Safeguard Network Slicing in 5G: A Learning Augmented Optimization Approach

Xiangle Cheng, Yulei Wu, *Senior Member, IEEE,* Geyong Min, *Member, IEEE,* Albert Y. Zomaya, *Fellow, IEEE,* and Xuming Fang, *Senior Member, IEEE*

***Abstract*—Network slicing, as a key 5G enabling technology, is promising to support with more flexibility, agility, and intelligence towards the provisioned services and infrastructure management. Fulfilling these tasks is challenging, as nowadays networks are increasingly heterogeneous, dynamic and large-dimensioned. This contradicts the dominant network slicing solutions that only customize immediate performance over one snapshot of the system in the literature. Instead, this paper first presents a two-stage slicing optimization model with time-averaged metrics to safeguard the network slicing in the dynamical networks, where prior environmental knowledge is absent but can be partially observed at runtime. Directly solving an off-line solution to this problem is intractable since the future system realizations are unknown before decisions. Therefore, we propose a learning augmented optimization approach with deep learning and Lyapunov stability theories. This enables the system to learn a safe slicing solution from both historical records and run-time observations. We prove that the proposed solution is always feasible and nearly optimal, up to a constant additive factor. Finally, we demonstrate up to 2.6× improvement in the simulation when compared with three state-of-the-art algorithms.**

***Index Terms*—Network slicing, 5G, deep learning, Lyapunov optimization**

I. Introduction

With the evolution of Software-Defined Networking (SDN) and Network Function Virtualization (NFV), 5G networks have advocated a revolutionary paradigm called network slicing [1] [2] to construct network services. Unlike the large deployment of dedicatedly built network devices in conventional networks, network slicing utilizes Virtual Network Functions (VNF) to implement individually optimized services on top of a shared physical infrastructure. This enables a more flexible, scalable and agile management towards the end-to-end physical resources, including communication and computation resources, radio spectrum, energy, *etc*.

Network slicing has been accepted as an integrated part of the latest 3GPP release [3], but its technical implementation is solution agnostic. With proper assumptions or knowledge on traffic patterns and networking environment, a myriad of slicing solutions based on classic optimization theories (*e.g.*, combinatorial/convex optimization) have been proposed (related surveys can be found in [1], [4]). As with the dominant implementations of network slicing in the literature, if the full knowledge of traffic and network states is assumed known, *e.g.*, observed states or Probability Distribution Functions (PDFs), we can always compute in offline an optimal or approximate slicing policy to best respond to these given network states.

However, with today's networks becoming increasingly dynamic, heterogeneous, and large-dimensioned, the real-time tracking and explicitly modelling of the networking environment are getting increasingly costly or even intractable. Examples can be easily found from these typical 5G scenarios, such as massive human-/machine-type connections, or dense small cells with millimeter-wave spectrum [5]. The system states in these emerging applications may evolve rapidly on nearly a millisecond order [6]. In these popular 5G cases, it is strongly vulnerable to environmental changes for the slicing schemes that customize the immediate performance over a given deterministic environment information in the literature. This requires that a slicing system should be able to make decisions in the absence of partial system state information while the resulted solution efficiencies are safeguarded across the whole running trajectory. Compared with the existing challenges addressed in the literature, *e.g.*, solving the NP-hard combinatorial slicing model, this leads to three additional challenges for the targeted network slicing problem:

- The slice operation environment frequently evolves with great uncertainties, and explicit environment knowledge/models are unavailable ahead of schedule;
- Timely control response, possibly on a millisecond order, is required in order to secure the system running as desired. This precludes the traditional scaling or migration based dynamic slicing solutions, which re-solve slicing models after each network change. This is because each attempt of solving a usually NP-hard slicing model in large-scale networks is costly and time-consuming;
- In addition to preserving the real-time performance and constraints, the chosen slicing policy should also be able to work properly across the whole system running trajectory and to safeguard the system performance in the long run.

In light of the above insights, it is essential for a slicing system to observe environment variations, learn uncertainties, and accordingly plan response actions properly. Therefore, in this paper, we aim to safeguard the network slicing in dynamic

Manuscript received April 15, 2019; revised February 03, 2020; accepted March 02, 2020. (*Corresponding authors: Yulei Wu and Geyong Min*.)

X. Cheng, Y. Wu and G. Min are with the College of Engineering, Mathematics and Physical Sciences, University of Exeter, Exeter EX4 4QF, U.K. E-mail: {xc272, y.l.wu, g.min}@exeter.ac.uk.

A.Y. Zomaya is with the School of Information Technologies, University of Sydney, Camperdown NSW 2006, Australia. E-mail: albert.zomaya@sydney.edu.au.

X. Fang is with the Key Lab of Info Coding & Transmission, Southwest Jiaotong University, Chengdu 610031, China. E-mail: xmfang@swjtu.edu.cn.

5G networks so that the returned slicing policy is always feasible and the resultant system performance is long-term safeguarded.

To address these challenges and achieve the desired objectives, we first present a two-stage slicing optimization model with time-averaged metrics. This provides a two-phase control to secure the slicing process: initial slice deployment and long-term slice operation. Directly solving an off-line solution to this problem is intractable since the future system realizations are unknown before decisions. Therefore, we propose a learning augmented optimization approach by intertwining deep learning and Lyapunov stability theories to deploy and operate network slices with both historical records and real-time observations. We prove that the proposed slicing solution is always feasible and nearly optimal, up to a constant additive factor. In the proposed solution, the involved combinatorial model only needs to be solved during the initial deployment phase, while the subsequent operation controls only involve solving a simple continuous program. Therefore, this solution is robust and agile to respond to any dynamics. In the simulation, we demonstrate up to 2.6× improvement when compared with the state-of-the-art baselines.

The major contributions of this paper can be summarized as follows:

- We formulate the safe network slicing problem in dynamic networks as a two-stage optimization model with time-averaged metrics. This provides a systematic support for the slicing systems to achieve robust deployment and prompt adaptation of sliced services in a dynamic environment.
- By intertwining the advanced deep learning and Sample Averaged Approximation (SAA) optimization tools, we present an approximate solution to deploy network slices with robustness under incomplete system knowledge. An analytical probabilistic bound is provided, which can be improved by increasing the sample size.
- We propose an online slice operation policy with misloading calibration, which enables the deployed network slices to learn from and adapt to real-time observations. Based on the Lyapunov stability theory, we present a proven better performance bound than the referenced online learning algorithm while maintaining a same solution feasibility.
- Finally, extensive simulation experiments are conducted with the settings in accordance with 5G expectations. Through the comparison with the incumbent network slicing solutions, we demonstrate the efficacy of the proposed learning augmented optimization approach for the targeted network slicing problem.

The rest of this paper is structured as follows. We first investigate the related works in Section II. The implementation of the safe network slicing optimization model is presented in Section III. In Section IV, we extend the classic SAA method with deep learning based predictions to learn an approximated slicing deployment policy from historical records. Then, the safe slice operation control solution is presented in Section V. The simulation results are summarized in Section VI. Finally, Section VII concludes this paper.

II. Related Work

Network slicing has been identified as the backbone of the rapidly evolving 5G technology [1]. By allowing different parties to instantiate and run software-based network services, this paradigm facilitates the development of service-tailed and truly differentiated services on top of a shared underlying network infrastructure. Gaining momentum from immense 5G applications [5], network slicing has been the focus of an ever-growing community of researchers.

A. Network Slice Modelling and Optimization

There are many deterministic network slicing modelling solutions reported in the preliminary studies, which extensively investigate the basic implementation of network slicing in static networks. These studies mainly focus on the optimization of the resource solutions with acceptable computing complexity under a given network state. Thus, these solutions are not directly applicable in the considered problem. Related works can be found in *e.g.*, [4], [8]- [10].

There also exist a few studies in the literature striving to address similar resource utility problems for dynamic NFV networks. Among these very few work, Ying *et al.* in [11] addressed the joint optimization problem of dynamic radio resources and computing resource. A robust resource allocation framework is presented in [12] with an iterative algorithm to auto-scale slices in response to the changing environment. With a similar motivation, the resource provisioning solution proposed by Li *et al.* [13] is proactive although their objective is to assign requests with bounded response time. This is achieved by using slice consolidation with timing abstraction, but the placement of slices is still based on deterministic models, and the instance migration of VNFs is involved when new requests arrive. Split/Merge [14] provides system support for achieving efficient, load-balanced elasticity when scaling in and out of virtual middleboxes. However, as aforementioned, scaling or migration based dynamic solutions are precluded in our problem. In our prior work in [15], we provided a stochastic solution to the similar problem and showed that explicit PDF models of environmental knowledge can contribute considerable improvements. In this paper, we release the dependence of explicit environmental models of our prior solution by exploiting the learning and online optimization tools. This makes the resulted slicing solution applicable to more generic settings.

B. Network Optimization with Online Learning Approaches

Recently, there is also an increasing traction on developing online solutions to handle dynamic networking problems. The pioneering works can be traced back to the studies that are based on online computation and competitive analysis [16]. Among the most related studies, Jia *et al.* in [17] investigated the online scaling problem of NFV service chains across geo-distributed datacenters. Their solution aims to handle time-varying traffic volumes and relies on the dynamic scaling

of VNF instances. Evan *et al.* in [18] studied the online embedding of virtual networks. Their goal is to select high-benefit requests meanwhile maximizing the likelihood that future requests can be embedded.

Other major attempts are the applications of Lyapunov optimization [19] and multi-armed bandits (MAB) theories [20] to model the resource allocation problems operating in dynamic systems. In this case, these classic online learning theories are exploited to make a sequence of control decisions with progressively learned knowledge about system dynamics and to optimize the long-term cumulative rewards.

For example, Mao *et al.* developed in [21] a Lyapunov optimization based dynamic computation offloading solution for mobile-edge computing applications. Huang *et al.* in [22] studied the learning-aided stochastic network optimization with dual learning and online queue-based control. Neely in [23] investigated the application of Lyapunov theories for the distributed stochastic sensor network problem. The applications of MAB can also be found in the problems of antenna beam selection [24], mobile edge computing [25], *etc*.

However, all these works are mainly focused on addressing the sequential online control problems under the imperfect system state information. This is achieved by repeatedly solving their base models with progressively collected new observations across time. In contrast, this paper highlights the stochastic combinatorial hardness of the considered problem. Herein, it is not supported to repeatedly solve the combinatorial models with new knowledge. Additionally, we attempt to intertwine the advanced learning and online optimization theories so that we can merge both empirical data experience and domain expertise to enhance the slicing solution.

C. Connections with Existing Optimization Approaches

In this paper, learning augmented optimization, is proposed to safeguard the optimization of 5G network slicing. As a combinatorial approach, we can find both connections and differences of this approach to the related existing ones.

Stochastic/robust optimization approaches [26] [27]: As the conventional ways to optimize the decision procedures under the presence of dynamics and uncertainties, this kind of approaches models the system uncertainties and dynamics from historical samples as certain stochastic processes or probabilistic events. Accordingly, appropriate actions are chosen by searching or planning under these constructed models. Owning to the solid stochastic theories, these approaches find successful applications in problems, *e.g.* [28], [29], [30].

Machine learning approaches [31]: However, explicit knowledge models of environment/events are not always available, and we need to observe and analyze the sample paths of the system to make the optimization decisions. In this case, the diverse data-driven machine learning tools are applicable to directly infer decisions from historical observations. Nevertheless, the data-driven philosophy that underpins machine learning inherently exhibits poor decision interpretation and accuracy. Although claimed nearly 90% of accuracy by latest deep learning models for optimization tasks *e.g.,* in [32], this is still far from the required reliability for carrier-grade network services.

Learning augmented optimization approach: Instead, we reject the exclusive choice between the above two approaches and advocate for a combinatorial approach which benefits from their complementary strengths. Concretely, learning augmented optimization aims to intergrate multiple artificial intelligence tools (*e.g.*, classic optimization, control theories and machine learning) into networking architecture to learn a holistic solution. As depicted in Fig. 1, such an approach, essentially, retains the basic optimization structure, but more learned knowledge from historical records and run-time observations are introduced to improve the optimization outcomes. The testaments in *e.g.,* [33], [34], [34] have demonstrated the promissing efficacy of deep learning based predictions for the tasks of VNF resource management. In this paper, we will exam the joint application of deep learning and online learning theories to improve the network slicing optimization in the long run.

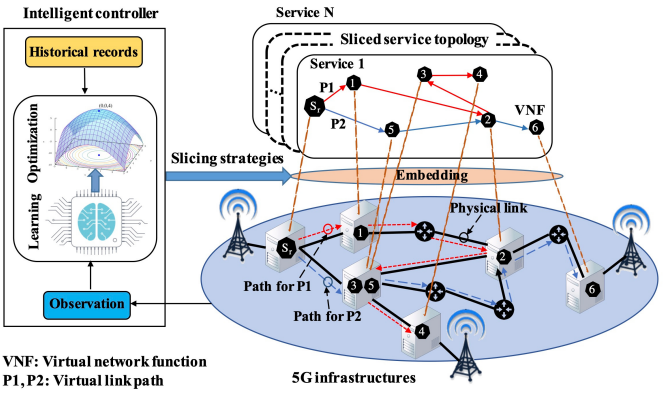


Fig. 1. A general framework of learning augmented optimization system for network slicing in 5G: The intelligent controller decides the VNF embedding and routing strategies based on the learning augmented optimization, and then the requested network slices are embedded and operated accordingly in the 5G infrastructures.

III. System Model

As portrayed in the system architecture in Fig. 1, a sliced network service is constructed with solely required VNFs that are chained in a specific order according to its service policies. The installation/teardown and operations of network slices are mutually independent, although the underlying infrastructures are shared. In this case, one critical task for network slicing is to plan the utilization of the shared resources with adherence to diverse service and resource constraints. This is non-trivial and more challenging than planning 4G network services due to the rapidly expanded network sizes as well as the frequently changing network environment in 5G.

Let us treat the 5G slicing system as a discrete-time stochastic system with time-varying resources c_t. Upon the given limited system resources, slicing requests with random resource demands d_t are required to schedule and control. At the beginning of each time slot $t \in \{0, 1, 2, \dots\}$, the system controller is able to observe a state update $\omega_t = (d_t, c_t)$, which specifies the independent realizations of current traffic and network states. The network slicing optimization problem

aims to decide a robust slicing policy at the beginning of every scheduling interval T so that the slicing system can be configured and operated with full respect to the desired objectives and constraints.

A. Preliminaries

Considering the system dynamics in this problem, we apply partial admission control policy. Accordingly, requests can be accepted (*i.e.*, deployed) but run with compromised flow rates rather than be directly rejected when the available resources at the decision time are not enough to fully meet the required demands. This ameliorates the over-/under-loading problem caused by partial observations at decision time. In this setting, the slicing policy can be split into two parts: slice deployment policy and slice control policy.

We first clearly define the two policies to be optimized in this paper as follows.

Definition 1 (**Slice deployment policy** $\pi_+ \in P_+$)*:* is a policy vector specifying which slice requests are accepted, where to instantiate and how to chain the required VNFs along with routing paths in the underlying physical networks.

Definition 2 (**Slice control policy** $\pi_- \in P_-$)*:* is a contingency plan for choosing a single running control action to adapt the flow rates of deployed slices to a given network event observation under the chosen slice deployment policy π_+.

where P_+, P_- are the solution spaces for the two policies, respectively.

In this paper, we denote a slicing policy $\pi := \{\pi_+, \pi_-\}$ to be ***safe*** if it is feasible across the runtime (*i.e.*, without violating any real-time or time-averaged constraints) while optimizing the long-term system performance.

B. The Deterministic Network Slicing Optimization Problem

As aforementioned, real-time scaling and migration of deployed slice instances are not considered. In this case, the slicing system can be sequentially controlled as a two-stage process: (i) decide a long-term feasible and robust slice deployment policy with an initial knowledge, and (ii) then under the given deployment, adaptively control the running flow rates allocated to the deployed network slices according to the real-time environment changing. As with the extensively studied slicing models in the literature, we can abstract, without loss of generality[1], the network slicing optimization problem under any given system state ω as the following **T**wo-**S**tage **S**licing (**T2S**) Program:

$$(\textbf{T2S model}) \quad \pi = \underset{\pi_+, \pi_-}{\operatorname{argmin}} \; f(\pi_+, \pi_-, \omega) \tag{1}$$

$$s.t. \qquad \pi_+ \in P_+ \text{ and } \pi_+ \text{ is binary} \tag{2}$$

$$\pi_- \in \underset{\pi_- \geq \mathbf{0}}{\operatorname{argmin}} \left\{ f(\pi_-, \omega | \pi_+) \middle| u_l(\pi_-, \omega) \leq 0, l = 1, 2, \ldots, L \right\} \tag{3}$$

where

- $\{f, u_l\}$: the objective function and a set of utility functions specifying the real-time constraints required for running the deployed network slices in the physical network. Their values are random, dependent on the realization of system state ω;
- P_+: the solution space to be defined by a set of constraints to guarantee that the chosen π_+ is a valid slice deployment policy (*e.g.*, satisfying flow conservation). A concrete example is provided in (32)–(34);
- π_+: a vector of decision variables indicating which physical nodes and link paths are used to construct the accepted network slices;
- π_-: real-time running flow rates allocated to the accepted network slices under the given slice deployment policy π_+ and system state ω;

As claimed in extensive existing studies *e.g.*, [12], under the given system state ω, solving this problem is NP-hard. By this token, many reported exact or approximated computing strategies in the literature can be used to solve this problem, such as dynamic programming [36] or integer relaxation [37]. However, the resulted solution can only customize the system performance that is best respond to the given system state. When the system evolves frequently and explicit models of the system environment are unavailable a-priori, the obtained slicing solutions through these traditional approaches are not safe across time.

Next, we safeguard the long-term performance of this **T2S** model by incorporating time-averaged performance metrics.

C. Safeguarding Long-Term Performance with Time-Averaged Metrics

Let us denote $\{\pi_-^t\}_{t=0}^{\infty}$ as the slice control policy across t and $f(\pi_+, \pi_-^t, \omega_t)$ as the time-variant objective function. Then, we define the time-averaged objective function as follows:

$$\bar{f}\left(\pi_+, \{\pi_-^t\}_{t=0}^{\infty}\right) = \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}_{\omega}\left[f(\pi_+, \pi_-^{\tau}, \omega_{\tau})\right] \tag{4}$$

Likewise, we define a set of time-averaged utility functions $\bar{g}_k, k \in \{1, 2, \ldots, K\}$ through its time-variant function $g_k(\pi_+, \pi_-^t, \omega_t)$ as follows:

$$\bar{g}_k\left(\pi_+, \{\pi_-^t\}_{t=0}^{\infty}\right) = \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}_{\omega}\left[g_k(\pi_+, \pi_-^{\tau}, \omega_{\tau})\right] \tag{5}$$

In order to safeguard the slicing optimization solutions in 5G, we first extend the **T2S** model in (1) with its time-averaged objective and constraints. This results in our new proposed **T**wo-**S**tage **S**afe **S**licing (**T3S**) model as follows:

$$(\textbf{T3S model}) \quad \pi = \underset{\pi_+, \{\pi_-^t\}_{t=0}^{\infty}}{\operatorname{argmin}} \; \bar{f}\left(\pi_+, \{\pi_-^t\}_{t=0}^{\infty}\right) \tag{6}$$

$$s.t. \quad \bar{g}_k\left(\pi_+, \{\pi_-^t\}_{t=0}^{\infty}\right) \leq \bar{c}_k, k \in \{1, 2, \ldots, K\} \tag{7}$$

$$\text{Constraints } (2) - (3), \forall t \tag{8}$$

where $\bar{c}_k$ is the upper bound (or cost budget) of the time-averaged utility $\bar{g}_k$.

[1]Equality constraints can be equivalently expressed through two inequality constraints.

In this new slicing model, three differences can be highlighted when compared with the existing system formulations. First, by optimizing the time-averaged objective, it captures the concerns on the long-term performance safety in the considered stochastic system. Second, with the time-averaged constraints in (7), this model provides a higher flexibility to consolidate both the long-term and real-time running requirements for deploying the sliced services in dynamic networks. Finally, the combinatorial solution for π_+ is optimized to respond to long-term system behaviours. This circumvents the complicated amendments to π_+ (*e.g.*, dynamic scaling or migrating deployed slice instances). Alternatively, the adaptation to real-time changes is handled through simplified continuous programs in the slice operation process.

A concrete application example of the **T3S** model is that network providers want to optimize their long-term revenues from the provisioned network slices. The available resource capacities are time-varying due to *e.g.*, wireless channel fluctuation, traffic variations [15], [38]. In this case, the providers need to keep the average operation cost for *e.g.*, energy or bandwidth within given budgets. Meanwhile, both the real-time (*e.g.*, delay, jitter) and long-term (*e.g.*, packet loss/service interruption rate) service qualities should meet the contracted service level agreements. This is fundamentally more complicated than the deterministic counterparts.

In **T3S** model, we need to infer at the beginning of every scheduling interval a robust slicing policy that not only has an impact on the immediate system performance but also on the future ones. However, the objective function $f(\pi_+, \pi_-^t, \omega_t)$, utility functions $g_k(\pi_+, \pi_-^t, \omega_t)$ and $g_l(\pi_-^t, \omega_t)$ are unknown before decisions. Thus it is intractable to directly solve an optimal off-line solution to this problem.

Based on the above analyses, we resort to enhance the traditional stochastic optimization algorithms with learning components to solve this problem in two sequential steps. Specifically, a Deep learning aided SAA (DSAA) approach is first introduced to learn a robust slice deployment policy from historical system records. This eschews frequently re-solving the complicated combinatorial program in **T3S**. Considering the imperfect matching of the deployment policy with real-time operation environment, an online adaptation scheme is presented in the subsequent slice operation process to further secure the slicing system.

IV. Robust Slice Deployment Aided with Deep Learning

As the inevitable absence of future system knowledge in the targeted problem, we resort to improve the slice deployment policy by learning from historical system records.

A. Predictions with Sequence-to-Sequence LSTM Networks

The Long Short-Term Memory's (LSTM) ability to successfully learn on data with long range temporal dependencies makes it a natural choice for this application. In our problem, we need to predict the system realizations of next W_t time window with the newly observed T time-series sequence about the system realizations during the previous scheduling interval. The prediction size W_t may differ with the random lifespans required by slice requests. Clearly, this is beyond the capability of regular LSTM networks but can be well captured by sequence-to-sequence (seq2seq) LSTM Networks [39]. Consequently, we present a seq2seq LSTM based Encoder-Decoder learning structure for such a prediction task. As depicted in Fig. 2, the Encoder reads T new time-series observations into the LSTM network to produce an Encoder Vector as the representations of current system states. Then, the Decoder reads the updated Encoder Vector to another LSTM network and sequentially generates W_t predictions.

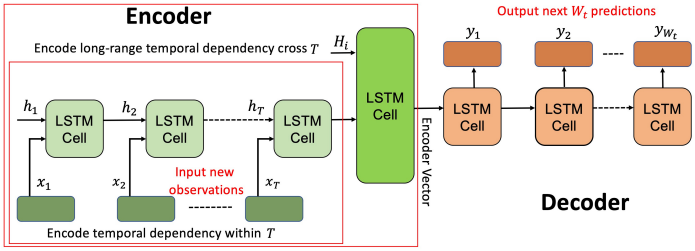


Fig. 2. Prediction with sequence to sequence LSTM networks.

B. Slice Deployment with Predictions

Let $\{\hat{\omega}_t\}_{t=1}^{W_t}$ be a learned system trajectory for the next W_t time window. Then, through approximating the cost and objective functions with the learned trajectory, we can get a **S**ampled reduction of the stochastic **T3S** model (**T4S**) as the following deterministic program:

$$(\textbf{T4S model})\ \pi_+ = \underset{\pi_+}{\operatorname{argmin}} \ \frac{1}{W_t+1}\sum_{t=0}^{W_t} f(\pi_+, \pi_-^t, \hat{\omega}_t) \tag{9}$$

$$\begin{cases} s.t. \\ \quad \frac{1}{W_t+1}\sum_{t=0}^{W_t} g_k(\pi_+, \pi_-^t, \hat{\omega}_t) \le \bar{c}_k, k = 1, 2, \ldots, K \\ \quad \pi_+ \in P_+ \text{ and } \pi_+ \text{ is binary} \\ \quad \pi_-^t \in \operatorname{argmin}_{\pi_-^t \ge \mathbf{0}} f(\pi_-^t, \hat{\omega}_t | \pi_+), \forall t = 0, 1, \ldots, W_t \\ \quad \begin{cases} s.t. \\ \quad\quad u_l(\pi_-^t, \hat{\omega}_t) \le 0, l = 1, 2, \ldots, L \end{cases} \end{cases} \tag{10}$$

In **T4S**, the reduction of a stochastic problem to its deterministic approximation resembles the SAA based techniques (*e.g.*, [22], [40], [41]). Instead of a direct application of SAA with random historical samples, the manipulation with learned outcomes in DSAA/**T4S**[2] reaps two-fold benefits: i) better extract the temporal dependencies of the underlying system evolution, and 2) save tremendous computation on unnecessary samples when solving the combinatorial model. As a deterministic problem, many existing algorithms (fractional rounding, heuristic or decomposition, as surveyed in [9]) can be invoked to solve an exact or suboptimal solution for the **T4S** model, which is out of the focus in this work.

The optimality of the learned deployment policy depends on both the prediction accuracy and the distribution consistency between historical samples and future outcomes. Since the objective function is measured in expectation, a strictly error-free prediction at each $t \in \{1 \cdots W_t\}$ is unnecessary. When

[2]DSAA and **T4S** are used interchangeably in this paper.

the underlying stochastic processes are ergodic[3], the simplest learning strategy would be a directly random sampling from the historical records. However, in order to preserve the distribution consistency in the behavioral patterns of dynamic systems, this requires a vast amount of samples so that all possible states (peak, normal and valley) and their transition patterns can be precisely captured. We validated in Section VI that the proposed solution gains remarkable improvement even with a random sampling based learning strategy. When the objective function f is linear with respect to ω_t (as the case in the Benchmark problem in Section VI-A), we can further get an analytical *probabilistic bounds* on the objective value that improves (in expectation) with increasing sample/prediction window size. This is presented in APPENDIX A.

Misloading discussion: A suboptimal policy π_+ solved by an approximate solver to (9) tends to accept less requests so that the given cost budget constraint will not be violated. An under-loaded system will not be able to harvest a better networking environment later after the decision. Reversely, the imperfect prediction accuracy may make the system overload, thus more cost will be devoted to maintaining the active of deployed slices. Both cases will lead to system degradation. Next, an online misloading (*i.e.*, under-/over-loading) calibration scheme will be introduced to ameliorate this issue.

V. Adaptive Online Slice Operation with Misloading Calibration

Once instantiated under the learned slice deployment policy $\hat{\pi}_+$ from the history records, the original **T3S** model reduces to a simplified stochastic continuous program, which aims to exert a slice running control process to further secure both the real-time and long-term system performances. Since the future realizations and PDF models of ω_t are unavailable a-prior, we now propose an online learning approach to solve the **T3S** for the slice **R**unning control policy (called **T3S-R** model hereafter). In the following, $\hat{\pi}_+$ is identified as a known parameter and will not be shown explicitly in the **T3S-R** problem. Additionally, we assume the **T3S-R** is convex on π^t_-, which is aligned with most of existing resource allocation problems for network slicing.

A. Online Running Control with Misloading Calibration

In an online process, a solution has the following structure: At the beginning of every slot t, the system controller observes a realization of ω_t, and then a slice running control π^t_- is derived accordingly. In order to achieve the long-term objective, we construct the online process with the theoretical support of virtual queuing networks [19].

Let us treat each time-averaged constraint in (7) as a virtual queuing process. For each constraint $k \in \{1, 2, \ldots, K\}$, define a virtual queue Q^t_k with initial condition $Q^0_k = 0$. The queue backlog updates over time via:

$$Q^{t+1}_k = \max\{Q^t_k + g_k(\pi^t_-, \omega_t) - \bar{c}_k, 0\} \tag{11}$$

[3]An ergodic process is a stochastic process whose behavior does not depend on the initial conditions and whose statistical properties do not vary with time [47].

The connection of such a queuing network with the **T3S-R** is that if we control to stabilize the queue Q^t_k, the average of the "arrival process" $g_k(\pi^t_-, \omega_t)$ must be less than or equal to the "service process" $\bar{c}_k$. Consequently, the resultant control sequence will be a feasible solution meeting the time-averaged constraint $\bar{g}_k\left(\hat{\pi}_+, \{\pi^t_-\}_{t=0}^{\infty}\right) \leq \bar{c}_k$.

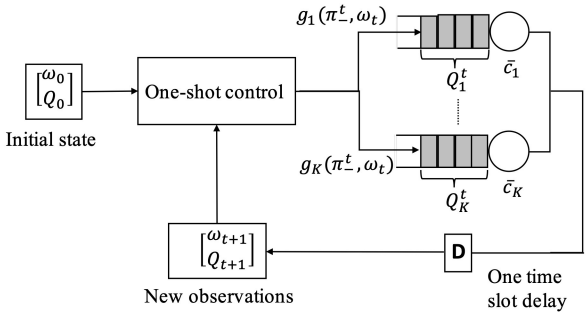


Fig. 3. Online slice running with the virtual queuing network.

Let $Q_t = [Q^t_1, Q^t_2, \ldots, Q^t_K]$ be the vector of queue backlogs, and define the *Lyapunov function* L_t as follows:

$$L_t = \frac{1}{2}\|Q_t\|^2 = \frac{1}{2}\sum_{k=1}^{K}[Q^t_k]^2 \tag{12}$$

The value L_t is a scalar measure of the size of the queue backlogs till t. If we take actions to consistently push this value down, then the queues will be stabilized (*i.e.*, satisfy the time-averaged constraints in (7)). Fig. 3 illustrates the online slice running control process with the virtual queuing network.

In this online framework, the system observes the current queuing state Q_t and new realizations of ω_t at the beginning of every slot t. Then, an one-shot control is exerted accordingly to adapt the slice running to the new observations. Motivated by the theory in [19], we implement the one-shot control by minimizing the following drift-plus-penalty expression:

$$\min_{\pi^t_- \geq \mathbf{0}} \quad Vf(\pi^t_-, \omega_t) + \sum_{k=1}^{K} Q^t_k g_k(\pi^t_-, \omega_t) \tag{13}$$

$$s.t. \qquad u_l(\pi^t_-, \omega_t) \leq 0, l = 1, 2, \ldots, L \tag{14}$$

where V is a non-negative weight that will be shown to affect the performance tradeoff.

Behind the one-shot control strategy is the intuition that more budget than the average $\bar{c}_k$ can be provided so that we can fully exploit the '*good*' state ω_t to collect a better objective value $f(\pi^t_-, \omega_t)$. Balanced by the surplus budget from the under utilization of $\bar{c}_k$ when ω_t is in '*bad*' quality, the time-averaged utility budget constraints in (7) can still be preserved.

With the convergence analysis in [42], the control strategy from (13) is known to provide an $O(1/V)$ approximation to the optimality of **T3S-R** with a convergence time of $O(V^2)$. However, considering the impacts of misloading resulted from the imperfect prediction accuracy and the approximation for solving **T4S** model, we introduce a misloading calibration scheme by extending the queue update function as following:

$$Q^{t+1}_k = \max\{Q^t_k + g_k(\pi^t_-, \omega_t) - (\bar{c}_k + \delta_k), 0\} \tag{15}$$

where $\delta_k = \max\{\bar{c}_k - \frac{1}{W_t+1}\sum_{\tau=0}^{W_t} g_k(\pi_-^\tau, \hat{\omega}_\tau), 0\}$.

δ_k measures the positive cost gap between the cost budget and the estimated utility due to the suboptimality of $\hat{\pi}_+$. With the calibration from δ_k, (15) imposes a positive utility offset to the available cost budget. This avoids the unnecessary penalty on the objective $f(\pi_-^t, \omega_t)$ in subsequent controls. As analyzed in Section V-B, such a calibration leads to a better convergence performance.

Finally, by putting all together, we are now ready to present the overall learning augmented optimization approach for the safe network slicing in Algorithm 1.

Algorithm 1 The proposed learning augmented optimization approach for safe network slicing.

Input: Historical system trajectory, network and requested slice topologies, V, W_t, T.

Output: Slice deployment policy $\hat{\pi}_+$ and online running flow rate π_-^t.

▷ *Slice deployment at the beginning of every T*

1: *(Historical Learning)* Generate the predictions of ω_t for the next W_t time slots.
2: *(Deployment Policy Approximation)* Infer the slice deployment policy $\hat{\pi}_+$ by solving **T4S** model under predictions $\{\hat{\omega}_t\}_{t=1}^{W_t}$.
3: Instantiate accepted network slices according to policy $\hat{\pi}_+$.

▷ *Online slice running control*

4: *(New Observations)* Collect new observations for ω_t, Q_t at the beginning of time slot t.
5: *(One-shot Control)* Decide the current slice running policy π_-^t by solving the deterministic problem defined in (13).
6: *(Queue Update)* Observe the resulting utility $g_k(\pi_-^t, \omega_t)$, and update virtual queues by (15).
7: Go to Step 4 if $t = t + 1$ is not the beginning of new scheduling interval.
8: Otherwise go to Step 1.

B. Theoretical Analysis

We first show the running constraint violation across iteration in Algorithm 1.

Lemma 1*:* Let $Q_k^0 = 0$ and Q_k^t updates according to (15). Then, Algorithm 1 satisfies: for all $t > 0$, i) Q_k^t is mean-rate stable (*i.e.*, $\mathbb{E}[Q_k^t]$ is upper bounded by a finite value), and ii)

$$\bar{g}_k\big(\hat{\pi}_+, \{\pi_-^\tau\}_{\tau=0}^{t-1}\big) - \bar{c}_k \le \mathbb{E}[Q_k^t]/t + \delta_k, \forall k = 1, 2, \ldots, K \quad (16)$$

Proof. The boundedness of $\mathbb{E}[Q_k^t]$ can be proved by contradiction. Assume Q_k^t is infinite large at some t. Then, in order to minimize the expression in (13), it must return a flow running rate with $\pi_-^t = \mathbf{0}$ and lead to the decrease of queue length for next slot, Q_k^{t+1}, by $\bar{c}_k + \delta_k$. This continues until all Q_k^t stabilize with certain finite queue backlogs. Consequently, under the control of Algorithm 1, it is impossible for Q_k^t to grow to infinity and thus all Q_k^t are mean-rate stable. As shown in (16), this property is useful to guarantee a declined constraint violation across iterations. Next, we prove the constraint violation in (16).

From (15), we have:

$$Q_k^{t+1} \ge Q_k^t + g_k(\pi_-^t, \omega_t) - (\bar{c}_k + \delta_k) \quad (17)$$

Summing over $\tau \in \{0, 1, \ldots, t-1\}$ gives:

$$Q_k^t - Q_k^0 \ge \sum_{\tau=0}^{t-1} g_k(\pi_-^\tau, \omega_\tau) - t(\bar{c}_k + \delta_k) \quad (18)$$

Dividing by t and using the fact that $Q_k^0 = 0$ gives

$$\frac{Q_k^t}{t} \ge \frac{1}{t}\sum_{\tau=0}^{t-1} g_k(\pi_-^\tau, \omega_\tau) - \bar{c}_k - \delta_k \quad (19)$$

Taking expectations and re-arranging terms yield (16). ■

The right side of (16) presents the running violation on the time-averaged constraints in (7). With the boundness of Q_k^t, it is clear that the term $\mathbb{E}[Q_k^t]/t$ vanishes as $t \to \infty$. Additionally, the added calibration offset only takes positive values in the event of system underloading. In this case, $\hat{\pi}_+$ tends to accept less loads, and the real-time running utility $g_k(\pi_-^t, \omega_t)$ is less likely to exceed the average budget $\bar{c}_k$. Consequently, the added calibration offset is inoffensive to the constraint in (7). This shows that Algorithm 1 maintains nearly same solution feasibility as the existing drift-plus-penalty algorithm [42]. Meanwhile, as we will shown below, the action of the misloading calibration can enhance Algorithm 1 with a better converged objective than the vanilla drift-plus-penalty algorithm.

Theorem 1*:* Let $\bar{f}\big(\{\hat{\pi}_-^t\}_{t=0}^{t-1}\big)$ be the achieved objective value under the online control sequence $\{\hat{\pi}_-^t\}_{t=0}^{t-1}$ by recursively solving program (13), and $\bar{f}^*$ be the optimum of **T3S-R** obtained by some '*genius*' decision maker who holds a complete knowledge about the true system trajectory $\{\omega_t\}_{t=0}^{t-1}$. Then, we have:

$$\bar{f}\big(\{\hat{\pi}_-^t\}_{t=0}^{t-1}\big) \le \bar{f}^* + \frac{1}{V}\big(B - \frac{1}{t}\sum_{\tau=0}^{t-1}\sum_{k=1}^{K}\mathbb{E}[Q_k^\tau]\delta_k\big) \quad (20)$$

where B is a positive constant that upper bounds the second moments of the "arrival" and "service" processes of Q_k^t as follows:

$$\frac{1}{2}\sum_{k=1}^{K}\mathbb{E}\Big[\big(g_k(\pi_-^t, \omega_t) - (\bar{c}_k + \delta_k)\big)^2\Big] \le B \quad (21)$$

Proof. This can be proved by extending the result in [42] with the misloading calibration scheme. Based on the objective function analysis in [42], we can bound the weighted-sum expression in (13) under the new queue update function in (15) as follows:

$$\mathbb{E}[\Delta_\tau] + V\mathbb{E}[f(\pi_-^\tau, \omega_\tau)] \le B + V\bar{f}^* - \sum_{k=1}^{K}\mathbb{E}[Q_k^\tau]\delta_k \quad (22)$$

where $\Delta_\tau = L_{\tau+1} - L_\tau$, called the *Lyapunov drift*.

Summing (22) over the first t slots gives:

$$\mathbb{E}[L_t] - \mathbb{E}[L_0] + V\sum_{\tau=0}^{t-1}\mathbb{E}[f(\pi_-^\tau, \omega_\tau)] \le (B + V\bar{f}^*)t - \sum_{\tau=0}^{t-1}\sum_{k=1}^{K}\mathbb{E}[Q_k^\tau]\delta_k \quad (23)$$

Dividing the above by Vt and using the fact that $\mathbb{E}[L_0] = 0$, $\mathbb{E}[L_t] \geq 0$, we have:

$$\bar{f}\big(\{\hat{\pi}_-^t\}_{t=0}^{t-1}\big) = \frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}[f(\hat{\pi}_-^\tau, \omega_\tau)] \tag{24}$$

$$\overset{\textbf{(a)}}{\leq} \frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}[f(\pi_-^\tau, \omega_\tau)] \tag{25}$$

$$\leq \bar{f}^* + \frac{1}{V}\big(B - \frac{1}{t}\sum_{\tau=0}^{t-1}\sum_{k=1}^{K}\mathbb{E}[Q_k^\tau]\delta_k\big) \tag{26}$$

where **(a)** follows because $\hat{\pi}_-^\tau$ is the optimal solution of program (13). *Proof* ends. ∎

With the fact that $\mathbb{E}[Q_k^\tau] \geq 0$ and $\delta_k \geq 0$, (20) shows that *Theorem* 1 always achieves a tighter running performance bound than the vanilla drift-plus-penalty algorithm. This particularly improves the performance when Q_k^τ spurts due to a sudden improvement of the system state when the system is under-loaded. This stands to the reason that it is less likely to break the time-averaged constraints in this case. Then, the action of misloading calibration provides a pullback on Q_k^t to avoid the unnecessary penalty on $f(\pi_-^t, \omega_t)$ in (13), which contributes the extra performance here.

VI. Performance Evaluation and Analysis

In this section, we use synthetic scenarios to evaluate the proposed solutions. Current BT's IP network topology within Europe[4] is considered as the network topology, from which arbitrary node is chosen as the destination of each network slice request. 5 nodes in the whole 21 nodes are randomly selected to act as the wireless access nodes. For each node in the network, a fixed number of computing resources is considered. Nevertheless, the available wireless transmission capacity within each access node is time-variant. Considering the long-term co-existence of 5G and legacy networks (3G/4G, *etc*.) and the tremendous changes among Line-of-Sight (LOS), non-LOS (NLOS) and outage stages in 5G wireless channels [6], we use the Rician fading and Rayleigh fading[5] [43] to emulate the network fluctuation. For each fading status, a fixed duration T_Δ is set, and the transition probabilities between statuses are equal. The channel parameters are configured so that the resulted capacity of each access node is on average within the envisioned capacity range for a 5G cell [44].

A. A Benchmarking Problem

As a benchmark, we extend the deterministic network slicing model in [12] to maximize the time-averaged system revenue while safeguarding the time-averaged link resource cost not exceeding a given budget. For clarity, the involved objective[6] and constraint functions are clearly defined as follows:

$$f(\cdot) = -\sum_{s\in S}\Big(\pi_-^{st}(b_s - \overbrace{\sum_{l\in\mathcal{L}^s, e\in\mathcal{E}}\pi_+^{le}k_e}^{\text{dynamic link cost}}) + \overbrace{\sum_{f\in\mathcal{F}^s, n\in\mathcal{N}}\pi_+^{fn}d_f k_n}^{\text{fixed node cost}}\Big) \tag{27}$$

$$g(\cdot) = \sum_{s\in S, l\in\mathcal{L}^s, e\in\mathcal{E}}\pi_-^{st}\pi_+^{le}k_e \tag{28}$$

$$u_l(\cdot): \begin{cases} \pi_-^{st} \leq \pi_+^s d_s, \forall s \in S & (29)\\ \sum_{l\in\mathcal{L}^s, s\in S}\pi_+^{le}\pi_-^{st} \leq c_e, \forall e \in \mathcal{E} & (30)\\ \sum_{s\in\{S|Sr(s)=n_a\}}\pi_-^{st} \leq c_{n_at}, \forall n_a \in N_{access} & (31)\end{cases}$$

and the solution space for a valid slice deployment policy $\pi_+ \in P_+$ is defined as follows:

$$P_+: \begin{cases} \sum_{f\in\mathcal{F}^s, s\in\mathcal{S}}\pi_+^{fn}d_f \leq c_n, \forall n \in \mathcal{N} & (32)\\ \sum_{n\in\mathcal{N}}\pi_+^{fn} = \pi_+^s, \forall f \in \mathcal{F}^s, s \in S & (33)\\ \sum_{e_{uv}\in O(u)}\pi_+^{l_{ij}e_{uv}} - \sum_{e_{vu}\in I(u)}\pi_+^{l_{ij}e_{vu}} = \pi_+^{iu} - \pi_+^{ju}, \\ \qquad \forall l_{ij} \in \mathcal{L}^s, s \in S, u \in \mathcal{N} & (34)\end{cases}$$

where

- $\{\mathcal{N}, \mathcal{E}\}$ are the node and directed link sets in the physical network, respectively;
- $\{\mathcal{L}^s, \mathcal{F}^s\}$ are the virtual link and VNF sets for slice request $s \in S$, respectively;
- $\pi_+ = \{\pi_+^s, \pi_+^{le}, \pi_+^{fn}\}, \forall s, l, e, f, n$ are binary variables that decide whether slice request $s \in S$ should be accepted, whether physical link e should be used to construct the virtual link l, and whether VNF f should be installed in physical node n, respectively;
- $\pi_- = \{\pi_-^{st}\}, \forall s, t$ are variables that decide the running flow rates allocated to s at t;
- $\{Sr(s), N_{access}\}$ are the source node of s and access node set, respectively;
- $\{c_n, c_e, c_{n_at}\}$ are the capacities of node, link and access resources, respectively;
- $\{b_s, k_e, k_n\}$ are the service benefit per unit rate and prices for using per unit physical link and node resources, respectively;
- $\{d_s, d_f\}$ are the demanded service rate of s, and the required computing resources to instantiate f, respectively;
- $\{O(u), I(u)\}$ are the outgoing and incidental links of physical node u, respectively;
- $\{e_{uv}, l_{ij}\}$ are the physical link connecting node v from node u and virtual link connecting VNF j from VNF i, respectively.

In this example, we aim to maximize the time-averaged system revenue $\bar{f}$ under a given time-averaged link cost budget $\bar{c}$ (*i.e.*, only one constraint, $\bar{g}\big(\pi_+, \{\pi_-^t\}_{t=0}^\infty\big) \leq \bar{c}$, for (7)). Specifically, Constraints (29) and (33) enforce the admission control

[4] http://www.topology-zoo.org/dataset.html

[5] But the proposed solution is not limited to any specific type of dynamics.

[6] Take minus to change to a minimization problem as defined in our models.

on correlated variables; (30) – (32) bound the capacities of corresponding link and node resources; (34) expresses the single-path flow conservation [45].

The proposed solution is compared with three existing reference algorithms, Current-Greedy (CG) [46] and Prediction Average Approximation (PAA) and Learning Augmented Optimization (LAO)+CG. In CG, slicing decisions are made only to optimize instant system revenue while guaranteeing that the resource cost constraint is not violated under the current network state. PAA can be viewed as an extension of SAA with next W_t predictions to approximate the deployment policy in **T4S**. The slice running control policies in PAA are then constructed to best respond to the predicted mean states. In contrast, in our proposed approach (LAO), the slice deployment policy is first derived through DSAA. Thereon, the deployed slices are adaptively operated with Lyapunov stability and misloading calibration scheme. In light of the separate efficacy of the slice deployment and running control policies in LAO, we further exam the combinations of the proposed slice deployment policy with CG based slice running control policy (for clarity, this is called LAO+CG).

We first evaluate the system by drawing system realizations randomly according to the corresponding fading distribution models. For simplyfication, we directly compare the performance of PAA under the approximation with W_t error-free predictions, which is an upper bound that PAA can achieve. We first implement the historical learning in LAO with a random sampling strategy (*i.e.*, directly sampling future realizations at random from upfront observations) to show the model robustness. All compared algorithms were solved under a same greedy solver, which greedily decides the placement policy π_+^s for each s through Benders' decomposition [40].

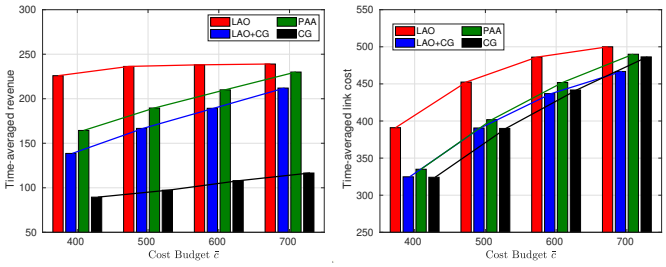
All algorithms are measured with the following four performance metrics: (1) time-averaged system revenue, (2) time-averaged system throughput, (3) time-averaged link cost, and (4) number of actively running slice services. The simulation parameters are shown in Table I.

TABLE I. Simulation Setup

Parameters	**Value**
$c_n, n \in \mathcal{N}$	Fixed with an initial value uniformly distributed within [5,10]
$c_e, e \in \mathcal{E}$	10*Gbps*
$[k_n, k_e, b_s]$	[10, 20/*Gbps*, 100/*Gbps*]
# of VNFs $\lvert\mathcal{F}^s\rvert$	3
Rate demand d_s	1–3 *Gbps*, uniformly distributed
Node resource demand d_f	1–3, uniformly distributed
Radio bandwidth	1 *GHz*
Rician factors	1*dB*
Power ratio of signal to noise plus interference	Rician: 31.3 *dB*; Rayleigh: 0 *dB*
Channel duration T_Δ	10
Scheduling interval T	20

B. Time-Averaged System Performance

We first test the time-averaged performance of compared algorithms when the system is converged. In the following experiments, We applied Poison arrival process with the arrive rate = 20 per T and average service lifespan $T_s = 40$ for the received network slice requests.



(a) (b)

Fig. 4. Performance comparisons with $V = 50, W_t = 40$: a) Time-averaged revenue, b) Time-averaged link cost.

In Fig. 4, we plot the compared time-averaged objective value (*i.e.*, system revenue) as a function of cost budget $\bar{c}$. We observe that the proposed slicing solution achieves an obvious system performance improvement over the compared algorithms in all cases. Moreover, under all cases, the converged time-averaged cost constraint is well preserved. Such improvement manifests particularly when the system is greatly limited by the cost budget constraint. For the best case, say $\bar{c} = 400$ in Fig. 4(a), LAO gains even up to 2.6×, 1.63× and 1.37× better performance than CG, LAO+CG and PAA, respectively. This can be interpreted as follows:

For CG, the slicing policies only best respond to the instant network states at the decision time. As a consequence, any over-optimistic decision will lead to high resource occupations for maintaining the over-loaded active slices. Likewise, any over-pessimistic decision will not be able to make full use of the available cost budget to improve the system performance. Benefited from the approximation with the partial future predictions, these situations are slightly alleviated in PAA. However, the predicted mean state information still fails to capture the long-term system evolution and then adapt itself efficiently. By contrast, with full respect to large history samples and learned dynamics, the proposed slicing solution well matches the system dynamics, which finally contributes the significant system improvements.

C. Solution Feasibility and Convergence Analysis

Fig. 5 presents the real-time average performance. The achieved numerical results in Fig. 5(c) show that all the compared algorithms meet the required time-averaged resource cost constraint throughout the running. However, this is achieved in CG and PAA by restricting the real-time resource cost at each running time slot according to the given observations and predicted mean state information, respectively. These drawbacks lead to very low utilization of the available cost budget. However, combined with the virtual queue based slice running policy in the proposed solution, the practical running link cost is below but very close to the given cost budget line. This confirms the solution feasibility of the proposed slicing policies, as analyzed in *Lemma* 1. On the other hand, we notice from both Fig. 4(b) and Fig. 5(c) that the available cost budget is always under-exploited for all compared algorithms.

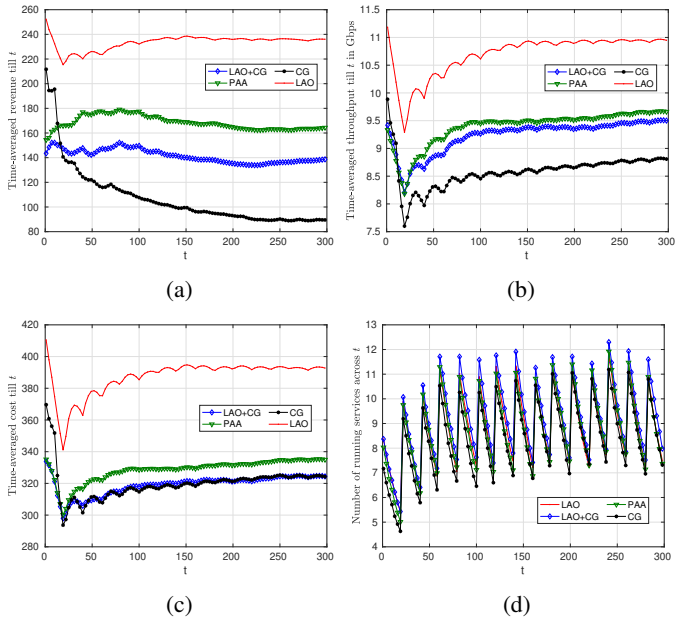


Fig. 5. Performance comparisons with $V = 50, W_t = 40, \bar{c} = 400$: a) Time-averaged system revenue till t, b) Time-averaged system throughput till t, c) Time-averaged link cost till t, d) Number of actively running services across t.

This is resulted from the suboptimality of the solution to the **T4S** program. When an approximate solver is applied during solving the **T4S** model, only solutions preserving the time-averaged cost constraint will be returned. Depending on the suboptimality gap of the applied approximate solver, the estimated cost is always inevitably lower than the given cost budget by a certain value.

Under a feasible slicing solution, Fig. 5(a) and Fig. 5(b) confirm that the benefits of network providers and the provisioned service throughput are all superior in LAO to the other three ones. In Fig. 5(d), the serrated changes on the number of actively running services across t are the results of the scheduling towards new slice requests and the service expiration of deployed slices. Fig. 5(d) shows that all compared algorithms maintains nearly same level of service loads when the overloading and underloading cases are averaged. However, it shows that a similar loading level in LAO contributes up to 2.6× better revenue than CG. This indicates that LAO coordinates the available resources for the competing requests in a more efficient manner. The degradation of the compared algorithms results from both the misloading during the slice deployment phase and the inefficient utilization of cost budget during the slice running control.

D. Parameter Effectiveness

We now test the parameter effectiveness of our proposal. Fig. 6 plots the running time-averaged performances under different V. It shows that the solution superiority of our proposal is preserved under all values of V. Moreover, all measured system metrics keep sustainable improvement as V goes larger, but the improvement gradually vanishes when V reaches a certain value.

Also revealed in Fig. 6, is the property that the running time-averaged system performance converges more quickly when controlled by a smaller V. These results are consistent with the theoretical analysis in Section V-B and [42]. The choice of a '*good*' V depends on the balancing requirements between the convergence speed and desired system performance, which is subject to practical applications.

In Fig. 6, we can also observe that there is always a descending slop during the start-up time along t axis. This is caused by the service expiration and tear-down of the deployed services with lifespan less than scheduling interval (*i.e.*, $T_s \leq T$). After a slow warm-up, more active services with lifespan $T_s > T$ will be accumulated and keep active in the following scheduling intervals. Since these services, once deployed, will have impacts throughout their lifespans, the service lifespan imposes a direct influence on the decision accuracy of the obtained slicing policies. In next section, we provide a more detailed analysis on the efficacy of different service lifespans.

E. Efficacy of Different Service Lifespan

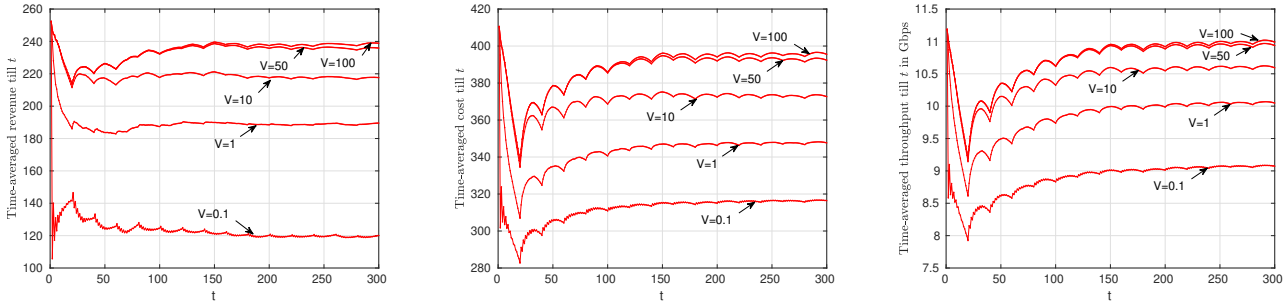
Intuitively, if the deployed services only possess a short lifespan, say $T_s < T$, the system will get under-loaded during the remaining time of a scheduling interval. In contrast, if $T_s \gg T$, say $T_s = \infty$, the system will keep loading new requests until the system is saturated during the peak networking status. To illustrate this, we examined three different lifespan settings $T_s = \{20, 50, 100\}$. The performance comparisons are shown in Fig. 7.

When services with short lifespans are largely loaded as the case of $T_s = 20$, Fig. 7 discloses that the system always maintains very low number of running services, resulting in the underutilization of available resources. Inversely, if it comes to the case of $T_s = 100$, more loaded services only lead to increased service competition and deteriorates the system performances. By contrast, $T_s = 50$ achieves the best control towards the misloading problem. In practice, the scheduling interval T is fine-tuned according to the features of provisioned services.

F. Efficacy of Different learning strategies

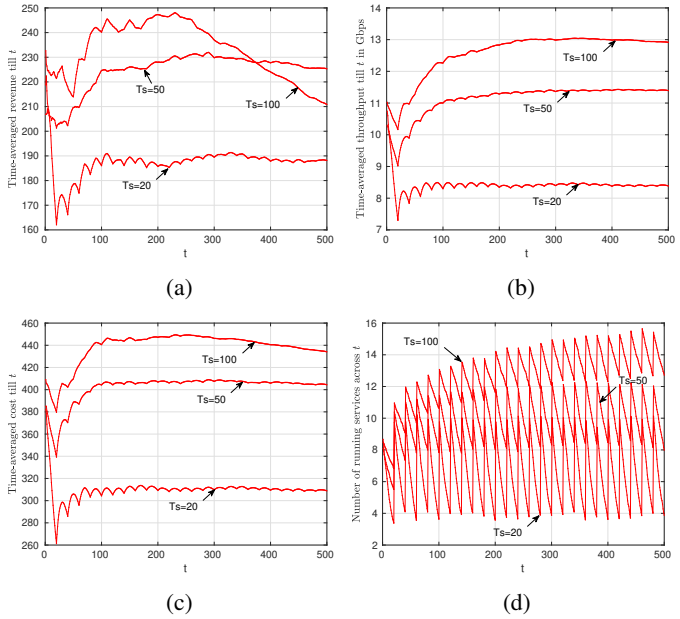
Next, we exam the efficacy of deep-learning based predictions. We first generated 10000 time-series system realizations according to the transition setting between Rician and Rayleigh fading to train the seq2seq LSTM model. The training is conducted with parameters: {3 hidden layers with size: 250, optimizor: Adam, 250 epochs, learning rate: 0.005}. Three learning strategies are tested: random sampling based learning strategy (i.e., LAO with SAA), LAO with perfect predictions, and LAO with the seq2seq LSTM based predictions. During test, sample/prediction window size W_t takes the average lifespans of existing requests to be schedulled as the value. Fig. 8 shows the results of all compared algorithms.

Fig. 8 reveals that the LSTM based predictions improve the system revenue by 10% over the SAA strategy, although their superiority over CG is both preserved. Moreover, provided with the perfect predictions, LAO with perfect predictions



(a) (b) (c)

Fig. 6. Performance comparisons of LAO under different V with $\bar{c} = 400, W_t = 40$: a) Time-averaged revenue till t, b) Time-averaged link cost till t, c) Timed-average throughput till t.



(a) (b) (c) (d)

Fig. 7. Performance comparisons of LAO under different T_s with $V = 50, T = 40, W_t = 40, \bar{c} = 400$: a) Time-averaged revenue till t, b) Time-averaged throughput till t, c) Time-averaged link cost till t, d) Number of actively running services across t.

achieves a better improvement gain by $\sim 25\%$ over the SAA strategy. However, the overall improvements of both LSTM based predictions and perfect predictions over the SAA strategy are not significant. This demonstrates that in systems with independent and identifical distribution, provided with enough samples, simple sampling based strategy is already efficient to approximate the system. The temporal-independent system realizations used in aboved experiments are often viewed as highly skewed and unpredictable. Consequently, instead of an accurate predictions for a short time window, samples that can better approximate the PDF of system realizations play a more important role.

VII. Conclusion

This paper has highlighted the performance safety problem of slicing operations when the environment is time-varying and difficult to track with explicit models due to its complexity and heterogeneity. Based on the advanced deep learning and online optimization, we have accordingly developed a learning augmented optimization approach to learn a safe slicing solution from both historical records and real-time observations. We have proved that the feasibility of proposed solution with a sub-optimality, up to a constant additive factor. Finally, we have demonstrated up to 2.6× improvement in the simulation when compared with the referenced algorithms. This work is a good start to stimulate the further researches on the innovative use of learning augmented optimization approaches for more dynamic and stochastic networking problems.

Appendix A
Probabilistic Bounds

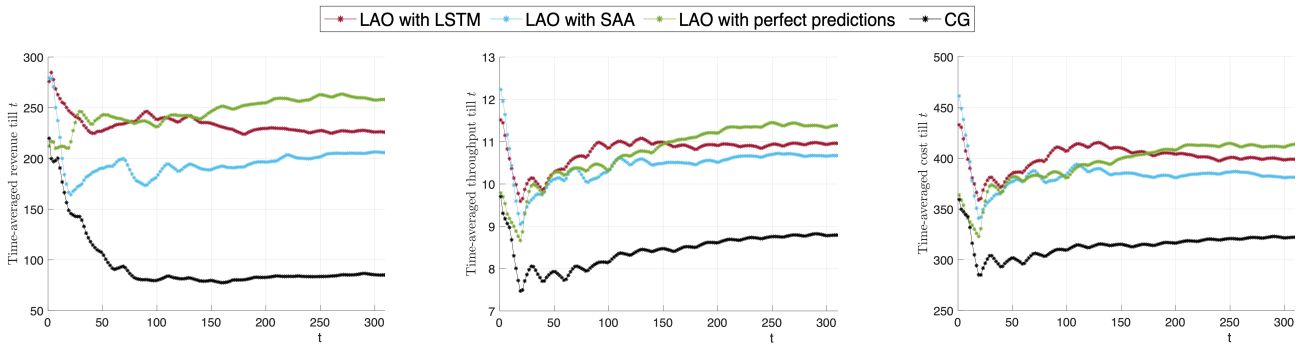
Proposition 1*:* Assume f is linear with respect to the ergodic process ω_t. Let π_+^* be the optimal solution of **T3S** achieved through some '*genius*' algorithm that knows the true system trajectory $\{\omega_t\}_{t=1}^{\infty}$ a-priori. Then, we have

$$\bar{f}\big(\{\pi_-^t\}_{t=0}^{W} \mid \check{\pi}_+\big) \geq \bar{f}\big(\pi_+^*, \{\pi_-^t\}_{t=0}^{\infty}\big) \geq \mathbb{E}\big[\bar{f}\big(\hat{\pi}_+^*, \{\pi_-^t\}_{t=0}^{W_t}\big)\big] \quad (35)$$

$$\mathbb{E}\big[\bar{f}\big(\hat{\pi}_+^*, \{\pi_-^t\}_{t=0}^{W_t+1}\big)\big] \geq \mathbb{E}\big[\bar{f}\big(\hat{\pi}_+^*, \{\pi_-^t\}_{t=0}^{W_t}\big)\big] \quad (36)$$

where $\check{\pi}_+$ is a fixed policy obtained by some approximation procedure, *e.g.*, solving **T4S** with W samples; $\hat{\pi}_+^*$ is the optimal solution of **T4S** under the samples $\{\hat{\omega}_t\}_{t=1}^{W_t}$.

Proof. First, under any feasible deployment policy, say $\check{\pi}_+$, the achievable value $\bar{f}\big(\{\pi_-^t\}_{t=0}^{\infty} \mid \check{\pi}_+\big)$ is clearly a *rigorous upper bound* on $\bar{f}\big(\pi_+^*, \{\pi_-^t\}_{t=0}^{\infty}\big)$. As presented in [40], $\bar{f}\big(\{\pi_-^t\}_{t=0}^{\infty} \mid \check{\pi}_+\big)$ can be estimated by solving the resultant continuous program under a large set of historical samples. Therefore, the left-side inequality in (35) can provides an upper bound estimation towards the true objective value. The right-side inequality of (35) shows a *probabilistic lower bound*, which can be calculated by solving a set of sampled instances of **T4S**. This is proved as follows.



(a) (b) (c)

Fig. 8. Performance comparisons with $V = 50, T_s = 40, \bar{c} = 400$: a) Time-averaged system revenue till t, b) Time-averaged system throughput till t, c) Time-averaged link cost till t.

According to the definition of (4), we can have

$$\bar{f}\big(\pi_+^*, \{\pi_-^t\}_{t=0}^{\infty}\big) = \min \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\big[f(\pi_+, \pi_-^\tau, \omega_\tau)\big] \tag{37}$$

$$= \min_{\pi} \mathbb{E}\big[\lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} f(\pi_+, \pi_-^\tau, \omega_\tau)\big] \tag{38}$$

$$\overset{(a)}{=} \min_{\pi} \frac{1}{W_t+1} \mathbb{E}\big[\sum_{t=0}^{W_t} f(\pi_+, \pi_-^t, \omega_t)\big] \tag{39}$$

$$\geq \mathbb{E}\big[\min_{\pi} \frac{1}{W_T+1} \sum_{t=0}^{W_t} f(\pi_+, \pi_-^t, \omega_t)\big] \tag{40}$$

$$\overset{(b)}{\geq} \mathbb{E}\big[\frac{1}{W_t+1} \sum_{t=0}^{W_t} f(\hat{\pi}_+^*, \pi_-^t, \omega_t)\big] \tag{41}$$

$$= \mathbb{E}[\bar{f}(\hat{\pi}_+^*, \{\pi_-^t\}_{t=0}^{W_t})] \tag{42}$$

where (a) follows with the ergodic theorem [47]; (b) follows since $\hat{\pi}_+^*$ is the optimal solution of the **T4S** model.

Next, we prove the monotonic feature of $\mathbb{E}\big[\bar{f}\big(\hat{\pi}_+^*, \{\pi_-^t\}_{t=0}^{W_t}\big)\big]$ with respect to W_t as follows:

$$\mathbb{E}\big[\bar{f}\big(\hat{\pi}_+^*, \{\pi_-^t\}_{t=0}^{W_t+1}\big)\big] = \mathbb{E}\big[\min_{\pi} \frac{1}{W_t+2} \sum_{t=0}^{W_t+1} f(\pi_+, \pi_-^t, \omega_t)\big] \tag{43}$$

$$= \mathbb{E}\big[\min_{\pi} \frac{1}{W_t+2} \sum_{t=0}^{W_t+1} \frac{1}{W_t+1} \sum_{\tau=0,\tau\neq t}^{W_t+1} f(\pi_+, \pi_-^\tau, \omega_\tau)\big] \tag{44}$$

$$\geq \frac{1}{W_t+2} \sum_{t=0}^{W_t+1} \mathbb{E}\big[\min_{\pi} \frac{1}{W_t+1} \sum_{\tau=0,\tau\neq t}^{W_t+1} f(\pi_+, \pi_-^\tau, \omega_\tau)\big] \tag{45}$$

$$= \frac{1}{W_t+2} \sum_{t=0}^{W_t+1} \mathbb{E}\big[\min_{\pi} \frac{1}{W_t+1} \sum_{\tau=0}^{W_t} f(\pi_+, \pi_-^\tau, \omega_\tau)\big] \tag{46}$$

$$\geq \mathbb{E}\big[\bar{f}\big(\hat{\pi}_+^*, \{\pi_-^t\}_{t=0}^{W_t}\big)\big] \tag{47}$$

Proof ends. ■

ACKNOWLEDGMENT

The work of Xiangle Cheng is partially supported by the China Scholarship Council for the study at the University of Exeter. This work is also partially supported by the UK EPSRC project (Grant No.: EP/R030863/1).

REFERENCES

[1] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network slicing & softwarization: A survey on principles, enabling technologies & solutions,", *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 2429–2453, third quarter, 2018.

[2] P. Rost *et al.*, "Network slicing to enable scalability and flexibility in 5G mobile networks," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 72—79, May 2017.

[3] 3GPP Release 15 specifications, http://www.3gpp.org/release-15, June, 2018.

[4] J. G. Herrera and J. F. Botero, "Resource allocation in NFV: A comprehensive survey," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 3, pp. 518–532, Sept. 2016.

[5] A. Osseiran *et al.*, "Scenarios for 5G mobile and wireless communications: the vision of the METIS project," *IEEE Commun. Mag.*, vol. 52, no. 5, pp. 26–35, May 2014.

[6] M. R. Akdeniz *et al.*, "Millimeter wave channel modeling and cellular capacity evaluation," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 6, pp. 1164–1179, Jun. 2014.

[7] R. Li *et. al.*, "Intelligent 5G: when cellular networks meet artificial intelligence," *IEEE Wireless Commun.*, vol. 24, no. 5, pp. 175–183, Oct. 2017.

[8] I. Benkacem, T. Taleb, M. Bagaa, and H. Flinck, "Optimal VNFs placement in CDN slicing over multi-cloud environment," in *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 616–627, Mar. 2018.

[9] Z. Liu, S. Wang, and Y. Wang, "Service function chaining resource allocation: A survey," *Cornell Univ. Library*, arXiv: 1608.00095, 2016. [Online]. Available: https://arxiv.org/pdf/1608.00095.pdf.

[10] T. Taleb, M. Bagaa, and A. Ksentini, "User mobility-aware virtual network function placement for virtual 5G network infrastructure," in *Proc. IEEE ICC*, Jun. 2015, pp. 3879—3884.

[11] Y. L. Lee, J. Loo, T. C. Chuah, and L. Wang, "Dynamic network slicing for multitenant heterogeneous cloud radio access networks," *IEEE Trans. Wireless Commu.*, vol. 17, no. 4, pp. 2146–2161, April 2018.

[12] M. Leconte *et. al.*, "A resource allocation framework for networkslicing," in *Proc. INFOCOM*, April 2018.

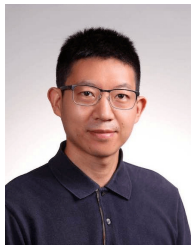
[13] Y. Li, L. T. X. Phan, and B. T. Loo, "Network functions virtualization with soft real-time guarantees," in *Proc. IEEE INFOCOM 2016*, Apr. 2016, pp. 1–9.

[14] S. Rajagopalan, D. Williams, H. Jamjoom, and A. Warfield, "Split/merge: System support for elastic execution in virtual middleboxes," in *Proc. 10th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2015, pp. 227–240.

[15] X. Cheng, Y. Wu, G. Min, and A. Y. Zomaya, "Network function virtualization in dynamic networks: a stochastic perspective," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2218–2232, Oct. 2018.

[16] A. Borodin and R. El-Yaniv, *Online Computation and Competitive Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 1998.
[17] Y. Jia *et al.*, "Online scaling of NFV service chains across geo-distributed datacenters," *IEEE/ACM Trans. Netw.*, vol. 26, no. 2, pp. 2008–2025, April 2018.
[18] G. Even, M. Medina, G. Schaffrath, and S. Schmid, "Competitive and deterministic embeddings of virtual networks," *Theoretical Computer Science*, vol. 496, pp. 184–194, July 2013.
[19] M. J. Neely, *Stochastic Network Optimization with Application to Communication and Queueing Systems*, San Rafael, CA, USA: Morgan and Calypool, 2010.
[20] Y. Gai, B. Krishnamachari, and R. Jain, "Combinatorial network optimization with unknown variables: multi-armed bandits with linear rewards and individual observations", *IEEE/ACM Trans. Netw.*, vol. 20, no. 5, pp. 1466–1478, Oct. 2012.
[21] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3604, Dec. 2016.
[22] L. Huang, M. Chen, and Y. Liu, "Learning-aided stochastic network optimization with state prediction," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1810–1820, Aug. 2018.
[23] M. J. Neely, "Distributed stochastic optimization via correlated scheduling," *IEEE/ACM Trans. Netw.*, vol. 24, no. 2, pp. 759—772, April 2016.
[24] M. Wang *et al.*, "Smart Exploration in HetNets: Minimizing Total Regret with mmWave," in *Proc. IEEE Int. Conf. Sens., Commun. Netw.*, June 2016.
[25] Y. Sun, S. Zhou, and J. Xu, "EMM: Energy-Aware Mobility Management for Mobile Edge Computing in Ultra Dense Networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2637–2646, Nov. 2017.
[26] S. Uryasev, P. M. Pardalos, *Stochastic Optimization: Algorithm and Applications*, Kluwer Academic: Norwell, MA, USA, 2001.
[27] A. Ben-Tal and A. Nemirovski, "Robust optimization – methodology and applications," *Math. Program.*, Ser. B, no. 92, pp. 453–480, 2002.
[28] S. Deb and P. Monogioudis, "Learning-based uplink interference management in 4G LTE cellular systems," *IEEE/ACM Trans. Netw.*, vol. 23, no. 2, pp. 398–411, April 2015.
[29] Q. Zheng *et al.*, "Delay-optimal virtualized radio resource scheduling in software-defined vehicular networks via stochastic learning," *IEEE Trans. Veh. Technol.*, vol. 65, no. 10, pp. 7857–7867, Oct. 2016.
[30] R. Wen *et al.*, "Robust network slicing in software-defined 5G networks," *Proc. GLOBECOM*, Dec. 2017, pp. 1–6.
[31] C. Jiang *et al.*, "Machine learning paradigms for next-generation wireless networks," *IEEE Wireless Commun.*, vol. 24, no. 2, pp. 98–105, April 2017.
[32] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer Networks," in *Proc. NIPS*, Dec. 2015.
[33] J. S. P. Roig, D. M. Gutierrez-Estevez, and D. Gunduz, "Management and orchestration of virtual network functions via deep reinforcement learning," *IEEE J. Sel. Areas Commun.*, in press, DOI: 10.1109/JSAC.2019.2959263, 2019.
[34] R. Mijumbi, *et al.*, "A connectionist approach to dynamic resource management for virtualised network functions," in *Proc. 12th IEEE/IFIP/ACM Int. Conf. Netw. Service Manag.*, 2016, pp. 1–9.
[35] A. S. Jacobs, "Artificial neural network model to predict affinity for virtual network functions," *IEEE/IFIP Network Operations and Management Symposium*, 2018.
[36] M. F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and O. C. M. B. Duarte, "Orchestrating virtualized network functions," in IEEE Trans. Netw. Service Manag., vol. 13, no. 4, pp. 725—739, Dec. 2016.
[37] G. Even, M. Rost, and S. Schmid, "An approximation algorithm for path computation and function placement in SDNs," In *Proc. SIROCCO*, July 2016, pp. 374–390.
[38] C. Liang and F. R. Yu, "Wireless network virtualization: a survey, some research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 358–380, first quarter 2015.
[39] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. NIPS*, 2014.
[40] H. M. Bidhandi and J. Patrick, "Accelerated sample average approximation method for two-stage stochastic programming with binary first-stage variables," *Appl. Math. Model.*, vol. 41, pp. 582–595, Jan. 2017.
[41] W.-K. Mak, D.P. Morton, and R.K. Wood, "Monte carlo bounding techniques for determining solution quality in stochastic programs," *Operations Research Letter*, vol. 24, no. 1, pp. 47–56, Feb. 1999.
[42] M. J. Neely, "A simple convergence time analysis of drift-plus penalty for stochastic optimization and convex programs," *arXiv preprint arXiv:1412.0791*, 2014. [Online]. Available: https://arxiv.org/pdf/1412.0791.pdf
[43] G. L. Stuber, *Principles of Mobile Communication*, Second Edition, Kluwer Academic Publishers, 2001.
[44] 5GMF White Paper, 5G Mobile Communications Systems for 2020 and beyond, 2016, [Online]. Available: http://5gmf.jp/wp/wp-content/uploads/2016/09/5GMF WP101 All.pdf.
[45] Y. Dinitz, N. Garg, M. X. Goemans, "On the single source unsplittable flow problem," *Proceedings of the 39th Symposium on the Foundations of Computer Science*, Palo Alto, CA, 1998, pp. 290–299.
[46] Y. Zhu and M. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *Proceedings of IEEE INFOCOM*, 2006.
[47] P. Z. Peebles, *Probability, Random Variables and Random Signal Principles*. New York, NY, USA: McGraw-Hill, 1993.

Xiangle Cheng received the M.Sc. degree in communication and information system from Southwest Jiaotong University, Chengdu, China in 2015. He is currently a Ph.D. candidate in Computer Science at the University of Exeter, UK. His research interests include 5G SDN/NFV, Network AI, Stochastic & Neural Combinatorial Optimization, Intelligent Wireless Networks and Mobile Computing, and Dynamic System Modelling and Performance Optimization.



Yulei Wu is currently a Senior Lecturer with the Department of Computer Science, College of Engineering, Mathematics and Physical Sciences, University of Exeter, U.K. He received the B.Sc. degree (1st Class Hons.) in Computer Science and the Ph.D. degree in Computing and Mathematics from the University of Bradford, U.K., in 2006 and 2010, respectively. His expertise is on networking and his main research interests include autonomous networks, intelligent networking technologies, edge computing, network slicing and softwarization, SDN/NFV, green networking, wireless networks, network security and privacy, and analytical modelling and optimization. His research has been supported by Engineering and Physical Sciences Research Council (EPSRC) of U.K., National Natural Science Foundation of China (NSFC), University's Innovation Platform and industry. He is an Editor of IEEE Transactions on Network and Service Management, Computer Networks (Elsevier) and IEEE Access. He is a Senior Member of the IEEE, and a Fellow of the HEA (Higher Education Academy).

Geyong Min is a Professor of High Performance Computing and Networking in the Department of Computer Science within the College of Engineering, Mathematics and Physical Sciences at the University of Exeter, United Kingdom. He received the PhD degree in Computing Science from the University of Glasgow, United Kingdom, in 2003, and the B.Sc. degree in Computer Science from Huazhong University of Science and Technology, China, in 1995. His research interests include Future Internet, Computer Networks, Wireless Communications, Multimedia Systems, Information Security, High Performance Computing, Ubiquitous Computing, Modelling and Performance Engineering.

Albert Y. Zomaya is the *Chair Professor of High Performance Computing & Networking* in the School of Information Technologies, University of Sydney, and he also serves as the Director of the Centre for Distributed and High Performance Computing. Professor Zomaya published more than 550 scientific papers and articles and is author, co-author or editor of more than 20 books. He is the Founding Editor in Chief of the *IEEE Transactions on Sustainable Computing* and serves as an associate editor for more than 20 leading journals. Professor Zomaya served as an Editor in Chief for the *IEEE Transactions on Computers* (2011-2014).

Professor Zomaya is the recipient of the *IEEE Technical Committee on Parallel Processing Outstanding Service Award* (2011), the *IEEE Technical Committee on Scalable Computing Medal for Excellence in Scalable Computing* (2011), and the *IEEE Computer Society Technical Achievement Award* (2014), and the *ACM MSWIM Reginald A. Fessenden Award* (2017). He is a Chartered Engineer, a Fellow of AAAS, IEEE, and IET. Professor Zomaya's research interests are in the areas of parallel and distributed computing and complex systems.

Xuming Fang received the B.E. degree in electrical engineering in 1984, the M.E. degree in computer engineering in 1989, and the Ph.D. degree in communication engineering in 1999 from Southwest Jiaotong University, Chengdu, China. He was a Faculty Member with the Department of Electrical Engineering, Tongji University, Shanghai, China, in 1984-1985. He then joined the School of Information Science and Technology, Southwest Jiaotong University, Chengdu, where he has been a Professor since 2001. He held visiting positions with the Institute of Railway Technology, Technical University at Berlin, Berlin, Germany, in 1998 - 1999, and with the Center for Advanced Telecommunication Systems and Services, University of Texas at Dallas, Richardson, in 2000 - 2001. He has published more than 200 high-quality research papers in journals and conference publications. He has authored or coauthored five books or textbooks. His research interests include wireless resource management, mmWave communications, and wireless communications for high speed railway. Dr. Fang is the editor of several journals including IEEE Transactions on Vehicular Technology.