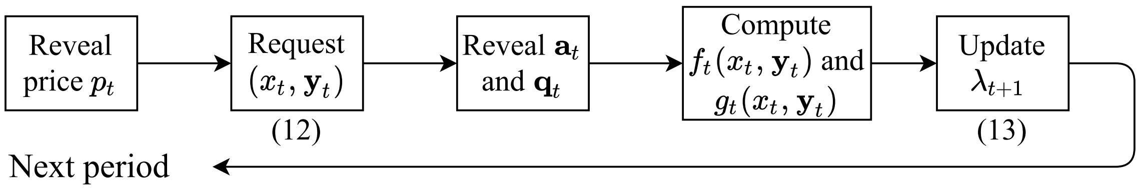

Reveal price p_t
Request $(x_t, \mathbf{y}_t)$
(12)
Reveal $\mathbf{a}_t$ and $\mathbf{q}_t$
Compute $f_t(x_t, \mathbf{y}_t)$ and $g_t(x_t, \mathbf{y}_t)$
Update λ_{t+1}
(13)
Next period

Figure 4.2: Online learning model and sequence of actions in the system.

separately.[3]

In detail, if the SP knew at the beginning of each period t, the price p_t, the demand $\boldsymbol{a}_t$ and the spot prices $\boldsymbol{q}_t$, it could find the optimal t-period decision $\boldsymbol{z}_t^\star = (x_t^\star, \boldsymbol{y}_t^\star)$ by solving:

$$(\mathbb{P}_t): \quad \min_{\{\boldsymbol{z}_t\}\in\Gamma^{K+1}} f_t(\boldsymbol{z}_t) \quad \text{s.t.} \quad g_t(\boldsymbol{z}_t) \leq 0. \tag{4.8}$$

Given that in practice this information is unavailable, our goal is to design an algorithm that finds the t-period reservation $(x_t, \boldsymbol{y}_t)$ in each period t, such that we achieve a good enough performance with respect to $\boldsymbol{z}_t^\star$. Formally, we define the *dynamic regret* and constraint *fit* metrics:

$$R_T = \sum_{t=1}^{T}\Big(f_t(\boldsymbol{z}_t) - f_t(\boldsymbol{z}_t^\star)\Big), \ V_T = \Big[\sum_{t=1}^{T} g_t(\boldsymbol{z}_t)\Big]_+,$$

which quantify respectively how well our policy $\{x_t, \boldsymbol{y}_t\}$ fairs against $\{x_t^\star, \boldsymbol{y}_t^\star\}$ and how much the constraints are violated on average. Note that we project the constraints onto $\mathbb{R}_+$ as we are interested to bound the excessive budget consumption.

Following the terminology in online learning, we state that our reservation algorithm achieves *no-regret* if both quantities grow sublinearly, i.e.,

$$\lim_{T\to\infty} \frac{R_T}{T} = 0, \lim_{T\to\infty} \frac{V_T}{T} = 0, \quad \forall \ \{f_t\}_{t=1}^T. \tag{4.9}$$

It is important to stress that this learning objective is more challenging than the respective *static* regret benchmark where we compare against policy $(x^\star, \boldsymbol{y}^\star)$ which is designed with hindsight but remains fixed across time. In other words it holds $R_T^s \leq R_T$ and by achieving $R_T = o(T)$ we ensure the same for R_T^s. We refer the interested reader to [1, 86] for a detailed discussion about benchmarks.

[3]In other words, a benchmark in $x^\star \in \mathcal{X}_{max}$ offers more degrees of freedom, hence potentially higher objective values that the learning policy cannot achieve without violating the constraints.

4.4 Online Reservation Policy

4.4.1 Online Learning for Reservations (OLR)

We proceed with the design of the online learning algorithm that implements the SP reservation policy. In detail, we define the Lagrangian:

$$\widetilde{L}_t(\boldsymbol{z}, \lambda) = f_t(\boldsymbol{z}) + \lambda g_t(\boldsymbol{z}) \tag{4.10}$$

where $\lambda \in \mathbb{R}_+$ is the Lagrange multiplier associated with constraint (4.7). It is easy to see that, for all t and $\lambda \in \mathbb{R}_+$, $\widetilde{L}_t$ is convex over $\boldsymbol{z}$, as it is the sum of a convex function f_t and an affine function g_t. Similarly, $\widetilde{L}_t$ is affine hence concave over λ. Therefore, we opt for a saddle-point methodology where we minimize the Lagrangian over $\boldsymbol{z}$ and then maximize it over λ. Instead of the online Lagrangian in (4.10), we optimize the modified Lagrangian with a linearized objective and a Euclidean regularizer with non-negative parameter ν:

$$L_t(\boldsymbol{z}, \lambda) = \nabla f_t(\boldsymbol{z}_t)^\top (\boldsymbol{z} - \boldsymbol{z}_t) + \lambda g_t(\boldsymbol{z}) + \frac{\|\boldsymbol{z} - \boldsymbol{z}_t\|^2}{2\nu}, \tag{4.11}$$

where note that in the first-order approximation of $f_t(\boldsymbol{z})$ we omit the term $f_t(\boldsymbol{z}_t)$ as it does not depend on either $\boldsymbol{z}$ or λ.

We can now describe our learning policy – please refer to Algorithm OLR and Fig. 4.2. At the beginning of each period t, the NO reveals the current reservation price p_t (step 2). Then, the SP decides its reservation policy $\boldsymbol{z}_t = (x_t, \boldsymbol{y}_t)$ by performing a primal update as follows (step 3):

$$\boldsymbol{z}_t = \arg\min_{\boldsymbol{z} \in \mathcal{Z}} L_{t-1}(\boldsymbol{z}, \lambda_t). \tag{4.12}$$

As explained above, the SP makes this decision while it does not have access to f_t or g_t, as is also evident from the time index of the Lagrangian in (4.12). After $\boldsymbol{z}_t$ is fixed, the demand and prices are revealed (steps 4-5) and the SP measures the performance $f_t(\boldsymbol{z}_t)$ and cost $g_t(\boldsymbol{z}_t)$. At the end of the period, the SP has access to the current Lagrangian (4.11), and

Algorithm OLR: Online Learning for Reservation

```
  Initialize:
  $\lambda_1 = 0, x_0 \in \Gamma, \boldsymbol{y}_0 \in \Gamma^K$, $\nu = \mu = \mathcal{O}(T^{-1/3})$
1 for $t = 1, \dots, T$ do
2     Observe the $t$-period price $p_t$
3     Decide $(x_t, \boldsymbol{y}_t)$ by solving (4.12)
4     Observe $\boldsymbol{a}_t$ and calculate $f_t(x_t, \boldsymbol{y}_t)$
5     Observe $\boldsymbol{q}_t$ and calculate $g_t(x_t, \boldsymbol{y}_t)$
6     Decide $\lambda_{t+1}$ by solving (4.13)
```

can update its dual variable by executing the dual gradient ascent with positive step-size μ:

$$\lambda_{t+1} = \big[\lambda_t + \mu \nabla_\lambda L_t(\boldsymbol{z}_t, \lambda)\big]_+ \tag{4.13}$$

These are the dual variables (or, shadow prices for (4.3)) that will assist the SP to select the new reservation bid. It is worth stressing that OLR is also applicable for markets where the advance-reservation prices p_t are revealed after step 3; this will become clear in the sequel.

4.4.2 Performance analysis

We start with the necessary model assumptions.

Assumption 4. *The NO prices are uniformly bounded, i.e., $p_t \in [0, p], \forall t$ and $q_k \in [0, q], \forall k$, where $p, q < \infty$.*

Assumption 5. *The utility parameters are uniformly bounded, i.e., $a_k \in [0, a]$, $\forall k$ where $a < \infty$.*

Assumption 6. *The set $\mathcal{Z} = \Gamma^{K+1}$ has bounded diameter, i.e., $E \triangleq D\sqrt{K+1}$.*

Moreover, we need to characterize the *variability* of the problem, i.e., how fast the constraints and the dynamic benchmark can change among successive periods. First, we define:

$$U_z = \sum_{t=1}^{T} \|\boldsymbol{z}_t^\star - \boldsymbol{z}_{t-1}^\star\|, \;\; U_g = \sum_{t=1}^{T} \max_{\boldsymbol{z} \in \mathcal{Z}} [g_t(\boldsymbol{z}) - g_{t-1}(\boldsymbol{z})]_+,$$

which measure this property for each problem realization. We define as well:

$$\widetilde{U}_g = \max_t \max_{\boldsymbol{z}\in\mathcal{Z}} [g_t(\boldsymbol{z}) - g_{t-1}(\boldsymbol{z})]_+$$

We see from (4.7) that $\widetilde{U}_g \leq D(p+Kq)$. However, in practice we expect $\widetilde{U}_g$ to be smaller as it is not reasonable for the NO (i.e., practical or acceptable from a regulatory perspective) to vary so drastically its pricing strategy in successive periods. As it will become clear below, $\widetilde{U}_g$ determines whether the SP can learn an efficient reservation policy.

Finally, we assume that all problem instances $(\mathbb{P}_t), \forall t$ admit a Slater vector, i.e. there is a vector $\tilde{\boldsymbol{z}} \in \mathcal{Z}$ such that:

$$g_t(\tilde{\boldsymbol{z}}) \leq -\epsilon, \ \forall t, \quad \text{with } \epsilon > 0. \tag{4.14}$$

The Slater constraint qualification, or weak Slater condition, is sufficient for strong duality [114], and is required for devising the algorithm's performance bounds. Moreover, it is related to the problem variability and specifically the following condition is required.

Assumption 7. ***strong Slater condition***. *The slack constant ϵ in the Slater condition* (4.14) *is such that it holds $\epsilon > \widetilde{U}_g$.*

The next Lemma characterizes the performance of OLR.

Lemma 2. *Under Assumptions 4-7, Algorithm OLR achieves the following regret and constraint violation bounds w.r.t. the dynamic benchmark policy $\{x_t^\star, \boldsymbol{y}_t^\star\}_{t=1}^T$:*

$$R_T \leq \frac{EU_z}{\nu} + \frac{\nu T G^2}{2} + \frac{(T+1)\mu M^2}{2} + \frac{E^2}{2\nu} + U_g\tilde{\lambda}$$

$$V_T \leq M + \frac{(2EG/\mu) + (E^2/2\nu\mu) + (M^2/2)}{\epsilon - \widetilde{U}_g},$$

$$\textit{where: } M \triangleq \max\{D(p+Kq) - B, B\}, \ \ G \triangleq a\sqrt{K(K+1)}$$

$$\tilde{\lambda} \triangleq \mu M + \frac{2EG + (E^2/2\nu) + (\mu M^2/2)}{\epsilon - \widetilde{U}_g}$$

4.4.3 Discussion

Complexity analsyis OLR. We stress there that the computational cost and memory requirements of the OLR algorithm are fairly low. At each period t, we need to solve the two sub-problems (4.12) and (4.13). The constraint function $g_t(\boldsymbol{z})$ being linear, the computational complexity of (4.12) is low and a closed form solution can be derived via the First Order Condition, as in (4.15) and (4.16). At each period t, we need to store the vectors $\boldsymbol{a}_{t-1}$, $\boldsymbol{y}_{t-1}$, $\boldsymbol{q}_{t-1}$ of length K and the scalars x_{t-1}, p_t and λ_t. Therefore, memory requirements are of $3K+3=\mathcal{O}(K)$. The solution to (4.15) runs in K operations (sum over $\mathcal{K}_{t-1}$) and the solution to (4.16) runs in 1 operation, for each $k \in \mathcal{K}_t$. Therefore, the running time for (4.12) is of $2K=\mathcal{O}(K)$. (4.13) only needs $K+1$ operations to compute $g_t(\boldsymbol{z}_t)$, hence runs in $\mathcal{O}(K)$.

It is important at this point to discuss the practical implications of this theoretical result for the problem at hand. First, note that a key ingredient of Algorithm OLR are the step sizes (or, *learning rates*) ν and μ. From Corollary 1 of [1], if these rates are selected such that:

$$\nu = \mu = \max\left\{\sqrt{\frac{U_z}{T}}, \sqrt{\frac{U_g}{T}}\right\},$$

then we obtain the following growth rate for the regret:

$$R_T = \mathcal{O}\left(\max\left\{\sqrt{U_z T}, \sqrt{U_g T}\right\}\right).$$

In order to asymptotically zeroize the average regret we need $R_T = o(T)$, and this condition is satisfied if $\max\{U_z, U_g\} = o(T)$, i.e., when the maximum variability of the benchmark and constraints across successive slots remains sublinear. When this condition cannot be verified in advance, one can select $\nu = \mu = T^{-1/3}$ to enforce:

$$R_T = \mathcal{O}\left(\max\left\{T^{\frac{1}{3}}U_g, T^{\frac{1}{3}}U_z, T^{\frac{2}{3}}\right\}\right), \quad V_T = \mathcal{O}\left(T^{\frac{2}{3}}\right).$$

Furthermore, the SP can adapt the steps ν and μ to the specific scenario and its priorities, namely the relative importance of achieving high performance or fast compliance with the average budget constraint. For instance, a “negative fit - high regret” situation means that

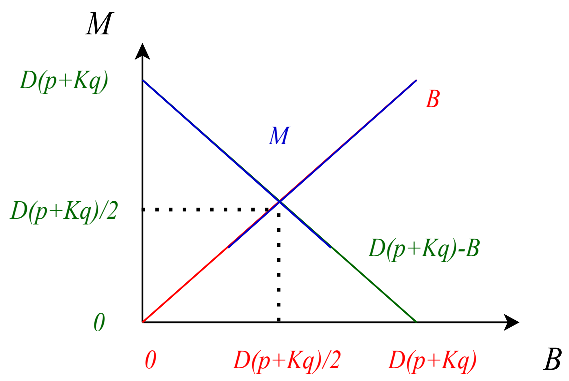


Figure 4.3: Impact on the committed budget, B, on the value of parameter M that affects the bounds of R_T and V_T.

the SP under-reserves and this can be rectified by tuning the steps.

It it also interesting to note that, for the specific performance and cost functions, we can obtain a closed form solution for the advance reservation decisions in each period, that reveal the intuition of this mechanism. Namely, we can use the First Order Condition for (4.12) and write:

$$x_t = \sum_{k \in \mathcal{K}_{t-1}} \frac{\nu a_k}{1 + x_{t-1} + y_k} - p_t \lambda_t \nu + x_{t-1}, \tag{4.15}$$

while ensuring that $x_t \in [0, D]$. The following important remarks stem from the above expression:

- A big ratio of demand over reservations in period $t-1$ favors a large reservation at period t;
- A large constraint violation in period $t-1$ favors a small reservation at period t;
- A large value of μ accentuates the effect of the violation if it is non-negative (more conservative reservations);
- A large value of ν favorites a large distance $|x_t - x_{t-1}|$.
- Conversely, a small ν reduces the distance $|x_t - x_{t-1}|$.

Similarly, we can obtain an analytical expression for the intra-period reservations:

$$y_k = \frac{\nu a_{k-K}}{1 + x_{t-1} + y_{k-K}} - q_{k-K} \lambda_t \nu + y_{k-K}, \quad k \in \mathcal{K}_t, \tag{4.16}$$

with $y_k \in [0, D]$, that allows us to understand how the learning rates, resource prices and reservations in the previous respective slots (index $k-K$) affect the reservations at slot k.

These remarks provide the SP with guidelines on how to adapt the step sizes ν and μ to its preferable criterion (performance or cost) but also to the specific market conditions.

Regarding the latter, it is crucial to observe that both the regret and constraint violation depend on the variability of NO's pricing. When the operator changes abruptly the prices among successive periods, i.e., in a superlinear fashion $U_g = O(T^{\beta})$ with $\beta > 1$, it is challenging for the SP to learn an optimal reservation strategy. The same holds for the needs of SP. For instance, if parameters $\{\boldsymbol{a}_t\}$ change so drastically that U_z grows superlinearly, then R_T/T might not converge. Observe also that the period length (number of slots K) affects directly both R_T and V_T. This is rather expected as the SP makes bidding decisions only at the beginning of each period. Hence, for larger K values the bidding depends on more unknown information – or, equivalently, the SP needs to learn more information. Furthermore, we can see that the relation of maximum prices, p and q, to the per-period budget B, affect through parameter M the bounds. These observations reveal the key market factors that shape the learning capability of the SP, and pave the road for possible regulatory interventions, or mutually-agreed guidelines among the SP and NO so as to increase the market efficiency.

Finally, we note that the SP can choose its budget so as to minimize parameter M, which will consequently shrink the upper bounds on the regret and fit. In Fig. 4.3, we plot the function $M = f(B) = \max\{D(p + Kq) - B, B\}$ and observe that the minimum $M = D(p + Kq)/2$ is attained at $B = D(p + Kq)/2$. We verify the effect of B on the performance of OLR in the numerical examples presented in Sec. 4.7.

4.5 Slice Orchestration and Reservation Policy

We consider next the case the SP decides the slice composition, i.e., the amount of each different type of resource, where the benefit from each resource can be time-varying and unknown. Consider, e.g., an SP that is unaware of the optimal computation, storage and bandwidth mix, as this depends on the type of user requests. We extend OLR to account for these decisions and discuss the new performance bounds.

4.5.1 Online Learning Reservations for Slice Orchestration

In this scenario, the SP makes multi-dimensional reservations using $\boldsymbol{x}_t, \boldsymbol{y}_k \in \mathit{\Gamma}_1 \times \ldots \times \mathit{\Gamma}_m$, where m is the number of resources comprising the slice. With a slight abuse of notation we define $\boldsymbol{y}_k = (y_{kj}, j = 1, \ldots, m)$, and $\boldsymbol{y}_t = (\boldsymbol{y}_k, k \in \mathcal{K}_t)$, where $\boldsymbol{y}_t \in \mathbb{R}^{mK}$. The benefit from each reservation $\boldsymbol{x}_t$ is quantified by the scalar $\boldsymbol{\theta}_t^\top \boldsymbol{x}_t$ (respectively, $\boldsymbol{\theta}_t^\top \boldsymbol{y}_k$), where the elements of $\boldsymbol{\theta}_t \in \mathbb{R}^m$ measure the contribution of each resource on performance. And, we allow this vector to change with time and be unknown when the reservations are decided. Similarly, the NO can charge a different (advance or spot) price for each type of resource, hence we define $\boldsymbol{p}_t = (p_{tj}, j = 1, \ldots, m)$ and $\boldsymbol{q}_k = (q_{kj}, j = 1, \ldots, m)$.

We first introduce the new slice-composition and reservation objective and cost functions:

$$f_t^\theta(\boldsymbol{x}_t, \boldsymbol{y}_t) = -\sum_{k \in K_t} a_k \log(\boldsymbol{\theta}_t^\top \boldsymbol{x}_t + \boldsymbol{\theta}_t^\top \boldsymbol{y}_k + 1), \tag{4.17}$$

$$g_t^\theta(\boldsymbol{x}_t, \boldsymbol{y}_t) = \boldsymbol{x}_t^\top \boldsymbol{p}_t + \sum_{k \in K_t} \boldsymbol{y}_k^\top \boldsymbol{q}_k - B, \tag{4.18}$$

and use the following modified Lagrange function:

$$L_t^\theta(\boldsymbol{z}, \lambda) = \nabla f_t^\theta(\boldsymbol{z}_t)^\top (\boldsymbol{z} - \boldsymbol{z}_t) + \lambda g_t^\theta(\boldsymbol{z}) + \frac{\|\boldsymbol{z} - \boldsymbol{z}_t\|^2}{2\nu}, \tag{4.19}$$

to update the primal and dual variables:

$$\boldsymbol{z}_t = \arg\min_{\boldsymbol{z} \in \mathcal{Z}_\theta} L_{t-1}^\theta(\boldsymbol{z}, \lambda_t), \tag{4.20}$$

$$\lambda_{t+1} = [\lambda_t + \mu \nabla_\lambda L_t^\theta(\boldsymbol{z}_t, \lambda)]_+. \tag{4.21}$$

Note that the prices are now vectors in $\mathbb{R}^m$, and the SP decision space is expanded to $\mathcal{Z}_\theta = (\mathit{\Gamma}_1 \times \ldots \times \mathit{\Gamma}_m)^{K+1}$, with $\mathit{\Gamma}_i = [0, D_i], i \leq m$.

The learning policy is implemented by running Algorithm OLR-SO. At the beginning of each period t, the NO reveals the current reservation price vector $\boldsymbol{p}_t \in \mathbb{R}^m$ (step 2). Then, the SP decides its reservation policy $\boldsymbol{z}_t = (\boldsymbol{x}_t, \boldsymbol{y}_t) \in \mathcal{Z}_\theta$ by solving (4.20). Subsequently, the demand $\boldsymbol{a}_t \in \mathbb{R}^K$, the contribution vector $\boldsymbol{\theta}_t \in \mathbb{R}^m$ and the spot prices $\boldsymbol{q}_k \in \mathbb{R}^m$ are revealed (steps 4-5), and the SP measures the performance $f_t^\theta(\boldsymbol{z}_t)$ and cost $g_t^\theta(\boldsymbol{z}_t)$. At the end of

Algorithm OLR-SO: OLR for Slice Orchestration

```
   Initialize:
   $\lambda_1 = 0, \boldsymbol{x}_0 \in \prod_{i=1}^m \Gamma_i, \boldsymbol{y}_0 \in (\prod_{i=1}^m \Gamma_i)^K$, $\nu = \mu = \mathcal{O}(T^{-1/3})$
1  for $t = 1, \dots, T$ do
2      Observe the $t$-period price $\boldsymbol{p}_t$
3      Decide $(\boldsymbol{x}_t, \boldsymbol{y}_t)$ by solving (4.20)
4      Observe $\boldsymbol{a}_t$, $\boldsymbol{\theta}_t$ and calculate $f_t^\theta(\boldsymbol{x}_t, \boldsymbol{y}_t)$ with (4.17)
5      Observe $\boldsymbol{q}_t$ and calculate $g_t^\theta(\boldsymbol{x}_t, \boldsymbol{y}_t)$ with (4.18)
6      Decide $\lambda_{t+1}$ by solving (4.21)
```

the period the SP gains access to the new Lagrangian (4.19) and updates its dual variable (4.21).

4.5.2 Performance Analysis

The performance of Algorithm OLR-SO is conditioned upon the following minimal assumptions that complement, or update, Assumptions 4-7 that we stated for OLR in Section 4.4:

Assumption 8. *The NO prices are uniformly bounded $\boldsymbol{p}_t \in \prod_{i=1}^m [0, p_i], \forall t$ and $\boldsymbol{q}_k \in \prod_{i=1}^m [0, q_i], \forall k$. Without loss of generality, we write $\boldsymbol{p}_t \in [0, p]^m, \forall t$ and $\boldsymbol{q}_k \in [0, q]^m, \forall k$, where $p = \max_i\{p_i\}$, $q = \max_i\{q_i\}$.*

Assumption 9. *The contribution parameters are uniformly bounded $\boldsymbol{\theta}_t \in \prod_{i=1}^m [0, \theta_i], \forall t$. Without loss of generality, we can write $\boldsymbol{\theta}_t \in [0, \theta]^m, \forall t$, where $\theta = \max_i\{\theta_i\}$*

Assumption 10. *The set $\mathcal{Z}_\theta = (\Gamma_1 \times \ldots \times \Gamma_m)^{K+1}$ has bounded diameter $F = \sqrt{K+1}\sqrt{D_1^2 + \ldots + D_m^2}$.*

The variability of this new problem can be defined using the following metrics:

$$U_z^\theta = \sum_{t=1}^T \|\boldsymbol{z}_t^\star - \boldsymbol{z}_{t-1}^\star\|, \quad U_g^\theta = \sum_{t=1}^T \max_{\boldsymbol{z} \in \mathcal{Z}_\theta} [g_t^\theta(\boldsymbol{z}) - g_{t-1}^\theta(\boldsymbol{z})]_+,$$

and we also define:

$$\widetilde{U}_g^\theta = \max_t \max_{\boldsymbol{z} \in \mathcal{Z}_\theta} [g_t^\theta(\boldsymbol{z}) - g_{t-1}^\theta(\boldsymbol{z})]_+.$$

We see from (4.18) that it holds $\widetilde{U}_g^\theta \leq (p + Kq) \sum_{i=1}^m D_i$.

We assume that the new problem:

$$(\mathbb{P}_t^\theta): \quad \min_{\boldsymbol{z}_t \in \mathcal{Z}_\theta} f_t^\theta(\boldsymbol{z}_t) \quad \text{s.t.} \quad g_t^\theta(\boldsymbol{z}_t) \leq 0. \tag{4.22}$$

admits a Slater vector $\tilde{\boldsymbol{z}} \in \mathcal{Z}_\theta$ where:

$$g_t(\tilde{\boldsymbol{z}}) \leq -\epsilon, \ \forall t, \tag{4.23}$$

for some $\epsilon > 0$, and which, as previously, is related to the problem's variability through the following condition:

Assumption 11. ***strong Slater condition****. The slack constant ϵ in the Slater condition* (4.23) *is such that it holds $\epsilon > \widetilde{U}_g^\theta$.*

The performance of OLR-SO is characterized next.

Lemma 3. *Under Assumptions 8-11, Algorithm OLR-SO achieves the following regret and constraint violation bounds w.r.t. the dynamic benchmark policy $\{\boldsymbol{x}_t^\star, \boldsymbol{y}_t^\star\}_{t=1}^T$:*

$$\begin{aligned}
&R_T \leq \frac{FU_z^\theta}{\nu} + \frac{\nu T G_\theta^2}{2} + \frac{(T+1)\mu M_\theta^2}{2} + \frac{F^2}{2\nu} + U_g^\theta \tilde{\lambda}_\theta \\
&V_T \leq M_\theta + \frac{(2FG_\theta/\mu) + (F^2/2\nu\mu) + (M_\theta^2/2)}{\epsilon - \widetilde{U}_g^\theta}, \\
&\textit{where: } M_\theta \triangleq \max\Big\{(p + Kq)\sum_{i=1}^m D_i - B, B\Big\}, \\
&\qquad G_\theta \triangleq a\theta\sqrt{mK(K+1)}, \\
&\qquad \tilde{\lambda}_\theta \triangleq \mu M_\theta + \frac{2FG_\theta + (F^2/2\nu) + (\mu M_\theta^2/2)}{\epsilon - \widetilde{U}_g^\theta}
\end{aligned}$$

4.5.3 Discussion

Complexity analysis OLR-SO. Here we discuss the complexity of the OLR-SO algorithm. Deriving the new Lagrangian (4.19) with regard to $\boldsymbol{z}$ is more challenging. The required running time to compute (4.20) is of $4Km^2 + m(K+1) + m(K+1) = \mathcal{O}(Km^2)$. The dominating term $4Km^2$ comes from the expression of the gradient $\nabla f_t^\theta(\boldsymbol{z}_t)$, that the reader can find in the Appendix B. The solution of (4.21) is much simpler, necessitating $m(K+$

$1) = \mathcal{O}(Km)$ time to compute $g_t^{\theta}(\boldsymbol{z}_t)$. As we need to store the vectors $\boldsymbol{a}_{t-1}$, $\boldsymbol{\theta}_{t-1}$, $\boldsymbol{x}_{t-1}$, $\boldsymbol{y}_k, k \in \mathcal{K}_{t-1}$, λ_t, $\boldsymbol{p}_{t-1}$ and $\boldsymbol{q}_k, k \in \mathcal{K}_{t-1}$, the memory requirements are of $K + m + m + Km + 1 + m + Km = \mathcal{O}(mK)$. As the OLR-SO algorithm increases complexity, it still lies in polynomial time.

The multi-dimensional problem has a higher variability on the constraints ($U_g^{\theta} \geq U_g$) and on the dynamic benchmark sequence ($U_z^{\theta} \geq U_z$). This represents an intuitive result since we deal: with larger reservations; vectors for the pricing scheme instead of scalars; and new unknown contribution parameter $\boldsymbol{\theta}_t$. In fact, we can quantify the effect of resource types m on the variability for the special – but important– case the prices $\{p_t\}_t, \{q_k\}_k$ are i.i.d. with uniform distribution in $[0, p]$ and $[0, q]$, respectively.

Lemma 4. *For sufficiently large value of T, it holds:*

$$U_g = \frac{TD(p + Kq)}{6}, \quad U_g^{\theta} = \frac{T(\sum_{i=1}^{m} D_i)(p + Kq)}{6}.$$

This result reveals that as we consider scenarios with larger m, i.e., more resource types comprising the slice, the respective variability parameter that affects the regret bounds increase proportionally to the diameter of $\mathcal{Z}_{\theta}$, i.e., depend both on the value of m and the respective upper bounds $D_i, i \leq m$.

Finally, it is interesting to note that parameters $\boldsymbol{p}_t$ and $\boldsymbol{\theta}_t$ can be revealed before or after the reservation $\boldsymbol{z}_t$, as the bounds in Lemma 2 hold whether the SP knows them or not. There, we notice that the contribution parameter $\boldsymbol{\theta}_t$ has more influence in the Lagrangian than the price $\boldsymbol{p}_t$. Indeed, $\boldsymbol{\theta}_t$ is present in the $K + 1$ scalar products derived from the gradient term, while $\boldsymbol{p}_t$ is present in only one scalar product: $\boldsymbol{p}_t^{\top}\boldsymbol{x}$. Therefore, the SP would rather know in advance the contributions of each resource to its utility than the on-demand price $\boldsymbol{p}_t$.

4.6 Mixed Time Scale Reservation Policy

Our second extension is a mixed time scale (MTS) reservation model, where the SP can update the slot reservations $\boldsymbol{y}_t$ of each period t, based on the demand $\boldsymbol{a}_t$ and spot prices $\boldsymbol{q}_t$

```
Algorithm OLR-MTS: OLR for Mixed Time Scale
   Initialize:
   $\lambda_1 = 0, x_0 \in \Gamma, \boldsymbol{y}_0 \in \Gamma^K$, $\nu = \mu = \mathcal{O}(T^{-1/3}), \widehat{\nu} = \nu, \widehat{\mu} = \mu$
 1 for $t = 1, \dots, T$ do
 2     Observe the $t$-period price $p_t$
 3     Decide $(x_t, \boldsymbol{y}_t)$ by solving (4.12)
 4     Calculate $B_t = (B - x_t p_t)/K$
 5     for $k = 1, \dots, K$ do
 6         Decide $y_k$ by solving (4.25) and replace $k$-th element of vector $\boldsymbol{y}_t$ by $y_k$
 7         Observe $a_k$, $q_k$
 8         Decide $\lambda_{k+1}$ by solving (4.26)
 9     Observe $\boldsymbol{a}_t$ and calculate $f_t(x_t, \boldsymbol{y}_t)$
10     Observe $\boldsymbol{q}_t$ and calculate $g_t(x_t, \boldsymbol{y}_t)$
```

it observes during that period. We first present the baseline scenario and then extend MTS for the slice composition case.

4.6.1 Online Learning for Mixed Time Scale Reservations

The first thing to note is that functions f_k and g_k for this slot-decision instance, are now:

$$f_k(y_k) = -a_k \log(x_t + y_k + 1), \quad g_k(y_k) = y_k q_k - B_t,$$

where x_t is treated as a parameter as it has been fixed in the beginning of t, and we have defined $B_t = (B - x_t p_t)/K$. Also, the t-slot Lagrangian is:

$$L_k(y, \widehat{\lambda}) = \nabla f_k(y_k)(y - y_k) + \widehat{\lambda} g_k(y) + \frac{(y - y_k)^2}{2\widehat{\nu}} \tag{4.24}$$

where $\widehat{\lambda} \in \mathbb{R}_+$ is the new dual variable.

The learning policy is summarized in Algorithm OLR-MTS. At the beginning of period t, the NO reveals the reservation price $p_t \in [0, p]$ (step 2). Then, the SP decides its reservation policy $\boldsymbol{z}_t = (x_t, \boldsymbol{y}_t)$ by solving (4.12) (step 3), where $\mathcal{Z} = \Gamma^{K+1}$. After x_t is fixed, the SP updates the slot decisions $y_k, \forall k \in \mathcal{K}_t$ (step 6), by executing:

$$y_k = \arg\min_{y \in \Gamma} L_{k-1}(y, \widehat{\lambda}_k). \tag{4.25}$$

Algorithm OLR-MTS-SO: OLR-MTS for Slice Orchestration

```
   Initialize:
   $\lambda_1 = 0, \boldsymbol{x}_0 \in \prod_{i=1}^m \Gamma_i, \boldsymbol{y}_0 \in (\prod_{i=1}^m \Gamma_i)^K$, $\nu = \mu = \mathcal{O}(T^{-1/3}), \widehat{\nu} = \nu, \widehat{\mu} = \mu$
1  for $t = 1, \dots, T$ do
2      Observe the $t$-period price $\boldsymbol{p}_t$
3      Decide $(\boldsymbol{x}_t, \boldsymbol{y}_t)$ by solving (4.20)
4      Calculate $B_t = (B - \boldsymbol{x}_t^\top \boldsymbol{p}_t)/K$
5      for $k = 1, \dots, K$ do
6          Decide $\boldsymbol{y}_k$ by solving (4.29) and replace $k$-th element of vector $\boldsymbol{y}_t$ by $\boldsymbol{y}_k$
7          Observe $a_k$, $\boldsymbol{q}_k$
8          Decide $\lambda_{k+1}$ by solving (4.30)
9      Observe $\{a_k\}_{k \in \mathcal{K}_t}$, $\boldsymbol{\theta}_t$ and calculate $f_t^\theta(\boldsymbol{x}_t, \boldsymbol{y}_t)$
10     Observe $\{\boldsymbol{q}_k\}_{k \in \mathcal{K}_t}$ and calculate $g_t^\theta(\boldsymbol{x}_t, \boldsymbol{y}_t)$
```

Then the demand a_k and the price q_k are revealed (step 7). At the end of the slot, the SP has access to the current Lagrangian (4.24) and can update its dual variable (step 8) by executing:

$$\widehat{\lambda}_{k+1} = [\widehat{\lambda}_k + \widehat{\mu} \nabla_\lambda L_k(y_k, \widehat{\lambda})]_+. \tag{4.26}$$

At the end of the period, the SP has observed all the demand $\boldsymbol{a}_t$ and price $\boldsymbol{q}_t$ parameters, and therefore can calculate the values $f_t(x_t, \boldsymbol{y}_t)$ and $g_t(x_t, \boldsymbol{y}_t)$ (steps 9 and 10). Parameters $\widehat{\nu}$ and $\widehat{\mu}$ are the steps for the intra-period decisions, and we set them to $\widehat{\nu}=\nu$ and $\widehat{\mu}=\mu$.

4.6.2 Online Learning for MTS with Slice Orchestration

Here, we combine the slice orchestration and the mixed time scale reservation models, where the SP can update the slot reservations $\boldsymbol{y}_k$'s of the period t, based on the demand a_k's and spot prices $\boldsymbol{q}_k$'s it observes during that period. The functions f_k^θ and g_k^θ for this slot-decision instance, are:

$$\begin{aligned} f_k^\theta(\boldsymbol{y}_k) &= -a_k \log(\boldsymbol{x}_t^\top \boldsymbol{\theta}_t + \boldsymbol{y}_k^\top \boldsymbol{\theta}_t + 1), \\ g_k^\theta(\boldsymbol{y}_k) &= \boldsymbol{y}_k^\top \boldsymbol{q}_k - B_t, \end{aligned} \tag{4.27}$$

where, again, $\boldsymbol{x}_t$ is a parameter and we define $B_t = (B - \boldsymbol{x}_t^\top \boldsymbol{p}_t)/K$. The t-slot Lagrangian is:

$$L_k^\theta(\boldsymbol{y}, \widehat{\lambda}) = \nabla f_k^\theta(\boldsymbol{y}_k)^\top (\boldsymbol{y} - \boldsymbol{y}_k) + \widehat{\lambda} g_k^\theta(\boldsymbol{y}) + \frac{||\boldsymbol{y} - \boldsymbol{y}_k||^2}{2\widehat{\nu}} \tag{4.28}$$

Then, the SP updates its reservation $\boldsymbol{y}_k$ for each slot k, and its dual variable after observing a_k and $\boldsymbol{q}_k$, by executing:

$$\boldsymbol{y}_k = \arg \min_{\boldsymbol{y} \in \prod_{i=1}^m \Gamma_i} L_{k-1}^\theta(\boldsymbol{y}, \widehat{\lambda}_k), \tag{4.29}$$

$$\widehat{\lambda}_{k+1} = [\widehat{\lambda_k} + \widehat{\mu} \nabla_\lambda L_k^\theta(\boldsymbol{y}_k, \widehat{\lambda})]_+. \tag{4.30}$$

At the end of the period, the SP has observed all demand $\{a_k\}_{k \in \mathcal{K}_t}$ and prices $\{\boldsymbol{q}_k\}_{k \in \mathcal{K}_t}$ parameters and the contribution vector $\boldsymbol{\theta}_t$, and hence can calculate $f_t^\theta(\boldsymbol{x}_t, \boldsymbol{y}_t)$ and $g_t^\theta(\boldsymbol{x}_t, \boldsymbol{y}_t)$. Parameters $\widehat{\nu}$ and $\widehat{\mu}$ are the steps for the intra-period decisions, and we use $\widehat{\nu} = \nu$ and $\widehat{\mu} = \mu$.

4.6.3 Discussion

Complexity analysis OLR-MTS, OLR-MTS-SO. We discuss now the complexity of the two mixed-time-scale algorithms. When compared to the OLR algorithm, the OLR-MTS needs $2K + 1$ more operations to compute the K updates (4.25) and (4.26), and to calculate B_t. Therefore, the running time is still of $\mathcal{O}(K)$. The memory requirements stay at $\mathcal{O}(K)$, as the OLR-MTS needs to store $4(K - 1)$ new variables. Similarly, the running time and memory requirements are the same for the OLR-SO and the OLR-MTS-SO algorithms, as the latter needs $2Km^2 + K$ more operations and $2(K - 1)m + 2(K - 1)$ more storage.

We start by giving an analytical expression for the intra-period reservation decision. Recall that (4.16) provided that expression for the single-time scale model. Here, we can write instead:

$$y_k = \frac{\widehat{\nu} a_{k-1}}{1 + x_{t-1} + y_{k-1}} - q_{k-1} \widehat{\lambda}_k \widehat{\nu} + y_{k-1}. \tag{4.31}$$

We observe that in this case the k-slot decision relies on the previous time-slot (a_{k-1}, q_{k-1}), while the respective decision in (4.16) utilizes information from the previous period (a_{k-K}, q_{k-K}). We expect that the gap between the past value we use and the current value

$|a_{k-1} - a_k|$ is smaller than $|a_{k-K} - a_k|$. This will allow the SP to take more accurate decisions in practice, when using these mixed time-scale updates. On the other hand, the single time scale approach induces smaller computation load, as we can update the decoupled variables in a parallel fashion. This is not the case for the MTS approach, where we need to follow a specific update order, i.e., $x_t, y_{(t-1)K+1}, y_{(t-1)K+2}, \dots, y_{(t-1)K+K}$. This, in turn, might affect the policy implementation for large problem instance and/or when the time-slots are very small.

4.7 Numerical Evaluation

We present a set of numerical tests that verify the learning efficacy of the proposed algorithms and highlight the impact of key system parameters on their performance.

4.7.1 Simulation setup

Scenario. We consider a network operator that offers slices that extend from the last hop wireless link to core computing servers. Thus, the SP can reserve: $y_{k1} \in [0, D_1]$ radio resources at the base stations (BSs); $y_{k2} \in [0, D_2]$ link capacity that connects the BSs to the servers; and $y_{k3} \in [0, D_3]$ processing capacity at the servers (i.e, $m = 3$). All variables are expressed in terms of the flow volume (Mbps) that the slice serves. The SP needs to serve a time-varying demand $a_k \in [0, 1]$ that models the normalized slice utilization – this can be calculated as the number of active users over the maximum number of users allowed in one slice so as to respect the SP's SLA. The achieved slice performance depends on reservation vectors $\boldsymbol{x}_t = (x_{t1}, x_{t2}, x_{t3})$ and $\boldsymbol{y}_k = (y_{k1}, y_{k2}, y_{k3})$ and is given by (4.17).

Traces. The random demand and cost parameters are created based on two cases. In the stationary *Case* 1, the demand a_k is uniformly distributed on $[0, a]$ and the prices p_t and q_k uniformly distributed on $[0, p] \cdot (K/c_f)$ and $[0, q]$, respectively. Here the advance-reservation price p_t is determined as a discount of the spot price q_k, which is tuned via parameter $c_f > 0$. In the non-stationary *Case* 2, we set:

$$\begin{aligned} a_k &= A_0 \sin(2\pi k/K_0) + \mathcal{U}[A_0, a], \\ q_k &= Q_0 \sin(2\pi k/K_0) + \mathcal{U}[Q_0, q], \\ p_t &= \big(P_0 \sin(2\pi t/K_0) + \mathcal{U}[P_0, p]\big)\frac{K}{c_f}, \end{aligned}$$

where $\mathcal{U}$ is the added uniform noise, e.g. $\mathcal{U}[A_0, a]$ follows a uniform distribution on the set $[A_0, a]$. K_0 is the period of the sine waves, A_0, Q_0, P_0 are the amplitude of the sine waves. Such design of the traces enforce $a_k \in [0, A_0 + a]$, $q_k \in [0, Q_0 + q]$ and $p_t \in [0, P_0 + p]$. This is a more challenging scenario for the SP which demonstrates the main advantage of the proposed learning-based reservation algorithms that can adapt to such non-stationary conditions.

Selection of the parameter values. We select by default $\nu = \mu = \widehat{\nu} = \widehat{\mu} = T^{-1/3} = 0.1$, for $T = 1000$, as these are recommended values for the learning rates in [1]. We take $D = 1$ and $a = p = q = 1$ to deal with normalized reservations (belonging to $[0, D] = [0, 1]$), and normalized traces for the stationary case. We limit our period length to $K = 3$, in order to fall under the scope of the sublinear regret (a high value of K increases significantly the regret bound). We select $c_f = 3$ and $K_0 = 5$ by default, as these parameters do not affect the theoretical bounds. For the non-stationary case, we aim to have the same amplitude for the sine wave and the uniform noise. By selecting $A_0 = P_0 = Q_0 = 0.5$, we ensure the sine wave has an amplitude of 1, from -0.5 to 0.5. The noise interval for each trace is $[0.5, 1.5]$, leading to the same amplitude of 1. The value of B is always selected to be $D(p + Kq)/2$, as it is the specific value that minimizes the theoretical bound in Lemma 1.

Technical implementation. The source code of our solutions, the dynamic benchmark, the validation of our regret bounds and the parameter sensitivity analysis can be found online[4]. We develop under Python 3.6 programming environment. We use the library scipy.optimize to solve the quadratic and linear problems of this chapter. As detailed above, we envision two cases (stationary and non-stationary) to evaluate the performance of our solutions in terms of regret and fit. We generate the traces with specific values of the

[4]https://github.com/jeanba19/OLR

Table 4.2: Simulation parameters

Symbol	Physical Meaning
ν	Step-size of the primal update
$\widehat{\nu}$	Step-size of the primal slot update
μ	Step-size of the dual update
$\widehat{\mu}$	Step-size of the dual slot update
p, q, a, θ	Upper bounds of the stationary traces
K_0	Period of the sine waves
P_0, Q_0, A_0, θ_0	Amplitude of the sine waves for the different traces
$P_0 + p, Q_0 + q, A_0 + a \ldots$	Upper bounds of the non-stationary traces
c_f	Discount factor of the period price wrt the spot price

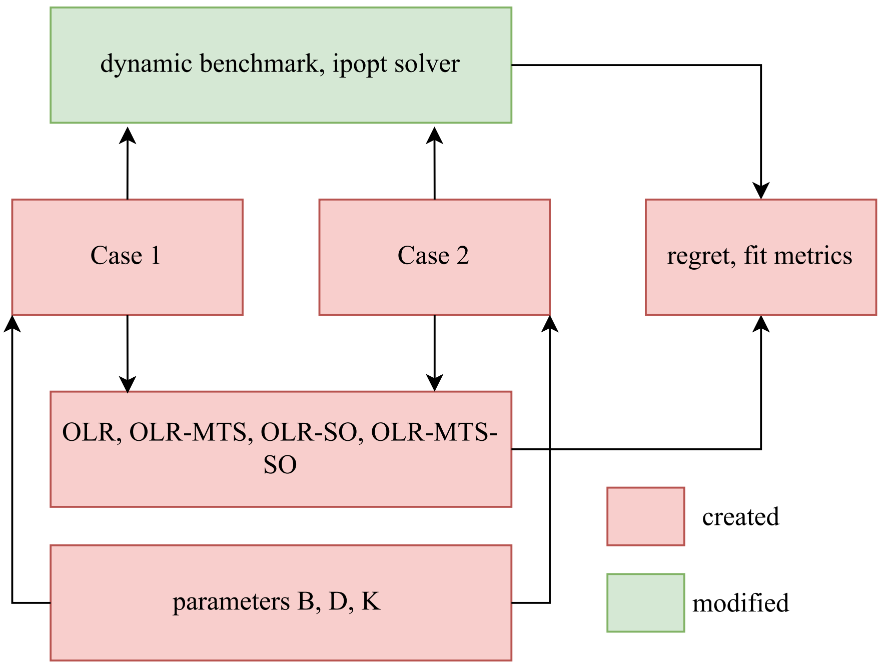


Figure 4.4: Architecture of our solution design

system parameters B, D or K. We then propose to analyze the sensitivity of our solutions against different values of these parameters. We provide an overview of the architecture of our design in Fig. 4.4.

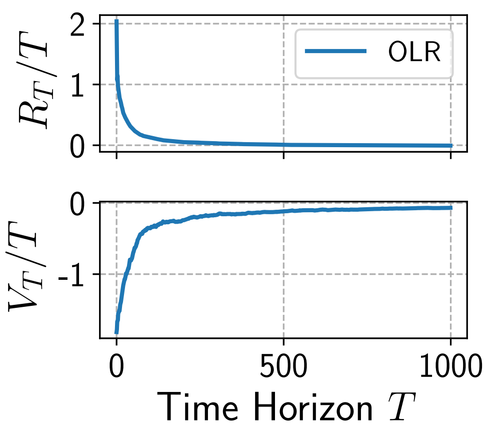
We design the OLR solution. We code the functions f_t and g_t from (4.6) and (4.7) respectively. We code the gradient and the lagrangian from (4.11). We then iterate and solve at each period the problem (4.12), which is quadratic, and the problem (4.13), which is linear. We store the total loss and constraint violation results. The OLR-MTS solution is the same except for, within each period, we solve the problems (4.25) and (4.26) for all the slots of the period. For the OLR-SO solution, we code the functions f_t^θ and g_t^θ from (4.17) and (4.18) respectively. We code the gradient and the lagrangian from (4.19). We then iterate and solve at each period the quadratic problem (4.20) and the linear problem (4.21). We store finally the loss and constraint violation results. The OLR-MTS-SO solution is almost the same: again, we solve the problems (4.29) and (4.30) for all the slots of the period.

For the dynamic benchmark, we use the ipopt convex solver, which implements an interior point method to find the optimal solution of a convex problem. Specifically, we build a class which takes as methods the objective function and its gradient, the constraint function and its jacobian. We create an instance of the class specifying the parameters B, m, K and D. We iterate and solve the sub-problem $(\mathbb{P}_t)$ at each round, using the ipopt solver which applies an interior point method to the instance of the class. Note that we also developed the competitive ratio, which is the optimal solution of the problem $(\mathbb{P})$, while the dynamic benchmark is the sequence of the solutions to the sub-problems $(\mathbb{P}_t)$. The competitive ratio analysis is not provided in this thesis and can be explored as future work.

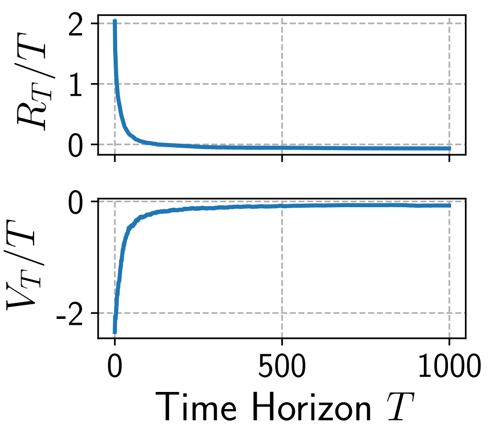
4.7.2 Evaluation of OLR and OLR-MTS

OLR Convergence. First, we verify that R_T and V_T grow sublinearly given that $U_g = o(T)$ and $U_z = o(T)$; see Figs. 4.5(a) and 4.5(b). In *Case* 1, we set the different parameters as shown in the respective caption, and select the optimal budget to be $B = D(p+Kq)/2 = 2$. Under these conditions, we observe in Fig. 4.5(a) the convergence of R_T/T and V_T/T. In *Case* 2, we set the parameters as shown in the respective caption, and select the optimal budget to be $B = D(p + P_0 + K(q + Q_0))/2 = 4\}$. We observe again in Fig. 4.5(b) the convergence of R_T/T and V_T/T. Note that for the evolution of the constraint violation we use $V_T = \sum_{t=1}^{T} g_t(\boldsymbol{z}_t)$ instead of $V_T = [\sum_{t=1}^{T} g_t(\boldsymbol{z}_t)]_+$, in order to study how the budget consumption evolves over time, even when it does not violate the respective bound.

Theory vs practice. In Figs. 4.6(a) and 4.6(b), we compare the convergence of the OLR solution to the convergence of the theoretical bounds (regret and fit). We observe that the solution always converges faster, as the theoretical bound represents the worst-case scenario. For the regret convergence, we zoom in the last 100 periods to point out that an horizon of $T = 1000$ might not be distant enough to observe the convergence to 0 of the theoretical bound. Indeed, we witness the slope of the theoretical regret bound slowly shrinking towards 0.

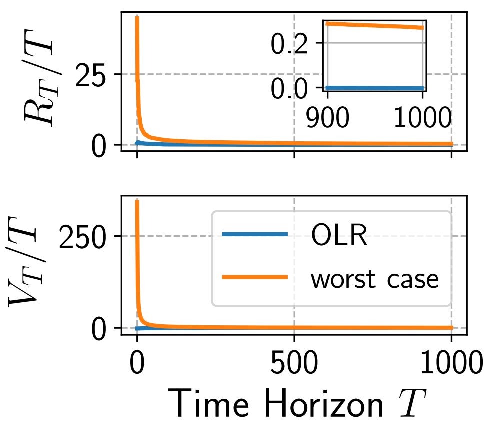


(a) Case 1

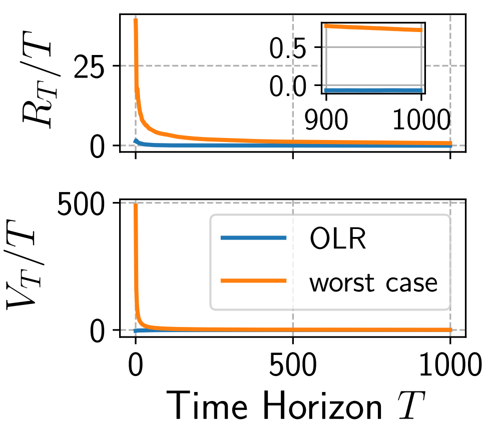


(b) Case 2

Figure 4.5: *Evolution of R_T/T and V_T/T.* Simulation parameters are set to $K=3$, $c_f=3$, $D=1$. (4.5(a)): $a=p=q=1$, $B=2$, $\nu=\mu=0.1$. (4.5(b)): $K_0=5$, $A_0=P_0=Q_0=0.5$, $a=p=q=1.5$, $B=4$, $\nu=\mu=0.1$.



(a) Case 1



(b) Case 2

Figure 4.6: *Evolution of R_T/T and V_T/T.* Simulation parameters are set to $K=3$, $c_f=3$, $D=1$. (4.6(a)): $a=p=q=1$, $B=2$, $\nu=\mu=0.1$. (4.6(b)): $K_0=5$, $A_0=P_0=Q_0=0.5$, $a=p=q=1.5$, $B=4$, $\nu=\mu=0.1$.

Comparison of OLR and OLR-MTS. In Figs. 4.7(a) and 4.7(b), we observe that OLR-MTS exhibits in practice a faster regret convergence than OLR. Namely, until $T = 100$, OLR-MTS achieves smaller regret, which is particularly important for problems that will run for small time horizons. Eventually, both algorithms reach asymptotically a zero average regret state. Indeed, we can see that both solutions show similar convergence, with a slight

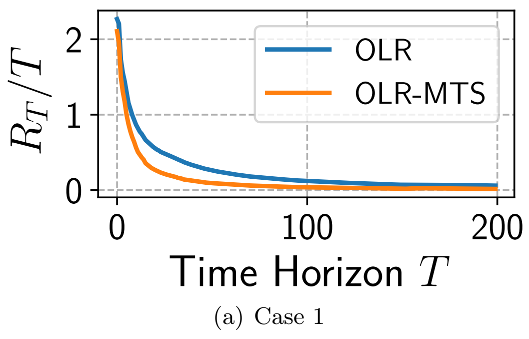


Figure 4.7: *Comparison of OLR and OLR-MTS*. Simulation parameters are $K=3$, $c_f=3$, $D=1$. (4.7(a)): $a=p=q=1$, $B=2$, $\nu=\widehat{\nu}=\mu=\widehat{\mu}=0.1$. (4.7(b)): $K_0=5$, $A_0=P_0=Q_0=0.5$, $a=p=q=1.5$, $B=4$, $\nu=\widehat{\nu}=\mu=\widehat{\mu}=0.1$.

advantage to the OLR-MTS that gets thinner as T increases. Both solutions satisfy the budget constraint, with a negative V_T/T at $T=200$ periods.

Influence of ν and μ. In Fig. 4.8, we observe the influence of the learning rates on the convergence of R_T/T and V_T/T. We set $\nu=0.1$ and use different values for μ, namely $\mu\in\{0.1,0.05,0.01\}$. As predicted in our analysis, a lower value of μ favorites over-reservation, hence the performance is better and the budget consumption is higher. Therefore, the SP can leverage these parameters, depending on whether it wishes to prioritize performance or the budget consumption constraint.

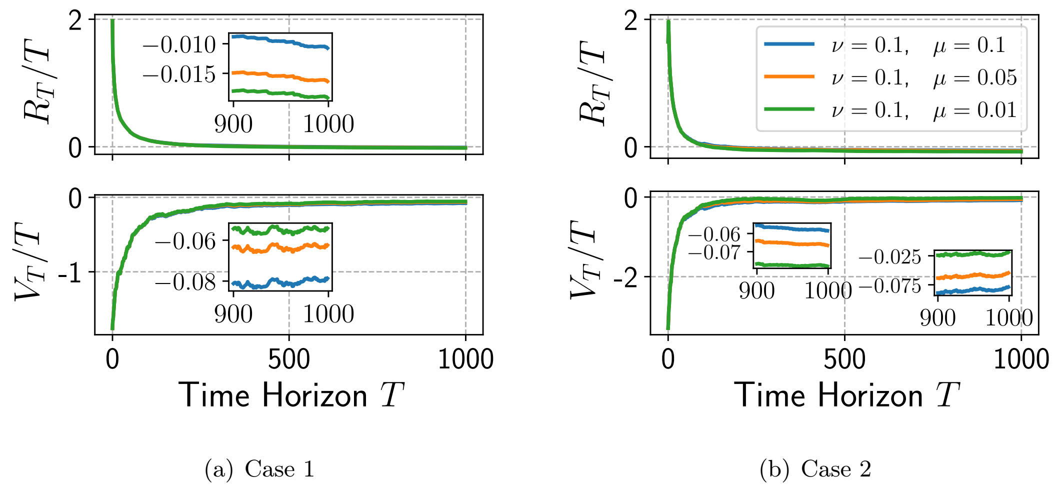


Figure 4.8: *Impact of Learning Rates*. Simulation parameters: $K=3$, $c_f=3$, $D=1$. (4.8(a)): $a=p=q=1$, $B=2$. (4.8(b)): $K_0=5$, $A_0=P_0=Q_0=0.5$, $a=p=q=1.5$, $B=4$

K and D sensitivity. In Figs. 4.9(a) and 4.9(b), as K increases, the regret performance drops in a linear fashion. This is quite reflective of the regret bound, where most terms are either linear or quadratic with regard to K. To fairly compare the performance of the OLR

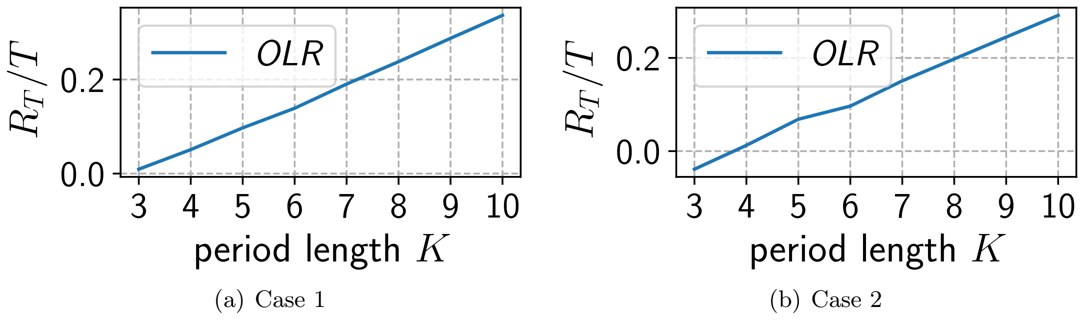


Figure 4.9: Simulation parameters are set to: 20 runs, $T\!=\!500$, $c_f\!=\!3$, $D\!=\!1$. (4.9(a)): $a\!=\!p\!=\!q\!=\!1$, $B = (K+1)/2$, $\nu = \mu = 0.1$. (4.9(b)): $K_0 = 5$, $A_0 = P_0 = Q_0\ = 0.5$, $a = p = q = 1.5$, $B = K+1$, $\nu\!=\!\mu\!=\!0.1$.

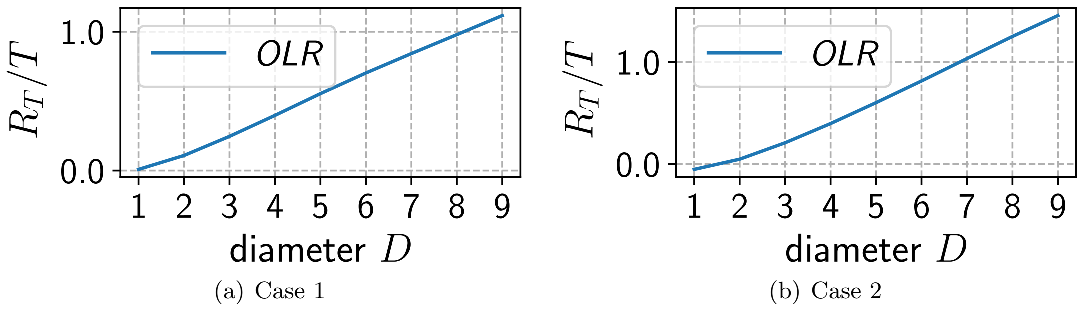


Figure 4.10: Simulation parameters are set to: 20 runs, $T\!=\!500$, $c_f\!=\!3$, $K\!=\!3$. (4.10(a)): $a\!=\!p\!=\!q\!=\!1$, $B\!=\!2D$, $\nu\!=\!\mu\!=\!0.1$. (4.10(b)): $K_0\!=\!5$, $A_0\!=\!P_0\!=\!Q_0\ \!=\!0.5$, $a\!=\!p\!=\!q\!=\!1.5$, $B\!=\!4D$, $\nu\!=\!\mu\!=\!0.1$.

for different values of K, we need to keep $B = D(p+Kq)/2$ at each point. The latter value of B gives the minimum for parameter M, henceforth the regret bound. This way we evaluate the performance of the OLR, holding that the regret bound is at its minimum at each point K. In Figs. 4.10(a) and 4.10(b), the regret performance worsens as D increases. We note that the solution is more sensitive to this parameter, as the performance is significantly worse for $D = 9$ than for $K = 9$ for instance. Again, we compare the OLR performance against different values of D, under the same condition that we use for each D a value of budget that minimizes the regret bound, i.e. $B = D(p+Kq)/2$. With our specific values, it leads in *Case* 1 to $B = D(1+3*1)/2 = 2D$, and in *Case* 2 to $B = D(0.5+1.5+3*(0.5+1.5))/2 = 4D$. We recall that in *Case* 2, $P_0 + p$, $Q_0 + q$ are the upper-bounds.

B **sensitivity.** Here we need to make a preliminary analysis on the impact of B on the optimal reservation policy. At every period, the dynamic benchmark achieves the best performance while respecting the budget constraint, i.e. reserving for a cost less or equal to B. In practice, the only case it spends less than B is when it over-provisions the slice, i.e. $\boldsymbol{z}_t^\star = [D, \dots, D]$ and still we have $D(p_t + q_{(t-1)K+1} + \ldots + q_{tK}) < B$. Thus, the higher the allocated budget TB is, the more the SP will over-provision and the bigger end-savings $TB - \sum_{t=1}^{T} \xi_t$ it will achieve, where $\xi_t = p_t x_t + \sum_{k \in \mathcal{K}_t} y_k q_k$.

Although this first observation is rather intuitive, it offers interesting insights. Let's take *Case* 1 and assume we over-provision the slices, i.e. $\boldsymbol{z} = [D, D, \ldots, D]$. We denote the probability to not violate the budget constraint B as:

$$\mathbb{P}\{Dp_t + Dq_{(t-1)K+1} + \ldots + Dq_{tK} \leq B\}$$

defined for $B \in [0, D(p + Kq)]$. The latter function is the cdf of the Irwin-Hall distribution [115] on $[0, D(p + Kq)]$. The proof follows from noticing that the random variable $p_t + q_{(t-1)K+1} + \ldots + q_{tK}$ follows the Irwin-Hall distribution on $[0, p + Kq)]$ (as a sum of independent uniform variables). This property can be generalized to the sum of independent variables following the same -symmetrical- distributions.

In Fig. 4.11, we count for $T = 500$ periods the number of times the dynamic benchmark over-provisions the slice. We run the simulation 20 times and take average values. We observe the shape of the Irwin-Hall distribution in both cases, although the theoretical result only stands for *Case* 1.

In Figs. 4.11(a) and 4.11(b), the variability U_z/T decreases and becomes very low for:

$$B \in \left[\frac{D(p+Kq)}{2}, D(p+Kq)\right] = [2, 4]$$

in *Case* 1, and:

$$B \in \left[\frac{D\big(p+P_0+K(q+Q_0)\big)}{2}, D\big(p+P_0+K(q+Q_0)\big)\right] = [4, 8]$$

in *Case* 2. Obviously for $B = 0$, the benchmark cannot reserve anything hence $U_z = 0$. It outlines that high B represents a trivial case, leading to very easy to learn optimal policy that consists of over-provisioning the slice. Therefore, in Fig. 4.12, we focus on a more challenging set to show the performance of our solutions: $B \in]0, D(p + Kq)/2] =]0, 2]$ in *Case* 1 and $B \in]0, D(p + P_0 + K(q + Q_0))/2] =]0, 4]$ in *Case* 2.

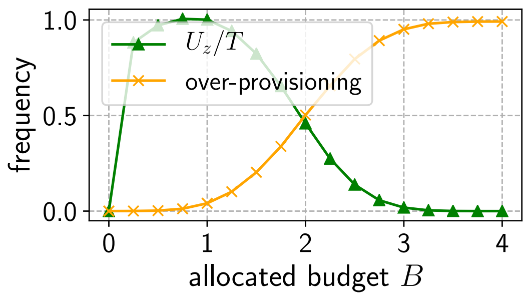
In Fig. 4.12, we focus on the performance ratio, which is the performance (given by

(4.2)) of our solution (OLR or OLR-MTS) over the performance of the benchmark, and on the violation ratio, which represents the importance of the violation through its proportion of the total allocated budget BT. We run the simulation 20 times and take average values. We observe performance and violation ratios at $T = 500$ periods.

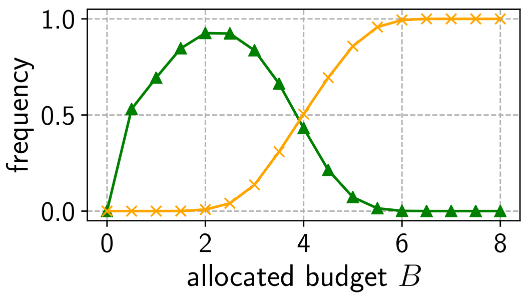
In Fig. 4.12(a), we observe that the solutions OLR and OLR-MTS achieve their best performance for $B = 2 = D(p + Kq)/2$. This is rather expected as $B = 2$ corresponds to the lowest variability on the benchmark U_z/T. It is also the value of B that minimizes the upper bound on the regret and fit, as we previously discussed in section IV. Surprisingly, however, the solutions achieve a very good performance at $B = 0.5 = D(p + Kq)/8$, while the variability U_z/T for this value is quite high. The respective violation is 3.44% of the total allocated budget, which is rather reasonable.

In Fig. 4.12(b), the solutions achieve their best performance of 1.023 at $B = 1 = D(p + P_0 + K(q + Q_0))/8$. The respective violation is 1.65% of the total allocated budget. This performance is truly amazing as it even surpasses the 1.016 performance achieved at $B = 4 = D(p+P_0+K(q+Q_0))/2$. For the latter B, the respective savings are 2.67% of the total allocated budget, which is less than the savings attained by the benchmark (9.19%).

Conclusively, we find out that two specific values of budget B lead to the best performance in both cases. While we expected the values $B = D(p + Kq)/2$ (*Case* 1) and $B = D(p + P_0 + K(q + Q_0))/2$ (*Case* 2) to show good performance, we are positively surprised to observe good performance for the values $B = D(p + Kq)/8$ (*Case* 1) and $B = D(p+P_0+K(q+Q_0))/8$ (*Case* 2). We will confirm in the slice orchestration case that these values of B are local optima of the performance for the SP.

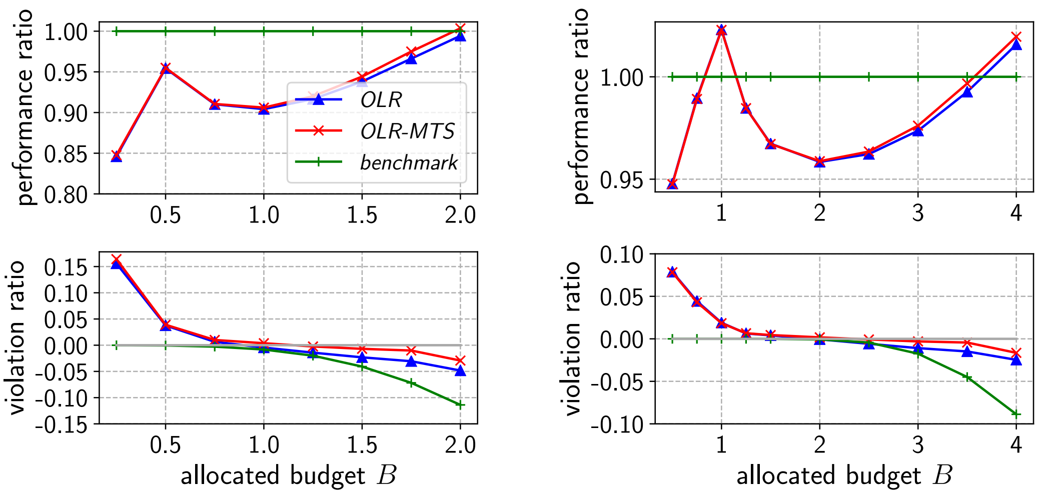


(a) Case 1



(b) Case 2

Figure 4.11: Simulation parameters are set to: 20 runs, $T = 500$, $K = 3$, $c_f = 3$, $D = 1$. (4.11(a)): $a = p = q = 1$. (4.11(b)): $K_0 = 5$, $A_0 = P_0 = Q_0 = 0.5$, $a = p = q = 1.5$.



(a) Case 1 (b) Case 2

Figure 4.12: Simulation parameters are set to: 20 runs, $T = 500$, $c_f = 3$, $D = 1$, $K = 3$. (4.12(a)): $a = p = q = 1$, $\nu = \widehat{\nu} = \mu = \widehat{\mu} = 0.1$. (4.12(b)): $K_0 = 5$, $A_0 = P_0 = Q_0 = 0.5$, $a = p = q = 1.5$, $\nu = \widehat{\nu} = \mu = \widehat{\mu} = 0.1$.

4.7.3 Evaluation of OLR-SO and OLR-MTS-SO

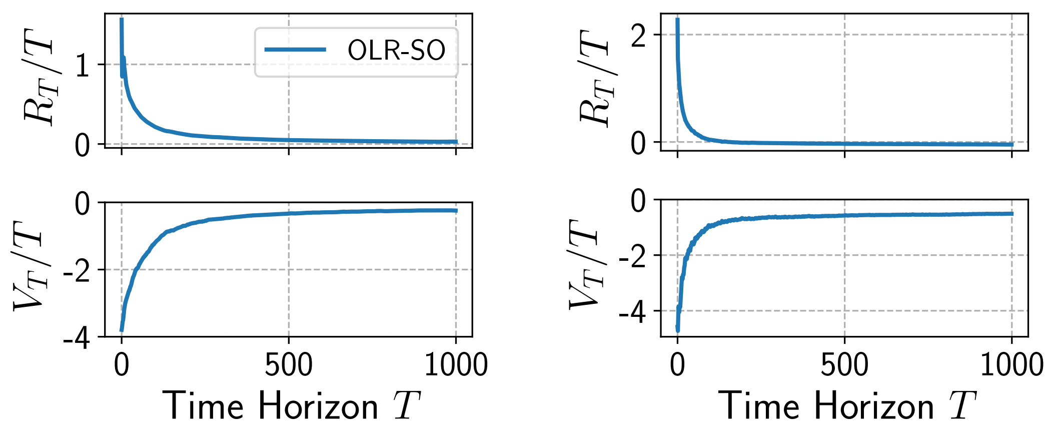
Traces. In the slice orchestration case the random demand and cost parameters are created based on two cases. In the stationary *Case* 1, the demand a_k is uniformly distributed on $[0, a]$ and the prices $\boldsymbol{p}_t$ and $\boldsymbol{q}_k$ are random vectors of dimension m following the uniform distributions $(K/c_f) * [\mathcal{U}[0, p], \dots, \mathcal{U}[0, p]]^\top$ and $[\mathcal{U}[0, q], \dots, \mathcal{U}[0, q]]^\top$, respectively. We define the slice composition vector $\boldsymbol{\theta}_t$ as the uniform random vector $[\mathcal{U}[0, \theta], \dots, \mathcal{U}[0, \theta]]^\top$ of dimension m. Here the advance-reservation price vector $\boldsymbol{p}_t$ is determined as a discount of the spot price vector $\boldsymbol{q}_k$. In our setting, we regulate the importance of such discount through parameter c_f. In the non-stationary *Case* 2, we set:

$$\begin{aligned}
a_k &= A_0 \sin(2\pi k/K_0) + \mathcal{U}[A_0, a], \\
\boldsymbol{q}_k &= Q_0 \sin(2\pi k/K_0) + [\mathcal{U}[Q_0, q], \dots, \mathcal{U}[Q_0, q]]^\top, \\
\boldsymbol{p}_t &= (P_0 \sin(2\pi t/K_0) + [\mathcal{U}[P_0, p], \dots, \mathcal{U}[P_0, p]]^\top)\frac{K}{c_f}, \\
\boldsymbol{\theta}_t &= \theta_0 \sin(2\pi t/K_0) + [\mathcal{U}[\theta_0, \theta], \dots, \mathcal{U}[\theta_0, \theta]]^\top
\end{aligned}$$

where, as before, parameters A_0, Q_0, P_0, θ_0 are the amplitudes of the sine waves and $\mathcal{U}$ is the added uniform noise on $[A_0, a]$, $[Q_0, q]$, etc. The sine part of $\boldsymbol{q}_k$, $\boldsymbol{p}_t$ and $\boldsymbol{\theta}_t$ is the offset that we add to the components of the uniform random vector.

OLR-SO convergence. First, we verify that R_T and V_T grow sublinearly given that $U_g^\theta = o(T)$ and $U_z^\theta = o(T)$; see Figs. 4.13(a) and 4.13(b). In *Case* 1, we select the system parameters shown in the respective caption, and for which the optimal budget is $B = (D_1+D_2+D_3)(p+Kq)/2 = 4$. We can observe in Fig. 4.13(a) the convergence of both R_T/T and V_T/T. In *Case* 2, we set $B = (D_1+D_2+D_3)(p+P_0+K(q+Q_0))/2 = 8\}$,Under these settings, we observe in Fig. 4.13(b) the convergence of R_T/T and V_T/T.

Theory vs practice. In Figs. 4.14(a) and 4.14(b), we plot the convergence of both the theoretical bound and the actual solution OLR-SO (regret and fit). Again, the solution converges faster than its worst-case scenario. Also, zooming in the last 100 periods show that the theoretical regret bound is slowly decaying towards 0.

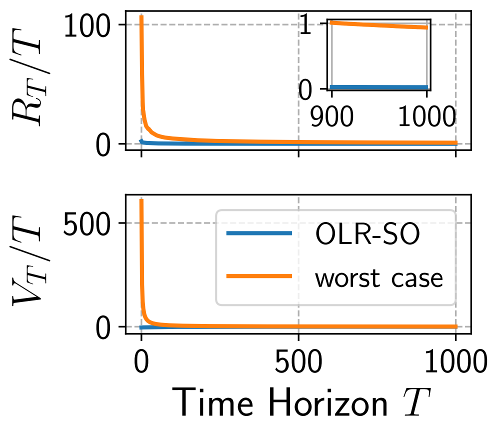


(a) Case 1 (b) Case 2

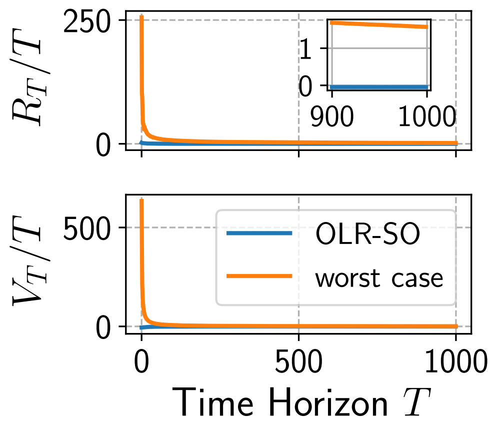
Figure 4.13: *Evolution of R_T/T and V_T/T.* Simulation parameters are set to $K = 3$, $c_f = 3$, $\boldsymbol{D} = [0.5, 1, 0.5]$, $m = 3$. (4.13(a)): $a = p = q = \theta = 1$, $B = 4$, $\nu = \mu = 0.1$. (4.13(b)): $K_0 = 5$, $A_0 = P_0 = Q_0 = \theta_0 = 0.5$, $a = p = q = \theta = 1.5$, $B = 8$, $\nu = \mu = 0.1$.

Comparison of OLR-SO and OLR-MTS-SO. In Figs. 4.15(a) and 4.15(b), we observe that the OLR-MTS-SO converges faster than the OLR-SO. This result is similar to the one of the previous subsection. We observe that until $T = 100$, the OLR-MTS-SO solution has better convergence for both R_T/T and V_T/T. Then, the two solutions show similar convergence, with a small advantage to the OLR-MTS-SO that vanishes as T increases.

Influence of ν and μ. In Fig. 4.16, we observe the influence of the learning rates

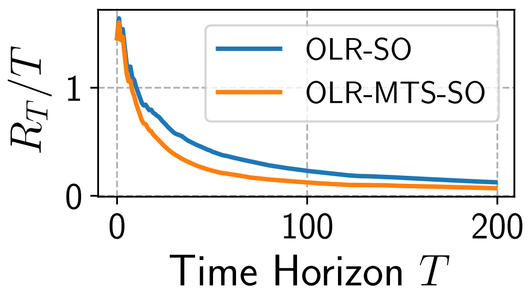


(a) Case 1

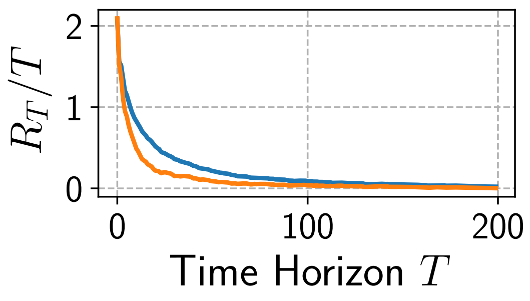


(b) Case 2

Figure 4.14: *Evolution of* R_T/T *and* V_T/T. Simulation parameters are set to $K = 3$, $c_f = 3$, $\boldsymbol{D} = [0.5, 1, 0.5]$, $m = 3$. (4.14(a)): $a = p = q = \theta = 1$, $B = 4$, $\nu = \mu = 0.1$. (4.14(b)): $K_0 = 5$, $A_0 = P_0 = Q_0 = \theta_0 = 0.5$, $a = p = q = \theta = 1.5$, $B = 8$, $\nu = \mu = 0.1$.



(a) Case 1

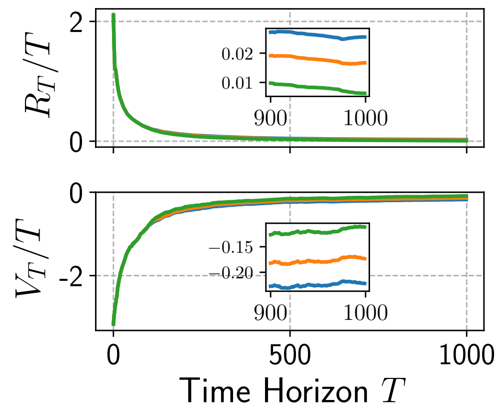


(b) Case 2

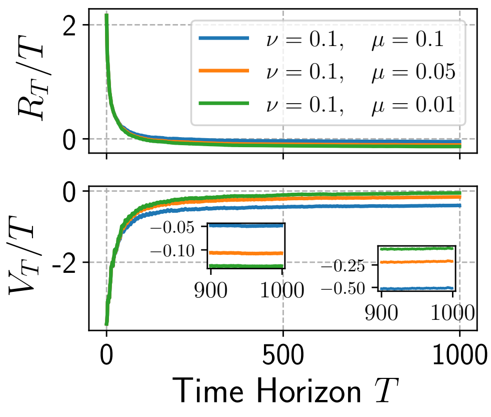
Figure 4.15: *Evolution of* R_T/T *and* V_T/T. Simulation parameters are set to $K = 3$, $c_f = 3$, $\boldsymbol{D} = [0.5, 1, 0.5]$, $m = 3$. (4.15(a)): $a = p = q = \theta = 1$, $B = 4$, $\nu = \widehat{\nu} = \mu = \widehat{\mu} = 0.1$. (4.15(b)): $K_0 = 5$, $A_0 = P_0 = Q_0 = \theta_0 = 0.5$, $a = p = q = \theta = 1.5$, $B = 8$, $\nu = \widehat{\nu} = \mu = \widehat{\mu} = 0.1$.

on the convergence of R_T/T and V_T/T. We proceed to keep $\nu = 0.1$ constant and to give the values $\{0.1, 0.05, 0.01\}$ to μ. As predicted in our analysis, a lower value of μ favorites over-reservation, therefore leads to better performance and higher budget consumption. We derive the same remark as previously, which is the SP can leverage these learning rates, depending on what it wishes to prioritize: either performance or budget.

***K* and *D* sensitivity.** In Figs. 4.17(a) and 4.17(b), we observe that the performance worsens faster against K than in the OLR case. This is not surprising, as each slot comprises m decisions, in lieu of only 1 in the OLR case. Once again, we compare the regret for different values of K choosing a budget value of $B = (\sum_i D_i)(p + Kq)/2$ which is equal to $K + 1$ in *Case* 1 and to $2(K + 1)$ in *Case* 2. In Figs. 4.18(a) and 4.18(b), $\boldsymbol{D}$ is a vector of dimension

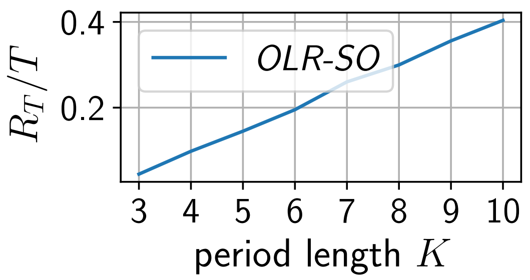


(a) Case 1

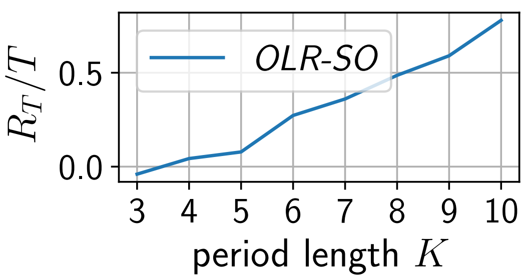


(b) Case 2

Figure 4.16: *Evolution of R_T/T and V_T/T.* Simulation parameters are set to $K = 3$, $c_f = 3$, $\boldsymbol{D} = [0.5, 1, 0.5]$, $m = 3$. (4.16(a)): $a = p = q = \theta = 1$, $B = 4$. (4.16(b)): $K_0 = 5$, $A_0 = P_0 = Q_0 = \theta_0 = 0.5$, $a = p = q = \theta = 1.5$, $B = 8$.



(a) Case 1

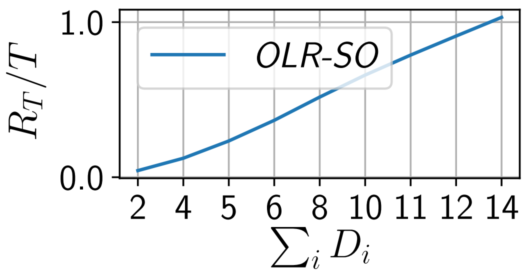


(b) Case 2

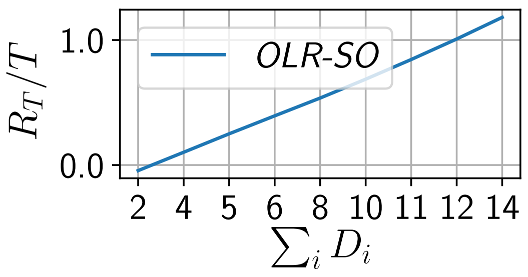
Figure 4.17: Simulation parameters are set to: 20 runs, $T = 500$, $c_f = 3$, $\boldsymbol{D} = [0.5, 1, 0.5]$, $m = 3$. (4.17(a)): $a = p = q = \theta = 1$, $B = K + 1$, $\nu == \mu = 0.1$. (4.17(b)): $K_0 = 5$, $A_0 = P_0 = Q_0 = \theta_0 = 0.5$, $a = p = q = \theta = 1.5$, $B = 2(K + 1)$, $\nu == \mu = 0.1$.

$m = 3$. We increment each item of the vector by 0.5 from $[0.5, 1, 0.5]$ to $[4.5, 5, 4.5]$. The x-axis on the graphs represent the sum of the items, starting from 2 to 14. Again, we choose the value $B = (\sum_i D_i)(p + Kq)/2$, that minimizes the regret bound in Lemma 2. Under our value settings, it leads to $B = 2(\sum_i D_i)$ in *Case* 1 and $B = 4(\sum_i D_i)$ in *Case* 2. We observe a rise of the regret, which is slower than in the OLR case, if we relate $\sum_i D_i$ to D.

B **sensitivity.** In this subsection, we confirm for the slice orchestration case our previous analysis on the parameter B. For *Case* 1, after analyzing the variability U_z^θ/T in Fig. 4.19(a), we focus on the performance and violation ratios for $B \in]0, 4]$. In Fig. 4.20(a), we observe two values of B, namely $B = (D_1 + D_2 + D_3)(p + Kq)/2 = 4$ and $B = (D_1 + D_2 + D_3)(p + Kq)/8 = 1$, that lead to the best performance of our solutions. For *Case* 2, we observe in Fig. 4.20(b) that $B = (D_1 + D_2 + D_3)(p + P_0 + K(q + Q_0))/2 = 8$ and



(a) Case 1

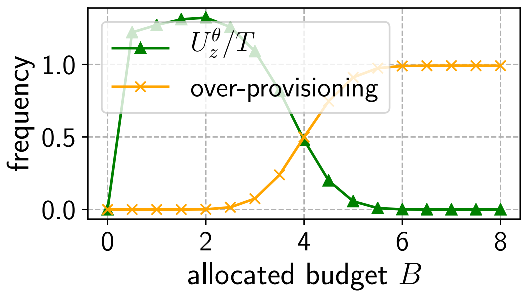


(b) Case 2

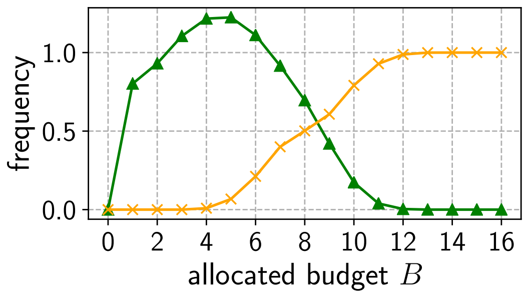
Figure 4.18: Simulation parameters are set to: 20 runs, $T=500$, $c_f=3$, $K=3$, $m=3$. (4.18(a)): $a=p=q=\theta=1$, $B=2(\sum_i D_i)$, $\nu=\mu=0.1$. (4.18(b)): $K_0=5$, $A_0=P_0=Q_0=\theta_0=0.5$, $a=p=q=\theta=1.5$, $B=4(\sum_i D_i)$, $\nu=\mu=0.1$.

$B=(D_1+D_2+D_3)(p+P_0+K(q+Q_0))/8=2$ show the best results in terms of performance. We look only at the interval $B\in]0,8]$, as for higher values of B, the variability U_z^θ/T is very low (see Fig. 4.19(b)) and the latter case is not challenging for the SP.

We conclude that a wealthy SP has more interest to choose a high value of B (for instance $](\sum_i D_i)(p+Kq)/2,(\sum_i D_i)(p+Kq)]$), as this decreases the variability of the dynamic benchmark sequence. The easy to learn optimal solution consists of over-provisioning the slice in most periods. For $B=(\sum_i D_i)(p+Kq)$, the over-provisioning strategy is the optimal solution. On the other hand, a modest SP can still choose wisely its parameter B, for instance $B=(\sum_i D_i)(p+Kq)/2$ or $B=(\sum_i D_i)(p+Kq)/8$. This represents a riskier scenario, as if the diverse bounds of the system model (D,p,q, etc.) are not known precisely, a small error in their estimation can lead the SP to miss the peak of performance observed at those precise values.

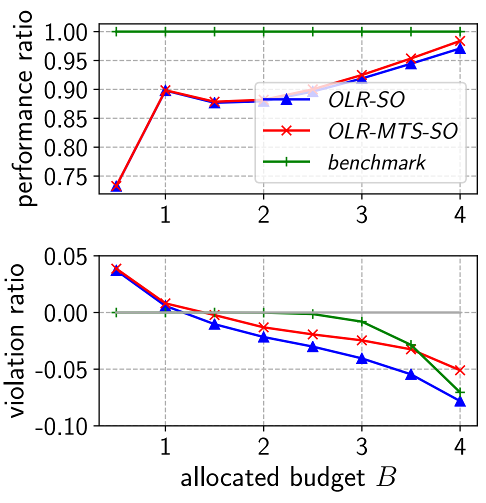


(a) Case 1

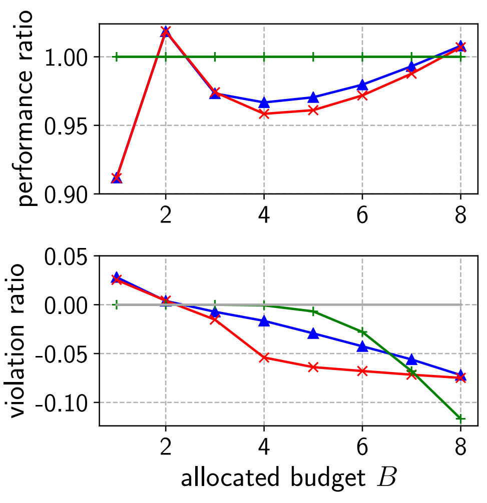


(b) Case 2

Figure 4.19: Simulation parameters are set to: 20 runs, $T=500$, $K=3$, $c_f=3$, $m=3$, $\boldsymbol{D}=[0.5,1,0.5]$. (4.19(a)): $a=p=q=\theta=1$. (4.19(b)): $K_0=5$, $A_0=P_0=Q_0=\theta_0=0.5$, $a=p=q=\theta=1.5$.



(a) Case 1



(b) Case 2

Figure 4.20: Simulation parameters are set to: 20 runs, $T=500$, $c_f=3$, $\boldsymbol{D}=[0.5,1,0.5]$, $m=3$, $K=3$. (4.20(a)): $a=p=q=\theta=1$, $\nu=\widehat{\nu}=\mu=\widehat{\mu}=0.1$. (4.20(b)): $K_0=5$, $A_0=P_0=Q_0=\theta_0=0.5$, $a=p=q=\theta=1.5$, $\nu=\widehat{\nu}=\mu=\widehat{\mu}=0.1$.

4.8 Conclusion & Discussion

The deployment of markets for virtualized network resources is becoming increasingly important and is expected to be among the key building blocks of future mobile networks. A prerequisite for their effective operation, however, is the ability of the service providers, the clients of these markets, to reserve resources in a way that maximizes their services' performance without exceeding the anticipated operating expenditures. In the new era of flexible market rules and volatile network conditions and user requirements, this reservation becomes an intricate problem that calls for new solution techniques.

Our approach is inspired by the celebrated online convex optimization paradigm, which is a bare bones optimization model with minimal assumptions regarding the involved cost functions. Hence, the designed algorithms enable the SP to learn how to reserve resources optimally, with guarantees that do not depend on the statistical properties of the involved random parameters. Namely, our policies are robust to arbitrary changes of the resource prices, oblivious to lack of this information when the reservations are made, and achieve optimal slice orchestration even when the SP needs are time-varying. These elements build a practical and general slicing framework with performance and budget guarantees.

While oblivious of the prices and the demand, and consequently perfectly robust to arbitrary changes, our decisions ought to benefit from accurate predictions on these unknown parameters. Recent works in online learning have investigated on *adaptive* and *optimistic* methods [116, 117]. These methods rely on predictions which, if accurate, can significantly improve the regret performance. We propose the reader to explore new solutions based on these methods in the next chapter.

5 Optimistic online reservation of virtualized resources

In this chapter, we propose a reservation-based learning solution capable to incorporate predictions that the SP can derive based on its partial knowledge of the environment. The approach complements our previous solutions which allowed the SP to make decisions while facing total uncertainty about the environment.

The work presented in this chapter corresponds to the contribution C3 which addresses RQ1 and RQ3 and is based on our conference paper "Reservation of Virtualized Resources with Optimistic Online Learning", currently under submission.

5.1 Introduction

In the previous chapter, we have designed four solutions for the SP to derive a complete online reservation policy, regardless of the evolution of its demand nor the prices of the available resources. These solutions give to the SP a complete reservation scheme to build an E2E network slice, robust to the arbitrary changes of the time-varying parameters. We have observed in chapter 3 that the SP receives a feedback from the NO at each step. The accumulated historical data have only been used in a minimal sense in chapter 4, as we take the previous slot/period values of the demand and prices to solve the convex optimization problem at the current slot/period, cf. equations (4.12) and (4.13). Therefore, we propose in this chapter a new solution, which would both provide performance guarantees and leverage historical data to extract accurate predictions. The latter can reduce the uncertainty of the problem and enhance the performance of the decisions. As the SP accumulates historical data about prices and demand, there exists the possibility to extract predictions for the next slot values based on previous window of the traces by using auto-regressive methods. Holt-Winters, Auto-Regressive Integrated Moving Average (ARIMA) or Neural Networks have been applied in [118] and in [27]. Albeit accurate, these methods do not provide performance guarantees. The ARMA-OGD algorithm presented in [119] is an accurate, robust and computationally low prediction model. It generates the predictions through an auto-regressive process, where the lag coefficients are updated using the online gradient descent (OGD) method, which has low time complexity. It also provides regret guarantees against the best Auto-Regressive Moving Average (ARMA) predictor with full hindsight of the future.

The solution is built upon the FTRL algorithm [120]. We develop an *optimistic* version of the FTRL, based on the solution of Mohri and Yang [116] and first introduced by Rakhlin and Sridharan [121], where the decision relies on an adaptive proximal regularizer term and the optimistic term of the next gradient prediction $\nabla\widehat{f}_{t+1}(\widehat{z}_{t+1})$. To apply a solution derived from the FTRL family, we must face an unconstrained minimization problem. Thus we reformulate the SP reservation problem in the next section to fall under the scope of unconstrained OCO.

The contribution of this chapter can be stated as follows:

- we formulate an optimization problem for the SP where it aims to maximize the leased slice utility and minimize the reservation cost in the long-term;
- to solve the reservation problem faced by the SP, we develop the OOLR solution which incorporates the *optimistic* prediction of the next slot gradient;
- we provide regret bound guarantees of $\mathcal{O}(\sqrt{T})$ for arbitrarily bad predictions and $\mathcal{O}(1)$ for perfect predictions;
- we implement a prediction module to assist our OOLR decision algorithm and we demonstrate good performance of the combined solution named OORLgrad;
- against real world data and non-stationary traces, our OOLRgrad solution outperforms the FTRL baseline. We extend our model to the situation the NO only fulfills part of the SP reservation request due to capacity constraints.

The SP can derive better reservation decisions by anticipating the future needs from its end users and the future prices of the network resources. To the best of our knowledge, this is the first prediction-assisted slice reservation model for SPs. Some works have derived similar models for the NO. In [50], network traffic information is leveraged to plan the capacity needed for each slice in a multi-tenant framework. In [122] cellular traffic prediction helps the allocation policy for the vehicular network slice. The approach in [118] employed an adaptive forecasting model of the elastic demand for network resources to perform slice allocation in Internet Access Services.

This chapter is structured as follows. We open with the reformulation of the convex programm previously defined in 4.3. Then, we detail the OOLR decision algorithm, providing performance guarantees. We also propose a model extension which tackles the situation where the NO cannot fulfill the totality of the SP request. Finally, we conduct an experimental scenario whereby we evaluate both our OOLR solution and the FTRL baseline against real-world data.

5.2 Problem formulation

5.2.1 Feasible set

The NO can impose upper limits on the requests of the SP. For instance, the reservation request for resource i must belong to the set $\mathit{\Gamma}_i = [0, D_i]$, where D_i is the limit imposed by the NO on resource i. Therefore, the SP request will belong to $\mathit{\Gamma}_1 \times \ldots \times \mathit{\Gamma}_m$, which we denote Δ. Such limitations arise from natural capacity constraints of the network, in charge of multiple services and its own needs. In some cases, the NO can be unable to fulfill the SP request, especially when the network is congested due to high users' demand load and heavy SPs requests. The NO must guarantee a certain SLA, which we relate to the respect of a certain threshold ratio of the requested amount resource. For instance, the NO must deliver at least $\alpha = 80\%$ of the desired capacity for the resource. We envision this scenario as an extension and assume from now that the NO must comply with the whole request if it belongs to $\mathit{\Gamma}_i$.

5.2.2 Unconstrained convex problem

In the previous chapter, we faced a constrained convex programm and thus the algorithm of Mohri and Yang [116] is not applicable. Thus we must reformulate this programm in an unconstrained manner while keeping the same information it contains about utility and cost derived from the reservation. The new unconstrained convex programm is:

$$
\begin{aligned}
(\mathbb{P}): \quad \max_{\{\boldsymbol{x}_t,\{\boldsymbol{y}_t\}\}_{t=1}^{T}} \sum_{t=1}^{T} \Big(& V a_t \log((\boldsymbol{x}_t+\boldsymbol{y}_t)^\top \boldsymbol{\theta}_t + 1) \\
& - (\boldsymbol{p}_t^\top \boldsymbol{x}_t + \boldsymbol{q}_t^\top \boldsymbol{y}_t)\Big) \qquad (5.1) \\
\text{s.t.} \quad \boldsymbol{y}_t \in & \Delta, \quad \forall t = 1, \dots, T, \qquad (5.2) \\
\boldsymbol{x}_t \in & \Delta, \quad \forall t = 1, \dots, T. \qquad (5.3)
\end{aligned}
$$

In Objective (5.1), we recognize the weighted sum of the slice performance (logarithmic term) and the payments (linear term). The latter term has a minus sign as the SP seeks to minimize its monetary cost. We sum over the number of slots T, as the goal is to maximize this weighted sum in the long-term. Constraints (5.2) and (5.3) ensure the decisions belong to the constraint convex set Δ. We define the hyper-parameter $V \geq 1$ which balance the influence between the two terms (utility term and cost term). The bigger V, the more we favor the slice utility in the detriment of the cost of reservation.

$(\mathbb{P})$ is a convex optimization problem but cannot be tackled directly due to the following challenges:

- the users' demand $\{a_t\}$ is unknown, time-varying and non-stationary;
- the unit prices $\{\boldsymbol{q}_t\}$ and $\{\boldsymbol{p}_t\}$, are unknown, time-varying and non-stationary;

Due to these challenges, the convex problem $(\mathbb{P})$ cannot be solved at $t = 1$ for the next T slots. Henceforth we define the vector function, at each slot t:

$$
\begin{aligned}
f_t(\boldsymbol{x}_t, \boldsymbol{y}_t) = -V a_t \log((\boldsymbol{x}_t+\boldsymbol{y}_t)^\top \boldsymbol{\theta}_t + 1) \qquad (5.4) \\
+(\boldsymbol{p}_t^\top \boldsymbol{x}_t + \boldsymbol{q}_t^\top \boldsymbol{y}_t) \qquad (5.5)
\end{aligned}
$$

The function f_t is convex which allows us to use the OCO framework. Our goal is to decide at each slot t the reservation plan $\boldsymbol{z}_t = (\boldsymbol{x}_t, \boldsymbol{y}_t)$ and achieve in the long term a sublinear regret as defined in equation (2.3.1).

5.3 Optimistic Online Learning for Reservation

5.3.1 Algorithm

Our approach is inspired from the FTRL policy, whereby the learner aims to minimize the loss on all past slots plus a regularization term:

$$\forall t, \boldsymbol{z}_{t+1} = \arg \min_{\boldsymbol{z} \in \Delta^2} \sum_{i=1}^{t} f_i(\boldsymbol{z}) + R(\boldsymbol{z}) \tag{5.6}$$

Due to the convexity of f_t, the following property holds:

$$f_t(\boldsymbol{z}_t) - f_t(\boldsymbol{z}^*) \leq \nabla f_t(\boldsymbol{z}_t)^\top (\boldsymbol{z}_t - \boldsymbol{z}^*) \tag{5.7}$$

which means that the regret against the functions $\{f_t\}$ is upper-bounded by the regret against their linearized form $\bar{f}_t(\boldsymbol{z}) = \nabla f_t(\boldsymbol{z}_t)^\top \boldsymbol{z}$ [117]. Consequently, the FTRL algorithm simplifies to:

$$\forall t, \boldsymbol{z}_{t+1} = \arg \min_{\boldsymbol{z} \in \Delta^2} \sum_{i=1}^{t} \nabla f_i(\boldsymbol{z}_i)^\top \boldsymbol{z} + R(\boldsymbol{z}) \tag{5.8}$$

In our approach, we consider an additional gradient term, which is the *optimistic* next slot gradient prediction $\nabla \widehat{f}_{t+1}(\widehat{\boldsymbol{z}}_{t+1})$. In the FTRL, the regularization function is quadratic $R(\boldsymbol{z}) = \frac{1}{2\eta}||\boldsymbol{z}||^2$. In contrast, we design a sequence of proximal regularizers:

$$\forall t = 1 \dots T, \quad r_t(\boldsymbol{z}) = \frac{\sigma_t}{2}||\boldsymbol{z} - \boldsymbol{z}_t||^2, \tag{5.9}$$

Algorithm OOLR: Optimistic Online Learning for Reservation

```
  Initialize:
  $z_1 \in \Delta^2, \sigma = 1, a_1, q_1, f_1(z_1)$
1 for $t = 1, \dots, T-1$ do
2     Observe the new prediction of the gradient $\nabla\widehat{f}_{t+1}(\widehat{z}_{t+1})$
3     Decide $z_{t+1}$ by solving (5.12)
4     Observe the demand $a_{t+1}$, the reservation price $p_{t+1}$, the spot price $q_{t+1}$, the
        contributions $\theta_{t+1}$
5     Calculate $f_{t+1}(z_{t+1})$ and $\nabla f_{t+1}(z_{t+1})$
6     Update $r_{1:t+1}(z)$ according to (5.9) and (5.10)
```

with $||.||$ the Euclidean norm. The regularizer parameters are:

$$\sigma_t = \sigma\Big(\sqrt{h_{1:t}} - \sqrt{h_{1:t-1}}\Big), \tag{5.10}$$

$$h_t = ||\nabla f_t(\boldsymbol{z}_t) - \nabla\widehat{f}_t(\widehat{\boldsymbol{z}}_t)||^2, \tag{5.11}$$

where $\sigma \geq 0$, and $h_{1:t} = \sum_{i=1}^t h_i$.

All the above lead to the final form of our algorithm decision step:

$$\boldsymbol{z}_{t+1} = \arg\min_{\boldsymbol{z}\in\Delta^2} \Big\{ r_{1:t}(\boldsymbol{z}) + \Big(\sum_{i=1}^t \nabla f_i(\boldsymbol{z}_i) + \nabla\widehat{f}_{t+1}(\widehat{\boldsymbol{z}}_{t+1})\Big)\top \boldsymbol{z}\Big\} \tag{5.12}$$

5.3.2 Performance analysis

We start with the necessary assumptions.

Assumption 12. *The sets Γ_i, $i = 1 \dots m$, are convex and compact, and it holds $|x| \leq D_i$, for any $x \in \Gamma_i$*[1].

Assumption 13. *The function f_t is convex.*

Assumption 14. *$\{r_t\}_{t=1}^T$ is a sequence of proximal non-negative functions.*

Assumption 15. *Prediction $\nabla\widehat{f}_{t+1}(\widehat{\boldsymbol{z}}_{t+1})$ is known at t.*

[1]Note that we can rename the set $\Gamma_1 \times \dots \times \Gamma_m$ as Δ and simply assume Δ is a compact convex set with diameter D. The design of the different sets Γ_i allows us to choose a different reservation restriction for each resource type.

Corollary 1. *Under Assumptions 12-15, we derive from [116, Theorem 1] and [123, Theorem 1] the following regret bound:*

$$\boxed{R(T) \leq \sqrt{\sum_{t=1}^{T} ||\nabla f_t(\boldsymbol{z}_t) - \nabla \widehat{f}_t(\widehat{\boldsymbol{z}}_t)||^2 (\frac{2}{\sigma} + \frac{\sigma}{2} 2D^2)}} \tag{5.13}$$

Proof. First let's remark that the function $h_{0:t} : \boldsymbol{z} \to r_{0:t}(\boldsymbol{z}) + (c_{1:t} + \tilde{c}_{t+1})^\top \boldsymbol{z}$ is 1-strongly convex, with respect to the norm $||.||_{(t)}$. It allows us to use [116, Theorem 1], which yields regret:

$$R(T) \leq r_{1:T}(\boldsymbol{z}^*) + \sum_{t=1}^{T} ||c_t - \tilde{c}_t||^2_{(t),*} \quad \forall \boldsymbol{z}^* \in \Delta^2 \tag{5.14}$$

Now, we define the norm $||x||_{(t)} = \sqrt{\sigma_{1:t}}||x||$, which has dual norm $||x||_{(t),*} = ||x||/\sqrt{\sigma_{1:t}}$. We remark that $\sigma_{1:t} = \sigma\sqrt{h_{1:t}}$, and starting from (5.14), we get:

$$\begin{aligned} R(T) &\leq \frac{\sigma}{2} \sum_{t=1}^{T} (\sqrt{h_{1:t}} - \sqrt{h}_{1:t-1}) ||\boldsymbol{z}^* - \boldsymbol{z}_t||^2 + \sum_{t=1}^{T} \frac{h_t}{\sigma\sqrt{h_{1:t}}} \qquad (5.15) \\ &\leq \frac{\sigma}{2} \sum_{t=1}^{T} (\sqrt{h_{1:t}} - \sqrt{h}_{1:t-1}) 2D^2 + \sum_{t=1}^{T} \frac{h_t}{\sigma\sqrt{h_{1:t}}} \end{aligned}$$

We use the first order definition of convexity on the square root function to get:

$$\begin{aligned} \sqrt{h_{1:t}} - \sqrt{h_{1:t-1}} &\leq \frac{1}{2\sqrt{h_{1:t}}} (h_{1:t} - h_{1:t-1}) \\ &= \frac{h_t}{2\sqrt{h_{1:t}}} \end{aligned}$$

Thus,

$$R(T) \leq \frac{\sigma}{4} \sum_{t=1}^{T} \frac{h_t}{\sqrt{h_{1:t}}} 2D^2 + \sum_{t=1}^{T} \frac{h_t}{\sigma\sqrt{h_{1:t}}} \tag{5.16}$$

From [124, Lemma 3.5], we have:

$$\sum_{t=1}^{T} \frac{h_t}{\sqrt{h_{1:t}}} \leq 2\sqrt{h_{1:t}} \tag{5.17}$$

Plugging this result into (5.16), it yields:

$$R(T) \leq \sqrt{h_{1:t}}(\frac{2}{\sigma} + \frac{\sigma}{2}2D^2) \tag{5.18}$$

$\square$

Remark 1. We observe that a certain value of σ can minimize the upper-bound on the regret, but one has to know the diameter of the decision set $\sqrt{2}D$. The very value of σ which minimizes the upper-bound is:

$$\sigma = \frac{\sqrt{2}}{D} \tag{5.19}$$

We re-write the upper bound:

$$\boxed{R(T) \leq 2\sqrt{2}D\sqrt{\sum_{t=1}^{T} ||\nabla f_t(\boldsymbol{z}_t) - \nabla \widehat{f}_t(\widehat{\boldsymbol{z}}_t)||^2}} \tag{5.20}$$

Remark 2. The regret bound is in $\mathcal{O}(\sqrt{T})$, and becomes *null* when the predictions are perfect, i.e. when $\forall t, \quad \nabla \widehat{f}_t(\widehat{\boldsymbol{z}}_t) = \nabla f_t(\boldsymbol{z}_t)$.

Remark 3. We implement an online learning prediction method [119] that learns how to predict the gradient with the regret $\mathcal{O}(2mGM\sqrt{T})$, where G and M are key constant in [119], and m is the number or resource types. Other prediction methods could be applied to the prediction of the gradient; however, this online learning method offers sublinear regret guarantees against all types of traces, even non-stationary.

5.3.3 Model Extension

We envision the case where the NO fails to fulfill the SP request in its entirety but must still comply with at least a certain ratio of what has been requested. Formally, we multiply the elements of the request vector $\boldsymbol{x}_t$ with a vector $\boldsymbol{\alpha}_t$ whose items belong to the set $[\alpha, 1]$, where the value of α is representative of the SLA the NO and the SP have agreed upon. Similarly, we multiply the request vector on the spot market $\boldsymbol{y}_t$ with a vector $\boldsymbol{\beta}_t$, whose items belong to $[\beta, 1]$.

We redefine the problem as:

$$(\mathbb{P})_{ext}: \quad \max_{\{\boldsymbol{x}_t,\{\boldsymbol{y}_t\}\}_{t=1}^T} \sum_{t=1}^{T} \Big(V a_t \log((\tilde{\boldsymbol{x}}_t + \tilde{\boldsymbol{y}}_t)^\top \boldsymbol{\theta}_t + 1) - (\boldsymbol{p}_t^\top \tilde{\boldsymbol{x}}_t + \boldsymbol{q}_t^\top \tilde{\boldsymbol{y}}_t) \Big) \tag{5.21}$$

$$\text{s.t.} \quad \boldsymbol{y}_t \in \Delta, \quad \forall t = 1, \ldots, T, \tag{5.22}$$

$$\boldsymbol{x}_t \in \Delta, \quad \forall t = 1, \ldots, T. \tag{5.23}$$

$$\tag{5.24}$$

where $\tilde{\boldsymbol{x}}_t = \boldsymbol{\alpha}_t \odot \boldsymbol{x}_t$ and $\tilde{\boldsymbol{y}}_t = \boldsymbol{\beta}_t \odot \boldsymbol{y}_t$. The notation $\odot$ corresponds to the element-wise multiplication of two vectors also called the Adamar product. The latter is a linear operator, as it is equivalent to the product $A\boldsymbol{x}_t$ ($B\boldsymbol{y}_t$), where A (B) is a diagonal matrix whose elements are the items of the vector $\boldsymbol{\alpha}_t$ ($\boldsymbol{\beta}_t$). Thus it conserves the convexity of the problem and we can still apply the same OOLR solution.

5.4 Numerical Evaluation

5.4.1 Experimental scenario

We consider an MVNO which aims to acquire network resources that constitute the end-to-end network slice dedicated to its specific network service. Confronted with unknown and evolving traces such as the users' demand, the prices, and contributions of the network resources, the MVNO will follow the online reservation strategy designed by our OOLR solution. We consider the base case where the MVNO faces the incoming demand at one BS and must reserve $m = 3$ types of resources to deliver its network service, encompassing radio resources at the BS, backhaul link capacity, and computing resources at the core. This base case falls under the scope of our system model.

To model user demand, we use a real-world data set that contains the aggregated traffic volumes seen across multiple BSs owned by a major MNO of Shanghai. We refer the reader to subsection 3.5.1 for more details about the data set we use.

We assume the network resources prices vary with non-stationary dynamics. We model such variations with an AR(1) (Auto-Regressive with 1 lag) process, the discrete-time equivalent of the Ornstein-Ulhenbeck (OU) process. This stochastic process is applied in financial mathematics to model stock prices. We model the contribution parameters -items of vector $\boldsymbol{\theta}_t$- as varying and non-stationary. Each item follows a seasonal trend (sine wave), with an offset and added OU stochastic process.

We compare the OOLRgrad solution to the FTRL baseline. The latter consists of the update as defined in equation (5.8). We introduce the parameter ζ to control the quality of different prediction models, where ζ is the average relative error rate of the prediction $\nabla\widehat{f}_{t+1}(\widehat{\boldsymbol{z}}_{t+1})$ against the real value $\nabla f_{t+1}(\boldsymbol{z}_{t+1})$. We set $\zeta = 0, 0.3$ and 4 to represent prediction models from perfect accuracy to arbitrarily bad. This allows us to introduce three OOLR baselines, with different levels of prediction accuracy.

5.4.2 Technical implementation

We present numerical tests that verifies the regret bound presented in section 5.3. We also compare the OOLR solution to the FTRL baseline. The solutions, available to the community at our public GitHub[2], are developed under Python 3.6 programming environment and we use scipy.optimize to solve the quadratic problems of the OOLR solution (5.12) and of the FTRL baseline (5.8). The ipopt convex solver of Python is applied to find the optimal static and dynamic solutions of the problem ($\mathbb{P}$).

We provide an overview of our simulation architecture in Fig. 5.1. We generate the traces of demand, prices and contributions which are the inputs of our online learning

[2]https://github.com/jeanba19/OOLRgrad

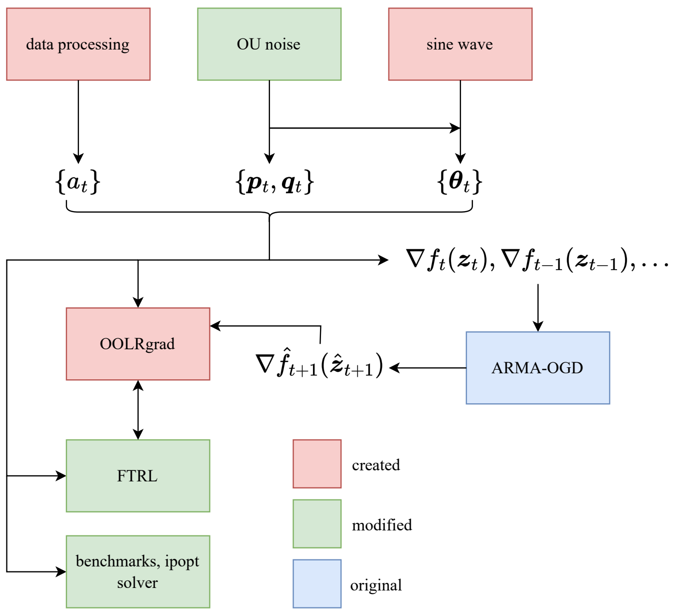

data processing
OU noise
sine wave
$\{a_t\}$
$\{\boldsymbol{p}_t, \boldsymbol{q}_t\}$
$\{\boldsymbol{\theta}_t\}$
$\nabla f_t(\boldsymbol{z}_t), \nabla f_{t-1}(\boldsymbol{z}_{t-1}), \ldots$
OOLRgrad
$\nabla \hat{f}_{t+1}(\hat{\boldsymbol{z}}_{t+1})$
ARMA-OGD
FTRL
benchmarks, ipopt solver
created
modified
original

Figure 5.1: Architecture of our simulation

solution OOLRgrad. In parallel, we compute the past gradient values based on these traces and we predict the next slot gradient value with the ARMA-OGD model. The prediction is incorporated into the OOLRgrad solution, which we then compare with the FTRL baseline.

5.4.3 Prediction module

The solution OOLR is *optimistic* in the sense it allows the SP to use the predicted gradient term $\nabla \widehat{f}_{t+1}(\widehat{z}_{t+1})$ of the next slot. In (5.13), we concluded that accurate predictions can greatly enhance the performance, as the regret bound goes from $\mathcal{O}(\sqrt{T})$ when predictions are arbitrarily bad to $\mathcal{O}(1)$ when predictions are almost perfect. This observation paves the way to the introduction of a prediction module, in support of our OOLR decision algorithm. We aim to find an accurate, robust and computationally low model. The algorithm ARMA-OGD created by Anava et al. in [119] presents these three key advantages. It consists of learning the AR(q) signal of the trace where the q lag coefficients are updated online at each slot by the gradient descent method. The algorithm guarantees that the total loss is no more on average than the loss of the best ARMA predictor with full hindsight.

First, we show in Fig. 5.2 that the model is accurate against two intricate signals. The SP demand is based on multiple latent factors, which makes the signal non-stationary and hard to predict. Yet, we observe the predicted signal is able to track the SP demand. The

$2m$ gradient items are composed of multiple signals, namely the SP demand, the prices and contributions of the network resources, as the reader can observe in the following expression of the gradient:

$$\nabla f_t(\boldsymbol{z}_t) = \begin{bmatrix} -V\frac{a_t\theta_{t,1}}{1+(\boldsymbol{x}_t+\boldsymbol{y}_t)^\top\boldsymbol{\theta}_t} + p_{t,1} \\ -V\frac{a_t\theta_{t,2}}{1+(\boldsymbol{x}_t+\boldsymbol{y}_t)^\top\boldsymbol{\theta}_t} + p_{t,2} \\ \vdots \\ -V\frac{a_t\theta_{t,m}}{1+(\boldsymbol{x}_t+\boldsymbol{y}_t)^\top\boldsymbol{\theta}_t} + p_{t,m} \\ -V\frac{a_t\theta_{t,1}}{1+(\boldsymbol{x}_t+\boldsymbol{y}_t)^\top\boldsymbol{\theta}_t} + q_{t,1} \\ -V\frac{a_t\theta_{t,2}}{1+(\boldsymbol{x}_t+\boldsymbol{y}_t)^\top\boldsymbol{\theta}_t} + q_{t,2} \\ \vdots \\ -V\frac{a_t\theta_{t,m}}{1+(\boldsymbol{x}_t+\boldsymbol{y}_t)^\top\boldsymbol{\theta}_t} + q_{t,m} \end{bmatrix} \tag{5.25}$$

Yet again, the model is able to give an accurate predicted signal. Secondly, ARMA-OGD provides guarantees of performance against all types of traces, which ensures its robustness. The total squared loss of the model is a $\mathcal{O}(\sqrt{T}) + Res$, where Res represents the residual squared loss of the best ARMA predictor with full hindsight of the target signal. We show in Fig. 5.3 the convergence of the average squared loss towards Res. Finally, the ARMA-OGD is based on the OGD update step, which is very low computationally and allows us to develop the algorithm alongside the OOLR solution. We insist here that the two combined solutions having both low time complexity allow the SP to take *optimistic decisions in real time.* There exists other models which employ advanced techniques such as Neural Networks that would obtain better accuracy than the ARMA-OGD. Nevertheless, these models necessitate an offline training phase, do not provide guarantees of performance, and have higher time complexity.

5.4.4 Impact of the quality of predictions

The SP can reserve $m = 3$ kinds of resources. We assume the NO sets the upper bound constraint to $D_i = 1, \forall i$. This means the SP reserves normalized values for each type of resource. Our goal is to maximize the SP utility while avoiding excessive reservation cost. We balance between the two terms (utility and cost) using the hyper-parameter V.We

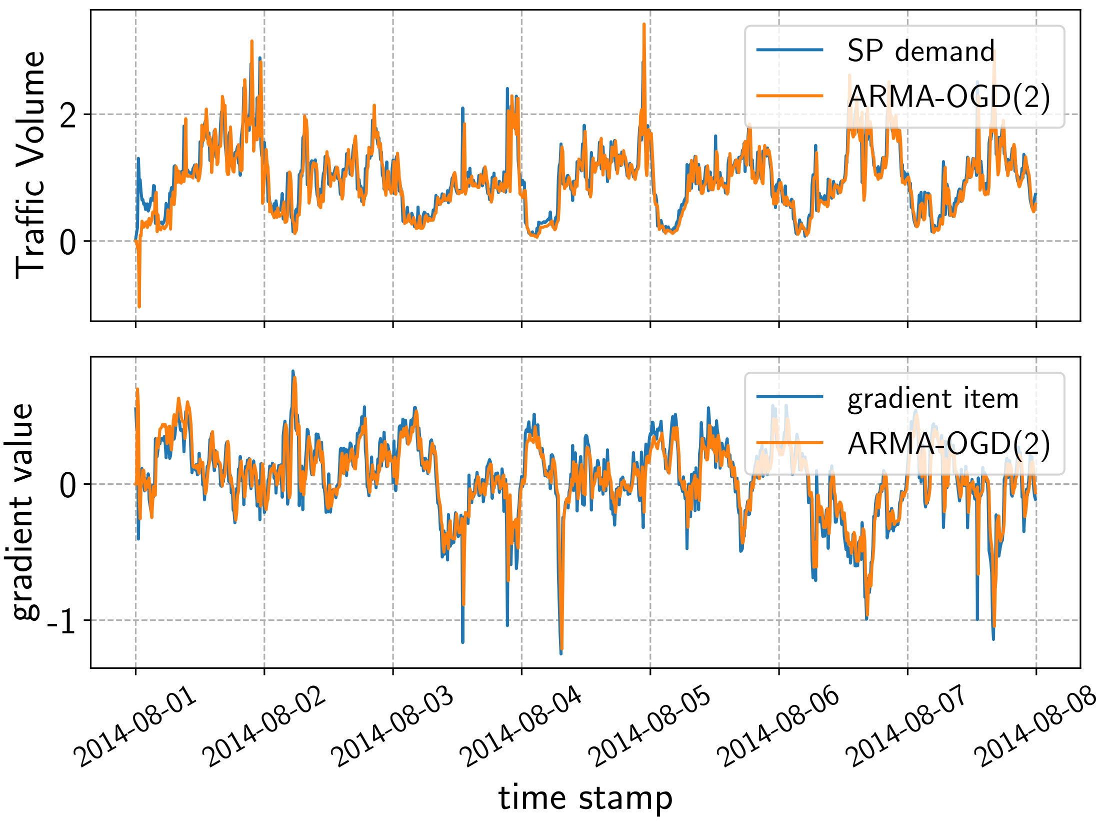


Figure 5.2: The x-axis encompasses the first week of August period. *Upper part:* the predicted signal against the real-world MVNO demand signal. The y-axis values are normalized. *Lower part:* the predicted signal against the first of the gradient $2m$ items.

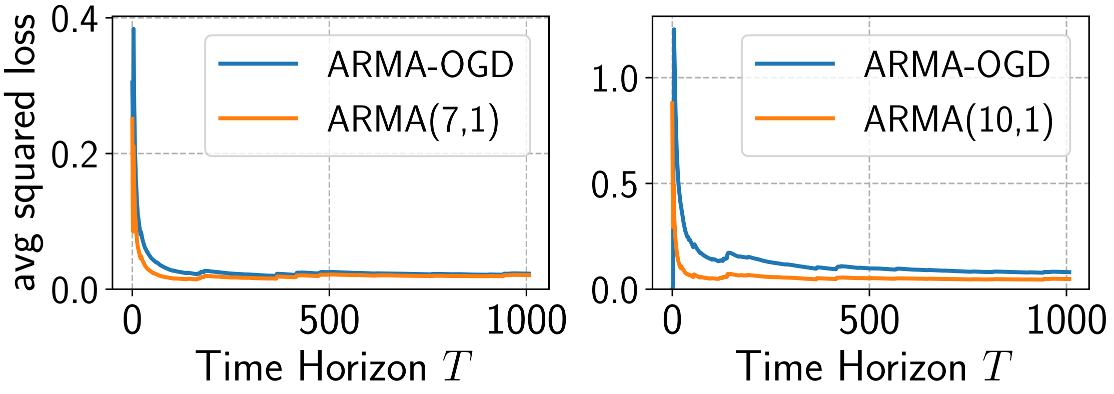
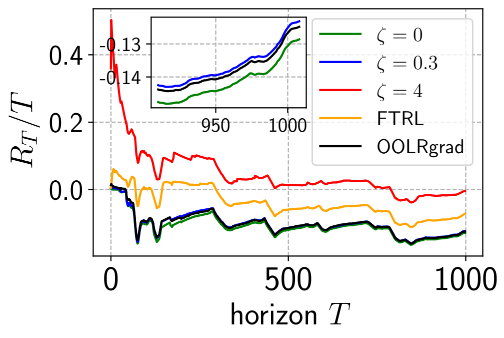


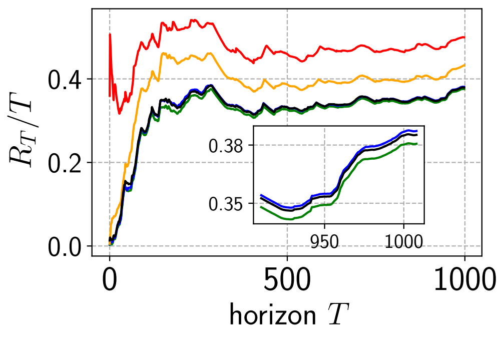
Figure 5.3: We evaluate the accuracy of the models on the first week of August. *Left side:* We observe the convergence of the average squared loss of the predicted gradient first item towards the best ARMA in hindsight. *Right side:* We observe the convergence of the average squared loss of the predicted MVNO demand toward the best ARMA in hindsight.

calibrate $V = 2$ to have both terms of the same order.

We call OOLRgrad the online decision algorithm OOLR because the prediction method ARMA-OGD is directly applied to the gradient items. We expect a regret bound of $\mathcal{O}(\sqrt{Res + \sqrt{T}})$. We show in Fig. 5.4(a) against the static benchmark the performance of the OOLRgrad solution, the classical FTRL algorithm with euclidean regularizer and the different OOLR models $\zeta = 0, 0.3$ and 4. We first observe the convergence of the average regret R_T/T towards 0 for the five models, which confirm the regret bound of $\mathcal{O}(\sqrt{T})$ even for arbitrarily bad predictions (represented by the OOLR $\zeta = 4$ model). Secondly, we observe a negative regret for the other four models, which confirm the $\mathcal{O}(1)$ regret bound when the predictions are accurate and the accumulated error $\sum_{t=1}^{T} ||\nabla f_t(\boldsymbol{z}_t) - \nabla \widehat{f}_t(\widehat{\boldsymbol{z}}_t)||^2$ is close to 0. Zooming in the last slots, we remark that our OOLRgrad solution based on the ARMA-OGD predictor shows better performance than the OOLR solution with a 70% accurate predictor ($\zeta = 0.3$) and is inferior to the OOLR with perfect predictor ($\zeta = 0$). The OORLgrad and the OOLR $\zeta = 0, 0.3$ solutions outperform the FTRL baseline, which shows that the incorporation of accurate predictions enhances the performance. One needs to be cautious as arbitrarily bad predictions (OOLR $\zeta = 4$) worsens the performance. In Fig. 5.4(b), we show the performance of the same solutions against the optimal benchmark. Against such competitive benchmark, the regret cannot be sublinear and thus the convergence of R_T/T towards 0 is not achieved. Nevertheless, we observe that the OOLRgrad solution displays good performance when compared to the different baselines.



(a) Static regret



(b) Dynamic regret

Figure 5.4: *Evolution of the regret R_T/T:* Horizon $T = 1008$, $m = 3$, $V = 2$, $\boldsymbol{D} = [1, 1, 1]$, $D = \sqrt{3}$, $\sigma = \sqrt{2}/D$.

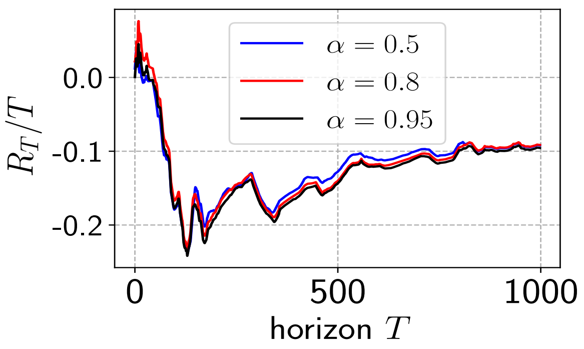


Figure 5.5: *Evolution of static R_T/T:* Horizon $T = 1008$, $m = 3$, $V = 2$, $\boldsymbol{D} = [1, 1, 1]$, $D = \sqrt{3}$, $\sigma = \sqrt{2}/D$.

5.4.5 Extension

Now we evaluate the OOLRgrad solution in the scenario where the NO is unable to fulfill the SP request in its entirety. We focus on a basic scenario in which the NO ensures a minimum ratio of α for all kinds of resources – in a more complex scenario the NO commits to a ratio of α_i for each resource i, where α_i are possibly different. Thus, for each resource i at slot t, the SP expect to receive a ratio $\alpha_{i,t}$ that belongs to the set $[\alpha, 1]$. We draw the $\{\alpha_{i,t}\}_t$ from the uniform distribution on $[\alpha, 1]$. We assume that the NO consistently deliver all requested spot resources, thus we keep $\beta = 1$. We observe in Fig. 5.5 the regret performance of the OOLRgrad solution for three different SLAs, which are $\alpha \in \{0.5, 0.8, 0.95\}$. We observe that the performance stays similar regardless of the SLA the SP has complied for, which implies our OOLRgrad solution is consistently applicable.

5.5 Conclusion

In this chapter, we introduced the OOLR algorithm that allows the SP to make reservations under uncertainty while incorporating predictions about the future gradient. We then proposed to combine this decision model with a prediction model, thus creating the OOLRgrad solution with better performance than the classical FTRL solution. The OORLgrad solution presents a regret bound of $\mathcal{O}(\sqrt{Res + \sqrt{T}}) \sim \mathcal{O}(\sqrt{\alpha T + \sqrt{T}}) \sim \mathcal{O}(\sqrt{T})$. The accumulated error Res of the best ARMA model with full hindsight grows with T. Depending on the nature of the traces, the parameter α can be very low and counterbalance the growth of T.

We focus here on a very volatile case with real world and non-stationary traces (see Fig. 5.2), and still observe a low value of α in Fig. 5.3. It would be interesting at this stage to find the family of traces for which $Res = \mathcal{O}(\sqrt{T})$, leading to an improved regret bound of $\mathcal{O}(T^{1/4})$.

6 Conclusions and future directions

This thesis has focused on the resource provisioning/reservation problem from the SP's perspective in virtualized communication networks. This chapter contains a summary of the main achievements of the thesis, discusses how to extend the research and presents the future trends of network slicing. Section 6.1 provides a high-level discussion of the achievements of the thesis, section 6.2 discusses the limitations of the thesis and investigates the potential directions for the future research, and section 6.3 presents the future challenges in 6G networks.

6.1 Thesis achievements

The main goal of this thesis has been to design the SP reservation policy of network slices. Specifically, we addressed three research questions – that we summarize below – to realize this goal. RQ1 focuses on the type of request the SP must formulate to acquire a complete E2E network slice. RQ2 considers the reservation of the SP in the context of RAN. RQ3 investigates the buy-sell relation between the SP and the NO in the network slicing market. We begun with a machine learning solution to address RQ2, i.e. the design of a resource reservation policy for the SP in the context of RAN. We then extended the scope of the reservation to multiple types of resources by addressing RQ1 with the design of an online reservation policy based on OCO. We addressed RQ3 in the same time by considering the price of the resources in the SP optimization process. We gave guarantees of performance, either experimental or mathematical, for each of our solutions. We relied on the tools of OCO to build analytic expressions of the worst performance our solutions could produce. We discussed trade-offs on the key parameters of our decision models.

We considered first the reservation of RAN resources to address RQ2. We applied supervised learning tools and characterized the optimal reservation policy to generate the labels for training. The DNN and LSTM solutions were trained offline and then validated online. Simulating different network traffic conditions, we evaluated the solutions against the ARIMA baseline. We defined three different metrics to assess the performance of our solutions: the MSE, the over-reservation and the under-reservation. The supervised solutions produced better results for multi-steps ahead reservations and reduced significantly the over-reservation. These solutions are promising, as they can perform decisions on the SP's requests a long time ahead. The SP can plan its needs of radio capacity at each BS and communicate them to the MNO. Ultimately, this type of request may render the resource allocation task of the MNO easier than high-level QoS specifications. The MNO would have to accept or reject each request through admission control. Nevertheless, they necessitate an offline training phase, which requires the access to historical data of traffic at multiple BSs of the network.

We then extended the scope of the reservation model to multiple types of resources (m in total). The optimization formulation allowed us to consider the cost of the reservation, which counter-balances the utility driven from such reservation, through a constrained convex program where the objective is to maximize the utility, under a budget constraint for the purchase of resources. Given such formulation, there exists a natural trade-off between the maximization of the utility with big reservations and the minimization of the cost with small reservations, that we were able to control through the hyper-parameter/learning rate μ of our solution algorithm. The designed solution is a primal-dual algorithm inspired by [1], where we minimize the Lagrangian for the primal update and we perform gradient ascent of the Lagrangian for the dual update. The basic algorithm was extended into two directions: the reservation of multiple types of resource that can compose a network slice, and the reservation of additional spot resources yielding to slice re-configuration. We provided regret and fit bounds. We thoroughly analyzed the budget parameter B. Unsurprisingly, a very high value of allocated budget provides the best result, which favors wealthy SPs. Nevertheless, we found that the value $B = (p + Kq)/2$ minimizes the regret bound, and the value $B = (p + Kq)/8$ brings forth a peak of performance, which would benefit a cost-saving strategy of the SP.

We finally expanded the model to include predictions into the main reservation decision solution. We applied an optimistic online learning solution to our problem by incorporating key prediction of the next slot gradient into the decision model. The typical solution is based on the FTRL algorithm and considers an additional *optimistic* term, which is the prediction. These optimistic methods provide for the static regret a bound of $\sqrt{T}$ when the accumulated prediction error is growing with T and a bound of $\mathcal{O}(1)$ when the predictions are perfect. We then associated the decision model with a prediction model based on online gradient descent. Our combined solution outperformed the FTRL algorithm and performed almost as good as if the decision model received perfect predictions of the next slot gradient. We evaluated the solution under real world settings and considered the case where the NO cannot fulfill the entire request of the SP due to resource scarcity/high number of requests. We observed that even with a 50% acceptance rate, the performance of the SP does not worsen. The derived optimistic solution, if given accurate predictions from a reliable model, yields to improved level of performance. Moreover, the performance of the solution is robust to congested traffic conditions that cause the NO to fail to address the request.

6.2 Futures directions of the research

The most immediate extension of our research is the sequel of chapter 5. The optimistic methods with dynamic regret analysis have been recently explored in the literature [125, 126]. Building on such solutions, e.g. the optimistic online mirror descent in [126], we can provide dynamic regret guarantees and derive new lemmas specific to our resource reservation problem. With no hindrance to the mathematical properties of the formulated problem, we can also refine the reservation model by considering more than two timescales. This will unlock the reservation of resources with different lifetimes, i.e. the lease of VMs typically lasts one hour [36], sub-channels for data packet transmission a few seconds [46]. Unlike [46, 47], our reservation model lacks a resource allocation policy. In the different chapters, we only consider the aggregated demand a_t of a set of users. We can modify this term in the objective by a set of individual users $i \in \mathcal{N}$ with specific demand a_t^i. The aggregated demand becomes $\sum_{i=1}^{|\mathcal{N}|} = a_t^i$. Once resources have been reserved for this equivalent term of aggregated demand, we can apply a dynamic resource allocation policy which maximizes the total utility and consider fairness between the users. We remark that only some SPs like MVNOs have a level of control on the network and can perform resource allocation.

The scope of the SP's perspective is limited in the context of network slicing. Taking a step back to see the broader picture from the perspective of the NO is an interesting in-depth study of our solutions. Considering multiple SPs using our reservation model to decide their requests, the NO would complete the admission control to maximize its own profit. This work could complement other approaches like [26], [17], where the authors consider requests following well-behaved probability distributions, e.g. exponential, uniform. We can evaluate the network utilization, the net profit of the NO, the net social welfare of the NO and SPs under these settings.

A promising direction of the current research would be to assess the performance of the developed solutions on the existing families of slices: URLLC, eMBB and mMTC, or on use cases like video streaming services. The general formulation in chapter 4 can be adapted to specific cases, as long as the convexity of the formulated problem is conserved. Specifically, we consider to apply the designed reservation model to the video streaming use case [127], where the problem is typically to maximize a utility function while complying with capacity or budget constraints.

6.3 Future trends in network slicing

In RAN slicing, the resource sharing remains the most challenging part faced by the operators. The dynamic sharing and allocation of radio resources renders impossible the isolation and poses reliability and safety issues, especially for critical services; while the static partition yields to under- and over-utilization as the demand from each SP varies over time. In chapter 3, we partly addressed this issue by anticipating the allocation of dedicated and isolated resources with in advance requests. Nevertheless, the reservations made long in advance are far from optimal. The virtualized RAN is a promising trend which pools several base station workloads on the same processing hardware [128]. Most BS functions, can be relocated to a central unit, while the radio units manage physical layer tasks and exchange data with the central unit [32]. The RAN/BS functions placement either at the radio units or the central unit and the routing path decisions between unit pairs can be optimized to obtain a cost efficient RAN [32]. This virtualization technique of RAN resources paves the way to the application of RAN slicing.

Another trend of the research is the design of efficient MEC networks. MEC (Multi-access Edge Computing or Mobile Edge Computing) consists of relocating computing resources close the RAN edge of the network. This new network architecture offers proximity (servers are close to the mobile end users), and lower latency (servers can process compute-intensive applications) [129]. The ETSI white paper [130] lists different envisioned scenarios: augmented reality, intelligent video acceleration, connected cars, IoT gateway. The MEC architecture offers a promising alternative to classical RAN with the regard to the development of these use cases. Nevertheless, the deployment of MEC networks poses real challenges with regard to the resource allocation. The number of users and applications grows and computing resources at the edge remain scarce. A distributed resource allocation adapting to the different objectives of the use cases and the different constraints of the network is the most interesting direction.

7 Appendices

7.1 Proof of Lemma 1

The proof of this lemma becomes trivial with the visual help of Fig. 7.1. In the first case, the sum of the MNO and the SP traces is less than the capacity c, thus the optimal reservation is zero. In the second case, the sum is more than c, with the MNO trace less than c. The optimal choice for the SP is to have $c - l_k$ served as best-effort traffic and thus must only reserve $s_k - (c - l_k)$. In the third case, the MNO trace is more than c. Thus the SP must reserve for its entire trace s_k. In the fourth case, the SP trace is more than c. As the SP cannot reserve more than the available capacity c, then the optimal reservation is c. This yields to the claim of lemma 1, *if* $s_k + l_k \leq c$ *then* $r_k^* = 0$, *else* $r_k^* = \min(\min(s_k - (c - l_k), s_k), c)$.

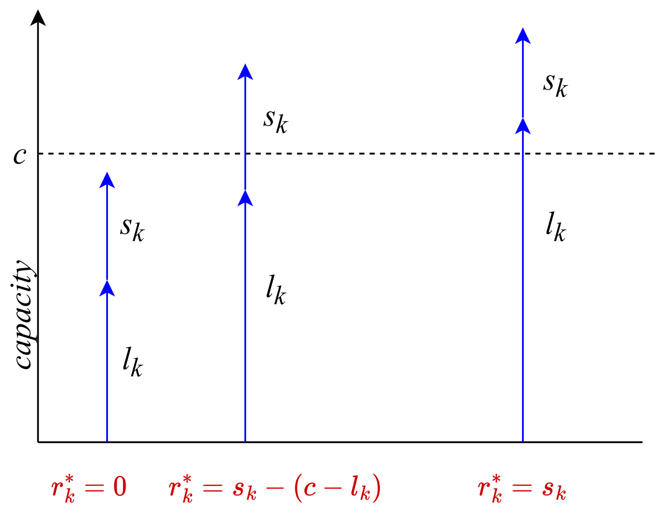


Figure 7.1: Different cases for the optimal reservation

7.2 Applicability of [1, Theorems 1 and 2] to our problem

In this section, we go through the details of the proofs in [1] to verify the applicability of their theorems to our case. This section does not provide any new contribution and is entirely based on the proofs presented in [1].

We first verify the applicability of Theorem 1. We start by the dual upper bound $\tilde{\lambda}$. From it, we will derive the upper bound on the fit V_T

Definition 1. *The dual drift is defined as* $\Delta(\lambda_t) := (\lambda_{t+1}^2 - \lambda_t^2)/2$.

We find an upper bound for the dual drift:

$$\begin{aligned}\lambda_{t+1}^2 &= [\lambda_t + \mu g_t(\boldsymbol{z}_t)]^{+2} \\ &\le (\lambda_t + \mu g_t(\boldsymbol{z}_t))^2 \\ &= \lambda_t^2 + 2\mu\lambda_t g_t(\boldsymbol{z}_t) + \mu^2 g_t(\boldsymbol{z}_t)^2\end{aligned} \tag{7.1}$$

Thus,

$$\Delta(\lambda_t) \le \mu\lambda_t g_t(\boldsymbol{z}_t) + \frac{\mu^2}{2} g_t(\boldsymbol{z}_t)^2 \tag{7.2}$$

We use the function $L_t(\boldsymbol{z}, \lambda_{t+1})$ defined in (4.11) and we recall that $\boldsymbol{z}_{t+1}$ is the optimal solution of the problem $\min_{\boldsymbol{z}\in\mathcal{Z}} L_t(\boldsymbol{z}, \lambda_{t+1})$. Thus, for any interior point $\tilde{\boldsymbol{z}}_t$, such that $g_t(\tilde{\boldsymbol{z}}_t) \le -\epsilon$, we have:

$$L_t(\boldsymbol{z}_{t+1}, \lambda_{t+1}) \le L_t(\tilde{\boldsymbol{z}}_t, \lambda_{t+1}) \tag{7.3}$$

which leads to:

$$\begin{aligned}
\lambda_{t+1} g_t(\boldsymbol{z}_{t+1}) &\overset{(a)}{\leq} \nabla f_t(\boldsymbol{z}_t)^\top (\tilde{\boldsymbol{z}}_t - \boldsymbol{z}_t) - \nabla f_t(\boldsymbol{z}_t)^\top (\boldsymbol{z}_{t+1} - \boldsymbol{z}_t) \\
&\quad + \lambda_{t+1} g_t(\tilde{\boldsymbol{z}}_t) + \frac{1}{2\nu} \|\tilde{\boldsymbol{z}}_t - \boldsymbol{z}_t\|^2 - \frac{1}{2\nu} \|\boldsymbol{z}_{t+1} - \boldsymbol{z}_t\|^2 \\
&\overset{(b)}{\leq} \nabla f_t(\boldsymbol{z}_t)^\top (\tilde{\boldsymbol{z}}_t - \boldsymbol{z}_t) - \nabla f_t(\boldsymbol{z}_t)^\top (\boldsymbol{z}_{t+1} - \boldsymbol{z}_t) \\
&\quad - \epsilon \lambda_{t+1} + \frac{1}{2\nu} \|\tilde{\boldsymbol{z}}_t - \boldsymbol{z}_t\|^2 - \frac{1}{2\nu} \|\boldsymbol{z}_{t+1} - \boldsymbol{z}_t\|^2 \\
&\overset{(c)}{\leq} \nabla f_t(\boldsymbol{z}_t)^\top (\tilde{\boldsymbol{z}}_t - \boldsymbol{z}_t) - \nabla f_t(\boldsymbol{z}_t)^\top (\boldsymbol{z}_{t+1} - \boldsymbol{z}_t) \\
&\quad - \epsilon \lambda_{t+1} + \frac{E^2}{2\nu} \\
&\overset{(d)}{\leq} \|\nabla f_t(\boldsymbol{z}_t)\| \, \|\tilde{\boldsymbol{z}}_t - \boldsymbol{z}_t\| + \|\nabla f_t(\boldsymbol{z}_t)\| \, \|\boldsymbol{z}_{t+1} - \boldsymbol{z}_t\| \\
&\quad - \epsilon \lambda_{t+1} + \frac{E^2}{2\nu} \\
&\overset{(e)}{\leq} 2GE - \epsilon \lambda_{t+1} + \frac{E^2}{2\nu}
\end{aligned} \tag{7.4}$$

where (a) follows from (7.3), rearranging the terms to isolate $\lambda_{t+1} g_t(\boldsymbol{z}_{t+1})$, (b) uses $g_t(\tilde{\boldsymbol{z}}_t) \leq -\epsilon$, (c) holds since $\mathcal{Z}$ confines $\|\tilde{\boldsymbol{z}}_t - \boldsymbol{z}_t\|^2 \leq E^2$ and $\|\boldsymbol{z}_{t+1} - \boldsymbol{z}_t\|^2 \geq 0$, (d) uses the Cauchy-Schwartz inequality twice, (e) leverages the upper bounds from (7.25) and (7.27).

Plugging (7.4) into (7.2), we have

$$\begin{aligned}
\Delta(\lambda_{t+1}) &\leq \mu \lambda_{t+1} g_{t+1}(\boldsymbol{z}_{t+1}) + \frac{\mu^2}{2} g_{t+1}(\boldsymbol{z}_{t+1})^2 \\
&\overset{(f)}{\leq} \mu(\lambda_{t+1} g_{t+1}(\boldsymbol{z}_{t+1}) - \lambda_{t+1} g_t(\boldsymbol{z}_{t+1})) \\
&\quad + \mu(2GE - \epsilon \lambda_{t+1} + \frac{E^2}{2\nu}) + \frac{\mu^2 M^2}{2} \\
&\overset{(g)}{\leq} \mu \lambda_{t+1} [g_{t+1}(\boldsymbol{z}_{t+1}) - g_t(\boldsymbol{z}_{t+1})]^+ - \mu \epsilon \lambda_{t+1} \\
&\quad + 2\mu GE + \frac{\mu E^2}{2\nu} + \frac{\mu^2 M^2}{2} \\
&\overset{(h)}{\leq} \mu \lambda_{t+1} (\tilde{U}_g - \epsilon) \\
&\quad + \mu(2GE + \frac{E^2}{2\nu} + \frac{\mu M^2}{2})
\end{aligned} \tag{7.5}$$

where (f) uses the trick to add and retrieve the same quantity $\lambda_{t+1} g_t(\boldsymbol{z}_{t+1})$ and leverages the bounds from (7.26) and (7.4), (g) holds since $\mu \lambda_{t+1} \geq 0$, (h) comes from the definition of $U_g^t = \max_{\boldsymbol{z} \in \mathcal{Z}} |[g_{t+1}(\boldsymbol{z}_{t+1}) - g_t(\boldsymbol{z}_{t+1})]^+|$ and $\forall t, \quad U_g^t \leq \tilde{U}_g$.

We finish the proof of the dual upper bound by contradiction. Let's presume $t+2$ is the first period for which the dual upper bound $\tilde{\lambda}$ as defined in Lemma 2 does not hold. Therefore,

$$|\lambda_{t+1}| \leq \mu M + \frac{2GE + E^2/(2\nu) + (\mu M^2)/2}{\epsilon - \tilde{U}_g} \tag{7.6}$$

and

$$|\lambda_{t+2}| > \mu M + \frac{2GE + E^2/(2\nu) + (\mu M^2)/2}{\epsilon - \tilde{U}_g} \tag{7.7}$$

Working on $|\lambda_{t+1}|$, we get

$$\begin{aligned} |\lambda_{t+1}| &= |\lambda_{t+2} - (\lambda_{t+2} - \lambda_{t+1})| \\ &\geq |\lambda_{t+2}| - |\lambda_{t+2} - \lambda_{t+1}| \\ &= |\lambda_{t+2}| - \left|[\lambda_{t+1} + \mu g_{t+1}(\boldsymbol{z}_{t+1})]^+ - \lambda_{t+1}\right| \\ &= |\lambda_{t+2}| - |\mu g_{t+1}(\boldsymbol{z}_{t+1})| \\ &\geq |\lambda_{t+2}| - \mu M \\ &> \frac{2GE + E^2/(2\nu) + (\mu M^2)/2}{\epsilon - \tilde{U}_g} \end{aligned} \tag{7.8}$$

If we multiply both sides by $\mu(\tilde{U}_g - \epsilon)$, which is strictly negative thanks to Assumption 7, (7.8) is equivalent to

$$\mu(\tilde{U}_g - \epsilon)\,|\lambda_{t+1}| < -\mu(2GE + \frac{E^2}{2\nu} + \frac{\mu M^2}{2}) \tag{7.9}$$

Passing all the terms on the left side, we deduce from (7.5)

$$\Delta(\lambda_{t+1}) < 0 \tag{7.10}$$

which means that $|\lambda_{t+2}| < |\lambda_{t+1}|$ and that contradicts our presumption! As we set $\lambda_1 = 0$, then $|\lambda_2| \leq \mu M$. Thus, for every $t \geq 1$, $|\lambda_t| \leq \tilde{\lambda}$ holds. The proof for the fit V_T

quickly follows:

$$\begin{aligned}\lambda_{T+1} &= [\lambda_T + \mu g_T(\boldsymbol{z}_T)]^+ \\ &\geq \lambda_T + \mu g_T(\boldsymbol{z}_T) \geq \lambda_1 + \sum_{t=1}^{T} \mu g_t(\boldsymbol{z}_t)\end{aligned} \tag{7.11}$$

Then we have

$$\sum_{t=1}^{T} g_t(\boldsymbol{z}_t) \leq \frac{\lambda_{T+1}}{\mu} \tag{7.12}$$

as $\lambda_1 = 0$. Non-negativity of λ_{T+1} implies

$$\begin{aligned}&[\sum_{t=1}^{T} g_t(\boldsymbol{z}_t)]^+ \leq \frac{\lambda_{T+1}}{\mu} \\ \Longleftrightarrow &\left|[\sum_{t=1}^{T} g_t(\boldsymbol{z}_t)]^+\right| \leq \frac{|\lambda_{T+1}|}{\mu} \\ \Longleftrightarrow &V_T \leq \frac{|\lambda_{T+1}|}{\mu} \leq \frac{\left|\tilde{\lambda}\right|}{\mu}\end{aligned} \tag{7.13}$$

which completes the proof and verifies our fit bound of Lemma 2.

Secondly, we verify the applicability of Theorem 2, which validates our regret bound of Lemma 2. It can be shown that $L_t(\boldsymbol{z}, \lambda)$, which we will denote as $L_t(\boldsymbol{z})$ for brevity, is $1/\nu$-strongly convex with regard to $\boldsymbol{z}$, which implies that for any $\boldsymbol{x}, \boldsymbol{y}$, we have [80, Chapter 2.1]

$$L_t(\boldsymbol{y}) \geq L_t(\boldsymbol{x}) + \nabla L_t(\boldsymbol{x})^\top (\boldsymbol{y} - \boldsymbol{x}) + \frac{1}{2\nu} \|\boldsymbol{y} - \boldsymbol{x}\|^2 \tag{7.14}$$

Since $\boldsymbol{z}_{t+1}$ minimizes the problem $\min_{\boldsymbol{z} \in \mathcal{Z}} L_t(\boldsymbol{z}, \lambda_{t+1})$, the optimality condition [80, Theorem 2.2] applies

$$\nabla L_t(\boldsymbol{z}_{t+1})^\top(\boldsymbol{y} - \boldsymbol{z}_{t+1}) \geq 0 \qquad \forall \boldsymbol{y} \in \mathcal{Z} \tag{7.15}$$

Setting $\boldsymbol{y} = \boldsymbol{z}_t^*$ and $\boldsymbol{x} = \boldsymbol{z}_{t+1}$ in (7.14), we have, using (7.15),

$$L_t(\boldsymbol{z}_t^*) \geq L_t(\boldsymbol{z}_{t+1}) + \frac{1}{2\nu}\left\|\boldsymbol{z}_t^* - \boldsymbol{z}_{t+1}\right\|^2 \tag{7.16}$$

Then, (7.16) leads to

$$\begin{aligned}
L_t(\boldsymbol{z}_{t+1}) &\leq L_t(\boldsymbol{z}_t^*) - \frac{1}{2\nu}\left\|\boldsymbol{z}_t^* - \boldsymbol{z}_{t+1}\right\|^2 \\
\iff f_t(\boldsymbol{z}_t) + L_t(\boldsymbol{z}_{t+1}) &\leq f_t(\boldsymbol{z}_t) + L_t(\boldsymbol{z}_t^*) \\
&\quad -\frac{1}{2\nu}\left\|\boldsymbol{z}_t^* - \boldsymbol{z}_{t+1}\right\|^2 \\
&= f_t(\boldsymbol{z}_t) + \nabla f_t(\boldsymbol{z}_t)^\top(\boldsymbol{z}_t^* - \boldsymbol{z}_t) \\
+\lambda_{t+1} g_t(\boldsymbol{z}_t^*) &+ \frac{1}{2\nu}(\left\|\boldsymbol{z}_t^* - \boldsymbol{z}_t\right\|^2 - \left\|\boldsymbol{z}_t^* - \boldsymbol{z}_{t+1}\right\|^2) \\
\overset{(a)}{\leq} f_t(\boldsymbol{z}_t^*) + 0 &+ \frac{1}{2\nu}(\left\|\boldsymbol{z}_t^* - \boldsymbol{z}_t\right\|^2 - \left\|\boldsymbol{z}_t^* - \boldsymbol{z}_{t+1}\right\|^2)
\end{aligned} \tag{7.17}$$

where (a) uses the convexity of f and set $\lambda_{t+1} g_t(\boldsymbol{z}_t^*)$ to 0. Indeed, $\lambda_{t+1} g_t(\boldsymbol{z}_t^*) \leq 0$ as $\lambda_{t+1} \geq 0$ and the per-slot optimal $\boldsymbol{z}_t^*$ is always feasible, i.e. $g_t(\boldsymbol{z}_t^*) \leq 0$.

We recall the expression of $L_t(\boldsymbol{z}_{t+1})$:

$$L_t(\boldsymbol{z}_{t+1}) = \nabla f_t(\boldsymbol{z}_t)\top(\boldsymbol{z}_{t+1} - \boldsymbol{z}_t) + \lambda_{t+1} g_t(\boldsymbol{z}_{t+1}) + \frac{\left\|\boldsymbol{z}_{t+1} - \boldsymbol{z}_t\right\|^2}{2\nu} \tag{7.18}$$

Next, we seek to bound the term $-\nabla f_t(\boldsymbol{z}_t)^\top(\boldsymbol{z}_{t+1} - \boldsymbol{z}_t)$ by

$$\begin{aligned}
-\nabla f_t(\boldsymbol{z}_t)^\top(\boldsymbol{z}_{t+1} - \boldsymbol{z}_t) &\overset{(b)}{\leq} \left\|\nabla f_t(\boldsymbol{z}_t)\right\| \left\|\boldsymbol{z}_{t+1} - \boldsymbol{z}_t\right\| \\
&\overset{(c)}{\leq} \frac{\left\|\nabla f_t(\boldsymbol{z}_t)\right\|^2}{2\eta} + \frac{\eta}{2}\left\|\boldsymbol{z}_{t+1} - \boldsymbol{z}_t\right\|^2 \\
&\overset{(d)}{\leq} \frac{G^2}{2\eta} + \frac{\eta}{2}\left\|\boldsymbol{z}_{t+1} - \boldsymbol{z}_t\right\|^2
\end{aligned} \tag{7.19}$$

where (b) uses Cauchy-Schwartz inequality, (c) is true for any arbitrary positive constant η, (d) uses the upper bound from (7.25). Plugging (7.19) into (7.17), we can isolate $f_t(\boldsymbol{z}_t)+\lambda_{t+1}g_t(\boldsymbol{z}_{t+1})$.

$$\begin{aligned}
f_t(\boldsymbol{z}_t)+\lambda_{t+1}g_t(\boldsymbol{z}_{t+1}) &\leq f_t(\boldsymbol{z}_t^*)+(\frac{\eta}{2}-\frac{1}{2\nu})\left\|\boldsymbol{z}_{t+1}-\boldsymbol{z}_t\right\|^2 \\
&+\frac{1}{2\nu}(\|\boldsymbol{z}_t^*-\boldsymbol{z}_t\|^2-\|\boldsymbol{z}_t^*-\boldsymbol{z}_{t+1}\|^2)+\frac{G^2}{2\eta} \\
&\overset{(e)}{=} f_t(\boldsymbol{z}_t^*)+\frac{1}{2\nu}(\|\boldsymbol{z}_t^*-\boldsymbol{z}_t\|^2-\|\boldsymbol{z}_t^*-\boldsymbol{z}_{t+1}\|^2) \\
&+\frac{\nu G^2}{2}
\end{aligned} \tag{7.20}$$

where (e) happens as we choose $\eta=1/\nu$ so that $\eta/2-1/2\nu=0$. Using the dual drift bound in (7.2), we have

$$\begin{aligned}
\frac{\Delta(\lambda_{t+1})}{\mu}+f_t(\boldsymbol{z}_t) &\leq \lambda_{t+1}(g_{t+1}(\boldsymbol{z}_{t+1})-g_t(\boldsymbol{z}_{t+1})) \\
&+f_t(\boldsymbol{z}_t)+\lambda_{t+1}g_t(\boldsymbol{z}_{t+1})+\frac{\mu}{2}g_{t+1}(\boldsymbol{z}_{t+1})^2 \\
&\overset{(f)}{\leq} \lambda_{t+1}[g_{t+1}(\boldsymbol{z}_{t+1})-g_t(\boldsymbol{z}_{t+1})]^+ + f_t(\boldsymbol{z}_t^*) \\
&+\frac{1}{2\nu}(\|\boldsymbol{z}_t^*-\boldsymbol{z}_t\|^2-\|\boldsymbol{z}_t^*-\boldsymbol{z}_{t+1}\|^2)+\frac{\nu G^2}{2} \\
&+\frac{\mu g_{t+1}(\boldsymbol{z}_{t+1})^2}{2} \\
&\overset{(g)}{\leq} f_t(\boldsymbol{z}_t^*)+\frac{1}{2\nu}(\|\boldsymbol{z}_t^*-\boldsymbol{z}_t\|^2-\|\boldsymbol{z}_t^*-\boldsymbol{z}_{t+1}\|^2) \\
&+\tilde{\lambda}U_g^t+\frac{\nu G^2}{2}+\frac{\mu M^2}{2}
\end{aligned} \tag{7.21}$$

where (f) uses the non-negativity of λ_{t+1} and (7.20), (g) comes from the upper bound $|[g_{t+1}(\boldsymbol{z}_{t+1})-g_t(\boldsymbol{z}_{t+1})]^+|\leq U_g^t$, the dual upper bound $\lambda_{t+1}\leq\tilde{\lambda}$ and (7.26).

By interpolating the norm terms, we have :

$$\begin{aligned}
&\|\boldsymbol{z}_t^* - \boldsymbol{z}_t\|^2 - \|\boldsymbol{z}_t^* - \boldsymbol{z}_{t+1}\|^2 \\
&= \|\boldsymbol{z}_t^* - \boldsymbol{z}_t\|^2 - \left\|\boldsymbol{z}_t - \boldsymbol{z}_{t-1}^*\right\|^2 + \left\|\boldsymbol{z}_t - \boldsymbol{z}_{t-1}^*\right\|^2 \\
&\quad - \|\boldsymbol{z}_t^* - \boldsymbol{z}_{t+1}\|^2 \\
&= \left\|\boldsymbol{z}_t^* - \boldsymbol{z}_{t-1}^*\right\| \left\|\boldsymbol{z}_t^* - 2\boldsymbol{z}_t + \boldsymbol{z}_{t-1}^*\right\| + \left\|\boldsymbol{z}_t - \boldsymbol{z}_{t-1}^*\right\|^2 \\
&\quad - \|\boldsymbol{z}_t^* - \boldsymbol{z}_{t+1}\|^2 \\
&\overset{(h)}{\le} \left\|\boldsymbol{z}_t^* - \boldsymbol{z}_{t-1}^*\right\| 2E + \left\|\boldsymbol{z}_t - \boldsymbol{z}_{t-1}^*\right\|^2 - \|\boldsymbol{z}_t^* - \boldsymbol{z}_{t+1}\|^2
\end{aligned} \tag{7.22}$$

where (h) follows from the diameter of $\mathcal{Z}$. Plugging (7.22) into (7.21), we have:

$$\begin{aligned}
&\frac{\Delta(\lambda_{t+1})}{\mu} + f_t(\boldsymbol{z}_t) \overset{(i)}{\le} f_t(\boldsymbol{z}_t^*) + \tilde{\lambda} U_g^t + \frac{\nu G^2}{2} \\
&+ \frac{\mu M^2}{2} + \frac{1}{2\nu}(2EU_z^t + \left\|\boldsymbol{z}_t - \boldsymbol{z}_{t-1}^*\right\|^2 - \|\boldsymbol{z}_t^* - \boldsymbol{z}_{t+1}\|^2)
\end{aligned} \tag{7.23}$$

where we use $\left\|\boldsymbol{z}_t^* - \boldsymbol{z}_{t-1}^*\right\| = U_z^t$ in the left-hand-side.

Summing up (7.23) over $t = 1...T$, and rearranging the terms, we find

$$\begin{aligned}
&\sum_{t=1}^{T} f_t(\boldsymbol{z}_t) - \sum_{t=1}^{T} f_t(\boldsymbol{z}_t^*) \le \tilde{\lambda} U_g + \frac{\nu G^2 T}{2} \\
&+ \frac{\mu T M^2}{2} + \frac{E U_z}{\nu} + \frac{1}{2\nu} \sum_{t=1}^{T} (\left\|\boldsymbol{z}_t - \boldsymbol{z}_{t-1}^*\right\|^2 - \|\boldsymbol{z}_t^* - \boldsymbol{z}_{t+1}\|^2) \\
&\quad - \sum_{t=1}^{T} \frac{\Delta(\lambda_{t+1})}{\mu} \\
&\overset{(j)}{=} \tilde{\lambda} U_g + \frac{\nu T G^2}{2} + \frac{\mu T M^2}{2} + \frac{E U_z}{\nu} \\
&+ \frac{1}{2\nu}(\|\boldsymbol{z}_1 - \boldsymbol{z}_0^*\|^2 - \|\boldsymbol{z}_T^* - \boldsymbol{z}_{T+1}\|^2) - \frac{1}{2\mu}(\lambda_{T+2}^2 - \lambda_2^2) \\
&\overset{(k)}{\le} \tilde{\lambda} U_g + \frac{\nu T G^2}{2} + \frac{\mu T M^2}{2} + \frac{E U_z}{\nu} \\
&\quad + \frac{E^2}{2\nu} + \frac{\mu M^2}{2}
\end{aligned} \tag{7.24}$$

where (j) comes from the telescoping sums, (k) uses $\|\boldsymbol{z}_T^* - \boldsymbol{z}_{T+1}\|^2 \ge 0$, $\lambda_{T+2}^2 \ge 0$,

$\|\boldsymbol{z}_1 - \boldsymbol{z}_0^*\|^2 \leq E^2$, $\lambda_2^2 \leq \mu^2 M^2$. This completes the proof and verifies our regret bound in Lemma 2.

Following the same steps and using the bounds presented in section 7.4, we verify the results of Lemma 3.

7.3 Proof of Lemma 2

Our starting point is Theorems 1 and 2 of [1]. First, we will prove that:

$$\|\nabla f_t(\boldsymbol{z}_t)\| \leq G \triangleq a\sqrt{K(K+1)}, \quad \forall x_t, y_k \in \Gamma, k \in \mathcal{K}_t. \tag{7.25}$$

Indeed, we can calculate the gradient vector:

$$\nabla f_t(\boldsymbol{z}_t) = \begin{bmatrix} -\sum_{k\in\mathcal{K}_t} \frac{a_k}{1+x_t+y_k} \\ -\frac{a_{(t-1)K+1}}{1+x_t+y_{(t-1)K+1}} \\ \vdots \\ -\frac{a_{tK}}{1+x_t+y_{tK}} \end{bmatrix}$$

where $\nabla f_t(\boldsymbol{z}_t) \in \mathbb{R}^{K+1}$. We can observe that the reservation variables appear only in the denominator of the gradient components, hence its norm is maximized when these variables are set equal to zero. Hence, we can write for the ℓ_2 norm:

$$\begin{aligned} \|\nabla f_t(\boldsymbol{z}_t)\|^2 \leq \|\nabla f_t(\mathbf{0})\|^2 = (\sum_{k\in\mathcal{K}_t} \frac{a_k}{1})^2 + \frac{a_{(t-1)K+1}^2}{1} + \ldots \\ + \frac{a_{tK}^2}{1} \leq (\sum_{k\in\mathcal{K}_t} a)^2 + a^2 + \ldots + a^2 \\ = K^2a^2 + Ka^2 = a^2K(K+1) \end{aligned}$$

which gives us the expression of the upper-bound G.

Second, for all $t, k \in \mathcal{K}_t$ and $x_t, y_k \in \Gamma$:

$$\begin{aligned}
0 \leq p_t x_t + \sum_{k \in \mathcal{K}_t} q_k y_k \leq pD + KqD \quad &\Rightarrow \\
-B \leq g_t(x_t, \boldsymbol{y}_t) \leq D(p + Kq) - B \;&\Rightarrow \\
|g_t(x_t, \boldsymbol{y}_t)| \leq \max\big\{D(p + Kq) - B, B\big\} = M. &
\end{aligned} \tag{7.26}$$

Finally, the ℓ_2 diameter E of $\mathcal{Z} = [0, D]^{K+1}$ is:

$$\begin{aligned}
\forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{Z}, \quad d(\boldsymbol{x}, \boldsymbol{y}) &\leq \sqrt{D^2 + D^2 + \ldots + D^2} \qquad (7.27) \\
&= D\sqrt{K + 1}
\end{aligned}$$

Replacing these bounds in [1, Theorem 1] we obtain the bounds of Lemma 2.

7.4 Proof of Lemma 3

Similarly, we first prove an upper bound on the gradient of the objective function, namely we will show that $\forall t, k \in \mathcal{K}_t, \boldsymbol{x}_t, \boldsymbol{y}_k \in \Gamma_1 \times \ldots \times \Gamma_m$, it holds:

$$\|\nabla f_t^\theta(\boldsymbol{z}_t)\| \leq G_\theta \triangleq a\theta\sqrt{mK(K+1)}.$$

The gradient vector is:

$$\nabla f_t^\theta(\boldsymbol{z}_t) = \begin{bmatrix} -\sum_{k \in \mathcal{K}_t} \frac{a_k \boldsymbol{\theta}_t}{1 + \boldsymbol{\theta}_t^\top \boldsymbol{x}_t + \boldsymbol{\theta}_t^\top \boldsymbol{y}_k} \\ -\frac{a_{(t-1)K+1} \boldsymbol{\theta}_t}{1 + \boldsymbol{\theta}_t^\top \boldsymbol{x}_t + \boldsymbol{\theta}_t^\top \boldsymbol{y}_{(t-1)K+1}} \\ \vdots \\ -\frac{a_{tK} \boldsymbol{\theta}_t}{1 + \boldsymbol{\theta}_t^\top \boldsymbol{x}_t + \boldsymbol{\theta}_t^\top \boldsymbol{y}_{tK}} \end{bmatrix}$$

and we maximize the norm by setting $\boldsymbol{z}_t = \mathbf{0}$ and using the maximum possible parameter values, namely $\boldsymbol{\theta}_t = [\theta, \ldots, \theta]^\top$ and $\forall k, a_k = a$. Then, we can write:

$$\begin{aligned}
\|\nabla f_t^\theta(\boldsymbol{z}_t)\|^2 \leq \|\nabla f_t^\theta(\mathbf{0})\|^2 &= \sum_{i=1}^{m} (\sum_{k \in \mathcal{K}_t} \frac{a_k}{1})^2 \theta_i^2 \\
&+ \sum_{i=1}^{m} \frac{(a_{(t-1)K+1})^2}{1} \theta_i^2 + \ldots + \sum_{i=1}^{m} \frac{(a_{tK})^2}{1} \theta_i^2 \\
&\leq m(\sum_{k \in \mathcal{K}_t} a)^2 \theta^2 + m a^2 \theta^2 + \ldots m a^2 \theta^2 \\
&\leq m K^2 a^2 \theta^2 + K m a^2 \theta^2 = (a\theta)^2 m K (K+1)
\end{aligned}$$

which gives us the expression of the bound G_θ.

Second, for all $t, k \in \mathcal{K}_t$ and $\boldsymbol{x}_t, \boldsymbol{y}_k \in \Gamma_1 \times \ldots \times \Gamma_m$:

$$\begin{aligned}
&0 \leq \boldsymbol{p}_t^\top \boldsymbol{x}_t + \sum_{k \in \mathcal{K}_t} \boldsymbol{q}_k^\top \boldsymbol{y}_k \leq \\
&pD_1 + \ldots + pD_m + K(qD_1 + \ldots + qD_m) \Rightarrow \\
&-B \leq g_t^\theta(\boldsymbol{x}_t, \{\boldsymbol{y}_k\}) \leq (D_1 + \ldots + D_m)(p + Kq) - B \Rightarrow \\
&|g_t^\theta(\boldsymbol{x}_t, \{\boldsymbol{y}_k\})| \leq \max\{(D_1 + \ldots + D_m)(p + Kq) - B, B\}
\end{aligned}$$

Finally, the diameter F of $\mathcal{Z} = [0, D_i]^{K+1} \times_{i=1}^{m}$ is:

$$\forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{Z}, \quad d(\boldsymbol{x}, \boldsymbol{y}) \leq \sqrt{(D_1^2 + \ldots + D_m^2)(K+1)} \tag{7.28}$$

$$= \sqrt{D_1^2 + \ldots + D_m^2}\sqrt{K+1} \tag{7.29}$$

And plugging these terms in [1, Theorem 1] we obtain Lemma 3.

7.5 Proof of Lemma 4

Let's prove the first part of the lemma, i.e. $U_g = TD(p+Kq)/6$. We denote $\boldsymbol{z} = [z_0, z_1, ..., z_K]^\top$ and $U_g = \sum_{t=1}^{T} U_g^t$, then:

$$\begin{aligned} U_g^t &= \max_{\boldsymbol{z}\in\mathcal{Z}} |[g_t(\boldsymbol{z}) - g_{t-1}(\boldsymbol{z})]_+| \\ &= \max_{\boldsymbol{z}\in\mathcal{Z}} |[(p_t - p_{t-1})z_0 + \sum_{k=1}^{K} (q_{(t-1)K+k} - q_{(t-2)K+k})z_k]_+| \\ &= \max_{\boldsymbol{z}\in\mathcal{Z}} |[Y_0 z_0 + \sum_{k=1}^{K} Y_k z_k]_+| \end{aligned} \tag{7.30}$$

where we have introduced the random variables $Y_0 = p_t - p_{t-1}$ and $Y_k = q_{(t-1)K+k} - q_{(t-2)K+k}$. As p_t and p_{t-1} follow the same uniform distribution on $[0,p]$, the random variable $Y_0 =$ follows a triangular distribution on $[-p,p]$, centered on 0. We observe the same for $Y_k, k \geq 1$, that follow the triangular distribution on $[-q,q]$, centered on 0.

The rule to determine the maximum wrt $\boldsymbol{z}$ is very simple: if the realization of Y_i is negative, then we select $z_i = 0$, otherwise we select $z_i = D$. This is because the operator $[.]_+$ nullify any negative total before we take the absolute value. Thus, we define the random variable $X_i = Y_i z_i$. If $Y_i \leq 0$, then $z_i = 0$, hence $X_i = 0$. Otherwise, $z_i = D$, then $X_i = y_i D$, hence $X_i \in]0, rD]$, where $r = p, q$. Therefore, $\mathbb{P}[X_i = 0] = 1/2$ and $X_{i|]0,rD]}$ follows a triangular distribution with mode 0.

$$\mathbb{E}[X_i] = (1/2)0 + (1/2)\mathbb{E}[X_{i|]0,rD]}] = \frac{1}{2}\frac{rD}{3} \tag{7.31}$$

Thus, $\mathbb{E}[X_0] = pD/6$ and $\forall k \geq 1, \mathbb{E}[X_k] = qD/6$. Hence:

$$\begin{aligned} \mathbb{E}[U_g^t] &= \mathbb{E}[X_0 + \sum_{k\geq 1} X_k] = (pD/6 + \sum_{k\geq 1} qD/6) \\ &= D(p+Kq)/6 \end{aligned} \tag{7.32}$$

With the weak law of large numbers, we finish the proof:

$$\frac{U_g}{T} = \frac{U_g^1 + \ldots + U_g^T}{T} \tag{7.33}$$

converges to the expected value of U_g^t: $U_g/T \to D(p+Kq)/6$ as $T \to \infty$. Thus, for sufficiently large T, we get:

$$U_g = TD(p+Kq)/6.$$

Now we prove the second part of the lemma which concerns the slice orchestration case, i.e. $U_g^\theta = T(\sum_{i=1}^m D_i)(p+Kq)/6$. We denote $\boldsymbol{z} = [z_0^1, \dots, z_0^m, z_1^1, \dots, z_1^m, \dots, z_K^1, \dots, z_K^m]$. The proof follows the exact same steps: we select $z_i^j = 0$ or D_j, based on the rule to determine the maximum. This leads to:

$$\mathbb{E}[U_g^{\theta,t}] = (D_1 + \ldots + D_m)(p+Kq)/6 \tag{7.34}$$

and finally to $U_g^\theta = T(D_1 + \ldots + D_m)(p+Kq)/6$.

Bibliography

[1] T. Chen, Q. Ling, and G. B. Giannakis, "An online convex optimization approach to proactive network resource allocation," *IEEE Transactions on Signal Processing*, vol. 65, no. 24, pp. 6350–6364, 2017.

[2] "Ericsson mobility report," November 2021.

[3] K. Samdanis, X. Costa-Perez, and V. Sciancalepore, "From network sharing to multi-tenancy: The 5G network slice broker," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 32–39, 2016.

[4] 3GPP TS 23.251, "Network Sharing; Architecture and Functional Description," March 2015.

[5] V. Sciancalepore, F. Cirillo, and X. Costa-Perez, "Slice as a service (SlaaS) optimal IoT slice resources orchestration," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*. IEEE, 2017, pp. 1–7.

[6] P. Rost, A. Banchs, I. Berberana, M. Breitbach, M. Doll, H. Droste, C. Mannweiler, M. A. Puente, K. Samdanis, and B. Sayadi, "Mobile network architecture evolution toward 5G," *IEEE Communications Magazine*, vol. 54, no. 5, pp. 84–91, 2016.

[7] M. Series, "IMT Vision–Framework and overall objectives of the future development of IMT for 2020 and beyond," *Recommendation ITU*, vol. 2083, p. 21, 2015.

[8] G. P. A. W. Group *et al.*, "View on 5G architecture," *White Paper, July*, 2016.

[9] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A survey on software-defined networking," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 27–51, 2014.

[10] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2014.

[11] B. Yi, X. Wang, K. Li, M. Huang *et al.*, "A comprehensive survey of network function virtualization," *Computer Networks*, vol. 133, pp. 212–262, 2018.

[12] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network slicing in 5G: Survey and challenges," *IEEE communications magazine*, vol. 55, no. 5, pp. 94–100, 2017.

[13] ETSI NFV ISG, "Network Function Virtualization (NFV) Management and Orchestration," November 2014.

[14] F. Z. Yousaf, V. Sciancalepore, M. Liebsch, and X. Costa-Perez, "MANOaaS: A multi-tenant NFV MANO for 5G network slices," *IEEE Communications Magazine*, vol. 57, no. 5, pp. 103–109, 2019.

[15] 3GPP TR 23.708, "Architecture Enhancement for Service Capability Exposure," June 2015.

[16] G. S. Paschos, M. A. Abdullah, and S. Vassilaras, "Network slicing with splittable flows is hard," in *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC).* IEEE, 2018, pp. 1788–1793.

[17] V. Sciancalepore, K. Samdanis, X. Costa-Perez, D. Bega, M. Gramaglia, and A. Banchs, "Mobile traffic forecasting for maximizing 5G network slicing resource utilization," in *IEEE INFOCOM 2017-IEEE conference on computer communications.* IEEE, 2017, pp. 1–9.

[18] P. Caballero, A. Banchs, G. De Veciana, X. Costa-Pérez, and A. Azcorra, "Network slicing for guaranteed rate services: Admission control and resource allocation games," *IEEE Transactions on Wireless Communications*, vol. 17, no. 10, pp. 6419–6432, 2018.

[19] B. Han, V. Sciancalepore, D. Feng, X. Costa-Perez, and H. D. Schotten, "A utility-driven multi-queue admission control solution for network slicing," in *IEEE INFOCOM 2019-IEEE conference on computer communications.* IEEE, 2019, pp. 55–63.

[20] M. Leconte, G. S. Paschos, P. Mertikopoulos, and U. C. Kozat, "A resource allocation framework for network slicing," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications.* IEEE, 2018, pp. 2177–2185.

[21] R. Su, D. Zhang, R. Venkatesan, Z. Gong, C. Li, F. Ding, F. Jiang, and Z. Zhu, "Resource allocation for network slicing in 5G telecommunication networks: A survey of principles and models," *IEEE Network*, vol. 33, no. 6, pp. 172–179, 2019.

[22] A. Banchs, G. de Veciana, V. Sciancalepore, and X. Costa-Perez, "Resource allocation for network slicing in mobile networks," *IEEE Access*, vol. 8, pp. 214 696–214 706, 2020.

[23] G. Wang, G. Feng, W. Tan, S. Qin, R. Wen, and S. Sun, "Resource allocation for network slices in 5G with network resource pricing," in *GLOBECOM 2017-2017 IEEE Global Communications Conference.* IEEE, 2017, pp. 1–6.

[24] N. Nikaein, E. Schiller, R. Favraud, K. Katsalis, D. Stavropoulos, I. Alyafawi, Z. Zhao, T. Braun, and T. Korakis, "Network store: Exploring slicing in future 5G networks," in *Proceedings of the 10th International Workshop on Mobility in the Evolving Internet Architecture*, 2015, pp. 8–13.

[25] X. Zhou, R. Li, T. Chen, and H. Zhang, "Network slicing as a service: enabling enterprises' own software-defined cellular networks," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 146–153, 2016.

[26] V. Sciancalepore, L. Zanzi, X. Costa-Perez, and A. Capone, "ONETS: online network slice broker from theory to practice," *IEEE Transactions on Wireless Communications*, vol. 21, no. 1, pp. 121–134, 2021.

[27] D. Bega, M. Gramaglia, A. Banchs, V. Sciancalepore, K. Samdanis, and X. Costa-Perez, "Optimising 5G infrastructure markets: The business of network slicing," in *IEEE INFOCOM 2017-IEEE conference on computer communications.* IEEE, 2017, pp. 1–9.

[28] C. Marquez, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, "How should I slice my network? A multi-service empirical evaluation of resource sharing efficiency," in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, 2018, pp. 191–206.

[29] C.-Y. Chang, N. Nikaein, R. Knopp, T. Spyropoulos, and S. S. Kumar, "FlexCRAN: A flexible functional split framework over ethernet fronthaul in Cloud-RAN," in *2017 IEEE International Conference on Communications (ICC).* IEEE, 2017, pp. 1–7.

[30] K. Katsalis, N. Nikaein, E. Schiller, A. Ksentini, and T. Braun, "Network slices toward 5G communications: Slicing the LTE network," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 146–154, 2017.

[31] C.-Y. Chang and N. Nikaein, "RAN runtime slicing system for flexible and dynamic service execution environment," *IEEE Access*, vol. 6, pp. 34 018–34 042, 2018.

[32] A. Garcia-Saavedra, X. Costa-Perez, D. J. Leith, and G. Iosifidis, "Fluidran: Optimized vran/mec orchestration," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications.* IEEE, 2018, pp. 2366–2374.

[33] G. Tseliou, F. Adelantado, and C. Verikoukis, "Scalable RAN virtualization in multi-tenant LTE-A heterogeneous networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 8, pp. 6651–6664, 2015.

[34] X. Costa-Pérez, J. Swetina, T. Guo, R. Mahindra, and S. Rangarajan, "Radio access network virtualization for future mobile carrier networks," *IEEE Communications Magazine*, vol. 51, no. 7, pp. 27–35, 2013.

[35] N. Reyhanian, H. Farmanbar, and Z.-Q. Luo, "Resource reservation in backhaul and radio access network with uncertain user demands," *IEEE Transactions on Vehicular Technology*, 2022.

[36] M. Khodak, L. Zheng, A. S. Lan, C. Joe-Wong, and M. Chiang, "Learning cloud dynamics to optimize spot instance bidding strategies," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications.* IEEE, 2018, pp. 2762–2770.

[37] P. Caballero, A. Banchs, G. De Veciana, and X. Costa-Pérez, "Network slicing games: Enabling customization in multi-tenant mobile networks," *IEEE/ACM Transactions on Networking*, vol. 27, no. 2, pp. 662–675, 2019.

[38] S. Vassilaras, L. Gkatzikis, N. Liakopoulos, I. N. Stiakogiannakis, M. Qi, L. Shi, L. Liu, M. Debbah, and G. S. Paschos, "The algorithmic aspects of network slicing," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 112–119, 2017.

[39] N. Liakopoulos, G. Paschos, and T. Spyropoulos, "No regret in cloud resources reservation with violation guarantees," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications.* IEEE, 2019, pp. 1747–1755.

[40] X. Zhang, C. Wu, Z. Li, and F. C. Lau, "Proactive VNF provisioning with multi-timescale cloud resources: Fusing online learning and online optimization," in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE, 2017, pp. 1–9.

[41] J. Martín-Pérez, F. Malandrino, C.-F. Chiasserini, and C. J. Bernardos, "OKpi: All-KPI network slicing through efficient resource allocation," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 804–813.

[42] G. Wang, G. Feng, T. Q. Quek, S. Qin, R. Wen, and W. Tan, "Reconfiguration in network slicing–Optimizing the profit and performance," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 591–605, 2019.

[43] J. X. Salvat, L. Zanzi, A. Garcia-Saavedra, V. Sciancalepore, and X. Costa-Perez, "Overbooking network slices through yield-driven end-to-end orchestration," in *Proceedings of the 14th International Conference on emerging Networking EXperiments and Technologies*, 2018, pp. 353–365.

[44] C. Marquez, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, "Resource sharing efficiency in network slicing," *IEEE Transactions on Network and Service Management*, vol. 16, no. 3, pp. 909–923, 2019.

[45] V. Sciancalepore, X. Costa-Perez, and A. Banchs, "RL-NSB: Reinforcement learning-based 5G network slice broker," *IEEE/ACM Transactions on Networking*, vol. 27, no. 4, pp. 1543–1557, 2019.

[46] Y. Zhang, S. Bi, and Y. A. Zhang, "Joint Spectrum Reservation and On-Demand Request for Mobile Virtual Network Operators," *IEEE Transactions on Communications*, vol. 66, no. 7, pp. 2966–2977, 2018.

[47] H. Zhang and V. W. S. Wong, "A Two-Timescale Approach for Network Slicing in C-RAN," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 6, pp. 6656–6669, 2020.

[48] C. Gutterman, E. Grinshpun, S. Sharma, and G. Zussman, "RAN resource usage prediction for a 5G slice broker," in *Proceedings of the twentieth ACM international symposium on mobile ad hoc networking and computing*, 2019, pp. 231–240.

[49] N. Van Huynh, D. T. Hoang, D. N. Nguyen, and E. Dutkiewicz, "Optimal and fast real-time resource slicing with deep dueling neural networks," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1455–1470, 2019.

[50] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, "DeepCog: Cognitive network management in sliced 5G networks with deep learning," in *IEEE INFOCOM 2019-IEEE conference on computer communications*. IEEE, 2019, pp. 280–288.

[51] J. Wang, J. Tang, Z. Xu, Y. Wang, G. Xue, X. Zhang, and D. Yang, "Spatiotemporal modeling and prediction in cellular networks: A big data enabled deep learning approach," in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE, 2017, pp. 1–9.

[52] C. Zhang and P. Patras, "Long-term mobile traffic forecasting using deep spatio-temporal neural networks," in *Proceedings of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2018, pp. 231–240.

[53] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, "AZTEC: Anticipatory capacity allocation for zero-touch network slicing," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications.* IEEE, 2020, pp. 794–803.

[54] M. R. Raza, C. Natalino, P. Öhlen, L. Wosinska, and P. Monti, "Reinforcement Learning for Slicing in a 5G Flexible RAN," *Journal of Lightwave Technology*, vol. 37, no. 20, pp. 5161–5169, 2019.

[55] F. Wei, G. Feng, Y. Sun, Y. Wang, S. Qin, and Y.-C. Liang, "Network slice reconfiguration by exploiting deep reinforcement learning with large action space," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2197–2211, 2020.

[56] A. Aijaz, "$\mathsf{Hap-SliceR}$: A Radio Resource Slicing Framework for 5G Networks With Haptic Communications," *IEEE Systems Journal*, vol. 12, no. 3, pp. 2285–2296, 2018.

[57] B. Han, J. Lianghai, and H. D. Schotten, "Slice as an evolutionary service: Genetic optimization for inter-slice resource management in 5G networks," *ieee access*, vol. 6, pp. 33 137–33 147, 2018.

[58] L. Zanzi, V. Sciancalepore, A. Garcia-Saavedra, and X. Costa-Pérez, "OVNES: Demonstrating 5G network slicing overbooking on real deployments," in *IEEE INFOCOM 2018-IEEE conference on computer communications workshops (INFOCOM WKSHPS).* IEEE, 2018, pp. 1–2.

[59] P. Sanders and C. Schulz, "Think locally, act globally: Highly balanced graph partitioning," in *International Symposium on Experimental Algorithms.* Springer, 2013, pp. 164–175.

[60] T. Ghazar and N. Samaan, "Pricing utility-based virtual networks," *IEEE Transactions on Network and Service Management*, vol. 10, no. 2, pp. 119–132, 2013.

[61] S. D'Oro, F. Restuccia, T. Melodia, and S. Palazzo, "Low-complexity distributed radio access network slicing: Algorithms and experimental results," *IEEE/ACM Transactions on Networking*, vol. 26, no. 6, pp. 2815–2828, 2018.

[62] U. Habiba and E. Hossain, "Auction mechanisms for virtualization in 5G cellular networks: Basics, trends, and open challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2264–2293, 2018.

[63] L. Liang, Y. Wu, G. Feng, X. Jian, and Y. Jia, "Online Auction-Based Resource Allocation for Service-Oriented Network Slicing," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 8063–8074, 2019.

[64] H. Halabian, "Distributed resource allocation optimization in 5G virtualized networks," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 3, pp. 627–642, 2019.

[65] M. Jiang, M. Condoluci, and T. Mahmoodi, "Network slicing in 5G: An auction-based model," in *2017 IEEE International Conference on Communications (ICC).* IEEE, 2017, pp. 1–6.

[66] P. D. Maciel, F. L. Verdi, P. Valsamas, I. Sakellariou, L. Mamatas, S. Petridou, P. Papadimitriou, D. Moura, A. I. Swapna, B. Pinheiro *et al.*, “A marketplace-based approach to cloud network slice composition across multiple domains,” in *2019 IEEE Conference on Network Softwarization (NetSoft).* IEEE, 2019, pp. 480–488.

[67] D. Bega, M. Gramaglia, A. Banchs, V. Sciancalepore, and X. Costa-Pérez, “A machine learning approach to 5G infrastructure market optimization,” *IEEE Transactions on Mobile Computing*, vol. 19, no. 3, pp. 498–512, 2019.

[68] L. Zanzi, A. Albanese, V. Sciancalepore, and X. Costa-Pérez, “NSBchain: A secure blockchain framework for network slicing brokerage,” in *ICC 2020-2020 IEEE International Conference on Communications (ICC).* IEEE, 2020, pp. 1–7.

[69] J. Backman, S. Yrjölä, K. Valtanen, and O. Mämmelä, “Blockchain network slice broker in 5G: Slice leasing in factory of the future use case,” in *2017 Internet of Things Business Models, Users, and Networks.* IEEE, 2017, pp. 1–8.

[70] N. Afraz and M. Ruffini, “5G network slice brokering: A distributed blockchain-based market,” in *2020 European Conference on Networks and Communications (EuCNC).* IEEE, 2020, pp. 23–27.

[71] “Compute Engine Pricing,” 2022, Google Cloud Platform. [Online]. Available: https://cloud.google.com/compute/

[72] “Google Cloud Platform,” 2022, Preemptible Virtual Machines. [Online]. Available: https://cloud.google.com/preemptible-vms/

[73] “Amazon EC2,” 2022, Reserved Instances. [Online]. Available: https://aws.amazon.com/ec2/purchasing-options/reserved-instances/

[74] “Amazon EC2,” 2022, Spot Instances. [Online]. Available: https://aws.amazon.com/ec2/spot/

[75] M. Zafer, Y. Song, and K.-W. Lee, “Optimal bids for spot vms in a cloud for deadline constrained jobs,” in *2012 IEEE Fifth International Conference on Cloud Computing.* IEEE, 2012, pp. 75–82.

[76] M. Lumpe, M. B. Chhetri, Q. B. Vo, and R. Kowalcyk, “On estimating minimum bids for Amazon EC2 spot instances,” in *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID).* IEEE, 2017, pp. 391–400.

[77] L. Zheng, C. Joe-Wong, C. W. Tan, M. Chiang, and X. Wang, “How to bid the cloud,” *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 71–84, 2015.

[78] P. Sharma, D. Irwin, and P. Shenoy, “How not to bid the cloud,” in *8th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 16)*, 2016.

[79] S. Shalev-Shwartz *et al.*, “Online learning and online convex optimization,” *Foundations and trends in Machine Learning*, vol. 4, no. 2, pp. 107–194, 2011.

[80] E. Hazan, “Introduction to Online Convex Optimization,” *Foundations and Trends in Optimization*, vol. 2, pp. 157–325, 2016.

[81] M. Zinkevich, “Online Convex Programming and Generalized Infinitesimal Gradient Ascent,” in *Proc. of ICML*, 2003.

[82] E. Hazan, A. Agarwal, and S. Kale, "Logarithmic regret algorithms for online convex optimization," *Machine Learning*, vol. 69, no. 2, pp. 169–192, 2007.

[83] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization.* Cambridge university press, 2004.

[84] M. Mahdavi, R. Jin, and T. Yang, "Trading regret for efficiency: online convex optimization with long term constraints," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 2503–2528, 2012.

[85] R. Jenatton, J. Huang, and C. Archambeau, "Adaptive algorithms for online convex optimization with long-term constraints," in *International Conference on Machine Learning.* PMLR, 2016, pp. 402–411.

[86] V. Valls, G. Iosifidis, D. Leith, and L. Tassiulas, "Online convex optimization with perturbed constraints: Optimal rates against stronger benchmarks," in *International Conference on Artificial Intelligence and Statistics.* PMLR, 2020, pp. 2885–2895.

[87] S. Mannor, J. N. Tsitsiklis, and J. Y. Yu, "Online Learning with Sample Path Constraints," *Journal of Machine Learning Research*, vol. 10, pp. 569–590, 2009.

[88] M. J. Neely and H. Yu, "Online convex optimization with time-varying constraints," *arXiv preprint arXiv:1702.04783*, 2017.

[89] H. Yu, M. Neely, and X. Wei, "Online convex optimization with stochastic constraints," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[90] A. Koppel, F. Y. Jakubiec, and A. Ribeiro, "A saddle point algorithm for networked online convex optimization," *IEEE Transactions on Signal Processing*, vol. 63, no. 19, pp. 5149–5164, 2015.

[91] N. Liakopoulos, A. Destounis, G. Paschos, T. Spyropoulos, and P. Mertikopoulos, "Cautious regret minimization: Online optimization with long-term budget constraints," in *International Conference on Machine Learning.* PMLR, 2019, pp. 3944–3952.

[92] A. Mokhtari, S. Shahrampour, A. Jadbabaie, and A. Ribeiro, "Online optimization in dynamic environments: Improved regret rates for strongly convex problems," in *2016 IEEE 55th Conference on Decision and Control (CDC).* IEEE, 2016, pp. 7195–7201.

[93] L. Zhang, T. Yang, J. Yi, R. Jin, and Z.-H. Zhou, "Improved dynamic regret for non-degenerate functions," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[94] T. Yang, L. Zhang, R. Jin, and J. Yi, "Tracking slowly moving clairvoyant: Optimal dynamic regret of online learning with true and noisy gradient," in *International Conference on Machine Learning.* PMLR, 2016, pp. 449–457.

[95] O. Besbes, Y. Gur, and A. Zeevi, "Non-stationary stochastic optimization," *Operations research*, vol. 63, no. 5, pp. 1227–1244, 2015.

[96] C.-K. Chiang, T. Yang, C.-J. Lee, M. Mahdavi, C.-J. Lu, R. Jin, and S. Zhu, "Online optimization with gradual variations," in *Conference on Learning Theory.* JMLR Workshop and Conference Proceedings, 2012, pp. 6–1.

[97] X. Cao and K. R. Liu, "Online convex optimization with time-varying constraints and bandit feedback," *IEEE Transactions on automatic control*, vol. 64, no. 7, pp. 2665–2680, 2018.

[98] A. D. Flaxman, A. T. Kalai, and H. B. McMahan, "Online convex optimization in the bandit setting: gradient descent without a gradient," *arXiv preprint cs/0408007*, 2004.

[99] A. Agarwal, O. Dekel, and L. Xiao, "Optimal Algorithms for Online Convex Optimization with Multi-Point Bandit Feedback," in *Colt.* Citeseer, 2010, pp. 28–40.

[100] X. Yi, X. Li, L. Xie, and K. H. Johansson, "Distributed online convex optimization with time-varying coupled inequality constraints," *IEEE Transactions on Signal Processing*, vol. 68, pp. 731–746, 2020.

[101] Q. Liu, W. Wu, L. Huang, and Z. Fang, "Simultaneously achieving sublinear regret and constraint violations for online convex optimization with time-varying constraints," *Performance Evaluation*, vol. 152, p. 102240, 2021.

[102] X. Ding, L. Chen, P. Zhou, Z. Xu, S. Wen, J. C. Lui, and H. Jin, "Dynamic online convex optimization with long-term constraints via virtual queue," *Information Sciences*, vol. 577, pp. 140–161, 2021.

[103] J.-B. Monteil, G. Iosifidis, and L. DaSilva, "No-Regret Slice Reservation Algorithms," in *ICC 2021-IEEE International Conference on Communications.* IEEE, 2021, pp. 1–7.

[104] J.-B. Monteil, G. Iosifidis, and L. Da Silva, "Learning-based Reservation of Virtualized Network Resources," *IEEE Transactions on Network and Service Management*, 2022.

[105] X. Shen, J. Gao, W. Wu, K. Lyu, M. Li, W. Zhuang, X. Li, and J. Rao, "AI-Assisted Network-Slicing Based Next-Generation Wireless Networks," *IEEE Open Journal of Vehicular Technology*, vol. 1, pp. 45–66, 2020.

[106] G. Dorffner, "Neural networks for time series processing," *Neural Network World*, vol. 6, pp. 447–468, 1996.

[107] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, 1997.

[108] L. Yunpeng, H. Di, B. Junpeng, and Q. Yong, "Multi-step ahead time series forecasting for different data patterns based on LSTM recurrent neural network," in *14th Web Information Systems and Applications Conference (WISA).* IEEE, 2017, pp. 305–310.

[109] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[110] A. Sang and S. Q. Li, "A predictability analysis of network traffic," in *Computer Networks*, July 2002, pp. 329–345.

[111] P. Cramton and L. Doyle, "Open access wireless markets," *Telecommunications Policy*, vol. 41, no. 5-6, pp. 379–390, 2017.

[112] S. Shenker, "Fundamental design issues for the future Internet," *IEEE Journal on selected areas in communications*, vol. 13, no. 7, pp. 1176–1188, 1995.

[113] S. G. Shakkottai and R. Srikant, *Network optimization and control.* Now Publishers Inc, 2008.

[114] D. Bertsekas, "Nonlinear Programming," *Athena Scientific*, vol. 3rd Ed., 2016.

[115] N. L. Johnson, S. Kotz, and N. Balakrishnan, *Continuous univariate distributions, volume 2.* John wiley & sons, 1995, vol. 289.

[116] M. Mohri and S. Yang, "Accelerating online convex optimization via adaptive prediction," in *Artificial Intelligence and Statistics.* PMLR, 2016, pp. 848–856.

[117] H. B. McMahan, "A survey of algorithms and analysis for adaptive online learning," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 3117–3166, 2017.

[118] D. H. Oliveira, T. P. de Araujo, and R. L. Gomes, "An adaptive forecasting model for slice allocation in softwarized networks," *IEEE TNSM*, vol. 18, no. 1, pp. 94–103, 2021.

[119] O. Anava, E. Hazan, S. Mannor, and O. Shamir, "Online learning for time series prediction," in *Conference on learning theory.* PMLR, 2013, pp. 172–184.

[120] S. Shalev-Shwartz and Y. Singer, "A primal-dual perspective of online learning algorithms," *Machine Learning*, vol. 69, no. 2, pp. 115–142, 2007.

[121] A. Rakhlin and K. Sridharan, "Online learning with predictable sequences," in *Conference on Learning Theory.* PMLR, 2013, pp. 993–1019.

[122] Y. Cui, X. Huang, D. Wu, and H. Zheng, "Machine learning based resource allocation strategy for network slicing in vehicular networks," in *International Conference on Comm. in China (ICCC).* IEEE, 2020, pp. 454–459.

[123] N. Mhaisen, G. Iosifidis, and D. Leith, "Online Caching with Optimistic Learning," *arXiv preprint arXiv:2202.10590*, 2022.

[124] P. Auer, N. Cesa-Bianchi, and C. Gentile, "Adaptive and self-confident on-line learning algorithms," *Journal of Computer and System Sciences*, vol. 64, no. 1, pp. 48–75, 2002.

[125] P. Zhao, Y.-J. Zhang, L. Zhang, and Z.-H. Zhou, "Dynamic regret of convex and smooth functions," *Advances in Neural Information Processing Systems*, vol. 33, pp. 12 510–12 520, 2020.

[126] ——, "Adaptivity and non-stationarity: Problem-dependent dynamic regret for online convex optimization," *arXiv preprint arXiv:2112.14368*, 2021.

[127] Y. Wang, M. Agarwal, T. Lan, and V. Aggarwal, "Learning-Based Online QoE Optimization in Multi-Agent Video Streaming," *Algorithms*, vol. 15, no. 7, p. 227, 2022.

[128] X. Foukas and B. Radunovic, "Concordia: Teaching the 5G vRAN to share compute," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, 2021, pp. 580–596.

[129] Q.-V. Pham, F. Fang, V. N. Ha, M. J. Piran, M. Le, L. B. Le, W.-J. Hwang, and Z. Ding, "A survey of multi-access edge computing in 5G and beyond: Fundamentals, technology integration, and state-of-the-art," *IEEE Access*, vol. 8, pp. 116 974–117 017, 2020.

[130] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, “Mobile edge computingâA key technology towards 5G,” *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.