Learning Resource Reservation with Predictions

***Abstract*—The virtualization of wireless networks enables new services to access network resources made available by the Network Operator (NO) through a Network Slicing market. The different service providers (SPs) have the opportunity to lease the network resources from the NO to constitute slices that address the demand of their specific network service. The goal of any SP is to maximize its service utility and minimize costs from leasing resources while facing uncertainties of the prices of the resources and the users' demand. In this paper, we propose a smart solution that allows the SP to decide its online reservation policy, which aims to maximize its service utility and minimize its cost of reservation simultaneously. We design the OOLR solution, a decision algorithm built upon the Follow-the-Regularized Leader (FTRL), that incorporates key predictions to assist the decision-making process. Our solution achieves a $\mathcal{O}(\sqrt{L_T T})$ regret bound where L_T represents the accumulated quadratic error of the predictions and T the horizon. We integrate a prediction model into the OOLR solution which leads to a $\mathcal{O}(T^{3/4})$ regret guarantee and we demonstrate through numerical results the efficacy of the combined models' solution.**

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

I. Introduction

A. Motivation

The virtualization of wireless networks has gained significant interest in recent studies, cf. [1], [2]. This new technology enables the development of the Network Slicing framework, where service providers (SPs) can lease virtualized network resources from the Network Operator (NO) to address the demand of their specific network service [3], [4]. Network Slicing promises to boost the utilization efficiency of the network resources by accommodating multiple and diverse SPs on the NO's infrastructure. This in turn brings new challenges: on the one hand the NO must accommodate heterogeneous slices on its network to satisfy diverse requirements of the SPs; on the other hand the SPs must request network resources or slice requirements in a smart and proactive way by anticipating their future demand.

The players are expected to operate in a real-time market, where the SPs can lease both computing and storage resources while the NO offers both in-advance reservation and on-the-fly spot opportunities. The modeling of such slicing market draws ideas from cloud marketplaces [5]–[8], where the Cloud Provider allows customers to bid for resources in the on-demand and spot markets [9], [10]. This market will allow the NO to proactively schedule the slice configuration based on the information coming from the in-advance reservation requests, but also offer the available spot resources dynamically, leading to slice re-configuration and boosting network utilization.

In this context, the SP must request the needed network resources while facing uncertainty about its own demand, and without knowing the pricing scheme of such resources managed by the NO [11]. We expect the NO reveals the prices along with the SP demand after the SP request. Therefore, the SP must decide its requests dynamically without the crucial information of the current and future SP demand and resource pricing. Additionally, we expect those to vary according to non-stationary patterns, as they might depend on multiple underlying factors, such as the other SPs requests, the NO internal needs, etc. We highlight here the necessity for the SP to build a decision model robust to uncertainty, while being able to use the NO's feedback about historical prices and demand.

The focus of this paper is to tackle exactly the problem faced by the SP. To this end, we propose to design a smart solution that allows the SP to decide its online reservation policy, which aims to maximize its service utility and minimize its cost of reservation simultaneously. We develop an online optimization tool able to incorporate predictions which will assist the reservation decision-making process of the SP. Our solution provides guarantees of performance even for arbitrarily bad predictions which may arise in volatile settings. We complement our decision model with an accurate, robust, and low-complexity prediction model. The combined solution achieves a $\mathcal{O}(T^{3/4})$ regret.

B. Methodology and Contributions

In the line of our previous works [12], [13], we consider a hybrid slicing market where the SP can reserve different types of resources, which will compose the reserved slice. The NO pricing scheme is unknown to the SP, which means the prices of the resources are revealed to the SP only after the SP makes the bid. Hence, the SP has to reserve the network resources without knowing the demand from its users and the prices of the network resources. The SP can leverage the feedback received from the NO to design accurate predictions that will assist the online reservation policy, which end goal is to maximize the slice performance and minimize the overall cost of reservation.

The SP receives feedback from the NO about its demand and the resource pricing after its reservation being made. As time passes, the SP accumulates historical data of those varying and potentially non-stationary signals. The latter information paves the way to the design of a prediction algorithm able to feed accurate predictions of the different signals into the reservation-based decision model.

We model the reservation problem faced by the SP as a learning problem, where the goal is to design a *no-regret online reservation policy*. We build our solution upon the standard OCO problem introduced in the seminal paper of Zinkevich [14], which aims to find the online policy $\{\boldsymbol{z}_t\}_{t=1}^{T}$

that minimizes the regret with respect to the best static solution:

$$R(T) = \sum_{t=1}^{T} f_t(\boldsymbol{z}_t) - \arg\min_{\boldsymbol{z}\in\mathcal{Z}} \sum_{t=1}^{T} f_t(\boldsymbol{z}). \quad (1)$$

At each slot t, the learner (the SP in our case) must decide $\boldsymbol{z}_t$ (the reservation vector) from a convex set $\mathcal{Z}$ without knowing the convex loss f_t. We say the online policy $\{\boldsymbol{z}_t\}_{t=1}^{T}$ has *no-regret* if the achieved regret is sublinear, i.e. $R(T) = o(T)$, in other words $\lim_{T\to\infty} R(T)/T = 0$.

We build our online learning solution (OOLR) upon the Follow-the-Regularized-Leader (FoReL) algorithm [15]. We develop an *optimistic* version of the FoRel, first introduced by Mohri and Yang [16], where the decision relies on an adaptive proximal regularizer term and the optimistic term of the next gradient prediction $\nabla\tilde{f}_{t+1}(\tilde{z}_{t+1})$. We opt for an accurate, robust and computationally low prediction model which we find in the work of Anava et al [17]: the ARMA-OGD algorithm. The latter generates the predictions through an AR(q) process, where the q lag coefficients are updated thanks to the online gradient descent (OGD) method, which has low time complexity. It also provides regret guarantees against the best ARMA predictor with full hindsight of the future $\{f_i\}_{i=t+1}^{T}$. The two combined models provide the SP with a unique solution which is competitive and presents strong guarantees.

We draw up the different contributions of this paper:

- we introduce a hybrid reservation model where a service provider can lease multiple kinds of resources from a network operator in an online manner;
- we formulate an optimization problem for the SP where it aims to maximize the leased slice utility and minimize the reservation cost in the long-term. To solve the reservation problem faced by the SP, we develop an online learning solution (OOLR) which incorporates the *optimistic* prediction of the next slot gradient;
- we provide regret bound guarantees of $\mathcal{O}(\sqrt{L_T T})$ for arbitrarily bad predictions and $\mathcal{O}(1)$ for perfect predictions;
- we implement a prediction module to assist our OOLR decision algorithm and we provide a theoretical regret bound for the combined solution of $\mathcal{O}(T^{3/4})$.
- we extend our model to the situation the NO only fulfills part of the SP reservation request due to capacity constraints.

II. Literature review

A. Adaptive Online Learning

In [18], [16],

B. Reservation of virtualized resources

In [19], the authors leveraged network traffic information to plan the capacity needed for each slice in a multi-tenant framework. Using a data-driven approach including C-RAN, MEC and core networks, their solution outperforms other state-of-the-art deep learning solutions [20], [21]. The paper [22] employed an adaptive forecasting model of the elastic demand for network resources to perform slice allocation in Internet Access Services. [23] 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 [24] 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, [25] studied the impact of slice overbooking; [26] employed predictive capacity allocation for improving the slice composition; and [27] focused on dynamic slicing via reinforcement learning.

Fewer works have considered the SP point-of-view. The paper [28] developed a two-time scale approach for the activation and the re-configuration of the slices while considering the reservation of both RAN and backhaul resources. [29] focuses on wireless spectrum considering two reservation schemes (in advance and on demand). In [30], the authors develop a two-stage approach for the resource reservation and the intra-slice resource allocation. These works presume a stationary environment where user statistics do not change and/or cost of resources are supposed constant. The paper [31] considers as well a mixed-time scale reservation model of long-term and short-term VM instances then used for VNF deployment. The authors propose three solutions for the VNF demand prediction, the short-term and long-term reservations of VMs. In [32], 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.

III. Model and Problem Statement

A. Network and Market Model

We consider a slotted system $\{1, \ldots, T\}$. A Network Operator (NO) sells virtualized resources to the service provider (SP), 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 radio resources, backhaul link capacity, computing resources, storage resources ($m = 4$). The SP can request a certain amount for each type of resource, which are then composing the end-to-end network slice dedicated for the users of the service. The SP must decide the reservation vector $\boldsymbol{x} = [x_1, \ldots, x_m]^\top$. Based on its reservation $\boldsymbol{x}$, the slice obtained by the SP will have a given capacity, which we assume to be a linear combination of the leased resources which contributions to the capacity are transcribed in the vector $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_m]^\top$. Therefore, the SP obtains a slice of capacity $\boldsymbol{x}^\top\boldsymbol{\theta}$. This assumption is general and can be found in related works such as [33] which considers a multi-resource allocation framework. Moreover, we make no assumptions about the contributions of the resources which we deem unknown and changing from slot to slot, potentially in a non-stationary manner. Therefore, at slot t the slice capacity is

equal to $\boldsymbol{x}_t^\top \boldsymbol{\theta}_t$, where $\boldsymbol{x}_t$ is the reservation vector for slot t, and $\boldsymbol{\theta}_t$ is the contribution vector.

Following the standard practice, we model the slice utility of the SP as an increasing concave function of the acquired slice capacity by using the logarithm function. For instance, the paper [23] provides the general form of α-fair utility functions:

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

where $\boldsymbol{z}$ is the reservation vector of slices. In [34], 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. Other reservation-based papers relate to the same kind of convex/concave objective function to minimize/maximize [24], [29], [32]. The logarithm function allows us to model as well the diminishing returns which naturally arise with the over-reservation of the network resources. For instance, the data rate is a logarithmic function of the spectrum; the additional revenue of the SP from more slice resources is typically diminishing.

The market operates in a hybrid model. At the beginning of each slot, the SP can lease network resources, plus additional resources on a spot market. We denote with $\boldsymbol{p}_t = [p_1, \ldots, p_m]_t^\top \in \mathbb{R}_+^m$ the unit price of the network resources; and we denote with $\boldsymbol{q}_t = [q_1, \ldots, q_m]_t^\top \in \mathbb{R}_+^m$ the unit price of the resources available in the spot market. At the time the SP makes a bid, both reservation and spot unit prices are unknown and are only revealed *after* the SP makes its decision. Moreover, the prices dynamically change following non-stationary patterns, which force the SP to use a reservation decision model robust to the price uncertainty. As prices are revealed along time to the SP, the latter acquires historical traces which render possible the prediction of the next slot values. We will detail how to use this information within our decision algorithm.

The NO can impose upper limits on the requests of the SP. For instance, the reservation request for resource i must belong to the set $\Gamma_i = [0, D_i]$. Therefore, the SP request will belong to $\Gamma_1 \times \ldots \times \Gamma_m$, which we denote Δ. Such limitations arise from natural capacity constraints of the network, in charge of multiple services and its own needs. In some cases, the NO can be unable to fulfill the SP request, especially when the network is congested due to high users' demand load and heavy SPs requests. The NO must guarantee a certain level of SLA, which we relate to the respect a certain threshold ratio of the requested amount resource. For instance, the NO must deliver at least $\alpha = 80\%$ of the desired capacity for the resource. We envision this scenario in the sequel, with a slight change to the main problem statement.

We now elucidate the SP reservation policy, which consists of the t-slot reservation decision $\boldsymbol{x}_t$ and the spot decision $\boldsymbol{y}_t$. At the beginning of each slot t, the SP decides its t-slot reservation plan $(\boldsymbol{x}_t, \boldsymbol{y}_t)$, and pays the price $\boldsymbol{p}_t^\top \boldsymbol{x}_t + \boldsymbol{q}_t^\top \boldsymbol{y}_t$ at the end of the slot. The SP's goal is to maximize the performance of its service, while avoiding excessive monetary costs. We model the slice utility function as a logarithmic concave increasing function on the slice capacity, weighted by the SP demand a_t, i.e. $a_t \log(1+\boldsymbol{\theta}_t^\top(\boldsymbol{x}_t+\boldsymbol{y}_t))$. The SP demand a_t is unknown at the beginning of slot t and we assume the SP accesses it at the end of the slot.

B. Problem statement

Putting the above together, the ideal reservation slice policy is the solution of the following convex program:

$$\begin{aligned} (\mathbb{P}): \quad \max_{\{\boldsymbol{x}_t, \{\boldsymbol{y}_t\}\}_{t=1}^T} & \sum_{t=1}^{T} \Big(V a_t \log((\boldsymbol{x}_t + \boldsymbol{y}_t)^\top \boldsymbol{\theta}_t + 1) \\ & - (\boldsymbol{p}_t^\top \boldsymbol{x}_t + \boldsymbol{q}_t^\top \boldsymbol{y}_t) \Big) && (3) \\ \text{s.t.} \quad & \boldsymbol{y}_t \in \Delta, \quad \forall t = 1, \ldots, T, && (4) \\ & \boldsymbol{x}_t \in \Delta, \quad \forall t = 1, \ldots, T. && (5) \\ & && (6) \end{aligned}$$

In Objective (22), we recognize the weighted sum of the slice performance (logarithmic term) and the payments (linear term). The latter term has a minus sign as the SP seeks to minimize its monetary cost. We sum over the number of slots T, as the goal is to maximize this weighted sum in the long-term. Constraints (23) and (24) ensure the decisions belong to the constraint convex set Δ. We define the hyper-parameter $V \geq 1$ which balance the influence between the two terms (utility term and cost term). The bigger V, the more we favor the slice utility in the detriment of the cost of reservation.

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

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

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

$$\begin{aligned} f_t(\boldsymbol{x}_t, \boldsymbol{y}_t) = & -V a_t \log((\boldsymbol{x}_t + \boldsymbol{y}_t)^\top \boldsymbol{\theta}_t + 1) && (7) \\ & + (\boldsymbol{p}_t^\top \boldsymbol{x}_t + \boldsymbol{q}_t^\top \boldsymbol{y}_t) && (8) \end{aligned}$$

The function f_t presents the key advantage to be convex which allows us to use the OCO framework. Our goal is to design the online reservation policy $\{\boldsymbol{z}_t\}_{t=1}^T$, where we denote $\boldsymbol{z}_t = (\boldsymbol{x}_t, \boldsymbol{y}_t) \quad \forall t$, able to give sublinear regret guarantees against an optimal policy. We define the latter in the next subsection.

C. Static Benchmark

The efficacy of our learning policy is mainly characterized by the benchmark to which we compare it (i.e. there exists different kind of benchmark, which are more or less stringent); and by the convergence rate of the learning policy loss relative to the chosen benchmark.

In our problem, we opt for a static benchmark, that consists of a unique reservation vector $\boldsymbol{z}^*$, optimally chosen over the period of evaluation, to minimize the loss.

More precisely, the optimal solution $\boldsymbol{z}^*$ is defined as:

$$\boldsymbol{z}^* = \arg \min_{\{\boldsymbol{x},\boldsymbol{y}\in\Delta\}} \sum_{t=1}^{T} f_t(\boldsymbol{x}, \boldsymbol{y}), \tag{9}$$

where the period of evaluation includes the slots $t = 1, \ldots, T$.

The benchmark has the pros to know *a priori* all the future prices $\{\boldsymbol{p}_t\}_{t=1}^T$, $\{\boldsymbol{q}_t\}_{t=1}^T$, the future demand $\{a_t\}_{t=1}^T$ and the future contribution vectors $\{\boldsymbol{\theta}_t\}_{t=1}^T$. It has the cons to reserve a unique vector, which is ideal *on average* (over the period of evaluation), but *sub-optimal* when considering the slots separately. Therefore, our algorithm can possibly outperform the benchmark, synonym of a negative regret.

Given that in practice, the information of the benchmark is unavailable, our goal is to design an algorithm that finds the reservation vector $\boldsymbol{z}_t = (\boldsymbol{x}_t, \boldsymbol{y}_t)$ at each slot t, such that the overall loss achieved (for $t = 1, \ldots, T$) is of the same order of the overall loss achieved by $\boldsymbol{z}^*$. Formally, we define the *static regret* as:

$$R(T) = \sum_{t=1}^{T} \Big(f_t(\boldsymbol{z}_t) - f_t(\boldsymbol{z}^*) \Big), \tag{10}$$

and the goal is to have $R(T)/T \to 0$ when $T \to \infty$.

IV. OOLR Solution

A. Basic assumptions

A1. The sets Γ_i, $i = 1 \ldots m$, are convex and compact, and it holds $|x| \leq D_i$, for any $x \in \Gamma_i$[1].

A2. The function f_t is convex.

A3. $\{r_t\}_{t=1}^T$ is a sequence of proximal non-negative functions.

A4. Prediction $\nabla \hat{f}_{t+1}(\hat{\boldsymbol{z}}_{t+1})$ is known at t.

A5. The function $r_{1:t}(\boldsymbol{z})$ is 1-strongly convex with respect to some norm $||.||_{(t)}$, that we will define in the sequel.

B. Performance analysis

Our approach is inspired from the *Follow-the-Regularized-Leader* (FTRL) policy.

First, we define the proximal regularizers:

$$\forall t = 1 \ldots T, \quad r_t(\boldsymbol{z}) = \frac{\sigma_t}{2} ||\boldsymbol{z} - \boldsymbol{z}_t||^2, \tag{11}$$

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

$$\sigma_t = \sigma \Big(\sqrt{h_{1:t}} - \sqrt{h_{1:t-1}} \Big), \tag{12}$$

$$h_t = ||\nabla f_t(\boldsymbol{z}_t) - \nabla \hat{f}_t(\hat{\boldsymbol{z}}_t)||^2, \tag{13}$$

where $\sigma \geq 0$, and $h_{1:t} = \sum_{i=1}^t h_i$. $\nabla \hat{f}_t(\hat{\boldsymbol{z}}_t)$ is the gradient prediction that we obtain with our prediction model.

The basic step of our algorithm is:

$$\boldsymbol{z}_{t+1} = \arg \min_{\boldsymbol{z}\in\Delta^2} \Big\{ r_{1:t}(\boldsymbol{z}) + \Big(\sum_{s=1}^{t} \nabla f_s(\boldsymbol{z}_s) + \nabla \hat{f}_{t+1}(\hat{\boldsymbol{z}}_{t+1}) \Big) \top \boldsymbol{z} \Big\} \tag{14}$$

Algorithm OOLR: Optimistic Online Learning for Reservation

Initialize:
$\boldsymbol{z}_1 \in \Delta^2, \sigma = 1, a_1, \boldsymbol{q}_1, f_1(\boldsymbol{z}_1)$
for $t = 1, \ldots, T-1$ **do**
- Observe the new prediction of the gradient $\nabla \hat{f}_{t+1}(\hat{\boldsymbol{z}}_{t+1})$
- Decide $\boldsymbol{z}_{t+1}$ by solving (14)
- Observe the demand a_{t+1}, the reservation price $\boldsymbol{p}_{t+1}$, the spot price $\boldsymbol{q}_{t+1}$, the contributions $\boldsymbol{\theta}_{t+1}$
- Calculate $f_{t+1}(\boldsymbol{z}_{t+1})$ and $\nabla f_{t+1}(\boldsymbol{z}_{t+1})$
- Update $r_{1:t+1}(\boldsymbol{z})$ according to (11) and (12)

For simplicity, we denote:

$$\begin{cases} \nabla f_t(\boldsymbol{z}_t) & = c_t \\ \nabla \hat{f}_t(\hat{\boldsymbol{z}}_t) & = \tilde{c}_t \end{cases}$$

Here we conduct the performance analysis on the regret of Algorithm *OOLR*. The main result is the following. *Algorithm OOLR ensures the regret bound:*

$$\boxed{R(T) \leq \sqrt{\sum_{t=1}^{T} ||c_t - \tilde{c}_t||^2} \Big(\frac{2}{\sigma} + \frac{\sigma}{2} 2D^2 T \Big)} \tag{15}$$

Proof. First let's remark that the function $h_{0:t} : \boldsymbol{z} \to r_{0:t}(\boldsymbol{z}) + (c_{1:t} + \tilde{c}_{t+1})^\top \boldsymbol{z}$ is 1-strongly convex, with respect to the norm $||.||_{(t)}$. It allows us to use [16, Theorem 1], which yields regret:

$$R(T) \leq r_{1:T}(\boldsymbol{z}^*) + \sum_{t=1}^{T} ||c_t - \tilde{c}_t||^2_{(t),*} \quad \forall \boldsymbol{z}^* \in \Delta^2 \tag{16}$$

□

Now, we define the norm $||x||_{(t)} = \sqrt{\sigma_{1:t}} ||x||$, which has dual norm $||x||_{(t),*} = ||x|| / \sqrt{\sigma_{1:t}}$. We remark that $\sigma_{1:t} = \sigma \sqrt{h_{1:t}}$, and starting from (16), we get:

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

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

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

Thus,

$$R(T) \leq \frac{\sigma}{4} \sum_{t=1}^{T} \frac{h_t}{\sqrt{h_{1:t}}} ||\boldsymbol{z}^* - \boldsymbol{z}_t||^2 + \sum_{t=1}^{T} \frac{h_t}{\sigma \sqrt{h_{1:t}}} \tag{17}$$

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

From [35, Lemma 3.5], we have:

$$\sum_{t=1}^{T} \frac{h_t}{\sqrt{h_{1:t}}} \leq 2\sqrt{h_{1:t}} \tag{18}$$

Plugging this result into (17), it yields:

$$R(T) \leq \sqrt{h_{1:t}}(\frac{2}{\sigma} + \frac{\sigma}{2}\sum_{t=1}^{T} ||\boldsymbol{z}^* - \boldsymbol{z}_t||^2) \tag{19}$$

We finish the proof by finding the following bound:

$$\sum_{t=1}^{T} ||\boldsymbol{z}^* - \boldsymbol{z}_t||^2 \leq \sum_{t=1}^{T} 2(D_1^2 + \ldots + D_m^2) = 2D^2T$$

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

$$\sigma = \frac{\sqrt{2}}{D\sqrt{T}} \tag{20}$$

We re-write the upper bound:

$$\boxed{R(T) \leq 2D\sqrt{2T}\sqrt{\sum_{t=1}^{T} ||c_t - \tilde{c}_t||^2}} \tag{21}$$

Remark 2. The regret bound is in $\mathcal{O}(\sqrt{L_T T})$, and becomes *null* when the predictions are perfect, i.e. when $\forall t, \quad \tilde{c}_t = c_t$.

Remark 3. We implement an online learning prediction method [17] that learns how to predict the gradient with the regret $\mathcal{O}(2mGM\sqrt{T})$, where G and M are key constant in [17]. The integration of this algorithm into our main OOLR solution leads to the regret of $\mathcal{O}(T^{3/4})$. Other prediction methods could be applied to the prediction of the gradient; however, this online learning method offers sublinear regret guarantees against all types of traces, even non-stationary.

Remark 4. We cannot set the value of σ like we did in (20) while ignoring the horizon T. If we ignore the latter, which is possible in some settings, we must set the value of σ according to the doubling trick. This consists in dividing the horizon T in $\log_2(T)$ periods, then iterate over the periods $k = 0, 1, \ldots, log_2(T)$, and set for the slots of period k which are $t = 2^k, 2^k + 1, \ldots, 2^{k+1} - 1$ the value:

$$\sigma = \frac{\sqrt{2}}{D\sqrt{2^{k+1}}}.$$

This method will conserve the $\mathcal{O}(T^{3/4})$ regret bound, which will worsen only by a constant multiplicative factor. We refer the reader to the proof [36, Section 2.3.1].

V. Model Extension

A. Problem statement

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

We redefine the problem as:

$$\begin{aligned}
(\mathbb{P}): \quad \max_{\{\boldsymbol{x}_t, \{\boldsymbol{y}_t\}\}_{t=1}^{T}} & \sum_{t=1}^{T} \Big(Va_t \log((\tilde{\boldsymbol{x}}_t + \tilde{\boldsymbol{y}}_t)^\top \boldsymbol{\theta}_t + 1) \\
& - (\boldsymbol{p}_t^\top \tilde{\boldsymbol{x}}_t + \boldsymbol{q}_t^\top \tilde{\boldsymbol{y}}_t) \Big) && (22) \\
\text{s.t.} \quad & \boldsymbol{y}_t \in \Delta, \quad \forall t = 1, \ldots, T, && (23) \\
& \boldsymbol{x}_t \in \Delta, \quad \forall t = 1, \ldots, T. && (24) \\
& && (25)
\end{aligned}$$

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

B. Solution

VI. Numerical evaluation

A. Description of the traces

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

To model the users' demand for the service, we use a real-world data set that contains the aggregated traffic volumes seen across multiple BSs owned by a major MNO of Shanghai. Traffic volumes have been recorded over a one-month period, spanning from Friday 1 August 2014 00:00 to Sunday 31 August 2014 23:50, with each recording averaged over a period of 10 minutes. Hence, there are 6 measurements per hour and a total of 4464 measurements for each BS over this period.

We assume the network resources prices vary with potentially non-stationary dynamics. We model such variations thanks to an AR(1) process, the discrete-time equivalent of the Ornstein-Ulhenbeck process. The latter is a stochastic process with applications in financial mathematics [37]. We model the contribution parameters -items of vector $\boldsymbol{\theta}_t$- as varying and non-stationary. Each item follows a seasonal trend (modeled through a sine wave), with a specific offset and added OU stochastic process.

B. Prediction module

The solution OOLR is *optimistic* in the sense it allows the SP to use the predicted gradient term $\nabla \hat{f}_{t+1}(\hat{\boldsymbol{z}}_{t+1})$ of the next slot. In (15), we concluded that accurate predictions can greatly enhance the performance, as the regret bound goes from $\mathcal{O}(\sqrt{L_T T})$ when predictions are arbitrarily bad to $\mathcal{O}(1)$ when predictions are almost perfect. This observation paves the way to the introduction of a prediction module, in support of our OOLR decision algorithm. We aim to find an accurate, robust and computationally low model. The algorithm ARMA-OGD created by Anava et al. in [17] presents these three key advantages. It consists of learning the AR(q) signal where the q lag coefficients are updated online at each slot thanks to the gradient descent method. The algorithm guarantees that the total loss is no more on average than the loss of the best ARMA predictor with full hindsight.

First, we show in Fig. 1 that the model is accurate against two intricate signals. The SP demand is based on multiple latent factors, which makes the signal non-stationary and hard to predict. Yet, we observe the predicted signal is able to track the SP demand. The $2m$ gradient items are composed of multiple signals, namely the SP demand, the prices and contributions of the network resources, as the reader can observe in the following expression of the gradient:

$$\nabla f_t(\boldsymbol{z}_t) = \begin{bmatrix} -V\frac{a_t\theta_{t,1}}{1+(\boldsymbol{x}_t+\boldsymbol{y}_t)^\top\boldsymbol{\theta}_t} + p_{t,1} \\ -V\frac{a_t\theta_{t,2}}{1+(\boldsymbol{x}_t+\boldsymbol{y}_t)^\top\boldsymbol{\theta}_t} + p_{t,2} \\ \vdots \\ -V\frac{a_t\theta_{t,m}}{1+(\boldsymbol{x}_t+\boldsymbol{y}_t)^\top\boldsymbol{\theta}_t} + p_{t,m} \\ -V\frac{a_t\theta_{t,1}}{1+(\boldsymbol{x}_t+\boldsymbol{y}_t)^\top\boldsymbol{\theta}_t} + q_{t,1} \\ -V\frac{a_t\theta_{t,2}}{1+(\boldsymbol{x}_t+\boldsymbol{y}_t)^\top\boldsymbol{\theta}_t} + q_{t,2} \\ \vdots \\ -V\frac{a_t\theta_{t,m}}{1+(\boldsymbol{x}_t+\boldsymbol{y}_t)^\top\boldsymbol{\theta}_t} + q_{t,m} \end{bmatrix} \quad (26)$$

Again, the model is able to give an accurate predicted signal. Secondly, ARMA-OGD provides guarantees of performance against all types of traces, which ensures its robustness. The total squared loss of the model is a $\mathcal{O}(\sqrt{T}) + Res$, where Res represents the residual squared loss of the best ARMA predictor with full hindsight of the target signal. We show in Fig. 2 the convergence of the average squared loss towards Res. Finally, the ARMA-OGD is based on the OGD update step, which is very low computationally and allows us to develop the algorithm alongside the OOLR solution. We insist here that the two combined solutions having both low time complexity allow the SP to take *optimistic decisions in real time*. Obviously there exists other models which employ advanced techniques such as Neural Networks that would obtain better accuracy than the ARMA-OGD. Nevertheless, these models necessitate an offline training phase, do not provide guarantees of robustness, and have higher time complexity.

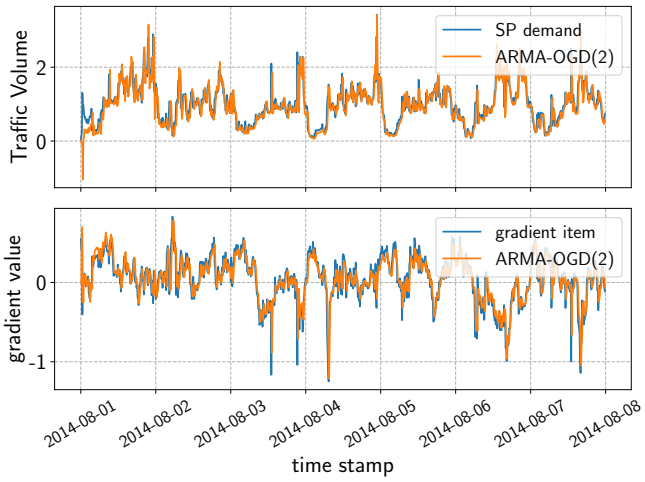

Fig. 1: The x-axis encompasses the first week of August period. *Upper part:* the predicted signal against the real-world MVNO demand signal. The y-axis values are normalized. *Lower part:* the predicted signal against the first of the gradient $2m$ items.

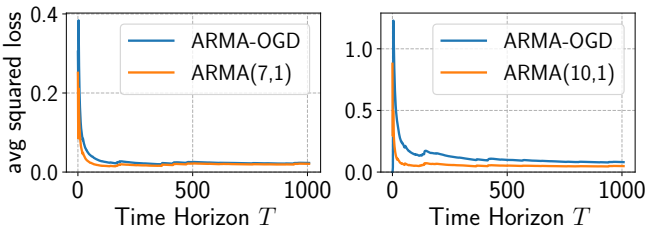

Fig. 2: We evaluate the accuracy of the models on the first week of August. *Left side:* We observe the convergence of the average squared loss of the predicted gradient first item towards the best ARMA in hindsight. *Right side:* We observe the convergence of the average squared loss of the predicted MVNO demand toward the best ARMA in hindsight.

C. Impact of the quality of predictions

We propose to evaluate the OOLR solution. The SP can reserve $m = 3$ kinds of resources. We assume the NO sets the upper bound constraint to $D_i = 1, \forall i$. This means the SP reserves normalized values for each type of resource. We recall from our system model that the slice capacity is a linear combination of those normalized values and the SP utility is a logarithmic function of the obtained slice capacity. Our goal is to maximize the SP utility while avoiding excessive reservation cost. We balance between the two terms (utility and cost) thanks to the hyper-parameter V. As one term is approximately $V * 2\log(1+X)$, the other term is X and X is around 1, we set $V = 2$ to have both terms of the same order.

We introduce the parameter ζ to control the quality of two prediction models, where ζ is the average relative error

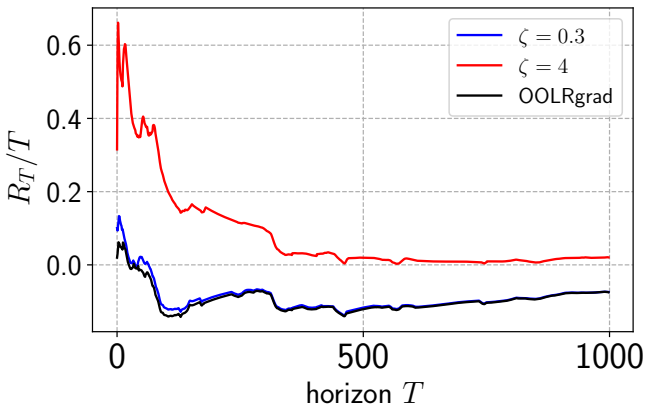

Fig. 3: *Evolution of* R_T/T*:* Horizon $T = 1008$, $m = 3$, $V = 2$, $\boldsymbol{D} = [1, 1, 1]$, $D = \sqrt{3}$, $\sigma = \sqrt{2}/D\sqrt{T}$.

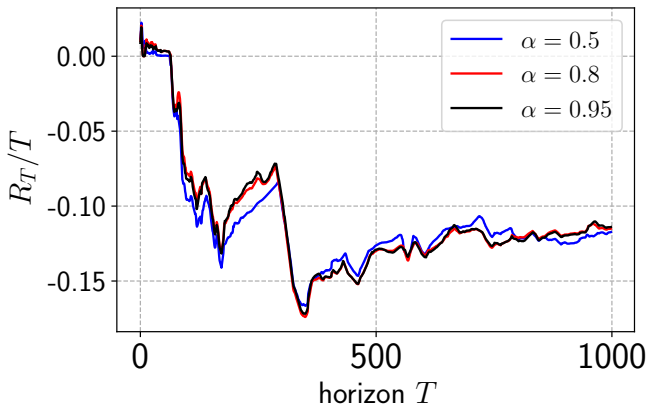

Fig. 4: *Evolution of* R_T/T*:* Horizon $T = 1008$, $m = 3$, $V = 2$, $\boldsymbol{D} = [1, 1, 1]$, $D = \sqrt{3}$, $\sigma = \sqrt{2}/D\sqrt{T}$.

rate of the prediction $\nabla \hat{f}_{t+1}(\hat{\boldsymbol{z}}_{t+1})$ against the real value $\nabla f_{t+1}(\boldsymbol{z}_{t+1})$. We set $\zeta = 0.3$ and 4 to respectively represent an accurate and inaccurate prediction model. We call OOLRgrad the online decision algorithm OOLR with the prediction method ARMA-OGD directly applied to the gradient items. We omit to show the OOLRsignal version where the prediction method is applied to the feedback signals (demand, prices, etc.) as it does not provide any regret bound guarantee. We expect a regret bound of $\mathcal{O}(\sqrt{Res + \sqrt{T}\sqrt{T}}) \sim \mathcal{O}(T^{3/4})$, as we can neglect the residual error Res of the best ARMA prediction model against $\sqrt{T}$. We show in Fig. 3 the regret performance of the three models $\zeta = 4$, $\zeta = 0.3$ and OOLRgrad. We first observe the convergence of the average regret R_T/T towards 0 for the three models, which confirm the regret bound of $\mathcal{O}(\sqrt{L_T T})$ even for arbitrarily bad predictions (represented by the $\zeta = 4$ model). Secondly, we observe a negative regret for the $\zeta = 0.3$ and OOLRgrad models, which confirm the $\mathcal{O}(1)$ regret bound when the predictions are accurate and the accumulated error $\sum_{t=1}^{T} ||c_t - \tilde{c}_t||^2$ is close to 0. We remark that our OOLRgrad solution based on the ARMA-OGD predictor shows similar performance than the OOLR solution with a 70% accurate predictor.

D. Extension

Now we evaluate the OOLR solution in the scenario where the NO is unable to fulfill the SP request in its entirety. We focus on a basic scenario in which the NO ensures a minimum ratio of α for all kinds of resources. One can envision a more complex scenario where the NO commits to a ratio of α_i for each resource i, where α_i are possibly different. Thus, for each resource i at slot t, the SP expect to receive a ratio $\alpha_{i,t}$ that belongs to the set $[\alpha, 1]$. We draw the $\{\alpha_{i,t}\}_t$ from the uniform distribution on $[\alpha, 1]$. We assume that the NO consistently deliver all requested spot resources, thus we keep $\beta = 1$.

We observe in Fig. 4 the regret performance of the OOLRgrad solution for three different SLAs, which are $\alpha \in \{0.5, 0.8, 0.95\}$. We observe that the performance stays similar regardless of the SLA the SP has complied for, which implies our OOLRgrad solution is consistently applicable.

E. Doubling trick

Appendix

We detail how we obtain the theoretical regret bound of the OOLRgrad solution. First, we apply the ARMA-OGD on each gradient item and we obtain the regret bound for item i:

$$\sum_{t=1}^{T} (\nabla_i f_t(\boldsymbol{z}_t) - \nabla_i \tilde{f}_t(\tilde{z}_t))^2 = Res + \mathcal{O}(GM\sqrt{T}),$$

where Res is the residual error of the best ARMA prediction method with full hindsight, G and M are key constant in [17]. Thus, we obtain the regret bound:

$$\sum_{t=1}^{T} ||\nabla f_t(\boldsymbol{z}_t) - \nabla \tilde{f}_t(\tilde{z}_t)||^2 = 2mRes + \mathcal{O}(2mGM\sqrt{T}).$$

Thus we rewrite the regret bound in (21) as:

$$R(T) = \mathcal{O}\Big(2D\sqrt{2T}\sqrt{2mRes + \mathcal{O}(2mGM\sqrt{T})}\Big).$$

Within the square root, we neglect $2mRes$ in front of $\mathcal{O}(2mGM\sqrt{T})$, hence we can simplify:

$$R(T) = \mathcal{O}(4D\sqrt{mGM}T^{3/4}) = \mathcal{O}(T^{3/4}).$$

References

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

[2] F. W. Murti, A. Garcia-Saavedra, X. Costa-Perez, and G. Iosifidis, "On the optimization of multi-cloud virtualized radio access networks," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–7.

[3] 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.

[4] 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.

[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. 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.

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

[11] 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.

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

[13] J.-B. Monteil, G. Iosifidis, and L. Da Silva, "Learning-based reservation of virtualized network resources," *IEEE Transactions on Network and Service Management*, 2022.

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

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

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

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

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

[19] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, "Deepcog: Cognitive network management in sliced 5g networks with deep learning," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 2019, pp. 280–288.

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

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

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

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] N. Reyhanian, H. Farmanbar, and Z.-Q. Luo, "Data-driven adaptive network resource slicing for multi-tenant networks," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 4715–4719.

[29] 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.

[30] 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.

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

[32] N. Liakopoulos, G. Paschos, and T. Spyropoulos, "No Regret in Cloud Resources Reservation with Violation Guarantees," in *Proc. of IEEE INFOCOM*, 2019.

[33] F. Fossati, S. Moretti, P. Perny, and S. Secci, "Multi-resource allocation for network slicing," *IEEE/ACM Transactions on Networking*, vol. 28, no. 3, pp. 1311–1324, 2020.

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

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

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

[37] O. E. Barndorff-Nielsen and N. Shephard, "Non-gaussian ornstein–uhlenbeck-based models and some of their uses in financial economics," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 2, pp. 167–241, 2001.