No-Regret Slice Reservation Algorithms

Jean-Baptiste Monteil*, George Iosifidis†, Luiz DaSilva‡
*School of Computer Science and Statistics, Trinity College Dublin
†Delft University of Technology, Netherlands
‡Department of Electrical and Computer Engineering, Virginia Tech, USA

***Abstract*—Emerging network slicing markets promise to boost the utilization of expensive network resources and to unleash the potential of over-the-top services. Their success, however, 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 slices in order to maximize their performance while not exceeding their budget. 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 a hypothetical policy that has knowledge of future demand and prices. We extend our framework for the scenario the SP decides dynamically its slice orchestration, where it additionally needs to learn which resource composition is performance - maximizing; and we propose a mixed-time scale scheme that allows the SP to leverage any spot-market information revealed between its reservations. We evaluate our learning framework and its extensions using a variety of simulation scenarios and following a detailed parameter sensitivity analysis.**

***Index Terms*—Online convex optimization, network slicing markets, virtualization, resource reservation, SP utility maximization, constrained optimization, duality.**

I. Introduction

Motivation. The increasing softwarization of wireless networks coupled with the proliferation of over-the-top service providers (SPs) which rely on network operators' infrastructure (NOs), have spurred numerous studies for network slicing solutions, cf. [1], [2]. For instance, researchers have proposed embedding algorithms for assisting NOs to accommodate heterogeneous slices [3]; and pricing mechanisms to maximize the operators' revenue from selling slices to different SPs [4]. These schemes are expected to operate in near-real time and enable the fine-grained (re-)allocation of network resources; hence, boosting their utilization efficiency. Yet, an aspect that has received less attention is how the SPs should request slices.

The envisioned network slicing markets draw ideas from pertinent cloud computing marketplaces [5]–[8], where cloud providers adapt dynamically their offered prices, and SPs complement their reservations with on-the-fly bidding in spot markets. This flexibility compounds the slice-reservation task of each SP which has to request resources without knowing the needs of its users; to decide between (lower-cost) advance reservation and (higher-cost) dynamic reservation; and to anticipate the future slice prices that, in turn, depend on the NO's internal needs and the requests of other SPs. If these decisions are ineffective, the SPs might over/under-reserve resources, which will lead to network under-utilization or induce prohibitively high servicing costs. And these effects can nullify the anticipated benefits of slicing.

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

Related Work. The design of slice markets is a relatively new research area. In [9], the authors proposed a mechanism for the NO to auction its sliced resources; [10] formulated slicing as a utility maximization problem; and [11] considered QoS metrics when serving the slices. Similarly, [12] studied the impact of slice overbooking; [13] employed predictive capacity allocation for improving the slice composition; and [14] focused on dynamic slicing via reinforcement learning. Fewer works consider the problem from the SP's point of view. In [15], the authors consider a hybrid reservation/spot market where the SP reservations are decided by solving a stochastic problem, and a similar mixed time scale reservation model was studied in [16]. Our previous work [17] employed demand and price predictions (via neural networks) to assist the SP's reservations, while [18] focused on slice reconfiguration costs.

Similar reservation problems have been considered in the context of cloud computing, cf. [19], using various reservation criteria, e.g., costs or task deadlines, [20]. However, all above reservation solutions make (often strong) assumptions about knowing the user demands and resource prices, or assume that these parameters follow stationary stochastic processes. Our approach is fundamentally different, as we design reservation policies that do not require the SPs to know in advance their needs and the charged prices, nor we make any assumptions about the evolution of these parameters. In essence, we treat slice reservation as a learning problem for the SP, and rely on the theory of online convex optimization (OCO) [21] to design algorithms that adapt the reservations to users' needs and the NO's (potentially arbitrary) pricing decisions. Our approach adapts recent *constrained* OCO algorithms, cf. [22], [23], which are particularly robust and practical.

Contributions. In detail, we consider a hybrid 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, where the latter are made available to the SP only after the bidding is decided. Hence,

in effect, the provider has to reserve slices without knowing the needs of its users or the total cost of its reservation. 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 an average budget constraint.

We formulate this process as a learning problem which allows the SP to implement a *no-regret slice reservation policy*. This means that, as time evolves, the SP is guaranteed to achieve the same performance with 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 algorithm relies on a primal-dual online iteration [22] which minimizes a Lagrangian function, and controls concurrently for performance and budget costs. We extend our framework to allow the SP determine the *composition* of its slices (as opposed only to its size), without even knowing what resource combination is performance-optimal. This is crucial when the qualitative features of user demand are volatile and/or unknown. And finally, we propose a mixed time-scale learning policy that exploits any price information that is revealed by the NO during each period, so as to improve the SP reservations.

Our contributions can be thus summarized as follows:

• We consider a general model, where the SP reserves slices in two time scales; determines the slice size and/or composition; and is oblivious to the user needs and NO prices.

• We design an online learning framework for slice reservations, that ensures sample-path asymptotically-optimal performance while respecting the SP budget constraints.

• We perform a battery of numerical tests using stationary and non-stationary parameter patterns. The results verify the robustness and efficacy of our learning-based algorithms.

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.

II. System Model and Problem Statement

Network & Market Model. We consider a mixed time-scale model with long periods that are further 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 sliced resources to service providers (SPs), and we denote with $\mathcal{H}$ the set of $d = |\mathcal{H}|$ different resources that comprise each slice. For instance $\mathcal{H}$ may include wireless spectrum, backhaul capacity, computing and storage resources ($d = 4$). We introduce the *bundling* vector $\boldsymbol{\theta} \in \mathbb{R}^d$ which determines the proportion 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 first the case that $\boldsymbol{\theta}$ is fixed, but we drop this assumption in Sec. IV.

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 that are both announced by the NO. Since $\boldsymbol{\theta}$ is given, these scalar prices suffice to model the slice cost. We also define the vector of spot prices for period t as $\boldsymbol{q}_t = \big(q_k, k = (t-1)K + 1, \ldots, tK\big)$. 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 that the NO can impose upper limits on the slice size each SP can lease. Such limitations arise naturally due to capacity constraints, or when the NO reserves resources for its needs.

Reservation Decisions. We focus on one SP and study its slice reservation policy. This consists of the t-period reservation of $x_t \in \mathbb{R}_+$ slice units and the per-slot reservations $y_k \in \mathbb{R}_+$ within t. Note that the actual reserved resources are $x_t\boldsymbol{\theta}$ and $y_k\boldsymbol{\theta}$ respectively, but we drop the bundling vector until Sec. IV. At the beginning of each period t the SP decides its t-period reservation plan $(x_t, \boldsymbol{y}_t)$, where $\boldsymbol{y}_t = \big(y_k, k = (t-1)K+1, \ldots, tK\big)$. The SP's goal is to maximize its service while not exceeding its monetary budget B. The service performance is quantified with a concave *utility* function increasing on the resources and modulated by parameter $a_k \geq 0$ that captures the users' needs in slot k. For example, a_k might represent the total user demand, their willingness to pay, and so on. We also define $\boldsymbol{a}_t = \big(a_k, k = (t-1)K + 1, \ldots, tK\big)$. The concavity of the utility models diminishing returns which arise naturally in these systems.[2]

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:

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

$$\text{s.t.} \quad \sum_{t=1}^{T} \big(x_t p_t + \boldsymbol{y}_t^\top \boldsymbol{q}_t\big) \leq BT, \tag{2}$$

$$y_k \in \Gamma, \quad \forall k = 1, \ldots, KT, \tag{3}$$

$$x_t \in \Gamma, \quad \forall t = 1, \ldots, T. \tag{4}$$

Objective (1) is the total utility that the SP achieves with its reservations, after a duration of T periods. Remark there that if the SP never reserves ($x_t = 0, \forall t, y_k = 0, \forall k$), its total utility equals to 0. Constraint (2) captures the total budget constraint of the SP; and (3), (4) confine the decision variables to a convex set that collects upper reservation bounds set by the NO, minimum slicing requirements set by the SP. For the

[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.

[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.

sequel, we define Γ as the segment $[0, D]$, where 0 and D correspond to the minimum and maximum reservation requests the SP can make. 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. This means the NO always has the resources available to satisfy such requests.

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

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

• *(Ch2)*: The spot $\{q_k\}$ and reservation prices $\{p_t\}$ 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 and a_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 is unknown; nor can be tackled with standard online optimization techniques as some parameters are revealed after the reservation decisions are made in each slot (*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. Therefore, we aim to solve the convex optimization problem in an online manner, for each t:

$$(\mathbb{P}_t): \min_{\{x_t, \boldsymbol{y}_t\}} f_t(x_t, \boldsymbol{y}_t) = -\sum_{k=(t-1)K+1}^{tK} a_k \log(x_t + y_k + 1), \quad (5)$$

$$\text{s.t.} \quad g_t(x_t, \boldsymbol{y}_t) = x_t p_t + \boldsymbol{y}_t^\top \boldsymbol{q}_t - B \leq 0, \quad (6)$$

$$\{x_t, \boldsymbol{y}_t\} \in \Gamma^{K+1}. \quad (7)$$

III. Online Reservation Policy

A. Benchmark Policy: What we try to learn

Learning how to reserve slices and solve optimally $(\mathbb{P})$ in a dynamic fashion is, unfortunately, impossible. Namely, it was proved in [24] that in the absence of further assumptions about the price and demand evolution, there is no learning algorithm that can (asymptotically) maximize the objective while satisfying constraint (2). In light of this result, we settle for a weaker benchmark for our learning algorithm.

If the SP knew, at the beginning of each period t, the price p_t, the demand $\boldsymbol{a}_t$ and the spot price vector $\boldsymbol{q}_t$, it could find the optimal t-period decision $(x_t^*, \boldsymbol{y}_t^*)$ by solving:

$$\min_{\{x_t, \boldsymbol{y}_t\} \in \Gamma^{K+1}} f_t(x_t, \boldsymbol{y}_t) \quad \text{s.t.} \quad g_t(x_t, \boldsymbol{y}_t) \leq 0. \quad (8)$$

Given that in practice this information is unavailable, our goal is to design an algorithm that finds the t-period reservation policies $(x_t, \boldsymbol{y}_t)$ in each period t, such that we achieve a good enough performance with respect to this benchmark (8). Formally, we define the *dynamic regret* and constraint *fit* metrics:

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

which quantify respectively how well our policy $\{x_t, \boldsymbol{y}_t\}$ fairs against $\{x_t^*, \boldsymbol{y}_t^*\}$ 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.,

$$\lim_{T \to \infty} \frac{R_T}{T} = 0, \lim_{T \to \infty} \frac{V_T}{T} = 0. \quad (9)$$

It is important to stress that this learning objective is more challenging than the respective *static* regret benchmark where we compare against policy $(x^*, \boldsymbol{y}^*)$ which is designed with hindsight but remains fixed across time, cf. [22], [23].

B. Online Learning for Reservations (OLR)

We proceed with the design of the online learning algorithm that implements the SP reservation policy. We collect all reservation variables in $\boldsymbol{z} = (x, \boldsymbol{y})$, and define the Lagrangian:

$$L_t(\boldsymbol{z}, \lambda) = f_t(\boldsymbol{z}) + \lambda g_t(\boldsymbol{z}) \quad (10)$$

where $\lambda \in \mathbb{R}_+$ is the Lagrange multiplier associated with the constraint (6). It is easy to see that, for all t, for all $\lambda \in \mathbb{R}_+$, 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, for all t, for all $\boldsymbol{z} \in \Gamma^{K+1}$, 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, in a Gauss-Siedel manner. Instead of the online Lagrangian in (10), we minimize/maximize the modified Lagrangian with a linearized objective and a Euclidean regularizer with non-negative parameter ν:

$$L_t(\boldsymbol{z}, \lambda) = \nabla f_t(\boldsymbol{z}_t)^\top (\boldsymbol{z} - \boldsymbol{z}_t) + \lambda g_t(\boldsymbol{z}) + \frac{\|\boldsymbol{z} - \boldsymbol{z}_t\|^2}{2\nu} \quad (11)$$

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. 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):

$$\boldsymbol{z}_t = \arg\min_{\boldsymbol{z} \in \mathcal{Z}} L_{t-1}(\boldsymbol{z}, \lambda_t), \quad (12)$$

where $\mathcal{Z} = \Gamma^{K+1}$. 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 (12). After $\boldsymbol{z}_t$ is fixed, the demand and prices are revealed (steps 4-5) and the SP measures the performance $f_t(\boldsymbol{z}_t)$ and cost $g_t(\boldsymbol{z}_t)$. At the end of the period, the SP has access to the current Lagrangian (11), and can update its dual variable by executing the dual gradient ascent with positive step-size μ:

$$\lambda_{t+1} = [\lambda_t + \mu \nabla_\lambda L_t(\boldsymbol{z}_t, \lambda)]_+ \quad (13)$$

These are the dual variables (or, shadow prices for (2)) that will assist the SP to select the new reservation bid. It is worth stressing that OLR also works if, for instance, the prices p_t are revealed after step 3; this will become clear in the sequel.

Algorithm OLR: Online Learning for Reservation

Initialize:

$\lambda_1 = 0, x_0 \in \Gamma, \boldsymbol{y}_0 \in \Gamma^K$, $\nu = \mu = T^{-1/3}$

for $t = 1, \ldots, T$ **do**

- Observe the t-period price p_t
- Decide $(x_t, \boldsymbol{y}_t)$ by solving (12)
- Observe $\boldsymbol{a}_t$ and calculate $f_t(x_t, \boldsymbol{y}_t)$
- Observe $\boldsymbol{q}_t$ and calculate $g_t(x_t, \boldsymbol{y}_t)$
- Decide λ_{t+1} by solving (13)

C. Performance Analysis

We start with the necessary model assumptions.

Assumption 1. *The NO prices are uniformly bounded and specifically it holds $p_t \in [0, p], \forall t$ and $q_k \in [0, q], \forall k$.*

Assumption 2. *The utility parameters are uniformly bounded and specifically it holds $a_k \in [0, a]$, $\forall k$.*

Assumption 3. *The set $\mathcal{Z} = \Gamma^{K+1}$ has bounded diameter E.*

For all t, for all $\boldsymbol{z}_t \in \mathcal{Z}$, we have

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

Using the assumptions and the definition of the gradient, we can further derive the following bound on the gradient:

$$\|\nabla f_t(\boldsymbol{z}_t)\| \leq \sqrt{a^2K^2 + Ka^2} = a\sqrt{K(K+1)}.$$

Also, the maximum distance between any two reservation policies $\boldsymbol{z}$ and $\boldsymbol{z}'$ is:

$$\|\boldsymbol{z} - \boldsymbol{z}'\| \leq E = D\sqrt{K+1}, \ \forall \boldsymbol{z}, \boldsymbol{z}' \in \mathcal{Z},$$

where D is simply the maximum reservation request the SP can submit. And we can accordingly bound the constraint:

$$|g_t(\boldsymbol{z}_t)| \leq \max\{D(p + Kq) - B, B\}, \ \forall t, \boldsymbol{z}_t \in \mathcal{Z}, \quad (14)$$

which follows from the definition of $g_t(\boldsymbol{z}_t)$ in (6).

For all t, we assume that Problem $(\mathbb{P}_t)$ naturally admits a Slater vector, i.e. there is a vector $\tilde{\boldsymbol{z}}$ such that:

$$g_t(\tilde{\boldsymbol{z}}) \leq -\epsilon, \ \forall t, \quad (15)$$

where $\epsilon > 0$. The Slater constraint qualification is sufficient to have strong duality. Therefore, if the minimizer $\boldsymbol{z}_t$ of the Lagrangian $\mathcal{L}_{t-1}(\boldsymbol{z}, \lambda_t^*)$ is primal feasible, it must be primal optimal (λ_t^* is the dual optimal). This ensures that $\boldsymbol{z}_t$ is an optimal point of $(\mathbb{P}_{t-1})$. In fact, we are interested in the largest value ϵ can obtain without violating the Slater condition (15).

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:

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

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

$$U_g = \max_t \max_{\boldsymbol{z} \in \mathcal{Z}} \|g_t(\boldsymbol{z}) - g_{t-1}(\boldsymbol{z})\|$$

We see from (6) that it holds $U_g \leq D(p + Kq)$. However, in practice we expect U_g to be much 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, U_g affects the ability of the SP to learn the optimal reservation policy, and the following assumption needs to hold.

Assumption 4. *The slack constant ϵ in the Slater condition (15) is such that it holds $\epsilon > U_g$.*

Under these assumptions, OLR performs as follows.

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

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

$$V_T \leq M + \frac{(2Ea\sqrt{K(K+1)}/\mu) + (E^2/2\nu\mu) + (M^2/2)}{\epsilon - U_g},$$

$$\textit{where}: \ M \triangleq \max\{D(p + Kq) - B, B\},$$

$$\tilde{\lambda} \triangleq \mu M + \frac{2Ea\sqrt{K(K+1)} + (E^2/2\nu) + (\mu M^2/2)}{\epsilon - U_g}$$

Proof. The expressions for V_T and R_T follow from Theorems 1 and 2 of [22], respectively, and the contribution of the Lemma is to quantify the different parameters and the upper bound of the dual variables, which is a key step in the analysis.

First, note that $\|\nabla f_t(x_t, y_k)\| \leq a\sqrt{K(K+1)}$, for all t and $x_t, y_k \in \Gamma$. Moreover, recall that we showed in (14) that the upper bound of the constraint is $|g_t(\boldsymbol{z})| \leq \max\{D(p + Kq) - B, B\}$. Replacing these quantities in [22] we obtain the respective bounds of Lemma 1.

Now, let us focus on $\tilde{\lambda}$. We first define $\Delta(\lambda_t) := (\lambda_{t+1}^2 - \lambda_t^2)/2$, and use [22, Lemma 1] to upper bounded as follows:

$$\Delta(\lambda_{t+1}) \leq \mu\lambda_{t+1}(U_g - \epsilon) + \mu\left(2a\sqrt{K(K+1)}E + \frac{E^2}{2\nu} + \frac{\mu M^2}{2}\right) \quad (16)$$

Next, we proceed to prove by contradiction the upper bound on λ_t. Let us assume that $t+2$ is the first period for which the dual upper bound $\tilde{\lambda}$ does not hold. Therefore,

$$\lambda_{t+1} \leq \mu M + \frac{2a\sqrt{K(K+1)}E + E^2/(2\nu) + (\mu M^2)/2}{\epsilon - U_g},$$

$$\text{and } \lambda_{t+2} > \mu M + \frac{2a\sqrt{K(K+1)}E + E^2/(2\nu) + (\mu M^2)/2}{\epsilon - U_g}.$$

Working on λ_{t+1}, we get:

$$\begin{aligned}|\lambda_{t+1}| &= |\lambda_{t+2} - (\lambda_{t+2} - \lambda_{t+1})| \\ &\geq |\lambda_{t+2}| - |\lambda_{t+2} - \lambda_{t+1}| \\ &= |\lambda_{t+2}| - |[\lambda_{t+1} + \mu g_{t+1}(\boldsymbol{z}_{t+1})]^+ - \lambda_{t+1}| \\ &= |\lambda_{t+2}| - |\mu g_{t+1}(\boldsymbol{z}_{t+1})| \\ &\geq |\lambda_{t+2}| - \mu M \\ &> \frac{2a\sqrt{K(K+1)}E + E^2/(2\nu) + (\mu M^2)/2}{\epsilon - U_g}\end{aligned} \tag{17}$$

If we multiply both sides with $\mu(U_g - \epsilon)$, which is strictly negative due to Assumption 4, (17) is equivalent to:

$$\mu(U_g - \epsilon)\lambda_{t+1} < -\mu\left(2a\sqrt{K(K+1)}E + \frac{E^2}{2\nu} + \frac{\mu M^2}{2}\right)$$

Passing all the terms on the left side, we deduce:

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

which yields $|\lambda_{t+2}| < |\lambda_{t+1}|$ and that contradicts our assumption. As we set $\lambda_1 = 0$, then $|\lambda_2| \leq \mu M$. Thus, for every $t \geq 1$, $|\lambda_t| \leq \tilde{\lambda}$ holds. □

Discussion. Let us now discuss the implications of this result. First, note that a key ingredient of Algorithm OLR are the step sizes (or, *learning rates*) ν and μ. From the Corollary 1 of [22], if:

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

then:

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

. To expect *zero-regret*, we need $R_T = o(T)$. We can easily show that if $\max\{U_z^T, U_g^T\} = o(T)$, we have $R_T = o(T)$.

Practically we do not have access to U_z^T and U_g^T, as this would signify that we know the optimal decisions and the prices beforehand. We can select $\nu = \mu = T^{-1/3}$ to obtain [22]:

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

Another interesting observation is that both the regret and constraint violation depend on the variability of NO's pricing strategy. If the operator changes abruptly the prices among successive periods, namely in a superlinear fashion, $U_g^T = O(T^\beta)$ with $\beta \geq 1$, then it is impossible for the SP to learn an optimal reservation strategy. The same holds for the needs of SP. If, for instance, parameters $\{\boldsymbol{a}_t\}$ change so drastically that U_z^T grows superlinearly, then R_T/T will not diminish. Observe also that the period length (number of slots K) affects directly the 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. Finally, we can see that the relation of maximum prices, p and q, to the available per-period budget B, affect through parameter M the bounds. These observations reveal the key factors that shape the learning capability of the SP, and pave the road for possible regulatory interventions, or, if you prefer, bilaterally agreed guidelines among the SP and NO so as to increase the efficiency of the market.

IV. Model & Algorithm Extensions

We consider two practical extensions which demonstrate the modeling power of our framework: *(i)* when the SP decides the *slice composition* where the benefit from each resource is time-varying; and *(ii)* mixed-time scale bidding where the SP updates its per-slot reservations *during each period* as it acquires information for the demand and spot prices.

Slice Orchestration (SO). The SP makes multi-dimensional reservations using $\boldsymbol{x}_t, \boldsymbol{y}_k \in \mathbb{R}^d$, where d is the number of resources comprising the slice. The benefit from each reservation $\boldsymbol{x}_t$ is quantified by the scalar $\boldsymbol{\theta}_t^\top \boldsymbol{x}_t$ ($\boldsymbol{\theta}_t^\top \boldsymbol{y}_k$, respectively), where the elements of $\boldsymbol{\theta}_t \in \mathbb{R}^d$ 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. 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. Our analysis can be extended to handle this richer scenario. Namely, we need to update the objective and cost functions defined in (5), (6) by:

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

$$g_t(\boldsymbol{x}_t, \{\boldsymbol{y}_k\}_{k=(t-1)K+1}^{tK}) = \boldsymbol{x}_t^\top \boldsymbol{p}_t + \sum_{k=(t-1)K+1}^{tK} \boldsymbol{y}_k^\top \boldsymbol{q}_k - B. \tag{19}$$

Then, Algorithm OLR needs to be slightly modified by replacing the primal update (step 3) with a similar update that finds the multidimensional reservations $\boldsymbol{x}_t, \boldsymbol{y}_k \in \mathbb{R}^d$; change the step 4 so as to observe both the $\boldsymbol{a}_t$ and the $\boldsymbol{\theta}_t$ vectors; and perform the dual update (13) using $g_t(\boldsymbol{x}_t, \{\boldsymbol{y}_k\})$.

Mixed-time-scale reservation (MTS). Our second extension is a mixed time scale 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. The functions f_k and g_k for this slot-decision instance, are:

$$f_k(y_k) = -a_k \log(x_t + y_k + 1), \quad g_k(y_k) = y_k q_k - B_{slot},$$

where note that x_t is a parameter here, as it has been fixed in the beginning of t, and we have defined $B_{slot} = (B - x_t p_t)/K$. Finally, the per-slot Lagrangian is:

$$\begin{aligned}L_k(y, \hat{\lambda}) = \nabla f_k(y_k)(y - y_k) + \hat{\lambda} g_k(y) \\ + \frac{(y - y_k)^2}{2\hat{\nu}}\end{aligned} \tag{20}$$

where $\hat{\lambda} \in \mathbb{R}_+$ is the new dual variable. Then, the SP updates its reservation y_k for each slot k, and its dual variable after observing a_k and q_k, by executing:

$$y_k = \arg\min_{y \in \Gamma} L_{k-1}(y, \hat{\lambda}_k), \quad \hat{\lambda}_{k+1} = [\hat{\lambda_k} + \hat{\mu}\nabla L_k(y_k, \hat{\lambda})]_+.$$

$\hat{\nu}$ and $\hat{\mu}$ are the steps for the intra-period decisions. We set in the next section $\hat{\nu} = \nu$ and $\hat{\mu} = \mu$. Algorithm OLR can be

amended with these updates (for all slot k) after step 3. We do not provide theoretical guarantees but we do verify next the performance gains of this refined approach.

V. Numerical Results

Simulation setup. We evaluate the performance of Algorithm OLR and its extensions in different scenarios. We consider a slicing market where the NO manages B cellular base stations connected through a backhaul network of 100 paths to $N=20$ core nodes with data processing and storage capabilities. Hence, the SP reserves slices with wireless and backhaul bandwidth, storage and CPU capacity, i.e., $d=4$. The maximum slice size of D units is determined by the most scarce of these resources. We study two scenarios based on the demand and price evolution:

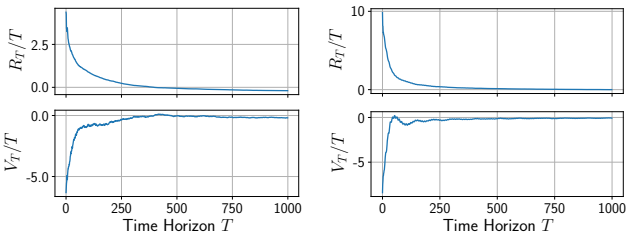
- *Case 1*: Parameters $\{a_k\}_k, \{p_t\}_t, \{q_k\}_k$ are random variables following a uniform distribution in the interval $[0,1]$.
- *Case 2*: Parameters $\{a_k\}_k, \{p_t\}_t, \{q_k\}_k$ are drawn from a non-stationary process. Namely, $a_k = \sin(2\pi k/K) + n_k$ with i.i.d. noise n_k uniformly distributed in $[1,2]$; $p_t = \sin(2\pi t/10) + \epsilon_t$, with ϵ_t uniformly distributed in $[1,2]$; $q_k = \sin(2\pi k/K) + \iota_k$, with ι_k uniformly distributed in $[1,2]$.

Performance Convergence. We start by evaluating the convergence of Algorithm OLR; see Fig. 1. We set the value of D so as to limit the reservations through the budget constraint.[3] We have observed that U_g^T and U_z^T gradually increase with D. High U_g^T and U_z^T can severely affect the performance and we are certain that $R_T = o(T)$ if $\max\{U_g^T, U_z^T\} = o(T)$. For *Case 1* and *Case 2*, we have selected $D=15$ and $D=5$ respectively to satisfy the latter condition on U_g^T and U_z^T. We observe that indeed the average regret R_T/T gradually diminishes to zero, and similarly the constraint violation V_T remains consistently below zero. Note that we do not project on $\mathbb{R}_+$ in order to provide a clearer view of the constraints.

Impact of parameter K. In Fig. 2, we compare Algorithm OLR with its mixed-time scale extension (OLR-MTS) for *Case 1* and *Case 2*, for different values of K, at a fixed time point. We observe a perceptible performance improvement for the MTS version, which is rather expected since the SP updates its decisions as new information becomes available. Indeed, the final R_T/T are significantly lower (especially in *Case 1*), and for the V_T/T we have better convergence. Moreover, we see that as K increases, so does the regret and violation, since the SP needs to tackle larger amounts of unknown information.

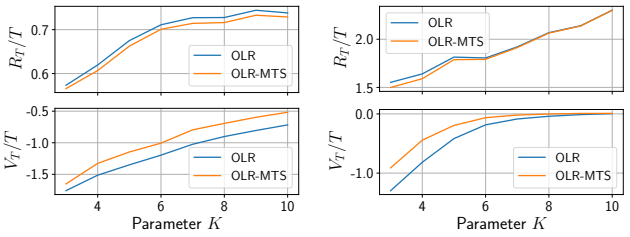
Evaluation of Slice Orchestration (SO). Next, we focus on the OLR-SO extension where the provider decides the slice composition. We model each of the d resource prices, i.e., the components of vectors $\boldsymbol{p}_t$ and $\boldsymbol{q}_k$, independently and according to the assumptions in *Case 1* and *Case 2*. The vector $\boldsymbol{\theta}_t$ is uniformly drawn from $[0,1]^3$ and updated at each t. In Fig. 3 we observe the convergence of the algorithm. Furthermore, we explore the sensitivity of OLR on K, in Fig. 4. We observe an increasing final regret against K. The final fit, which is negative, approaches 0 with rising K in *Case 1*. This means that the final budget savings are lower for longer periods.

[3]Note that if D limits the reservation, then the best strategy simplifies to constantly over-provision the slice ($\boldsymbol{z} = [D, D, ..., D]$).



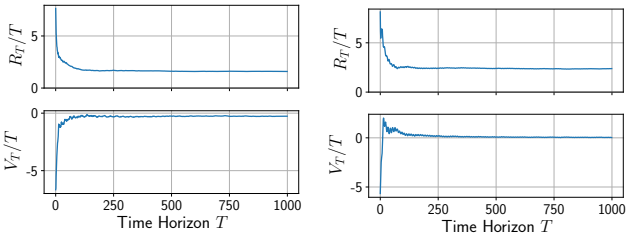
(a) Case 1 (b) Case 2

Fig. 1: *Evolution of* R_T/T *and* V_T/T. Simulation parameters are set to $T=1000$, $K=5$, $B=10$. (1a): $D=15$, $\nu=0.8, \mu=0.08$. (1b): $D=5$, $\nu=0.1, \mu=0.01$



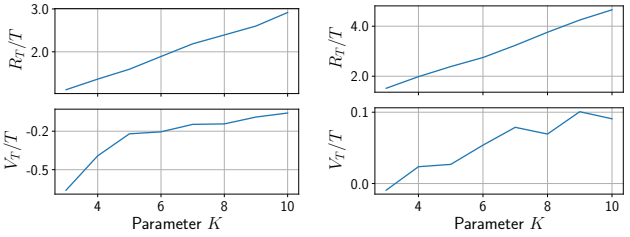
(a) Case 1 (b) Case 2

Fig. 2: *Sensitivity of* R_T/T *and* V_T/T *on* K. We plot the values of R_T/T and V_T/T at $T=1000$ while K changes. (2a): $B=10$, $D=30$, $\nu=0.12, \mu=0.04$. (2b): $B=10$, $D=20$, $\nu=0.6, \mu=0.06$.



(a) Case 1 (b) Case 2

Fig. 3: *Evolution of* R_T/T *and* V_T/T *for OLR-SO*. We use $T=1000$, $K=5$, $B=10$ and $d=3$. (3a): $\boldsymbol{d}=[20,20,20]$, $\nu=0.9$ and $\mu=0.03$. (3b): $\boldsymbol{d}=[5,5,5]$, $\nu=0.1$ and $\mu=0.01$.



(a) Case 1 (b) Case 2

Fig. 4: *Sensitivity of OLR-SO on* K. We used $T=1000$, $B=10$. (4a): $\boldsymbol{d}=[20,20,20]$, $\nu=0.9$ and $\mu=0.03$. (4b): $\boldsymbol{d}=[5,5,5]$, $\nu=0.1$ and $\mu=0.01$.

In *Case 2*, asymptotic violations are very small and tend to increase with rising K.

VI. Conclusions

The potential of network slicing markets can be only unleashed if service providers are able to reserve resources effectively. To that end, we proposed a set of slice reservation

policies, based on the theory of online convex optimization, which enable the SP to learn how to reserve resources optimally. Our policies are robust to arbitrary changes of the resource prices, oblivious to lack of this information when the reservations are made, and can achieve optimal slice orchestration even when the SP needs are unknown and time-varying. These key elements build a practical and general slicing framework with performance and budget guarantees.

VII. Acknowledgments

This work was supported by the European Commission through Grant No. 856709 (5Growth) and Grant No. 101017109 (DAEMON); and by SFI through Grant No. SFI 17/CDA/4760 and Grant No. 17/NSFC/5224.

References

[1] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network Slicing in 5G: Survey and Challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 94–100, 2017.

[2] K. Samdanis, X. Costa-Perez, and V. Sciancalepore, "From Network Sharing to Multi-tenancy: The 5G Network Slice Broker," *IEEE Communications Magazine*, vol. 54, pp. 32–39, 2016.

[3] S. Vassilaras, L. Gkatzikis, N. Liakopoulos, I. N. Stiakogiannakis, M. Qi, L. Shi, L. Liu, M. Debbah, and G. S. Paschos, "The Algorithmic Aspects of Network Slicing," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 112–119, 2017.

[4] U. Habiba, and E. Hossain, "Auction Mechanisms for Virtualization in 5G Cellular Networks: Basics, Trends, and Open Challenges," *IEEE Communication Surveys and Tutorials*, vol. 20, 2018.

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

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

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

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

[9] M. Jiang, M. Condoluci, and T. Mahmoodi, "Network slicing in 5G: an Auction-Based Model," in *Proc. of IEEE ICC*, 2017.

[10] M. Leconte, G. S. Paschos, P. Mertikopoulos, and U. C. Kozat, "A Resource Allocation Framework for Network Slicing," in *Proc. of IEEE INFOCOM*, 2018, pp. 2177–2185.

[11] J. Martín-Peréz, F. Malandrino, C. F. Chiasserini, and C. J. Bernardos, "OKpi: All-KPI Network Slicing Through Efficient Resource Allocation," in *Proc. of IEEE INFOCOM*, 2020, pp. 804–813.

[12] 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. of ACM CoNEXT*, 2018, pp. 353–365.

[13] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, "AZTEC: Anticipatory Capacity Allocation for Zero-Touch Network Slicing," in *Proc. of IEEE INFOCOM*, 2020, pp. 794–803.

[14] N. Van Huynh, D. Thai Hoang, D. N. Nguyen, and E. Dutkiewicz, "Optimal and Fast Real-Time Resource Slicing With Deep Dueling Neural Networks," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1455–1470, 2019.

[15] Y. Zhang, S. Bi, and Y. J. Angela Zhang, "Joint Spectrum Reservation and On-demand Request for Mobile Virtual Network Operators," *IEEE Trans. on Communications*, vol. 66, 2018.

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

[17] 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 *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2020, pp. 1–6.

[18] 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 Transactions on Network and Service Management*, vol. 16, no. 2, pp. 591–605, 2019.

[19] L. Zheng, C. Joe-Wong, C. W. Tang, M. Chiang, and X. Wang, "How to Bid the Cloud," in *Proc. of ACM SIGCOMM*, 2015.

[20] M. Khodak, L. Zheng, A. S. Lan, C. Joe-Wong, and M. Chiang, "Learning Cloud Dynamics to Optimize Spot Instance Bidding Strategies," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 2018, pp. 2762–2770.

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

[22] T. Chen, Q. Ling, and G. B. Giannakis, "An Online Convex Optimization Approach to Proactive Network Resource Allocation," *IEEE Transactions on Signal Processing*, vol. 65, no. 24, pp. 6350–6364, 2017.

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

[24] S. Mannor, J. N. Tsitsiklis, and J. Y. Yu, "Online learning with sample path constraints." *Journal of Machine Learning Research*, vol. 10, pp. 569–590, 2009.