Learning-based Reservation of Virtualized Network Resources

Monteil, Jean Baptiste; Iosifidis, George; Da Silva, Luiz

DOI
10.1109/TNSM.2022.3144774

Publication date
2022

Document Version
Final published version

Published in
IEEE Transactions on Network and Service Management

Learning-Based Reservation of Virtualized Network Resources

Jean-Baptiste Monteil, George Iosifidis, *Member, IEEE*, and Luiz A. DaSilva, *Fellow, IEEE*

***Abstract*—Network slicing markets have the potential to increase significantly the utilization of virtualized network resources and facilitate the low-cost deployment of over-the-top services. However, their success is conditioned on the service providers (SPs) being able to bid effectively for the virtualized resources. In this paper, we consider a hybrid advance-reservation and spot slice market and study how the SPs should reserve resources to maximize their services' performance while not violating a time-average budget threshold. We consider this problem in its general form where the SP demand and slice prices are time-varying and revealed only after the reservations are decided. We develop a learning-based framework, using the theory of online convex optimization, that allows the SP to employ a no-regret reservation policy, i.e., achieve the same performance with an oracle that has full access to all future demand and prices. We extend the framework to the scenario where the SP decides dynamically its slice orchestration and hence needs to learn the performance-maximizing resource composition; and we further develop a mixed-time scale scheme that allows the SP to leverage spot-market information that is revealed between successive reservations. The proposed learning framework is evaluated using representative simulation scenarios that highlight its efficacy as well as the impact of key system and algorithm parameters.**

***Index Terms*—Online convex optimization, online learning, regret analysis, network slicing, network virtualization.**

I. Introduction

A. Motivation

THE INCREASING softwarization of mobile networks coupled with the proliferation of over-the-top service providers (SPs) which rely on network operators' infrastructure (NOs), have spurred numerous studies for the design of network virtualization markets, cf. [2], [3]. For instance, researchers have proposed embedding algorithms for assisting the NOs to accommodate heterogeneous slice requests [4], and devised pricing schemes aiming to maximize the operators' revenue from selling slices to SPs [5]. These solutions are expected to operate in near-real time and therefore enable the fine-grained (re-)allocation of network resources. This is a key step towards boosting the utilization efficiency of future mobile networks. Nevertheless, an aspect that has received less attention is how the SPs should request slices in these dynamic markets.

Recent forward-looking slicing market models draw ideas from the successful cloud computing marketplaces [6]–[9] that offer both in-advance reservation and on-the-spot bidding opportunities for computing and storage resources. Such hybrid markets will allow the NOs to proactively schedule their network operation using the information about the submitted advance reservations, but also to offer dynamically resources that have just become available. And similarly, they can enable the SPs to reserve resources with guarantees, but also to lease additional (often cheaper) resources tailored to the fast-changing demand of their users. However, this flexibility comes at a cost, as it confounds the already daunting slice-reservation task of the service providers.

Namely, each SP that participates in such a hybrid network market needs to request resources without knowing accurately the needs of its users; and to decide between (lower-cost) advance reservation and (higher-cost) dynamic reservations. Furthermore, the SP decisions are often made without access to future slice prices that, naturally, depend on the NO's internal needs and on the requests submitted by other SPs. In this complex and dynamic environment, the SP reservations need to be carefully designed in order to avoid over/under-reservation of the resources that can lead to network under-utilization or induce prohibitively high servicing costs. Clearly, inefficient slice reservation decisions can nullify the anticipated benefits of network virtualization and slicing.

Such dynamic environment paves the way to the design of smart online solutions for the SP reservations. We propose online optimization algorithms so that the SP learns how to proactively reserve resources in an online manner, adapting to the variations of the demand and of the pricing of resources. Our proposal falls under the scope of online convex optimization (OCO), which we detail in the sequel.

The focus of this paper is to tackle this problem by studying optimal slice reservation from the perspective of service providers. Our key aim is to design online reservation policies which an SP can employ to maximize the performance of its service while not exceeding the monetary budget it has committed for this purpose. This is an

Manuscript received 30 June 2021; revised 30 November 2021; accepted 6 January 2022. Date of publication 20 January 2022; date of current version 12 October 2022. This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant number 13/RC/2077_P2. It was also supported by the European Commission through Grant No. 101017109 (DAEMON). Part of this work appears in the proceedings of IEEE ICC 2021, doi: 10.1109/ICC42927.2021.9500530. The associate editor coordinating the review of this article and approving it for publication was N. Kamiyama. *(Corresponding author: Jean-Baptiste Monteil.)*

Jean-Baptiste Monteil is with the School of Computer Science and Statistics, Trinity College Dublin, Dublin 2, D02 PN40 Ireland (e-mail: jmonteil@tcd.ie).

George Iosifidis is with the ENS Group, Delft University of Technology, 2600 AA Delft, The Netherlands (e-mail: g.iosifidis@tudelft.nl).

Luiz A. DaSilva is with the Commonwealth Cyber Initiative, Virginia Tech, Arlington, VA 22203 USA (e-mail: ldasilva@vt.edu).

Digital Object Identifier 10.1109/TNSM.2022.3144774

important step towards unleashing the full potential of slicing markets.

B. Methodology and Contributions

We consider a hybrid network virtualization (or, slicing) market with advance-reservation and spot-bidding options, where an SP can request resources from a network operator in the beginning of each period and update its reservation at each slot within every period. The NO is allowed to change arbitrarily both its reservation and spot prices, which are made available to the SP only after the bidding is decided. Hence, in effect, the SP has to reserve slices without knowing the needs of its users or the overall cost of its reservation, as indeed happens often in practice. In this dynamic environment, the SP aims to maximize a general utility function of the slice resources, which reflects its service performance, while not violating a time-average budget constraint.

We model the market operation as a learning problem which allows the SP to implement a *no-regret slice reservation policy*. This means that, as time evolves, the SP achieves performance equal to that of an ideal benchmark policy that one could design only with hindsight, i.e., having access to all future demand and cost values. Our framework builds on the theory of online convex optimization (OCO) which was introduced in the seminal paper of Zinkevich [10] and initiated a new approach for the design of online learning algorithms, cf. [11]. The key advantage of OCO-based algorithms is their robustness with respect to the unknown problem parameters, e.g., cost functions. Indeed, they offer worst-case performance bounds that hold even if these parameters are selected strategically by an adversary. This makes them particularly attractive for the problem at hand where the interacting decisions of the service providers and the network operator are likely to create a non-stationary and highly volatile market.

The more intricate OCO model is applied here where the learning process involves both a network performance function and a time-average budget constraint. To tackle this constrained learning problem, we leverage a primal-dual online iteration [12]–[14] that achieves sublinear performance loss (i.e., *regret*) and sublinear constraint violation (i.e., *fit*) with respect to the optimal in hindsight decisions (benchmark). And the SP can prioritize one over the other metric by tuning properly the respective learning rates. The framework is extended to allow the SP to determine the *composition* of its slice, i.e., decide the exact amount of each resource (spectrum, backhaul capacity, etc.) that comprise the requested slice, without knowing in advance the performance-optimal resource combination. This is crucial when the qualitative features of user demand are volatile or unknown. Finally, we propose a mixed time-scale learning policy where the SP can exploit any price information that is revealed by the NO during each period. This practical modification is crucial as it improves significantly the performance of the reservation policy.

The contributions of this work can be therefore summarized as follows.

- 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. To the best of our knowledge, this is the first online learning model for budgeted slice reservation.
- 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.
- We analyze the impact of key system parameters on the learning performance, and discuss the implications for the design of such network virtualization markets.
- 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.

Paper Organization: The reminder of the paper is organized as follows. Section II reviews the related works and OCO background. Section III presents the system model and problem formulation. The first online learning solution (OLR) is presented in Section IV, while Sections V and VI introduce two practical extensions, namely the slice orchestration (OLR-SO) and the mixed time scale reservation models (OLR-MTS). Section VII verifies the efficacy of the proposed policies in a series of numerical tests and Section VIII concludes the study. All the proofs are placed in the Appendix.

II. Background and Related Work

A. Reservation of Virtualized Resources

The paper [15] studied the problem of allocating network and computing resources to a set of slices in order to maximize a system-wide utility function, namely to enforce fair resource allocation across slices. In [16] the authors proposed a static optimization framework for embedding VNF chains (interpreted as slices) in a shared network. Their key contribution is the formulated problem which accounts for reliability, delay and other slice requirements. Albeit detailed and rigorous, this analysis considers the various system parameters and requests to be known. Similarly, [17] studied the impact of slice overbooking; [18] employed predictive capacity allocation for improving the slice composition; and [19] focused on dynamic slicing via reinforcement learning. Unlike these works, we make no assumptions regarding the availability or the statistical properties of the prices and user needs.

Fewer works consider the problem from the SP point of view. In [20] and [21], the authors study a hybrid reservation and spot market where the SP reservations are decided by solving a stochastic problem. This, however, presumes a stationary environment, an assumption that is likely to fail when multiple SPs bid strategically and the NO adapts the prices accordingly. Our previous work [22] employed demand and price predictions (via neural networks) to assist the SP reservations; while [23] focused on slice reconfiguration costs.

In [24], the authors consider a reservation model, where a customer at each slot requests different types of cloud resources aiming to minimize a fixed and known cost function while satisfying a time-average unknown demand constraint. This model, however, is not suitable for the considered hybrid markets where the prices are volatile and hence the cost functions change dynamically.

B. Slicing Markets

The design of markets for virtualized network resources is a relatively new research area. The survey in [5] provides an overview of auction theory-based slicing solutions and [25] proposed mechanisms for the NO to auction its sliced resources. Unlike these works, we consider a dynamic pricing scheme that is more practical as it does not require to run any type of auction. Importantly, our model is based on already-deployed and widely-used market models in cloud computing ecosystems, e.g., see [7], [9].

Prior works that focus on such hybrid cloud market models have studied spot pricing models and devised intelligent bidding strategies for the buyers [26]–[30]. For instance, in [29] 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 [30] 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. Our work differs in that we make no assumption for the spot pricing model of the operator, and our reservation algorithm offers performance guarantees for any possible pricing scheme and demand pattern. This is crucial as in practice the operator might as well revise and adapt its pricing policy in the presence of strategic bidders.

C. Background on Constrained OCO

The standard OCO problem [10] considers a series of initially-unknown convex cost functions $\{f_t(\boldsymbol{x})\}_t$ and a convex compact set $\mathcal{X}$, and asks to find the decisions $\{\boldsymbol{x}_t\}_t \in \mathcal{X}$ such that the total loss, or *static regret*:

$$R_T^s = \sum_{t=1}^{T} f_t(\boldsymbol{x}_t) - f_t(\boldsymbol{x}^\star),$$

grows sublinearly w.r.t. to the optimal-in-hindsight benchmark $\boldsymbol{x}^\star = \arg\min_{\boldsymbol{x}\in\mathcal{X}} \sum_{t=1}^{T} f_t(\boldsymbol{x})$. This ensures the learning algorithm performs on average as well as the benchmark. A practical extension is to enrich $\mathcal{X}$ with a time-average budget constraint $\sum_{t=1}^{T} g_t(\boldsymbol{x}_t) \leq 0$, where $g_t(\boldsymbol{x}_t)$ is a convex function, that needs also to grow sublinearly in order to achieve zero average constraint violation (or, *fit*). This constrained OCO problem is in fact impossible to tackle in the general case where the benchmark is selected from the set:

$$\mathcal{X}_T^{max} = \left\{ \boldsymbol{x} \in \mathcal{X} : \sum_{t=1}^{T} g_t(\boldsymbol{x}) \leq 0 \right\}, \tag{1}$$

TABLE I
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
Γ	$\Gamma = [0, D]$, 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

as proved in [31]. Hence, follow-up works considered simpler settings where $g_t(\boldsymbol{x}) = g(\boldsymbol{x}), \forall t$ [13], [14]; constraints that are only linearly-perturbed [32]; or less demanding benchmarks that are selected from the restricted set $\mathcal{X}_T = \{\boldsymbol{x} \in \mathcal{X} : g_t(\boldsymbol{x}) \leq 0, \forall t\}$, see [33], [34].

A different line of research assesses the learning policy using the *dynamic regret* metric where the set benchmark is $\boldsymbol{x}_t^\star = \arg\min_{\boldsymbol{x}\in\mathcal{X}} f_t(\boldsymbol{x})$. This is a more demanding benchmark than the static regret. Recent works showed that it is possible to achieve both sublinear dynamic regret and fit under certain assumptions on the variability of the problem [35], [36]. Namely, it is required to have sublinear accumulated variations of the dynamic benchmark sequence $\{\boldsymbol{x}_t^\star\}_t$, which in turn imposes constraints on the slot-by-slot variability of the cost and constraint functions. We consider this more demanding learning objective here and characterize the conditions under which it is achievable. Our approach is inspired by [12], that proposed a primal-dual algorithm which runs on the Lagrangian relaxation of the constrained OCO problem. We tailor this idea to account for our problem-specific requirements and assumptions; and extend it to handle jointly the reservation and the slice composition decisions, both in the single and in the mixed time scale instances.

III. System Model and Problem Statement

Notation: We use bold typeface for vectors, $\boldsymbol{a}$, and vector transpose is denoted $\boldsymbol{a}^\top$. 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 $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_T$. Sets are denoted with calligraphic capital letters, e.g., $\mathcal{M}$. The projection onto the non-negative orthant is denoted $[\cdot]_+$, and $\|\cdot\|$ is the ℓ_2 norm. The key notation symbols are summarized in Table I below.

A. Network & Market 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] A network operator (NO) sells virtualized resources to service providers (SPs),

[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.

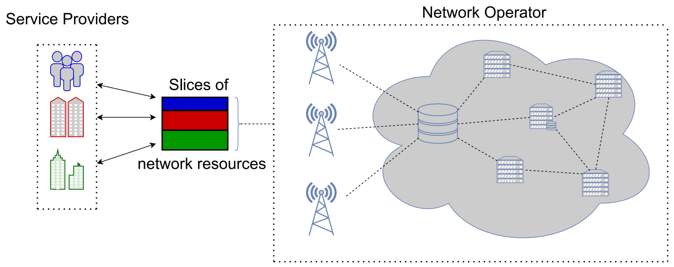


Fig. 1. A Network Operator (NO) leases different types of resources, e.g., wireless capacity, storage capacity and edge computing capacity, to different types of Service Providers (SPs) that offer over-the-top services to their users.

see Fig. 1, and we denote with $\mathcal{H}$ the set 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. We consider initially $\boldsymbol{\theta}$ to be fixed, but we drop this assumption in Section V.

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 $\boldsymbol{q}_t = (q_k, k \in \mathcal{K}_t)$ where $\mathcal{K}_t = \{(t-1)K+1, \ldots, tK\}$. 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 $\Gamma = [0, D]$. 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.

B. 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 Section V. At the beginning of each period t the SP decides its t-period reservation plan $(x_t, \boldsymbol{y}_t)$, where $\boldsymbol{y}_t = (y_k, k \in \mathcal{K}_t)$. 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 $\boldsymbol{a}_t = (a_k, k \in \mathcal{K}_t)$. 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 [15], [37], [38] and references therein. These papers provide the general form of α-fair utility function, namely:

$$f(\boldsymbol{z}) = \begin{cases} \frac{\boldsymbol{z}^{1-\alpha}}{1-\alpha} & \alpha \neq 1 \\ \log(\boldsymbol{z}) & \alpha = 1 \end{cases} \tag{2}$$

In [39], 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 [16], [20], [24].

C. Problem Statement

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:

$$(\mathbb{P}): \max_{\{x_t, \boldsymbol{y}_t\}_{t=1}^T} \sum_{t=1}^{T} \sum_{k\in\mathcal{K}_t} a_k \log(x_t + y_k + 1) \tag{3}$$

$$\text{s.t.} \sum_{t=1}^{T} \left(x_t p_t + \boldsymbol{y}_t^\top \boldsymbol{q}_t \right) \leq BT, \tag{4}$$

$$y_k \in \Gamma, \quad \forall k = 1, \ldots, KT, \tag{5}$$

$$x_t \in \Gamma, \quad \forall t = 1, \ldots, T. \tag{6}$$

Objective (3) is the total utility that the SP achieves with its reservations, after a duration of T periods. Constraint (4) captures the total budget constraint of the SP; and (5), (6) 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:

$$f_t(x_t, \boldsymbol{y}_t) = -\sum_{k\in\mathcal{K}_t} a_k \log(x_t + y_k + 1), \tag{7}$$

$$g_t(x_t, \boldsymbol{y}_t) = x_t p_t + \boldsymbol{y}_t^\top \boldsymbol{q}_t - B. \tag{8}$$

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.

[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.

• *(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.

D. 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 [31], constrained OCO problems like $(\mathbb{P})$ cannot learn efficiently to perform as benchmarks from set $\mathcal{X}_T^{max}$ (1). 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 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:

$$(\mathbb{P}_t): \min_{\{\boldsymbol{z}_t\}\in\Gamma^{K+1}} f_t(\boldsymbol{z}_t) \quad \text{s.t.} \quad g_t(\boldsymbol{z}_t) \le 0. \tag{9}$$

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:

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

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.,

$$\lim_{T\to\infty} \frac{R_T}{T} = 0, \; \lim_{T\to\infty} \frac{V_T}{T} = 0, \quad \forall \{f_t\}_{t=1}^T. \tag{10}$$

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 $R_T^s \le R_T$ and by achieving $R_T = o(T)$ we ensure the same for R_T^s. We refer the interested reader to [12], [32] for a detailed discussion about benchmarks.

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

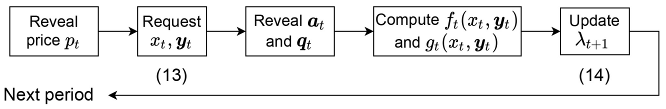

Reveal price p_t
Request $x_t, \boldsymbol{y}_t$
(13)
Reveal $\boldsymbol{a}_t$ and $\boldsymbol{q}_t$
Compute $f_t(x_t, \boldsymbol{y}_t)$ and $g_t(x_t, \boldsymbol{y}_t)$
Update λ_{t+1}
(14)
Next period

Fig. 2. Online learning model and sequence of actions in the system.

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 $t = 1, \ldots, T$ do
2     Observe the $t$-period price $p_t$
3     Decide $(x_t, \boldsymbol{y}_t)$ by solving (13)
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 (14)
```

IV. Online Reservation Policy

A. 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:

$$\widetilde{L}_t(\boldsymbol{z}, \lambda) = f_t(\boldsymbol{z}) + \lambda g_t(\boldsymbol{z}) \tag{11}$$

where $\lambda \in \mathbb{R}_+$ is the Lagrange multiplier associated with constraint (8). 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 (11), we optimize the modified Lagrangian with a linearized objective and a Euclidean regularizer with non-negative parameter ν:

$$L_t(\boldsymbol{z}, \lambda) = \nabla f_t(\boldsymbol{z}_t)^\top(\boldsymbol{z} - \boldsymbol{z}_t) + \lambda g_t(\boldsymbol{z}) + \frac{\|\boldsymbol{z} - \boldsymbol{z}_t\|^2}{2\nu}, \tag{12}$$

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. 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):

$$\boldsymbol{z}_t = \arg\min_{\boldsymbol{z}\in\mathcal{Z}} L_{t-1}(\boldsymbol{z}, \lambda_t). \tag{13}$$

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 (13). 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 (12), and can update its dual variable by executing the dual gradient ascent with positive step-size μ:

$$\lambda_{t+1} = [\lambda_t + \mu \nabla_\lambda L_t(\boldsymbol{z}_t, \lambda)]_+ \tag{14}$$

These are the dual variables (or, shadow prices for (4)) 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.

B. Performance Analysis

We start with the necessary model assumptions.

Assumption 1: 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 2: The utility parameters are uniformly bounded, i.e., $a_k \in [0,a]$, $\forall k$ where $a < \infty$.

Assumption 3: 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:

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

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 (8) 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:

$$g_t(\tilde{\boldsymbol{z}}) \leq -\epsilon, \ \forall t, \quad \text{with } \epsilon > 0. \tag{15}$$

The Slater constraint qualification is sufficient for strong duality [40], 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 4: The slack constant ϵ in the Slater condition (15) is such that it holds $\epsilon > \widetilde{U}_g$.

The next Lemma characterizes the performance of OLR.

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

$$R_T \leq \frac{EU_z}{\nu} + \frac{\nu T G^2}{2} + \frac{(T+1)\mu M^2}{2} + \frac{E^2}{2\nu} + U_g\tilde{\lambda}$$
$$V_T \leq M + \frac{(2EG/\mu) + (E^2/2\nu\mu) + (M^2/2)}{\epsilon - \widetilde{U}_g},$$
$$\text{where: } M \triangleq \max\{D(p+Kq) - B, B\},$$
$$G \triangleq a\sqrt{K(K+1)}$$
$$\tilde{\lambda} \triangleq \mu M + \frac{2EG + (E^2/2\nu) + (\mu M^2/2)}{\epsilon - \widetilde{U}_g}.$$

C. 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 (13) and (14). The constraint function $g_t(\boldsymbol{z})$ being linear, the computational complexity of (13) is low and a closed form solution can be derived via the First Order Condition, as in (16) and (17). 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 (16) runs in K operations (sum over $\mathcal{K}_{t-1}$) and the solution to (17) runs in 1 operation, for each $k \in \mathcal{K}_t$. Therefore, the running time for (13) is of $2K = \mathcal{O}(K)$. Equation (14) 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 [12, Corollary 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 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 (13) and write:

$$x_t = \sum_{k\in\mathcal{K}_{t-1}} \frac{\nu a_k}{1 + x_{t-1} + y_k} - p_t\lambda_t\nu + x_{t-1}, \tag{16}$$

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 $|x_t - x_{t-1}|$.

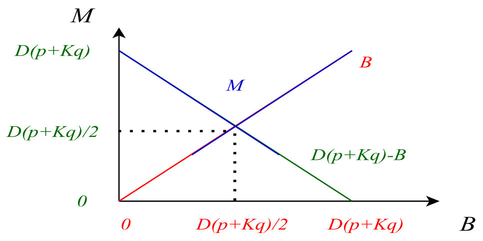


Fig. 3. Impact of the committed budget, *B*, on the value of parameter *M* that affects the bounds of R_T and V_T.

- Conversely, a small ν reduces the distance $|x_t - x_{t-1}|$.

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

$$y_k = \frac{\nu a_{k-K}}{1 + x_{t-1} + y_{k-K}} - q_{k-K}\lambda_t \nu + y_{k-K}, \quad k \in \mathcal{K}_t, \tag{17}$$

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. 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 Section VII.

V. 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.

A. Online Learning Reservations for Slice Orchestration

In this scenario, the SP makes multi-dimensional reservations using $\boldsymbol{x}_t, \boldsymbol{y}_k \in \Gamma_1 \times \ldots \times \Gamma_m$, 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:

$$f_t^\theta(\boldsymbol{x}_t, \boldsymbol{y}_t) = -\sum_{k \in K_t} a_k \log(\boldsymbol{\theta}_t^\top \boldsymbol{x}_t + \boldsymbol{\theta}_t^\top \boldsymbol{y}_k + 1), \tag{18}$$

$$g_t^\theta(\boldsymbol{x}_t, \boldsymbol{y}_t) = \boldsymbol{x}_t^\top \boldsymbol{p}_t + \sum_{k \in K_t} \boldsymbol{y}_k^\top \boldsymbol{q}_k - B, \tag{19}$$

and use the following modified Lagrange function:

$$L_t^\theta(\boldsymbol{z}, \lambda) = \nabla f_t^\theta(\boldsymbol{z}_t)^\top (\boldsymbol{z} - \boldsymbol{z}_t) + \lambda g_t^\theta(\boldsymbol{z}) + \frac{\|\boldsymbol{z} - \boldsymbol{z}_t\|^2}{2\nu}, \tag{20}$$

to update the primal and dual variables:

$$\boldsymbol{z}_t = \arg\min_{\boldsymbol{z} \in \mathcal{Z}_\theta} L_{t-1}^\theta(\boldsymbol{z}, \lambda_t), \tag{21}$$

$$\lambda_{t+1} = \Big[\lambda_t + \mu \nabla_\lambda L_t^\theta(\boldsymbol{z}_t, \lambda)\Big]_+. \tag{22}$$

Note that the prices are now vectors in $\mathbb{R}^m$, and the SP decision space is expanded to $\mathcal{Z}_\theta = (\Gamma_1 \times \ldots \times \Gamma_m)^{K+1}$, with $\Gamma_i = [0, D_i], i \leq m$.

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 (21). 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 the period the SP gains access to the new Lagrangian (20) and updates its dual variable (22).

B. Performance Analysis

The performance of Algorithm OLR-SO is conditioned upon the following minimal assumptions that complement, or update, Assumptions 1-4 that we stated for OLR in Section IV.

Assumption 5: 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 $p = \max_i\{p_i\}$, $q = \max_i\{q_i\}$.

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 $t = 1, \ldots, T$ do
2     Observe the $t$-period price $\boldsymbol{p}_t$
3     Decide $(\boldsymbol{x}_t, \boldsymbol{y}_t)$ by solving (21)
4     Observe $\boldsymbol{a}_t$, $\boldsymbol{\theta}_t$ and calculate $f_t^\theta(\boldsymbol{x}_t, \boldsymbol{y}_t)$ with (18)
5     Observe $\boldsymbol{q}_t$ and calculate $g_t^\theta(\boldsymbol{x}_t, \boldsymbol{y}_t)$ with (19)
6     Decide $\lambda_{t+1}$ by solving (22)
```

Assumption 6: 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 7: 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:

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

and we also define:

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

We see from (19) that it holds $\widetilde{U}_g^\theta \le (p + Kq) \sum_{i=1}^m D_i$.

We assume that the new problem:

$$\left(\mathbb{P}_t^\theta\right) : \min_{\boldsymbol{z}_t \in \mathcal{Z}_\theta} f_t^\theta(\boldsymbol{z}_t) \quad \text{s.t.} \quad g_t^\theta(\boldsymbol{z}_t) \le 0. \tag{23}$$

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

$$g_t(\tilde{\boldsymbol{z}}) \le -\epsilon, \; \forall t, \tag{24}$$

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

Assumption 8: The slack constant ϵ in the Slater condition (24) is such that it holds $\epsilon > \widetilde{U}_g^\theta$.

The performance of OLR-SO is characterized next.

Lemma 2: Under Assumptions 5-8, Algorithm OLR-SO achieves the following regret and constraint violation bounds w.r.t. the dynamic benchmark policy $\{\boldsymbol{x}_t^\star, \boldsymbol{y}_t^\star\}_{t=1}^T$:

$$R_T \le \frac{F U_z^\theta}{\nu} + \frac{\nu T G_\theta^2}{2} + \frac{(T+1)\mu M_\theta^2}{2} + \frac{F^2}{2\nu} + U_g^\theta \tilde{\lambda}_\theta$$

$$V_T \le M_\theta + \frac{(2FG_\theta/\mu) + (F^2/2\nu\mu) + (M_\theta^2/2)}{\epsilon - \widetilde{U}_g^\theta},$$

$$\text{where: } M_\theta \triangleq \max\left\{ (p + Kq) \sum_{i=1}^m D_i - B, B \right\},$$

$$G_\theta \triangleq a\theta\sqrt{mK(K+1)},$$

$$\tilde{\lambda}_\theta \triangleq \mu M_\theta + \frac{2FG_\theta + (F^2/2\nu) + (\mu M_\theta^2/2)}{\epsilon - \widetilde{U}_g^\theta}.$$

C. Discussion

Complexity Analysis OLR-SO: Here we discuss the complexity of the OLR-SO algorithm. Deriving the new Lagrangian (20) with regard to $\boldsymbol{z}$ is more challenging. The required running time to compute (21) 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 Appendix B. The solution of (22) 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 ($U_g^\theta \ge U_g$) and on the dynamic benchmark sequence ($U_z^\theta \ge U_z$). 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 3: 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 $D_i, i \le m$.

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 $\boldsymbol{p}_t$.

VI. 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 $\boldsymbol{q}_t$ it observes during that period. We first present the baseline scenario and then extend MTS for the slice composition case.

A. 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:

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

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

Algorithm OLR-MTS: OLR for Mixed Time Scale

```
Initialize:
  $\lambda_1 = 0, x_0 \in \Gamma, \boldsymbol{y}_0 \in \Gamma^K, \nu = \mu = \mathcal{O}(T^{-1/3}), \hat{\nu} = \nu, \hat{\mu} = \mu$
1  for $t = 1, \dots, T$ do
2      Observe the $t$-period price $p_t$
3      Decide $(x_t, \boldsymbol{y}_t)$ by solving (13)
4      Calculate $B_t = (B - x_t p_t)/K$
5      for $k = 1, \dots, K$ do
6          Decide $y_k$ by solving (26) 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 (27)
9      Observe $\boldsymbol{a}_t$ and calculate $f_t(x_t, \boldsymbol{y}_t)$
10     Observe $\boldsymbol{q}_t$ and calculate $g_t(x_t, \boldsymbol{y}_t)$
```

Also, the t-slot Lagrangian is:

$$L_k\left(y, \hat{\lambda}\right) = \nabla f_k(y_k)(y - y_k) + \hat{\lambda} g_k(y) + \frac{(y - y_k)^2}{2\hat{\nu}} \quad (25)$$

where $\hat{\lambda} \in \mathbb{R}_+$ 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 (13) (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:

$$y_k = \arg\min_{y \in \Gamma} L_{k-1}\left(y, \hat{\lambda}_k\right). \quad (26)$$

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 (25) and can update its dual variable (step 8) by executing:

$$\hat{\lambda}_{k+1} = \left[\hat{\lambda}_k + \hat{\mu} \nabla_\lambda L_k\left(y_k, \hat{\lambda}\right)\right]_+. \quad (27)$$

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 $\hat{\nu}$ and $\hat{\mu}$ are the steps for the intra-period decisions, and we set them to $\hat{\nu} = \nu$ and $\hat{\mu} = \mu$.

B. 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:

$$\begin{aligned} f_k^\theta(\boldsymbol{y}_k) &= -a_k \log\left(\boldsymbol{x}_t^\top \boldsymbol{\theta}_t + \boldsymbol{y}_k^\top \boldsymbol{\theta}_t + 1\right), \\ g_k^\theta(\boldsymbol{y}_k) &= \boldsymbol{y}_k^\top \boldsymbol{q}_k - B_t, \end{aligned} \quad (28)$$

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:

$$L_k^\theta\left(\boldsymbol{y}, \hat{\lambda}\right) = \nabla f_k^\theta(\boldsymbol{y}_k)^\top (\boldsymbol{y} - \boldsymbol{y}_k) + \hat{\lambda} g_k^\theta(\boldsymbol{y}) + \frac{||\boldsymbol{y} - \boldsymbol{y}_k||^2}{2\hat{\nu}} \quad (29)$$

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

```
Initialize:
  $\lambda_1 = 0, \boldsymbol{x}_0 \in \prod_{i=1}^m \Gamma_i, \boldsymbol{y}_0 \in (\prod_{i=1}^m \Gamma_i)^K, \nu = \mu = \mathcal{O}(T^{-1/3}), \hat{\nu} = \nu, \hat{\mu} = \mu$
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 (21)
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 (30) 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 (31)
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 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:

$$\boldsymbol{y}_k = \arg\min_{\boldsymbol{y} \in \prod_{i=1}^m \Gamma_i} L_{k-1}^\theta\left(\boldsymbol{y}, \hat{\lambda}_k\right), \quad (30)$$

$$\hat{\lambda}_{k+1} = \left[\hat{\lambda}_k + \hat{\mu} \nabla_\lambda L_k^\theta\left(\boldsymbol{y}_k, \hat{\lambda}\right)\right]_+. \quad (31)$$

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 $\hat{\nu}$ and $\hat{\mu}$ are the steps for the intra-period decisions, and we use $\hat{\nu} = \nu$ and $\hat{\mu} = \mu$.

C. 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 (26) and (27), 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 (17) provided that expression for the single-time scale model. Here, we can write instead:

$$y_k = \frac{\hat{\nu} a_{k-1}}{1 + x_{t-1} + y_{k-1}} - q_{k-1} \hat{\lambda}_k \hat{\nu} + y_{k-1}. \quad (32)$$

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 (17) 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., $x_t, y_{(t-1)K+1}, y_{(t-1)K+2}, \cdots, y_{(t-1)K+K}$. This, in turn, might affect the policy implementation for large problem instance and/or when the time-slots are very small.

VII. 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.

A. 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 (18).

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:

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

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 $\nu = \mu = \hat{\nu} = \hat{\mu} = T^{-1/3} = 0.1$, for $T = 1000$, as these are recommended values for the learning rates in [12]. 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.

TABLE II
SIMULATION PARAMETERS

Symbol	Physical Meaning
ν	Step-size of the primal update
$\hat{\nu}$	Step-size of the primal slot update
μ	Step-size of the dual update
$\hat{\mu}$	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
$P_0 + p, Q_0 + q, A_0 + a \dots$	Upper bounds of the non-stationary traces
c_f	Discount factor of the period price wrt the spot price

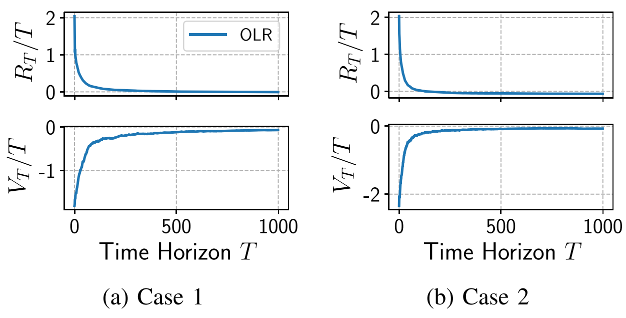
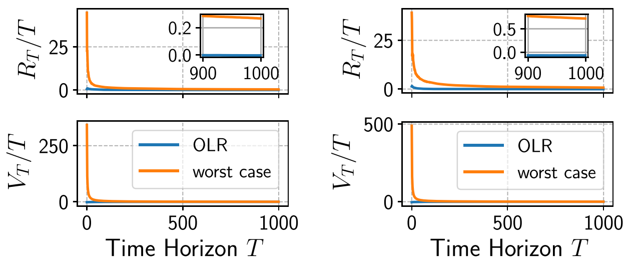


Fig. 4. *Evolution of R_T/T and V_T/T.* Simulation parameters are set to $K = 3$, $c_f = 3$, $D = 1$. (4(a)): $a = p = q = 1$, $B = 2$, $\nu = \mu = 0.1$. (4(b)): $K_0 = 5$, $A_0 = P_0 = Q_0 = 0.5$, $a = p = q = 1.5$, $B = 4$, $\nu = \mu = 0.1$.

B. Evaluation of OLR and OLR-MTS

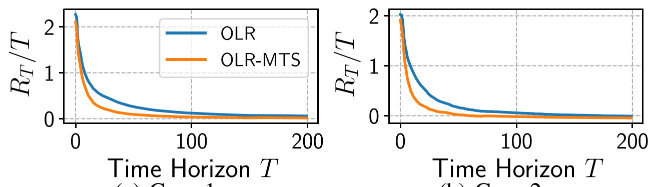
OLR Convergence: First, we verify that R_T and V_T grow sublinearly given that $U_g = o(T)$ and $U_z = o(T)$; see Figs. 4(a) and 4(b). In *Case* 1, we set the different parameters as shown in the respective caption, and select the optimal budget to be $B = D(p + Kq)/2 = 2$. Under these conditions, we observe in Fig. 4(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(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. 5(a) and 5(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.



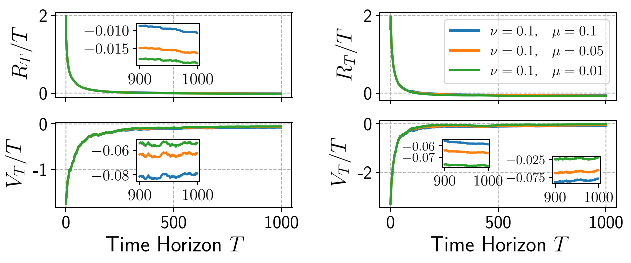
(a) Case 1 (b) Case 2

Fig. 5. *Evolution of R_T/T and V_T/T.* Simulation parameters are set to $K = 3$, $c_f = 3$, $D = 1$. (5(a)): $a = p = q = 1$, $B = 2$, $\nu = \mu = 0.1$. (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

Fig. 6. *Comparison of OLR and OLR-MTS.* Simulation parameters are $K = 3$, $c_f = 3$, $D = 1$. (6(a)): $a = p = q = 1$, $B = 2$, $\nu = \hat{\nu} = \mu = \hat{\mu} = 0.1$. (6(b)): $K_0 = 5$, $A_0 = P_0 = Q_0 = 0.5$, $a = p = q = 1.5$, $B = 4$, $\nu = \hat{\nu} = \mu = \hat{\mu} = 0.1$.



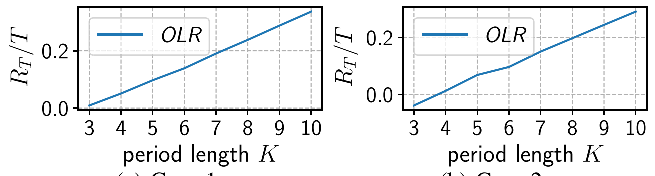
(a) Case 1 (b) Case 2

Fig. 7. *Impact of Learning Rates.* Simulation parameters: $K = 3$, $c_f = 3$, $D = 1$. (7(a)): $a = p = q = 1$, $B = 2$. (7(b)): $K_0 = 5$, $A_0 = P_0 = Q_0 = 0.5$, $a = p = q = 1.5$, $B = 4$.

Comparison of OLR and OLR-MTS: In Figs. 6(a) and 6(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 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.

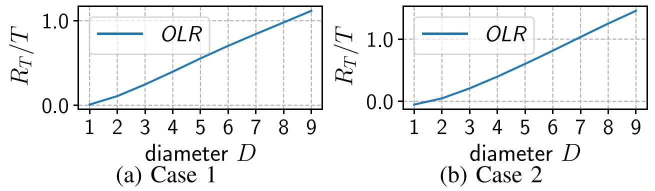
Influence of ν and μ: In Fig. 7, we observe the influence of the learning rates on the convergence of R_T/T and V_T/T. We set $\nu = 0.1$ and use different values for μ, namely $\mu \in \{0.1, 0.05, 0.01\}$. As predicted in our analysis, a lower value of μ favorites over-reservation, hence the performance 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.

K and D Sensitivity: In Figs. 8(a) and 8(b), as K increases, the regret performance drops in a linear fashion. This is quite



(a) Case 1 (b) Case 2

Fig. 8. Simulation parameters are set to: 20 runs, $T = 500$, $c_f = 3$, $D = 1$. (8(a)): $a = p = q = 1$, $B = (K + 1)/2$, $\nu = \mu = 0.1$. (8(b)): $K_0 = 5$, $A_0 = P_0 = Q_0 = 0.5$, $a = p = q = 1.5$, $B = K + 1$, $\nu = \mu = 0.1$.



(a) Case 1 (b) Case 2

Fig. 9. Simulation parameters are set to: 20 runs, $T = 500$, $c_f = 3$, $K = 3$. (9(a)): $a = p = q = 1$, $B = 2D$, $\nu = \mu = 0.1$. (9(b)): $K_0 = 5$, $A_0 = P_0 = Q_0 = 0.5$, $a = p = q = 1.5$, $B = 4D$, $\nu = \mu = 0.1$.

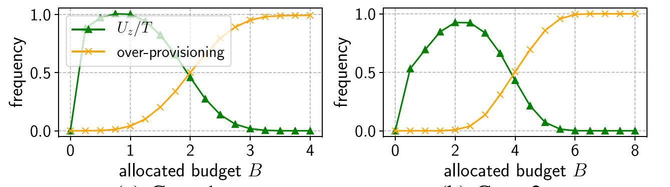
reflective of the regret bound, where most terms are either linear or quadratic with regard to K. To fairly compare the performance of the OLR for different values of K, we need to keep $B = D(p + Kq)/2$ at each point. The latter value of B gives the minimum for parameter M, henceforth the regret bound. This way we evaluate the performance of the OLR, holding that the regret bound is at its minimum at each point K. In Figs. 9(a) and 9(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., $\boldsymbol{z}_t^\star = [D, \ldots, D]$ 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:

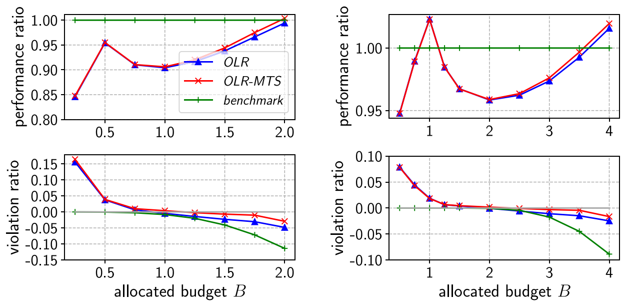
$$\mathbb{P}\Big\{ Dp_t + Dq_{(t-1)K+1} + \ldots + Dq_{tK} \leq B \Big\}$$

defined for $B \in [0, D(p + Kq)]$. The latter function is the cdf of the Irwin-Hall distribution [41] 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



(a) Case 1 (b) Case 2

Fig. 10. Simulation parameters are set to: 20 runs, $T = 500$, $K = 3$, $c_f = 3$, $D = 1$. (10(a)): $a = p = q = 1$. (10(b)): $K_0 = 5$, $A_0 = P_0 = Q_0 = 0.5$, $a = p = q = 1.5$.



(a) Case 1 (b) Case 2

Fig. 11. Simulation parameters are set to: 20 runs, $T = 500$, $c_f = 3$, $D = 1$, $K = 3$. (11(a)): $a = p = q = 1$, $\nu = \hat{\nu} = \mu = \hat{\mu} = 0.1$. (11(b)): $K_0 = 5$, $A_0 = P_0 = Q_0 = 0.5$, $a = p = q = 1.5$, $\nu = \hat{\nu} = \mu = \hat{\mu} = 0.1$.

$[0, p + Kq)]$ (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. 10, 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. 10(a) and 10(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:

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

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. 11, 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. 11, we focus on the performance ratio, which is the performance (given by (3)) 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. 11(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. 11(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.

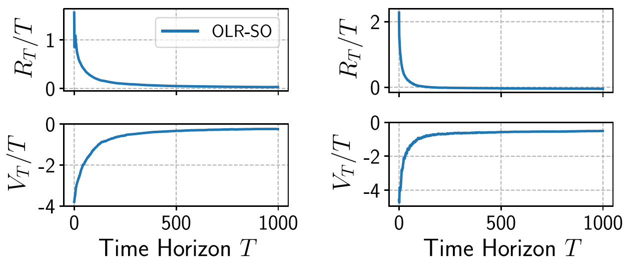
C. Evaluation of OLR-SO and OLR-MTS-SO

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:

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

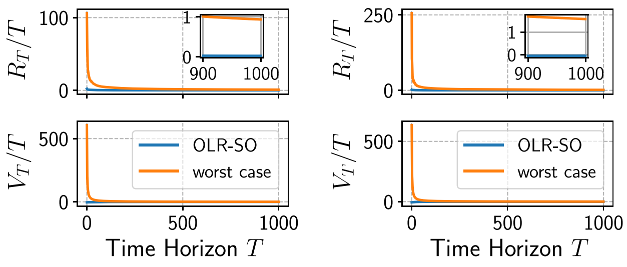
where, as before, parameters A_0, Q_0, P_0, θ_0 are the amplitudes of the sine waves and $\mathcal{U}$ is the added uniform noise on $[A_0, a]$, $[Q_0, q]$, etc. The sine part of $\boldsymbol{q}_k$, $\boldsymbol{p}_t$ and $\boldsymbol{\theta}_t$ is the offset that we add to the components of the uniform 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. 12(a) and 12(b). In *Case* 1, we select the system parameters shown in the respective caption, and for which the optimal budget is $B = (D_1 + D_2 + D_3)(p + Kq)/2 = 4$.



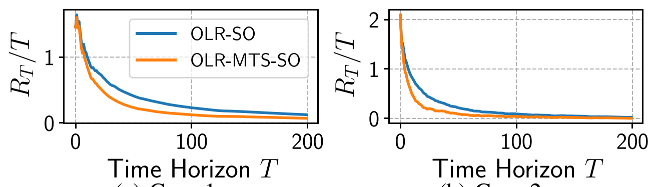
(a) Case 1 (b) Case 2

Fig. 12. *Evolution of R_T/T and V_T/T.* Simulation parameters are set to $K = 3$, $c_f = 3$, $\boldsymbol{D} = [0.5,\ 1,\ 0.5]$, $m = 3$. (12(a)): $a = p = q = \theta = 1$, $B = 4$, $\nu = \mu = 0.1$. (12(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

Fig. 13. *Evolution of R_T/T and V_T/T.* Simulation parameters are set to $K = 3$, $c_f = 3$, $\boldsymbol{D} = [0.5, 1, 0.5]$, $m = 3$. (13(a)): $a = p = q = \theta = 1$, $B = 4$, $\nu = \mu = 0.1$. (13(b)): $K_0 = 5$, $A_0 = P_0 = Q_0 = \theta_0 = 0.5$, $a = p = q = \theta = 1.5$, $B = 8$, $\nu = \mu = 0.1$.



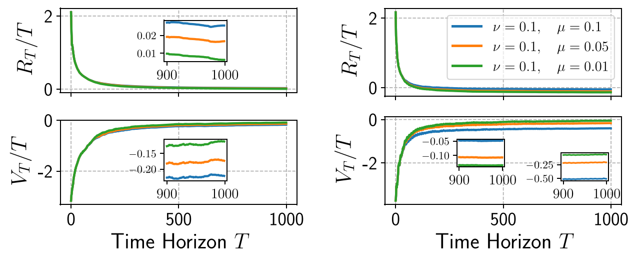
(a) Case 1 (b) Case 2

Fig. 14. *Evolution of R_T/T and V_T/T.* Simulation parameters are set to $K = 3$, $c_f = 3$, $\boldsymbol{D} = [0.5, 1, 0.5]$, $m = 3$ (14(a)): $a = p = q = \theta = 1$, $B = 4$, $\nu = \hat{\nu} = \mu = \hat{\mu} = 0.1$. (14(b)): $K_0 = 5$, $A_0 = P_0 = Q_0 = \theta_0 = 0.5$, $a = p = q = \theta = 1.5$, $B = 8$, $\nu = \hat{\nu} = \mu = \hat{\mu} = 0.1$.

We can observe in Fig. 12(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. 12(b) the convergence of R_T/T and V_T/T.

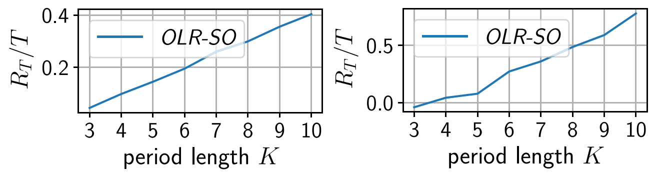
Theory vs Practice: In Figs. 13(a) and 13(b), we plot the convergence of both the theoretical bound and the actual solution OLR-SO (regret and fit). Again, the solution converges faster than its worst-case scenario. Also, zooming in the last 100 periods show that the theoretical regret bound is slowly decaying towards 0.

Comparison of OLR-SO and OLR-MTS-SO: In Figs. 14(a) and 14(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.



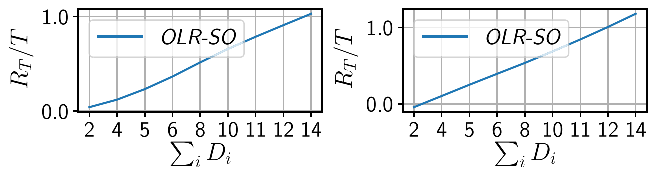
(a) Case 1 (b) Case 2

Fig. 15. *Evolution of R_T/T and V_T/T.* Simulation parameters are set to $K = 3$, $c_f = 3$, $\boldsymbol{D} = [0.5, 1, 0.5]$, $m = 3$. (15(a)): $a = p = q = \theta = 1$, $B = 4$. (15(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

Fig. 16. Simulation parameters are set to: 20 runs, $T = 500$, $c_f = 3$, $\boldsymbol{D} = [0.5, 1, 0.5]$, $m = 3$. (16(a)): $a = p = q = \theta = 1$, $B = K + 1$, $\nu == \mu = 0.1$. (16(b)): $K_0 = 5$, $A_0 = P_0 = Q_0 = \theta_0 = 0.5$, $a = p = q = \theta = 1.5$, $B = 2(K + 1)$, $\nu == \mu = 0.1$.

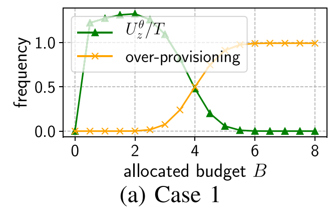


(a) Case 1 (b) Case 2

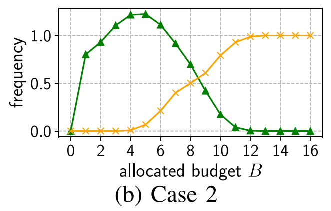
Fig. 17. Simulation parameters are set to: 20 runs, $T = 500$, $c_f = 3$, $K = 3$, $m = 3$. (17(a)): $a = p = q = \theta = 1$, $B = 2(\sum_i D_i)$, $\nu = \mu = 0.1$. (17(b)): $K_0 = 5$, $A_0 = P_0 = Q_0 = \theta_0 = 0.5$, $a = p = q = \theta = 1.5$, $B = 4(\sum_i D_i)$, $\nu = \mu = 0.1$.

Influence of ν and μ: In Fig. 15, we observe the influence of the learning rates 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. 16(a) and 16(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. 17(a) and 17(b), $\boldsymbol{D}$ is a vector of dimension $m = 3$. 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

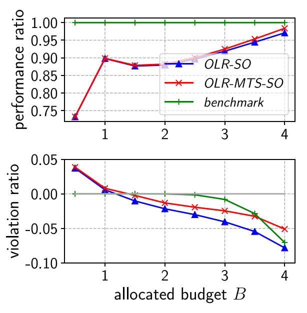


(a) Case 1

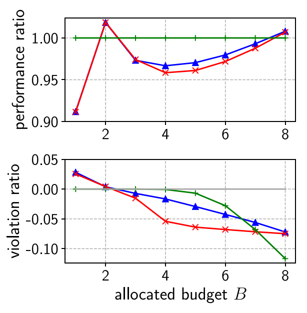


(b) Case 2

Fig. 18. Simulation parameters are set to: 20 runs, $T = 500$, $K = 3$, $c_f = 3$, $m = 3$, $\boldsymbol{D} = [0.5, 1, 0.5]$. (18(a)): $a = p = q = \theta = 1$. (18(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

Fig. 19. Simulation parameters are set to: 20 runs, $T = 500$, $c_f = 3$, $\boldsymbol{D} = [0.5, 1, 0.5]$, $m = 3$, $K = 3$. (19(a)): $a = p = q = \theta = 1$, $\nu = \hat{\nu} = \mu = \hat{\mu} = 0.1$. (19(b)): $K_0 = 5$, $A_0 = P_0 = Q_0 = \theta_0 = 0.5$, $a = p = q = \theta = 1.5$, $\nu = \hat{\nu} = \mu = \hat{\mu} = 0.1$.

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. 18(a), we focus on the performance and violation ratios for $B \in]0, 4]$. In Fig. 19(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. 19(b) that $B = (D_1 + D_2 + D_3)(p + P_0 + K(q + Q_0))/2 = 8$ and $B = (D_1 + D_2 + D_3)(p + P_0 + K(q + Q_0))/8 = 2$ 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. 18(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.

VIII. Conclusion

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.

Appendix A
Proof of Lemma 1

Our starting point is [12, Ths. 1 and 2]. First, we will prove that:

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

Indeed, we can calculate the gradient vector:

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

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:

$$\|\nabla f_t(\boldsymbol{z}_t)\|^2 \leq \|\nabla f_t(\boldsymbol{0})\|^2 = \left(\sum_{k\in\mathcal{K}_t} \frac{a_k}{1}\right)^2 + \frac{a^2_{(t-1)K+1}}{1}$$
$$+ \ldots + \frac{a^2_{tK}}{1} \leq \left(\sum_{k\in\mathcal{K}_t} a\right)^2 + a^2 + \cdots + a^2$$
$$= K^2a^2 + Ka^2 = a^2K(K+1)$$

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

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

$$0 \leq p_t x_t + \sum_{k\in\mathcal{K}_t} q_k y_k \leq pD + KqD$$
$$\Rightarrow -B \leq g_t(x_t, \boldsymbol{y}_t) \leq D(p + Kq) - B$$

$$\Rightarrow |g_t(x_t, \boldsymbol{y}_t)| \leq \max\{D(p+Kq)-B, B\}.$$

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

$$\forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{Z}, \quad d(\boldsymbol{x}, \boldsymbol{y}) \leq \sqrt{D^2 + D^2 + \cdots + D^2} \quad (33)$$
$$= D\sqrt{K+1} \quad (34)$$

Replacing these bounds in [12, Th. 1] we obtain the bounds of Lemma 1.

Appendix B
Proof of Lemma 2

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:

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

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 $\boldsymbol{z}_t = \boldsymbol{0}$ 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:

$$\left\|\nabla f_t^\theta(\boldsymbol{z}_t)\right\|^2 \leq \left\|\nabla f_t^\theta(\boldsymbol{0})\right\|^2 = \sum_{i=1}^m \left(\sum_{k\in\mathcal{K}_t} \frac{a_k}{1}\right)^2 \theta_i^2$$
$$+ \sum_{i=1}^m \frac{\left(a_{(t-1)K+1}\right)^2}{1}\theta_i^2 + \cdots + \sum_{i=1}^m \frac{(a_{tK})^2}{1}\theta_i^2$$
$$\leq m\left(\sum_{k\in\mathcal{K}_t} a\right)^2 \theta^2 + ma^2\theta^2 + \ldots ma^2\theta^2$$
$$\leq mK^2a^2\theta^2 + Kma^2\theta^2 = (a\theta)^2 mK(K+1)$$

which gives us the expression of the bound G_θ.

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

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

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

$$\forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{Z}, \quad d(\boldsymbol{x}, \boldsymbol{y}) \leq \sqrt{\left(D_1^2 + \cdots + D_m^2\right)(K+1)} \quad (35)$$
$$= \sqrt{D_1^2 + \cdots + D_m^2}\sqrt{K+1} \quad (36)$$

And plugging these terms in [12, Th. 1] we obtain Lemma 2.

Appendix C
Proof of Lemma 3

We denote $\boldsymbol{z} = [z_0, z_1, \ldots, z_K]^\top$ and $U_g = \sum_{t=1}^T U_g^t$, then:

$$U_g^t = \max_{\boldsymbol{z}\in\mathcal{Z}} \left|[g_t(\boldsymbol{z}) - g_{t-1}(\boldsymbol{z})]_+\right|$$
$$= \max_{\boldsymbol{z}\in\mathcal{Z}} \left|\left[(p_t - p_{t-1})z_0 + \sum_{k=1}^K \left(q_{(t-1)K+k} - q_{(t-2)K+k}\right)z_k\right]_+\right|$$
$$= \max_{\boldsymbol{z}\in\mathcal{Z}} \left|\left[Y_0 z_0 + \sum_{k=1}^K Y_k z_k\right]_+\right| \quad (37)$$

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.

$$\mathbb{E}[X_i] = (1/2)0 + (1/2)\mathbb{E}\left[X_{i|]0,rD]}\right] = \frac{1}{2}\frac{rD}{3} \quad (38)$$

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

$$\mathbb{E}\left[U_g^t\right] = \mathbb{E}\left[X_0 + \sum_{k\geq 1} X_k\right] = \left(pD/6 + \sum_{k\geq 1} qD/6\right)$$
$$= D(p+Kq)/6 \quad (39)$$

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

$$\frac{U_g}{T} = \frac{U_g^1 + \cdots + U_g^T}{T} \quad (40)$$

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:

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

We denote $\boldsymbol{z} = [z_0^1, \ldots, z_0^m, z_1^1, \ldots, z_1^m, \ldots, z_K^1, \ldots, z_K^m]$. 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:

$$\mathbb{E}\left[U_g^{\theta,t}\right] = (D_1 + \cdots + D_m)(p+Kq)/6 \quad (41)$$

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

References

[1] J. B. Monteil, G. Iosifidis, and L. DaSilva, "No-regret slice reservation algorithms," in *Proc. IEEE ICC*, 2021, pp. 1–7.

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

[3] K. Samdanis, X. Costa-Perez, and V. Sciancalepore, "From network sharing to multi-tenancy: The 5G network slice broker," *IEEE Commun. Mag.*, vol. 54, pp. 32–39, Jul. 2016.

[4] S. Vassilaras *et al.*, "The algorithmic aspects of network slicing," *IEEE Commun. Mag.*, vol. 55, no. 8, pp. 112–119, Aug. 2017.

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

[6] "Compute Engine Pricing." Google Cloud Platform. 2021. [Online]. Available: https://cloud.google.com/compute/

[7] "Google Cloud Platform." Preemptible Virtual Machines. 2021. [Online]. Available: https://cloud.google.com/preemptible-vms/

[8] "Amazon EC2." Reserved Instances. 2021. [Online]. Available: https://aws.amazon.com/ec2/purchasing-options/reserved-instances/

[9] "Amazon EC2." Spot Instances. 2021. [Online]. Available: https://aws.amazon.com/ec2/spot/

[10] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *Proc. ICML*, 2003, pp. 928–935.

[11] E. Hazan, "Introduction to online convex optimization," *Found. Trends Optim.*, vol. 2, nos. 3–4, pp. 157–325, 2016.

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

[13] M. Mahdavi, R. Jin, and T. Yang, "Trading regret for efficiency: Online convex optimization with long term constraints," *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 2503–2528, 2012.

[14] R. Jenatton, J. C. Huang, and C. Archambeau, "Adaptive algorithms for online convex optimization with long-term constraints," in *Proc. ICML*, 2016, pp. 1–9.

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

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

[17] 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 *Proc. ACM CoNEXT*, 2018, pp. 353–365.

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

[19] 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 J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1455–1470, Jun. 2019.

[20] Y. Zhang, S. Bi, and Y. J. A. Zhang, "Joint spectrum reservation and on-demand request for mobile virtual network operators," *IEEE Trans. Commun.*, vol. 66, no. 7, pp. 2966–2977, Jul. 2018.

[21] H. Zhang and V. W. S. Wong, "A two-timescale approach for network slicing in C-RAN," *IEEE Trans. Veh. Technol.*, vol. 69, no. 6, pp. 6656–6669, Jun. 2020.

[22] J. Monteil, J. Hribar, P. Barnard, Y. Li, and L. A. DaSilva, "Resource reservation within sliced 5G networks: A cost-reduction strategy for service providers," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, 2020, pp. 1–6.

[23] G. Wang, G. Feng, T. Q. S. Quek, S. Qin, R. Wen, and W. Tan, "Reconfiguration in network slicing—Optimizing the profit and performance," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 2, pp. 591–605, Jun. 2019.

[24] N. Liakopoulos, G. Paschos, and T. Spyropoulos, "No regret in cloud resources reservation with violation guarantees," in *Proc. IEEE INFOCOM*, 2019, pp. 1–34.

[25] M. Jiang, M. Condoluci, and T. Mahmoodi, "Network slicing in 5G: An auction-based model," in *Proc. IEEE ICC*, 2017, pp. 1–6.

[26] M. Zafer, Y. Song, and K.-W. Lee, "Optimal bids for spot VMs in a cloud for deadline constrained jobs," in *Proc. IEEE CLOUD*, 2012, pp. 75–82.

[27] M. Lumpe, M. B. Chhetri, Q. B. Vo, and R. Kowalczyk, "On estimating minimum bids for Amazon EC2 spot instances," in *Proc. IEEE/ACM CCGrid*, 2017, pp. 391–400.

[28] P. Sharma, D. Irwin, and P. Shenoy, "How not to bid the cloud," in *Proc. HotCloud*, 2016, pp. 1–6.

[29] M. Khodak, L. Zheng, A. S. Lan, C. Joe-Wong, and M. Chiang, "Learning cloud dynamics to optimize spot instance bidding strategies," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, 2018, pp. 2762–2770.

[30] L. Zheng, C. Joe-Wong, C. W. Tang, M. Chiang, and X. Wang, "How to bid the cloud," in *Proc. ACM SIGCOMM*, 2015, pp. 71–84.

[31] S. Mannor, J. N. Tsitsiklis, and J. Y. Yu, "Online learning with sample path constraints," *J. Mach. Learn. Res.*, vol. 10, pp. 569–590, Jun. 2009.

[32] V. Valls, G. Iosifidis, D. Leith, and L. Tassiulas, "Online convex optimization with perturbed constraints: Optimal rates against stronger benchmarks," in *Proc. AISTATS*, 2020, pp. 1–10.

[33] N. Liakopoulos, A. Destounis, G. Paschos, T. Spyropoulos, and P. Mertikopoulos, "Cautious regret minimization: Online optimization with long-term budget constraints," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 3944–3952.

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

[35] X. Cao and K. R. Liu, "Online convex optimization with time-varying constraints and bandit feedback," *IEEE Trans. Autom. Control*, vol. 64, no. 7, pp. 2665–2680, Jul. 2019.

[36] X. Yi, X. Li, L. Xie, and K. H. Johansson, "Distributed online convex optimization with time-varying coupled inequality constraints," *IEEE Trans. Signal Process.*, vol. 68, no. 1, pp. 731–746, Jan. 2020.

[37] S. Shenker, "Fundamental design issues for the future Internet," *IEEE J. Sel. Areas Commun.*, vol. 13, no. 7, pp. 1176–1188, Sep. 1995.

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

[39] S. G. Shakkottai and R. Srikant, *Network Optimization and Control*. Hanover, MA, USA: Now Publ., Inc., 2008.

[40] D. Bertsekas, *Nonlinear Programming*, vol. 3. Belmont, MA, USA: Athena Sci., 2016.

[41] N. L. Johnson, S. Kotz, and N. Balakrishnan, *Continuous Univariate Distributions*, vol. 2. New York, NY, USA: Wiley, 1995, p. 289.

Jean-Baptiste Monteil received the Diploma degree in electronics and telecommunications engineering from the Telecom SudParis School in 2018. He is currently pursuing the Ph.D. degree with Trinity College Dublin. His research interests include machine learning, online learning and online convex optimization, with applications to network slicing, and software-defined networks.

George Iosifidis (Member, IEEE) received the Diploma degree in electronics and telecommunications engineering from Greek Air Force Academy in 2000, and the Ph.D. degree from University of Thessaly, Greece, in 2012. He was a Postdoctoral researcher with CERTH, Greece, from 2012 to 2014; Yale University, USA, from 2014 to 2016; and an Assistant Professor with Trinity College Dublin from 2016 to 2020. He is currently an Assistant Professor with the Delft University of Technology. His research interests lie in the broad area of network optimization and economics, with a recent focus on edge computing, data caching systems, and sharing-economy platforms. He serves as an Editor for IEEE TRANSACTIONS ON COMMUNICATIONS, IEEE/ACM TRANSACTIONS ON NETWORKING, and IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING.

Luiz A. DaSilva (Fellow, IEEE) is the Executive Director of the Commonwealth Cyber Initiative and the Bradley Professor of Cybersecurity with Virginia Tech. Previously, he held the Chair of Telecommunications with Trinity College Dublin, where he served as the Director of CONNECT, the Science Foundation Ireland Research Center for Future Networks and Communications. His research focuses on distributed and adaptive resource management in wireless networks, and in particular radio resource sharing, dynamic spectrum access, and the application of game theory and machine learning to wireless networks. He is an IEEE Fellow for contributions to cognitive networking and to resource management in wireless networks.