Leveraging Multiplexing Gain in Network Slice Bundles

Qian Xu, Xiang Yan, Kui Wu, Jianping Wang, Kejie Lu, *Senior Member, IEEE*, and Weiwei Wu

***Abstract*—In this paper, we propose strategies and develop solutions for a network service provider (NSP) to cost-effectively provision and manage a large number of network slices. Specifically, we propose a novel framework, namely, network slice bundling, in which (1) an NSP can allocate resources and create multiple network slice bundles in advance, (2) a network slice request can be quickly instantiated in the bundle that supports its service requirements, and (3) network slices in the same bundle can share the resources and achieve a multiplexing gain by leveraging the stochastic behaviors of resource usage. Within this framework, we focus on a core problem, which is how to leverage the multiplexing gain to maximize the utility by optimally assigning multiple network slices to a set of pre-defined bundles. We formulate an optimization problem and theoretically analyze the irregularity of constraints and the difficulty of the problem. We develop a novel reinforcement learning (RL) based slice assignment solution. Finally, we conduct extensive data-driven simulation experiments. The numerical results confirm that the proposed solution can efficiently solve the network slice assignment problem and achieve significantly higher utility than the best baseline algorithm.**

***Index Terms*—Multiplexing gain, network slice bundles, quality-of-service, reinforcement learning.**

I. Introduction

IT is expected that mobile data traffic will increase sevenfold between 2017 and 2022 with a compound annual growth rate of 47% [1]. Coupled with booming Internet traffic are diverse services demanding different quality-of-service (QoS) requirements: from delay-tolerant content delivery applications to delay-sensitive real-time cyber-physical system (CPS) and virtual/augmented/mixed reality (VR/AR/MR) applications [2]. To address such a challenge, network slicing [3], [4] has been proposed to facilitate sustainable business models in future networking markets [5]. With network slicing, a logic network (a.k.a., network slice) can be created through a set of virtual network functions (VNFs) with sufficient resources to provide performance guarantee on shared physical infrastructure.

The advantages of network slicing in enabling various services of diverse QoS requirements have been widely recognized by 5 G industries. A joint study conducted by BT and Ericsson has demonstrated that network slicing in Internet of Things (IoT) services can bring an overall 150% increased economic benefit [6]. The efficiency and revenue achieved by network slicing make it a fundamental block to leverage the potential of 5 G. Still, there are several challenges to be tackled in order to achieve network slice as a service in reality.

Firstly, although it is desirable to create each network slice according to its QoS requirements, it is very difficult to quickly provision a network slice on-demand because the Network service provider (NSP) must provision and orchestrate many resources, including networking, computing, and functionality [7]. To address this challenge, a recent IETF draft [7] recommends a practical provisioning scheme, in which an NSP can define some network slice templates in advance, and a network slice tenant can choose the best template that fits its QoS requirements. Nevertheless, such a scheme may lead to over-provisioning of resources for each network slice, which is expensive and not scalable.

Secondly, the QoS guarantee for network slices is still a major challenge, especially when multiple network slices are operating using the same infrastructure resources and the traffic of each network slice is fluctuating. To this end, most previous work proposed adaptive solutions for the dynamic traffic in a network slice [8]–[10]. However, the reconfiguration of the network slice may lead to instability, service disruptions, and performance degradation [11].

To efficiently create multiple network slices on-demand, we propose a novel network slice provisioning model, namely, network slice bundling.[1] In this framework, an NSP can allocate and orchestrate resources in advance, which can help

Manuscript received March 5, 2020; revised August 18, 2020 and October 1, 2020; accepted October 10, 2020. Date of publication October 15, 2020; date of current version March 17, 2021. The work was supported in part by a grant from Hong Kong Research Grant Council under GRF project 11216618, Natural Sciences and Engineering Research Council of Canada (NSERC) under RGPIN-2018-03896, National Science Foundation under Grant CNS-1730325, and a grant from ZTE Corporation. Recommended for acceptance by Prof. Kun Yang *(Corresponding author: Jianping Wang.)*

Qian Xu is with the Institute of Cyberspace Security and College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023, China (e-mail: qianxu@zjut.edu.cn).

Jianping Wang is with the Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong 999077, China (e-mail: jianwang@cityu.edu.hk).

Xiang Yan is with the Department of Computer Science, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: yxghost@sjtu.edu.cn).

Kui Wu is with the Department of Computer Science, University of Victoria, Victoria, BC V8P 5C2, Canada (e-mail: wkui@uvic.ca).

Kejie Lu is with the Department of Computer Science and Engineering, University of Puerto Rico at Mayagüez, Mayagüez, PR 00680, Puerto Rico (e-mail: kejie.lu@upr.edu).

Weiwei Wu is with the Department of Computer Science, Southeast University, Nanjing 210000, China (e-mail: weiweiwu@seu.edu.cn).

Digital Object Identifier 10.1109/TNSE.2020.3031347

[1]Note that the definition of network slice bundle is different from that in [5] where a bundle is to group a set of vertical services.

NSP to ease the management and orchestration of resources for network slices. Although a similar idea has been adopted in the ready-made network slice in [7], our framework is novel in that each network slice bundle can accommodate multiple network slices that cover similar areas and have similar QoS requirements. In this manner, a network slice request can be assigned to an appropriate bundle that can guarantee its QoS requirements. Moreover, multiple network slices in the same bundle can share the resources and thus can further exploit the *multiplexing gain*, i.e., given the stochastic nature of traffic from different network slices, multiplexing makes it possible for two or more combined network slices to consume less resources than the summation of resources needed individually.

In this paper, as the first step to validate the proposed bundling framework, we focus on a fundamental issue, i.e., how to maximize the utility of NSP by optimally assigning multiple network slices to a set of pre-defined bundles. Specifically, we consider that a network slice request can be assigned to one of multiple bundles that provide exact or higher QoS guarantees than the requirement. We also assume that each network slice is associated with a stochastic model for traffic arrival in the time domain.[2] Moreover, we consider a practical but tough scenario that the arrival models for any two network slices are correlated.

Although there are many existing studies on different aspects of the multiplexing gain in various communication and networking systems [12]–[14], the multiplexing gain problem *in the context of network slice bundle* is unique due to the following main reasons:

1) A network slice request may be admitted into a bundle that supports a higher QoS level than needed. Therefore, the aggregation may upgrade the QoS of some requests in the bundle, which may result in using more resources than multiplexing gain. Thus the framework of network slice bundle, if not managed carefully, would naturally lead to resource waste.
2) A network slice bundle is pre-configured with allocated total resources. In this setting, the problem of admitting a network slice request seems similar to a *stochastic* version of multiple knapsack problem, but it has a unique feature that voids all previous solutions (Section IV). To be specific, when two network slice requests, say A and B, are admitted into a bundle, the total resources needed to guarantee the QoS of both requests is not simply the resources needed to guarantee the QoS of A plus the resources needed to guarantee the QoS of B. The actually needed resources may be smaller or even larger than the sum, depending on the statistical profiles of A and B.
3) There is no unified analytical model that can be directly applied to estimate the total resources when a network slice bundle accepts two or more network slice requests with correlated traffic.

[2]Since the problem is already very complicated, we will investigate the time domain models in this study.

To address the above issues, we first formulate a general optimal slice assignment problem, namely G-OSA, that aims to maximize the total utility of NSP. Based on the formulation, we further define an optimization problem, namely, f-OSA, when the traffic of network slices are characterized by correlated fractional Brownian motion (fBm) models. Next, we show that f-OSA is an NP-hard combinatorial optimization problem that has irregular constraints, which are more complex and cannot be solved with existing combinatorial optimization algorithms. For instance, existing schemes such as greedy algorithms, local search, simulated annealing, etc. are more adapted to particular structures of problems, e.g., Euclidean TSPs and knapsack, and usually need extensive parameter tuning and domain expertise [15]. Inspired by recent learning-based methods for solving NP-hard problems [15], [16], we propose a novel reinforcement learning based approach that can automatically search for good results based on rewards (Section V).

In a short summary, the concept of network slice bundles is proposed to simplify the management of network slicing. The utility of NSaaS is maximized by leveraging multiplexing gain among network slice requests which is highly related to the network slice assignment method. The reinforcement learning approach is proposed to solve optimization problem. We tackle these challenges and make the following contributions:

- We formulate an optimization problem for assigning network slice requests into appropriate bundles so that the total utility of NSP can be maximized. This general model considers the constraints on QoS of each network slice request and the capacity of each bundle. In addition, it presents a general framework, within which a flexible traffic model, fractional Brownian motion (fBm), is adopted to showcase the calculation of total resources needed for guaranteeing the QoS of all requests in a bundle.
- We analyze the special difficulty in the fBm-based optimal slice assignment problem that does not allow a polynomial-time approximation solution. We then propose a reinforcement learning (RL) approach, RL-Assign, that uses Policy Gradient to tackle this difficulty.
- With extensive trace-driven simulation, we demonstrate the superior performance of RL-Assign over four baselines: First-Fit, Best-Fit,a branch-and-bound method MTM [17], and a constraint integer programming approach CIP [18].

The rest of the paper is organized as follows. We introduce the network slice bundling framework in Section II. We formulate an optimization problem for assigning network slice requests into network slice bundles in Section III. The difficulty in the optimal slice assignment problem is addressed in Section IV, and reinforcement learning approach, RL-Assign, is presented in Section V. Section VI shows the performance evaluation. We discuss related work in Section VII. Section VIII concludes the paper.

II. A Network Slice Bundling Framework

In this section, we first present the background of network slice provisioning. We then introduce the network slice

bundling framework. Finally, we use an example to demonstrate the applicability of the framework and the multiplexing gain within the proposed framework.

A. An Overview of Network Slice Provisioning

To support potential applications with diverse needs, a network slice must be provisioned with different types of resources [7]: networking, computing, and functionality. To provision resources for network slices, IETF [7] states that network slices can be created as follows:

1) A ready-made network slice is designed by the NSP and it can be created in advance. A network slice tenant can select a ready-made network slice that best supports its requirements.
2) A custom-made network slice is created by the NSP according to the requirements specified by the tenant.
3) A semi-custom-made network slice is designed by the NSP in advance but can be tuned according to the tenant's needs.

B. The Network Slice Bundling Framework

From the previous introduction, we first note that a network slice needs various resources and thus it can be very complicated to provision a new network slice according to the requirements of a tenant. Therefore, to simplify network slice provisioning, it has been recommended that an NSP can define the details of multiple network slices in advance, and allow tenants to select the best ones that fit their needs. Such a scheme is practical because an NSP can design appropriate network slices based on the analysis of historical network slice requests and the prediction of demands for future network slices.

Although the aforementioned scheme can quickly instantiate a network slice for a tenant, there are some major issues. First, from the perspective of a tenant, a pre-defined network slice may occupy more resources than it needs so the tenant may pay more to the NSP. Secondly, from the perspective of the NSP, it is unclear how to optimally utilize the resources to exploit the potential multiplexing gain. For example, when one network slice has a high volume of traffic, another network slice may have low traffic volume. In such a case, the total traffic volume may be much lower than the sum of the peak traffic volume of these two network slices.

To address the above issues, we can exploit two major opportunities. First, if an NSP can define network slices in advance, then there may be multiple tenants who are interested in creating the same type of network slices. As shown in Fig. 1, multiple tenants may want to provision network slices for industrial Internet-of-things (IIoT) that cover the same industrial park. Secondly, network slices that share common features, e.g., the same types of resources organized with a similar workflow, may have some different service requirements. For example, in Fig. 1, several network slices for video services may cover the same road area in a city for different purposes, e.g., surveillance and autonomous driving, and thus require different delay guarantees.

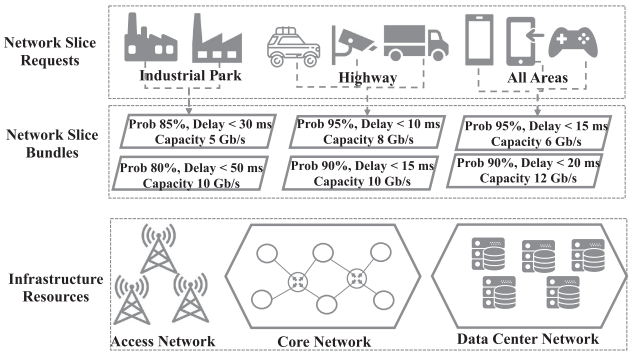


Fig. 1. Network Slice Bundling Framework.

Based on the analysis above, we propose a novel network slice bundling framework. Specifically, in this framework, an NSP can create multiple network slice bundles in advance, each of which has sufficient resources to guarantee one type of service level on a set of functionalities and can accommodate multiple network slices. Network slices that use same type of functionalities can be grouped into the same bundle. Consequently, when a tenant requests to create a network slice with certain service requirements, the NSP can choose the best bundle and quickly instantiate the network slice using the available resources in the bundle. Finally, during the operation stage, all network slices that are in the same bundle can share the resources so that the multiplexing gain can be leveraged to improve the utility of NSP.

To implement network slice bundling, one key issue to be solved is to assign network slice requests to its most appropriate network slice bundle so that the utility of NSP is maximized. In the rest of this paper, we will focus on such a problem.

C. An Example Case Study

As an example of our framework, Fig. 1 shows that an NSP has provisioned six network slice bundles: two for the industrial park areas, two for the highway areas, and two for all areas. Moreover, each bundle is associated with a capacity of traffic volume, a delay requirement, and a probability to guarantee the service level.

Specifically, for the highway areas, the first bundle has a capacity of 8 Gb/s and can guarantee that 95% of packets have less than 10 ms end-to-end delay, while the second bundle has a capacity of 10 Gb/s and can guarantee that 90% of packets have less than 15 ms end-to-end delay. For such a setting, we assume that there are 6 network slice requests waiting to be assigned into these two bundles, and each request has a peak arrival rate and a delay requirement, as shown in Fig. 2.

To assign these requests to the two bundles, we may have two naïve allocation methods: *First-Fit* and *Best-Fit*. Note that for fair comparison of different methods in the same setting, we do not consider adaptive resource re-allocation between bundles.

In the *First-Fit* method, network slice requests are assigned to the bundle which firstly meets their QoS requirements. We begin to assign the requests to bundle 1 until it cannot hold

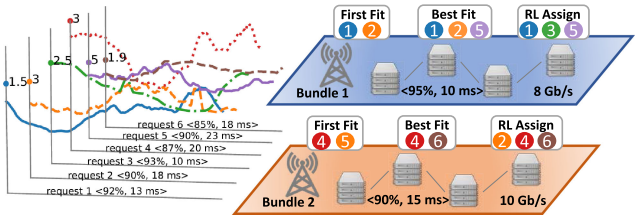


Fig. 2. An example of assigning network slice requests to bundles.

any more requests. Then we assign the requests to bundle 2 until it cannot hold any more requests. The assignment result is shown in the "First-Fit" part in Fig. 2. As a result, network slice requests 1 and 2 are allocated to bundle 1, and network slice requests 4 and 5 are allocated to bundle 2.

In the *Best-Fit* method, network slice requests are assigned to the bundle of the least available resources which meets their requirements. This heuristic is aimed at maximizing a bundle's resource utilization. As a result, network slice requests 1, 2, and 5 are allocated to bundle 1, and network slice requests 4 and 6 are allocated to bundle 2.

Using our reinforcement learning (RL) based method developed in this paper, we obtain the result shown in the "*RL Assign*" part of Fig. 2. Obviously, the bundles can host more requests using our strategy. Though the sum of maximal required capacity of network slice requests exceeds the capacity of the network slice bundle, the performance of each network slice requests can still be guaranteed due to the multiplexing gain. Intuitively, the bursts of traffic arrivals are smoothed out by aggregating negatively correlated traffic patterns, which reduces the total capacity required to guarantee QoS. In this way, resource utilization is always high without the cost of dynamic slice resizing.

III. System Model and Problem Formulation

In this section, we first describe the system model, then formulate the general network slice request assignment problem, which assigns each network slice request to an appropriate bundle to maximize the total utility. At last, we present the model with fractional Brownian motion traffic model to showcase the resource needed by requests in a bundle.

A. Network Slice Bundle

We assume that NSaaS service provider divides all its available resources into a set of bundles, where each bundle offers a set of functionalities and provides a certain amount of resources to meet a given QoS requirement, including any particular networking/computing/functional resources relevant to the functionalities. The consideration for this practical assumption is that each bundle only handles a set of functionalities, thus, functionalities can be pre-deployed which can ease the management overhead of NSP. Moreover, given the resources allocated to each bundle and its offered set of functionalities, the QoS that each bundle can guarantee can be determined.

The same set of functionalities may be offered in multiple bundles so that applications requiring the same set of functionalities but with different QoS requirements can be accommodated by different bundles. The consideration for this assumption is to maximize the utility of NSP. Applications with the same set of functionalities may have different QoS requirements, e.g., some may be very delay sensitive, while some are not. If all of them are admitted to one network slice bundle, the ones with low QoS requirements may enjoy high QoS guarantee while paying according to their required QoS, which will reduce the utility of NSP.

Under such network slice bundle creation assumptions, applications with different sets of functionalities, will be accommodated to different network slice bundles, while applications with the same set of functionalities may or may not be accommodated to the same network service bundle depending on whether aggregating them into one bundle will improve the utility of the NSP.

Given a set of network slice bundles and network slice requests, the general problem of assigning network slice requests to the appropriate bundles to maximize the utility of NSP thus can be decomposed to assign network slice requests with the same set of functionalities to a set of network slice bundles offering the corresponding set of functionalities independently. Therefore, in the rest discussion, we only consider how to assign a set of network slice requests with the same set of functionalities to a set of bundles offering the same set of functionalities.

We use $\{C_1, \ldots, C_m\}$ to denote the capacity of the m bundles, respectively, and use $\{< p^1, T^1 >, \ldots, < p^m, T^m >\}$ to denote the QoS guaranteed by the m bundles, respectively. The capacity is the number of packets which can be processed per second on the network slice bundle, which is pre-determined by the resources allocated to this bundle. The QoS $< p^i, T^i >$ means that $p^i(\%)$ packets passing through a network slice from bundle i have less than T^i end-to-end delay.

B. Service Requests

Assume that there are n network slice requests, denoted as $\{< A_1, p_1, T_1, w_1 >, \ldots, < A_n, p_n, T_n, w_n >\}$, respectively. Here, A_j denotes the traffic arrivals and $< p_j, T_j >$ the QoS requirement of network slice request j. By accepting the service request j, the NSP gains a utility w_j.

We assume that both the traffic pattern from a network slice request and the correlation between traffic patterns in different network slice requests can be estimated [19]. This assumption is widely accepted in traffic prediction work where the prediction models are based on spatial and temporal patterns captured from real-world observations [20]–[22]. As shown in Fig. 3, the real-world traffic arrivals for GEANT-CPS service at Krakow, Poland and Akai-Cluster service at Cambridge US have a similar pattern and are highly correlated, e.g., due to the time difference, traffic amount at US area is high when the amount at Poland area is low. Note that these preliminary analysis results of daily traffic pattern and pattern diversity indeed exist among all service requests. Our model fully utilizes the temporal difference of traffic arrivals of service requests.

We then use these two requests to illustrate the multiplexing gain of bundling them together. Assume that the QoS

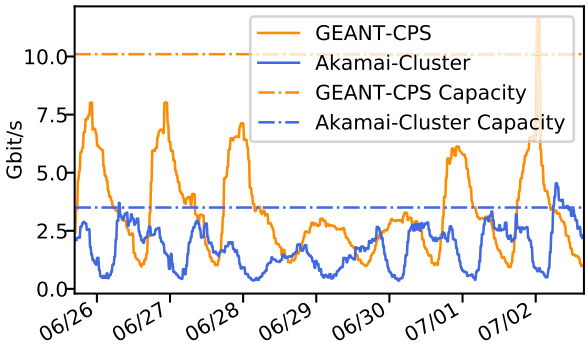


Fig. 3. One-week average out traffic with GEANT-CPS at Krakow, Poland and Akamai-Cluster at Cambridge, US.

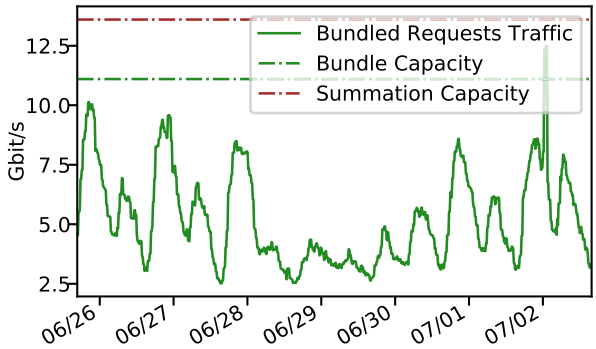


Fig. 4. Multiplexing gain of bundle GEANT-CPS request and Akamai-Cluster request together.

requirement of GEANT-CPS request is that 95% of packets have less than 20 ms end-to-end delay while the QoS requirement of Akamai-Cluster request is that 90% packets have less than 30 ms end-to-end delay. The capacity required by each individual network slice request is shown in Fig. 3 as 10.1 Gbit/s and 3.5 Gbit/s, respectively. As shown in Fig. 4, when we bundle these two requests in a network slice bundle whose QoS is set as $< 95\%, 20ms >$, the required capacity to guarantee the performance of the bundled request is 11.1 Gbit/s, which saves 2.5 Gbit/s capacity for NSP. Obviously, when traffic of requests are negatively correlated, the multiplexing gain achieved by bundling requests together increases the utility of NSP while QoS of some requests may be upgraded.

Based on the above assumption of system configuration and service requests, the problem that an NSP faces is: how to assign the network slice requests into different bundles such that the total utility is maximized subject to both capacity and QoS constraints? We call this problem optimal slice assignment (OSA) problem. In what follows, we first provide a general formulation of OSA, and then investigate a detailed formulation under a general Internet traffic model.

C. The General Form of Optimal Slice Assignment Problem

To meet the QoS constraint, a network slice request can only be assigned to a bundle which offers higher QoS than what the request demands. In other words, network slice request j can only be assigned to the bundle i when $p_j \leq p^i$ and $T_j \geq T^i$.

To meet the capacity constraint, the total amount of resources required by the network slices in a bundle should be below the capacity of the bundle. For a network slice request j allocated to bundle i, its allocated capacity is calculated as $f(A_j, p^i, T^i)$, which depends on specific traffic pattern and will be discussed in Section III-D. Since the QoS that a network slice will receive is at least what is requested or higher than that when it is assigned to a network slice bundle, the required capacity is naturally increased, i.e., $f(A_j, p^i, T^i) \geq f(A_j, p_j, T_j)$. In other words, NSP costs more resources in order to get utility w_j. However, due to the correlated traffic among different requests, the capacity required for the bundled requests may be smaller than the summation of capacities individually required, so-called multiplexing gain. Let x_{ij} be a binary variable which equals 1 if request j is assigned into bundle i, and equals 0, otherwise. Multiplexing gain should be a function of the aggregated traffic and the QoS defined by bundle i. The total multiplexing gain for network slice bundles can be derived as:

$$Gain = \sum_{j=1}^{n} f(A_j, p_j, T_j) - \sum_{i=1}^{m} f\left(\sum_{j=1}^{n} A_j x_{ij}, p^i, T^i\right) \quad (1)$$

Only when $Gain > 0$, the NSP can benefit from multiplexing gain since less amount of resources is consumed to achieve utility $\sum_{j=1}^{n} w_j$. In addition, minimizing the amount of resources for assigning requests to bundles equals to maximizing multiplexing gain since the summation of resources for individual request is fixed.

Consider our assumption that network slice bundles are predeployed with determined resources, we formulate the general form of OSA problem as follows:

Problem 1 (General Optimal Slice Assignment (G-OSA)):

$$\begin{aligned} \max \ & \sum_{i=1}^{m}\sum_{j=1}^{n} w_j x_{ij} \\ \text{s.t. } & f(\sum_{j=1}^{n} A_j x_{ij}, p^i, T^i) \leq C^i, \quad \forall i \in [1,m] \\ & x_{ij} = 0 \text{ if } \ p_j \geq p^i, T_j \leq T^i, \forall i \in [1,m], \forall j \in [1,n] \\ & \sum_{i=1}^{m} x_{ij} \leq 1, \quad \forall j \in [1,n] \\ & x_{ij} \in \{0,1\}, \quad \forall i \in [1,m], \forall j \in [1,n] \end{aligned} \quad (2)$$

The constraint $f(\sum_{j=i}^{n} A_j x_{ij}, p^i, T^i) \leq C^i$ ensures capacity constraint for each bundle. The constraint $p_j \leq p^i, T_j \geq T^i$ ensures that the QoS of each network slice request is guaranteed, and the last constraint $x_{ij} \in \{0,1\}$ ensures that each request is assigned to at most one bundle.

The object of G-OSA problem is to maximize the utilities of accepting network slice requests into network slice bundles with limited resources. The optimal solution is achieved when the resources required for the bundled requests are the minimum. In other words, the max utility of NSaaS is achieved by leveraging the multiplexing gain to make full use of resources.

D. Optimal Slice Assignment With Fractional Brownian Motion (fBm)

G-OSA is not solvable until the concrete form of function f is given. To derive f, we need to (a) profile traffic arrivals A_j and (b) estimate the required capacity based on the traffic arrivals and the QoS requirement.

1) Traffic Profile: While Poisson traffic has been broadly used in telecommunication networks, Internet traffic generally cannot be modelled as Poisson traffic [23]. Due to this consideration, we adopt fractional Brownian Motion (fBm) traffic model [23], [24], which is considered as a general model of Internet traffic and can well capture both long-range dependence as well as self-similarity of network traffic [23]–[25]. To make the paper self-contained, we introduce fBm traffic model as follows.

Definition 1 (fractional Brownian motion (fBm)): Let $A(0,\tau)$ be the amount of traffic that arrives during the interval $(0,\tau)$. An arrival process $\{A(0,\tau):\tau\geq 0\}$ is said to be fractional Brownian motion with Hurst parameter H if

1) $\{A(0,\tau):\tau\geq 0\}$ has stationary increments;
2) $\forall\tau\geq 0$, $\{A(0,\tau):\tau\geq 0\}$ is normally distributed with mean μ and variance $\sigma^2\tau^{2H}$;
3) Denote the increments of $A(0,\tau)$ as $Z(k)=A(0,k+1)-A(0,k), k\geq 0$. The autocorrelation function of $Z(k)$, $\rho_Z(k)$, satisfies

$$\rho_Z(k)=\frac{1}{2}\{|k+1|^{2H}+|k-1|^{2H}-2\ k^{2H}\}$$

It has been shown [26] that when $H>\frac{1}{2}$, the increments of $A(0,\tau)$ are long-range dependent. In the rest of the paper, we assume that traffic arrivals in each network slice follow fBm.

2) Capacity Estimation: Due to the stochastic nature of fBm and the rich dynamics that fBm can model [24]–[26], it is generally difficult to analyze the delay of a queueing system with fBm input. To handle the difficulty, we resort to the theory of effective bandwidth [27] to estimate the required capacity for a network slice with fBm traffic.

Effective bandwidth is the theory developed to provide a measurement of resource usage, taking account of the statistical characteristics and QoS requirements for diverse traffic patterns. Combined with the large deviation theory [28], effective bandwidth can be used to approximate the probability of overflow given the buffer size and capacity in a finite system [26], [29]. On the other side, the required capacity can be derived once other factors have been determined. In particular, effective bandwidth function can be estimated statistically with real traffic arrival described in discrete time, where the statistic characteristics like correlation can be considered. While a lot of research work studied the relationship between service capacity and traffic process, they usually assume that two traffic flows are independent. As pointed out in Section III-B, traffic arrivals from the network slices may be correlated. As such, their correlation should be considered since the total capacity is allocated to meet the QoS of the aggregated traffic.

The effective bandwidth of traffic $A(0,\tau)$ is defined as:

Definition 2: Effective Bandwidth

$$\alpha_A(\theta,\tau)=\frac{1}{\theta\tau}\log E[e^{\theta A(0,\tau)}] \tag{3}$$

Denote the rate function of overflow as I. The large deviations rate function I for overflow in a discrete time queue with capacity C and buffer size B for input process $A(0,\tau)$ can be calculated as [26]:

$$I=\inf_{\tau\geq 0}\sup_{\theta\geq 0}\theta(B+C\tau)-\theta\tau\alpha_A(\theta,\tau) \tag{4}$$

where the optimal θ^*, τ^* in the above equation are referred to as the *critical point* of the system. At the critical point, the traffic flow is regarded as a constant traffic stream of rate $\alpha_A(\theta^*,\tau^*)$ equivalently, and I is used to approximate (i.e., the most likely value) $-\log P(\text{overflow})$, where $P(\text{overflow})$ denotes the probability of overflow [26].

Suppose that the traffic arrivals follow fBm with mean arrival rate μ, variance σ^2, and Hurst parameter H. The effective bandwidth of fractional Brownian motion is:

$$\alpha_A(\theta,\tau)=\mu+\frac{\theta\sigma^2}{2}\tau^{2H-1}. \tag{5}$$

The critic points of fractional Brownian motion are

$$\tau^*=\frac{B}{C-\mu}\frac{H}{1-H} \tag{6}$$

$$\theta^*=\frac{B+(C-\mu)\tau^*}{\sigma^2(\tau^*)^{2H}} \tag{7}$$

For simplicity, denote $P(\text{overflow})=p$. The capacity C can be derived as:

$$C=\frac{2H\sqrt{2(1-H)^2\sigma^2(-log(p))BH}}{2H\sqrt{B^2}(1-H)}+\mu \tag{8}$$

With the above equation, we can derive the capacity required for each network slice request. When multiple requests with common Hurst parameter[3] H are aggregated, the aggregated traffic also follows fractional Brownian motion where $\mu_{agg}=\sum_{j=1}^{n}\mu_j$ and $\sigma^2_{agg}=\sum_{j=1}^{n}\sigma_j^2+2\sum_{j\neq k}cov_{jk}$. To ensure the QoS, the buffer size of each network slice bundle is approximated as $B_i=C^iT^i$. Thus, the first constraint function in G-OSA, which is to meet the total capacity constraint of each bundle and ensures the performance of all

[3]We ignore the case where traffic arrivals with different Hurst parameters are aggregated. This omission is reasonable because a network slice is for one specific service type and the traffic for the same service type is considered to have similar traffic pattern.

requests in the bundle, can be formulated as:

$$\sum_{j=i}^{n} \mu_j x_{ij} + \eta_i \left(\sum_{j=1}^{n} x_{ij}\sigma_j^2 + 2 \sum_{j \neq k} x_{ij}x_{ik} cov(A_j, A_k) \right)^{\frac{1}{2H}} \leq C^i \quad \forall i \in [1, m] \tag{9}$$

where $\eta_i = \frac{2H\sqrt{2(1-H)^2(-log(p^i))}B_i H}{2H\sqrt{B_i^2}(1-H)}$, determined by the prior knowledge of the network slice bundle configurations and the traffic type.

To simplify the above equation, we rewrite it in the vector form as follows:

$$\boldsymbol{x_i}^T \boldsymbol{U} + \eta_i (\boldsymbol{x_i}^T V \boldsymbol{x_i})^{\frac{1}{2H}} \leq C^i \tag{10}$$

where $\boldsymbol{x_i}^T = [x_{i1}, \ldots, x_{in}]$, $\boldsymbol{U} = [\mu_1, \ldots, \mu_n]^T$, and V is symmetric matrix defined as follows

$$V = \begin{bmatrix} Var(A_1) & \cdots & cov(A_1, A_n) \\ \vdots & \ddots & \vdots \\ cov(A_n, A_1) & \cdots & Var(A_n) \end{bmatrix} \tag{11}$$

*3) **OSA With Fbm**:* After we have analyzed the traffic arrivals and derived the capacity needed to meet QoS, Problem 1 can be recast in the following more concrete form:

Problem 2: **fBm-based Optimal Slice Assignment (f-OSA)**:

$$\max \sum_{i=1}^{m} \sum_{j=1}^{n} w_j x_{ij} \tag{12}$$

$$\text{s.t. } \boldsymbol{x_i}^T U + \eta_i (\boldsymbol{x_i}^T V \boldsymbol{x_i})^{\frac{1}{2H}} \leq C^i, \forall i \in [1, m] \tag{13}$$

$$x_{ij} = 0 \text{ if } p_j \geq p^i, T_j \leq T^i, \forall i \in [1, m], \forall j \in [1, n] \tag{14}$$

$$\sum_{i=1}^{m} x_{ij} \leq 1, \quad \forall j \in [1, n] \tag{15}$$

$$x_{ij} \in \{0, 1\}, \quad \forall i \in [1, m], \forall j \in [1, n] \tag{16}$$

To maximize the utility for NSP, the assignment strategy $\boldsymbol{x}$ needs to assign negative-correlated requests in a bundle to save resources so that more requests can be accommodated, i.e., to maximize the multiplexing gain through minimizing $\boldsymbol{x_i}^T V \boldsymbol{x_i}$. In what follows, we first show the difficulty of solving f-OSA and then propose a reinforcement learning-based solution.

IV. On the Hardness of f-OSA

In this section, we analyze the difficulty in the fBm-based optimal slice assignment problem developed above in terms of the irregularity of constraints and the hardness of approximating in polynomial-time.

A. Irregularity of Constraints

We note that the left hand side of constraints (13) is an irregular function of U and V.

It is easy to see that when two network slice requests are assigned into the same bundle, if the Hurst parameter is 0.5, the left hand side of (13) is a linear function of $\boldsymbol{U}$ and V, meaning that the total resource demand for the two requests is linear to the resource allocated to each individual request. Nevertheless, if the Hurst parameter is larger (or smaller) than 0.5, the left hand side of (13) is a convex (or concave) function of V. Using two slice requests as an example, Figure 5 illustrates the curvature of the left hand side of (13) with different Hurst parameters and different variances/covariances. We can conclude that the constraint (13) is quite irregular (i.e., it could be linear, concave, or convex depending on the Hurst parameter). The total required resource is highly coupled with the Hurst parameter of the traffic patterns and the correlation in the traffic from network slices.

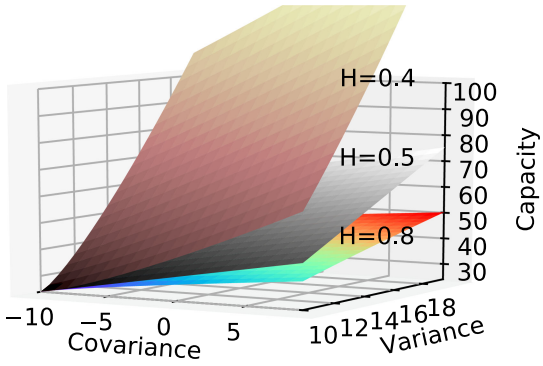


Fig. 5. Required capacity under different variances/covariances with different Hurst parameters (mean arrival rates $\mu_1 = 12$ and $\mu_2 = 12$).

B. Hardness of Approximating f-OSA

The *easiest* form of the f-OSA problem (i.e., the constraint (13) is linear when Hurst parameter = 0.5) has the similar form with multiply-constrained (multiple) knapsack problem (MCKP) [30]. f-OSA is constrained by resources and implicitly by traffic correlations, as defined in (13), as well as by QoS, as defined in (14). It is well known that the 0-1 variant version of MCKP (for any fixed $m \geq 2$ where m is the number of constraints) is NP-complete and does not even have a polynomial-time approximation scheme (PTAS) unless P=NP [31]–[33].

Indeed, the general case of f-OSA is even harder than MCKP due to the irregular constraints posed by (13), where the amount of resources required for network slice requests is determined by an irregular function of their traffic arrivals. As shown in Section IV-A, such irregularity not only makes a close-form theoretical analysis hopeless, but also makes accurate approximation extremely hard.

Due to the above difficulties and inspired by the success of reinforcement learning in solving NP-hard problems and combinatorial optimization problem [34], [35], in the next section we develop a reinforcement learning (RL) based solution for f-OSA, called RL-Assign. RL-Assign, as we shall show later, can effectively handle the complexity introduced by traffic correlation and the irregular resource-traffic constraint function and obtain much better solutions than other baseline methods.

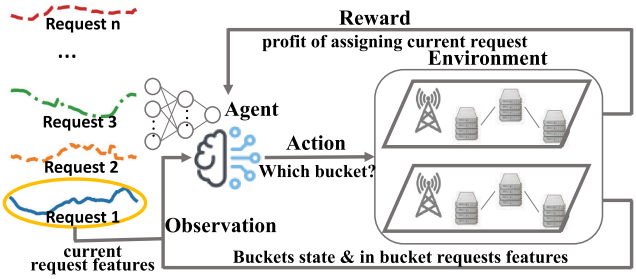


Fig. 6. Architecture of RL-Assign.

V. The Reinforcement Learning Approach: RL-Assign

In this section, we first give an overview of our approach with definitions of the basic elements of reinforcement learning, and then introduce the details of RL-Assign using Policy Gradient.

A. Overview of RL-Assign

RL-Assign is designed to work in the reinforcement learning framework as shown in Fig. 6. The *agent* is to make *action* for each network slice request according to the *observation* of the features of the to-be-assigned requests, the bundles' states, as well as the features of the requests already assigned in the bundles. The *environment* interacts with the agent to correct bad actions and encourage good actions through the *reward*. The problem solved by RL-Assign is equivalent to the f-OSA problem, where the environment corresponds to the constraints, the actions correspond to the solution x, and the total reward corresponds to the objective function.

Agent: The agent makes a network slicing assignment strategy (also called policy), which inputs the request's features and bundles state, and outputs the action that to which bundle the request is assigned. The mapping function between the observations to the action, when it is too complicated, is usually approximated with a neural network. When the agent explores the possible actions, it gets feedback from the environment which measures how good the strategy is. After the exploitation and exploration, the agent is able to learn a strategy that tries to assign the network slice requests to the bundles to gain the highest utility. That is, the agent works as the brain to solve the assignment problem.

Observations: The observations are to find the useful and relevant data from the environment for agent to make decisions. Based on the constraint functions in f-OSA, the state of the environment is mostly affected by Equation (13), where the required capacity of the network slice requests cannot exceed the pre-defined capacity of the network slice bundles. We can decompose Equation (13) into three parts: (1) the features of the to-be-assigned request corresponding to $\boldsymbol{x_i}^T U$, (2) the features related with the other requests in the bundle corresponding to $\boldsymbol{x_i}^T V \boldsymbol{x_i}$, and (3) the feature of the remaining capacity of the bundle corresponding to $\bar{C}^i = C^i - \boldsymbol{x_i}^T U - \eta_i (\boldsymbol{x_i}^T V \boldsymbol{x_i})^{\frac{1}{2H}}$. Equation (13) is the overall constraint for all requests in each bundle.

For the observations of a to-be-assigned network slice request, one of the basic features is the mean of the traffic arrivals of the request, denoted as μ_j. To help the agent learn better, we also need variance σ_j^2, utility w_j in the features. Note that in Equation (13), the variance is included in V.

The next part is the features related to other requests in the network slice bundle. Intuitively, in order to accommodate more requests in the bundle, the agent is more likely to assign the request to the bundle where the sum of the covariances is smaller. Thus, we set the $\boldsymbol{x_i}^T V[j], i \in [1, m]$ for the second part of the observations for request j. The $\boldsymbol{x_i}$'s, $i \in [1, m]$ have included the actions of the assigned network slice requests.

The last part of the observations is straightforward. It is the remaining capacity of the bundle. To help the agent learn more effectively, we use one trick to quickly find out the remaining capacities. In the aforementioned constraint functions, the network slice bundle will not be chosen if it cannot satisfy the QoS requirement for the network slice request, i.e., $x_{ij} = 0$ if $p_j \geq p^i, T_j \leq T^i$. Instead of modifying the action after it is done by the agent, we can prevent this case in advance with the remaining capacity of the bundles set as 0 if it cannot meet the performance requirement of the request.

Action: The action for the network slice requests is made by the agent according to the observations. We use a sequence of integer number $1, 2, \ldots, m$ to denote the bundle number and 0 as request rejection. That is, $x_{ij} = 1$ if and only if $a_j = i$ and $i \neq 0$. For a given action, the agent needs to double check whether the action is valid by evaluating the remaining capacity of the assigned network slice bundle. In practice, each time the agent has an observation, it provides the probability of choosing each action. It sorts all the actions in decreasing order with respect to the probabilities, then double check whether the first choice is valid by evaluating the remaining capacity of the assigned network slice bundle. If the remaining capacity is not positive, the action will be changed to the second best action and be checked again. Such process repeats until any action is valid or all actions fail, in which case the agent chooses $a_j = 0$.

Reward: The reward from the environment works as a critic for the agent. Once the network slice request j is assigned to a bundle, the step reward r_j is set as the utility of admitting this request. Otherwise, the reward r_j is set as 0 (i.e., no utility gain). The sum of utilities of assigned network slice requests is denoted as the total reward, which is equal to the objective function of f-OSA.

Transition: The agent starts the whole process with an observation OBS_1 with respect to the first to-be-assigned request, i.e., request 1, and takes an action a_1 accordingly. After the agent takes an action a_j according to each observation OBS_j, i.e., assigning request j to some specific bundle or rejecting the request, the observation transits to another observation OBS_{j+1} corresponding to the next to-be-assigned request $j+1$. In precise, for request $j+1$, Equation (13) is still decomposed into three parts: (1) the features of request $j+1$ corresponding to $\boldsymbol{x_i}^T U$, (2) the features related to other

requests in the bundle corresponding to ${\boldsymbol{x_i}}^T V \boldsymbol{x_i}$ (remain the same in OBS_{j+1}), and (3) the feature of the remaining capacity of the bundle after request j is assigned, corresponding to $\bar{C}^i = C^i - {\boldsymbol{x_i}}^T U - \eta({\boldsymbol{x_i}}^T V \boldsymbol{x_i})^{\frac{1}{2H}}$. Such a process is a Markov decision process since the next observation can be decided by only the current observation and the corresponding action.

B. The Details of RL Algorithms

Reinforcement learning problems are generally studied as a discrete-time Markov decision problem (MDP), defined by $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, where $\mathcal{S} \subseteq \mathbb{R}_n$ is an n dimensional state space, $\mathcal{A} \subseteq \mathbb{R}_m$ an m action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}_+$ a transitional probability function, and $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ a reward function. The *agent* receives observation $s \in \mathcal{S}$ from the environment, generates action $a \in \mathcal{A}$ by policy π_θ where θ are parameters of the neural network approximating the policy, and then interacts with the environment receiving reward $r \in \mathbb{R}$. The goal of the agent is to study a good policy π_θ which can maximize the total reward of the problem.

Following the above framework, each decision that RL-Assign makes is considered as a step of the whole assignment problem. Thus, the interaction sequence can be fully described as $s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_n, a_n, r_n$, which is referred as one *episode*. The common approach to learning the optimal policy π_θ can be classified into two categories, (a) value-based approaches which aim to learn the state-action value and then to select the optimal actions accordingly, and (b) policy-based approaches which learn the policy directly.

RL-Assign uses the classical policy-based approach, Policy Gradient, based on three reasons: (1) Policy Gradient can handle large-scale problems, where the performance of value-based approaches become worse as the dimension of action space increases, since these approaches require more training samples to estimate the Q-value accurately (this has been validated when the number of slice bundles is larger than 10). (2) It can learn the stochastic policy which is useful for balancing the exploration/exploitation trade-off. (3) It has fewer hyper-parameters and does not need to estimate Q values. Most advanced policy-based approaches, like A3C and DDPG, estimate Q values as critic to guide the gradient of the policy during the episode. However, the convergence of Q values is a severe problem during the training process, requiring a great effort to optimize the hyper-parameters to accurately estimate Q values. Overall, Policy Gradient is a simple and suitable algorithm for our problem. We next introduce the detail of Policy Gradient algorithm in the following.

Policy Gradient Algorithm: The policy is a mapping between the action to the specific state, which is usually modeled with a parameterized function with respect to θ, $\pi_\theta(a|s)$. The value of the reward function under the policy can be defined as $J(\theta) = \sum_{j=1}^{n} \sum_{a \in \mathcal{A}} \pi_\theta(a|s_j) r_j(s_j, a) = E_{\pi(\theta)}[\sum_j r_j(s_j, a_j)]$. Then, the Policy Gradient algorithm maximizes the expected total reward by repeatedly estimating the gradient $\nabla J(\theta) = E_{\pi(\theta)}[\sum_{j=1}^{n} r_j \nabla_\theta \ln \pi_\theta(a_j|s_j)]$.

Neural Networks: Since neural networks can approximate any continuous differential function, we use the set of parameters of a neural network, i.e., weights and biases, to parameterize the policy π_θ. The design of the neural network architecture follows the commonly-used policy network, which inputs the observations and outputs the actions. The neural network is composed of four layers, observation-layer1-layer2-actions, with fully-connected neurons. We set the number of neurons in the inner two layers as 64, and use the Rectified Linear Unit (ReLU) as the activation function.[4]

Algorithm 1: RL-Assign Algorithm.

Input: Network slice bundle capacity C, network slice requests A and utility W, learning rate lr
Output: Optimal policy π_θ
Initialize the policy parameter θ at random
while Not Converged **do**
  for network slice requests j = 1, n **do**
    Choose action a_j by policy π with probability $1 - \epsilon = 0.9$ and random otherwise.
    Check whether a_j is valid by evaluating the remaining capacity $\bar{C}^{a_j}$ of the bundle.
    IF $\bar{C}^{a_j} > 0$ **then**
      reward $r_j = w_j$
    else
      for k=2, n **do**
        Change action to the k_{th} best action
        if $\bar{C}^{a_j} > 0$ **then**
          reward $r_j = w_j$
          continue
        else
          Set the action as $a_j = 0$
    Feed the episode $s_1, a_1, r_1, r_2, \ldots, s_n, a_n, r_n$ to policy network and update the policy parameter $\theta \leftarrow \theta + lr \sum_{j=1}^{n} r_j \nabla_\theta \ln \pi_\theta(a_j|s_j)$

Training Process: The training process of policy gradient algorithm consists of both exploration and exploitation through ϵ-greedy approach. In precise, the learning agent exploits its current optimal action for each state with a probability of $1 - \epsilon$ while executes a random action with a probability of ϵ. While the former one is the standard exploitation for a learned policy, the later one helps the agent avoid getting stuck in a local optimum by exploring random actions to see if they lead to better reward. As for the optimization process for the parameterized neural network, we adopt the Adam optimizer among the stochastic gradient descent (SGD) methods since empirical results demonstrate that Adam works well in practice and compares favorably to other stochastic optimization methods [36]. The converging condition of the training process is set as the average reward of the latest 10 episodes no longer increases in consecutive 50 times. The training process in each episode is summarized in Algorithm 1.

It is worth mentioning that the RL approach presented above may be applied to other traffic types, as long as we can derive the way of calculating the corresponding capacity constraint to replace (13). We just need to simply plug the new constraint into our RL framework.

[4]As common in nearly all deep learning-based solutions, the design of neural network is empirical.

TABLE I
QoS OFFERED IN DIFFERENT NETWORK SLICE BUNDLES

N	Bundle 1 <95%, 10>	Bundle 2 <90%, 15>	Bundle 3 <90%, 20>	Bundle 4 <85%, 25>	Bundle 5 <80%, 50>
C	Bundle 1 <95%, 1>	Bundle 2 <95%, 5>	Bundle 3 <90%, 10>	Bundle 4 <90%, 15>	Bundle 5 <90%, 20>
	Bundle 6 <85%, 25>	Bundle 7 <80%, 30>	Bundle 8 <80%, 35>	Bundle 9 <75%, 40>	Bundle 10 <70%, 50>

Note: "N" part is the QoS offered in the network slice bundles for NORDUnet. "C" part is the QoS of the bundles for cellular traffic.

VI. PERFORMANCE EVALUATION

In this section, we evaluate the performance of RL-Assign algorithm over four baselines, namely, First-Fit, Best-Fit, and MTM, respectively, in two datasets.

A. Experiment Setting

Traffic Dataset: We use two datasets with different scales for evaluation: (1) The 24 hours real time NORDUnet traffic statistics ("Max out" traffic data) [37] of 46 flows starting from 00:00 to 23:59, April 16, 2019; (2) The week-long cellular traffic [38] generated in a median-size city of China from August 19, 2012 to August 26, 2012.

Network Slice Bundles (1) *NORDUNet Traffic:* Assume that NSP provides 5 network slice bundles, each has 100 Gb/s capacity. The QoS guaranteed by the network slice bundles are in the form that $p(\%)$ of packets have no longer than $T(\mathrm{ms})$ end-to-end delay, denoted as $< p, T >$. We listed the QoS guaranteed by each network slice bundle in Table I. (2) *Cellular traffic:* We set 10 network slice bundles to accommodate the traffic, each with capacity 100, 100, 70, 50, 50, 40, 50, 30, 30, 30 GB/s respectively. The QoS provided by each network slice bundle is also listed in Table I.

Network Slice Requests: We take the traffic of each flow in both datasets as the traffic of one network slice request. Therefore, in the simulation, we have 46 network slice requests in NORDUnet and randomly choose 165 network slice requests in cellular traffic.

The QoS required for each request in NORDUnet/Cellular is randomly set in a range such that at least one bundle for NORDUnet/Cellular listed in Table I can accommodate the network slice request with its required QoS.

Assuming that the traffic arrivals in the slice request j have mean μ_j and standard deviation σ_j, we set its utility as $k_j(\mu_j + \sigma_j + z)$, where k_j is a scaling parameter in $[0.8, 1.5]$ related to the QoS of the requests and z is a random noise randomly varying in the range of $[0, 10]$. The setting of k_j reflects that higher QoS requirement will bring higher utility for NSP while the mean μ_j and variance σ_j reflect larger traffic volume will also bring higher utility. The introduction of z allows us to observe the robustness of our algorithm when slice utilities vary.

There are potentially other ways to set the slice utility, e.g., uniformly random in a range or all the same. Nevertheless, while the final test results w.r.t. the total utility of accepted slices are different with different ways of setting utility, our algorithm consistently outperforms the four baseline methods (Section VI-C). We omit those tests due to space limitation.

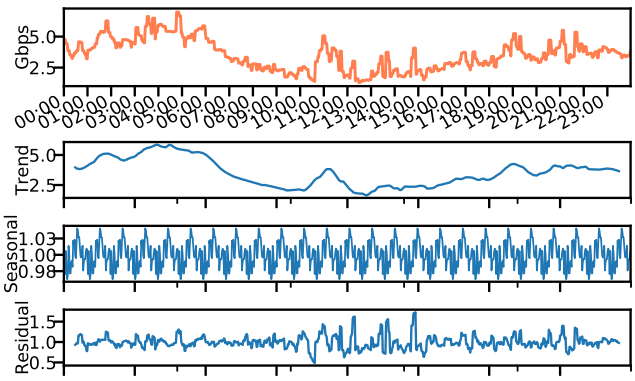


Fig. 7. Time series decomposition of "Akamai-Cluster-daymax" traffic.

B. Pre-Processing

To extract the correlation among the network slice requests, we use time series analysis tool to decompose the time series traffics into three additive components: main traffic trend, seasonality which is repeated in a short-term cycle, and residual which is random variation. The decomposing result is demonstrated in Fig. 7. We calculate the covariance matrix V based on the trend component of traffic in each network slice request.

C. Baseline Algorithms

We set First-Fit algorithm, Best-Fit algorithm, and MTM algorithm as the baseline algorithms for comparison. Since the capacity constraints developed in the f-OSA problem can not be applied in these baseline algorithms, we set the capacity of each network slice request according to its QoS requirement and its traffic arrivals per unit time. The traffic arrivals of a network slice request is a time series sequence with traffic arrival increments recorded per unit time. We set the capacity that can only hold $p(\%)$ of per unit time traffic arrival increments. However, such setting of capacity of each request cannot guarantee the QoS since the queueing delay could go to infinity. The total utility achieved by the baseline algorithms under the aforementioned capacity assignment may be higher than the actual achievable one because of the underestimation of the required capacity. In other words, we compare the performance of our algorithm with the upper bound performance of the baseline algorithms.

First-Fit: First-Fit algorithm is to assign the network slice requests to the first bundle which meets its capacity requirement and QoS requirement.

Best-Fit: Best-Fit algorithm is to assign the requests to the network slice bundle with least remaining capacity under the consideration of the QoS requirements.

MTM: Branch and bound algorithms are traditional algorithms for multiple knapsack problem [39]. We implement the most commonly used one, MTM algorithm by Martello and Toth (1981) [17] in our network slice request assignment problem. We can formulate the f-OSA problem as a standard

TABLE II
UTILITIES AND NUMBERS OF ADMITTED REQUESTS OF THE BASELINES AND RL-ASSIGN WITH DIFFERENT NETWORK SLICE REQUESTS

		First-Fit	Best-Fit	MTM	CIP	RL-Assign
NORDUnet	utility	368	476	499	499	**571**
	No.	35	39	35	33	**45**
Cellular	utility	290	431	NULL	NULL	**1493**
	No.	18	24	NULL	NULL	**87**

Note: NULL means the algorithm is unable to solve the problem due to the scale limitation.

multiple knapsack problem with determined capacity constraints. Briefly speaking, MTM algorithm is to use tree-search technique and lower bound to determine the branches to follow. The detail process is omitted here.

CIP: Constraint integer programming (CIP) is a novel way to combine constraint programming (CP) and mixed integer programming (MIP) methodologies [18]. We can formulate the f-OSA problem as an integer programming problem. We then use PySCIPOpt package [40] which is the interface from Python to CIP solver suite to get the optimal solution.

D. Experiment Result

We evaluate our system under the NORDUnet and Abilene traffic set specifically. For each set of network slice requests, we use the baseline algorithms as well as our solution to assign them into the given network slice bundles. For each algorithm, we repeat the experiments 15 times to make the results more convincing. The average total utility achieved when the utilities of the requests are resource based is demonstrated in Table II. Our system outperforms all other methods in both sets of requests. In NORDUnet set, First-Fit algorithm performs worst since it simply puts the requests according to the given order, Best-Fit algorithm performs a bit better, while MTM and CIP have better performance than them. MTM and CIP have found the optimal assignment strategy to get the maximal utility when the capacity constraints are set by the $p\%$ threshold. None of these algorithms considers the correlation among the network slice requests, which loses the utility brought by leveraging the multiplexing gain.

To be more specific, the total utility gained by RL-Assign can be improved by 14.4% and 246.4% while the number of admitted requests can be enhanced by 15.4% and 262.5% in NORDUnet and cellular traffic request set, respectively. The utility enhancement is achieved by the multiplexing gain of bundling correlated requests, which is 84.8 GB for NORDUnet request set and 1754.9 GB for cellular traffic request set. The benefit of multiplexing is much more significant in cellular traffic request set since peak arrivals in week-long requests are sharper and vary over time. Note that the utilities of admitted requests are different so that the improvement on the number of admitted requests does not equal to the improvement on the overall utility. Moreover, the MTM algorithm and the CIP approach fail to handle the Knapsack problem with scale of 10 bundles and 108 requests, which is not practical in real use.

Furthermore, we evaluate the QoS achieved by each network slice bundle with the assigned network slice requests. The lifetime of the assigned network slice requests in the network slice bundle is simulated in Network Simulator 3 (NS3) with real time traffics and specific bundle capacity. The simulation results show that the end-to-end delay of all the packets admitted to a bundle is lower than their specified worst-case delay. That is, our algorithm can guarantee the QoS of all requests.

E. Convergence and Time Complexity

To show the learning process of our system, we show one of the training processes of the RL-Assign system in Fig. 8 (a) & (b) for two sets of network slice requests respectively. Observed from the figures, the average total reward per 100 episodes increases with the process of training. Note that it is natural that the total reward in every episode fluctuated with the Policy Gradient algorithm due to the adoption of epsilon-greedy approach and SGD algorithm. It is the process of exploration that will help improve the policy. The optimal policy is learned when the algorithm converges. We then choose the actions in the episode with the highest total rewards during the whole training process as the assignment decision of our problem.

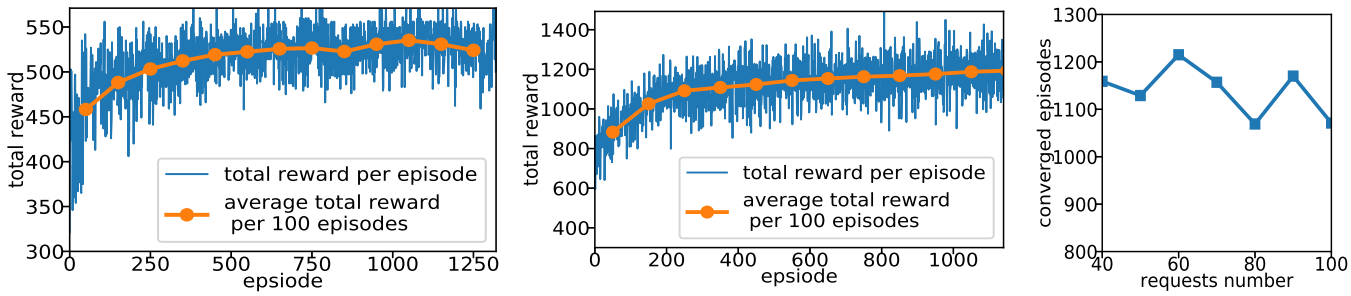
We now discuss the complexity of our algorithm, which is vitally important in practical use of the algorithm. Since it is impossible to get the theoretical analysis of the complexity of Policy Gradient algorithm, we illustrate the empirical result of our algorithm by comparing the converged episodes under different number of requests. We randomly select requests among the cellular traffic set from 40 to 100, and assign them to bundles with the same QoS configuration in Table I. Since our algorithm is specific for the case that resources of the bundles are limited, we reduce the capacity of each bundle to 50 Gbit/s. As shown in the Figure 8 (c), the converged episodes are stable with the increase of requests. We can conclude that our algorithm is robust and efficient.

In summary, our system is effective, efficient, and scalable, which can be applied to a system with tens of network slice bundles and hundreds of network slice requests.

VII. RELATED WORK

In this section, we review the literature on network slice management and the work applying reinforcement learning on problems in similar context, but different to our problem.

The majority of prior research about network slicing management can be classified into two categories: (1) Network slice resource planning for given network slice requests; (2) Network slice request admission control with allocated resource. In the first category, the literature can be further divided into the traffic forecasts [21], [41] and flexible resource allocation approaches [42], [43]. [21] proposes deep learning techniques with spatiotemporal modeling and prediction in cellular networks based on big data. Similarly, [41] forecasts the capacity needed to accommodate future traffic demands with deep learning architecture, considering the balance between the resource



(a) Training process for NORDUnet requests (b) Training process for cellular traffic requests (c) Converged episodes with different number of requests

Fig. 8. Training process of RL-Assign and the empirical converge episodes.

over-provisioning and service requirement violations. These studies demonstrate the importance of modeling and predicting the pattern of the traffic load which exhibits high spatial and temporal variance. On the other hand, with the assumption that traffic profiles are given, the work in [43] addresses the problem of placing and chaining of virtual network functions, and proposes a flexible resource allocation approach that takes the service latency into account. Later, [42] extends the work into edge and core network slice framework. In the second category, network slice admission control policies are studied to increase the utility of the NSaaS with limited available resources. The concept of verticals which can request and lease resources from NSaaS dynamically is firstly proposed in [44]. Based on this concept, [45] designs an admission control and orchestration optimization by modelling the slice queuing behavior. [46] finds a Nash equilibrium for dynamically sharing resources across slices.

Different from the aforementioned work, our work quantifies the multiplexing gain based on the characteristic of traffic patterns, which can be used in both resource planning and admission control optimization. We leverage the multiplexing gain through the concept of "network slice bundle". Though some other work also studies multiplexing in network slicing [14], [47], they focus on the dynamical resource allocation in a network consisting of a pool of resources, e.g., base stations that are shared by a set of operators to network slices. They show that higher multiplexing gain can be obtained when the spatial loads of network slices are more imbalanced while multiplexing temporal heterogeneous traffic loads has not been studied.

We tackle our proposed problem with reinforcement learning. Some pioneer work used reinforcement learning approaches in solving traditional knapsack problem [34], [48], but those solutions are not applicable in our problem, since our problem has much more complex constraints. Thus our work differs from [34], [48] in that we design effective input features and neural networks to adopt the reinforcement learning approach to our problem.

VIII. Conclusion

In conclusion, we develop a general model for assigning network slice requests into appropriate network slice bundles, so that the total utility of NSaaS service provider can be maximized as well as the constraint of the QoS of each network slice request is satisfied. After analyzing the hardness of the problem that cannot get the polynomial approximation, we propose a reinforcement learning approach, RL-Assign, to achieve efficient multiplexing gain in assigning network slice requests to network slice bundles with performance guarantee. Extensive trace-driven simulation shows the superior performance of RL-Assign over four baseline algorithms: First-Fit, Best- Fit, MTM, and CIP. The utility of the NSP can be improved by 14.4% and 246.4% while the number of admitted requests can be enhanced by 15.4% and 265.2% in different request scenarios. These findings provide a potential mechanism for NSP to efficiently configure the network slice bundles and assign the network slice requests to bundles with the maximal utility.

References

[1] Cisco, "Cisco visual networking index: Global mobile data traffic forecast update, 2017-2022," Feb. 2019. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html

[2] K. Makhijani *et al.*, "Network slicing use cases: Network customization and differentiated services," Jun. 2017. [Online]. Available: https://tools.ietf.org/id/draft-netslices-usecases-00.html

[3] 5GAmericas.org, "Network slicing for 5G networks & services," Nov. 2016. [Online]. Available: http://www.5gamericas.org/files/1414/8052/9095/5G_Americas_Network_Slicing_11.21_Final.pdf

[4] H. Zhang, N. Liu, X. Chu, K. Long, A.-H. Aghvami, and V. C. M. Leung, "Network slicing based 5G and future mobile networks: Mobility, resource management, and challenges," *IEEE Commun. Mag.*, vol. 55, no. 8, pp. 138–145, Aug. 2017.

[5] GSMA, "Network slicing: Use case requirements," Apr. 2018. [Online]. Available: https://www.gsma.com/futurenetworks/wp-content/uploads/2018/04/NS-Final.pdf

[6] Ericsson. Scalable network opportunities. [Online]. Available: https://www.ericsson.com/4a45a8/assets/local/digital-services/trending/scalable-network/executive_guide_network_slicing.pdf.

[7] IETF. Network slice provision models. [Online]. Available: https://tools.ietf.org/pdf/draft-homma-slice-provision-models-00.pdf (2019/02/01).

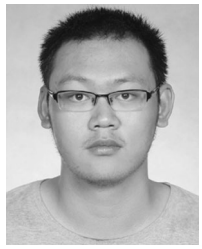
[8] P. Caballero, A. Banchs, G. De Veciana, and X. Costa-Pérez, "Network slicing games: Enabling customization in multi-tenant mobile networks," *IEEE/ACM Trans. Netw. (TON)*, vol. 27, no. 2, pp. 662–675, Apr. 2019.

[9] I. Afolabi, J. Prados, M. Bagaa, T. Taleb, and P. Ameigeiras, "Dynamic resource provisioning of a scalable e2e network slicing orchestration system," *IEEE Trans. Mobile Comput.*, vol. 19, no. 11, pp. 2594–2608, Nov. 2020.

[10] D. M. Gutierrez-Estevez *et al.*, "Artificial intelligence for elastic management and orchestration of 5g networks," *IEEE Wireless Commun.*, vol. 26, no. 5, pp. 134–141, Oct. 2019.

[11] K. A. Noghani, A. Kassler, and J. Taheri, "On the cost-optimality trade-off for service function chain reconfiguration," 2019, *arXiv:1910.01881*.
[12] C. Li, A. Burchard, and J. Liebeherr, "A network calculus with effective bandwidth," *IEEE/ACM Trans. Netw.*, vol. 15, no. 6, pp. 1442–1453, Dec. 2007.
[13] Y. Zaki, L. Zhao, C. Goerg, and A. Timm-Giel, "LTE mobile network virtualization," *Mobile Netw. Appl.*, vol. 16, no. 4, pp. 424–432, Aug. 2011.
[14] J. Zheng, P. Caballero, G. de Veciana, S. J. Baek, and A. Banchs, "Statistical multiplexing and traffic shaping games for network slicing," *IEEE/ACM Trans. Netw.*, vol. 26, no. 6, pp. 2528–2541, Dec. 2018.
[15] Z. Li, Q. Chen, and V. Koltun, "Combinatorial optimization with graph convolutional networks and guided tree search," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 539–548.
[16] M. Gasse, D. Chételat, N. Ferroni, L. Charlin, and A. Lodi, "Exact combinatorial optimization with graph convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 15 554–15 566.
[17] S. Martello and P. Toth, "A bound and bound algorithm for the zero-one multiple knapsack problem," *Discrete Appl. Math.*, vol. 3, no. 4, pp. 275–288, Nov. 1981.
[18] T. Achterberg, T. Berthold, T. Koch, and K. Wolter, "Constraint integer programming: A new approach to integrate cp and mip," in *Proc. Integr. AI OR Techn. Constraint Program. Combinatorial Optim. Problems*. 2008, pp. 6–10.
[19] J. Zhang, Y. Xiang, Y. Wang, W. Zhou, Y. Xiang, and Y. Guan, "Network traffic classification using correlation information," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 1, pp. 104–117, Jan. 2013.
[20] L. Nie, D. Jiang, S. Yu, and H. Song, "Network traffic prediction based on deep belief network in wireless mesh backbone networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, 2017, pp. 1–5.
[21] J. Wang *et al.*, "Spatiotemporal modeling and prediction in cellular networks: A big data enabled deep learning approach," in *Proc. IEEE Int. Conf. Comput. Commun.*, May 2017, pp. 1–9.
[22] C. Zhang and P. Patras, "Long-term mobile traffic forecasting using deep spatio-temporal neural networks," in *Proc. Eighteenth ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*. ACM, 2018, pp. 231–240.
[23] M. Mandjes, *Large Deviations Gaussian Queues: Modelling Communication Networks*. Wiley, 2007.
[24] I. Norros, "On the use of fractional Brownian motion in the theory of connectionless networks," *IEEE J. Sel. Areas Commun.*, vol. 13, no. 6, pp. 953–962, Aug. 1995.
[25] K. Park and W. Willinger, "Self-similar network traffic: An overview," in *Proc. Self-Similar Netw. Traffic Perform. Eval.* New York, USA: John Wiley & Sons, Inc., 2000, pp. 1–38.
[26] P. Rabinovitch, "Statistical estimation of effective bandwidth," Ph.D. dissertation, Carleton University, 2000.
[27] F. P. Kelly, "Notes on effective bandwidths," in *Proc. Stochastic Netw.: Theory Appl., Roy. Stat. Soc. Lecture Notes Series, 4*, F. P. Kelly, S. Zachary, and I. Ziedins, Eds. Oxford University Press, 1996, pp. 141–168.
[28] H. Touchette, "A basic introduction to large deviations: Theory, applications, simulations," Jun. 2011. [Online]. Available: http://arxiv.org/abs/1106.4146
[29] C.-S. Chang, *Perform. Guarantees Commun. Netw.*, ser. Telecommunication Networks and Computer Systems. New York, USA: Springer, 2000.
[30] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Springer, 2004.
[31] D. Zuckerman, "NP-complete problems have a version that's hard to approximate," in *Proc. Eighth Annu. Structure Complexity Theory Conf.* Comput. Soc. Press, 1993, pp. 305–312.
[32] G. Gens and E. Levner, "Complexity of approximation algorithms for combinatorial problems: A survey," *ACM SIGACT News*, vol. 12, no. 3, pp. 52–65, 1980.
[33] B. Korte and R. Schrader, "On the existence of fast approximation schemes," in *Proc. Nonlinear Program. 4*. Academic Press, Jan. 1981, pp. 415–437.
[34] L. Tran-Thanh, A. Chapman, A. Rogers, and N. R. Jennings, "Knapsack based optimal policies for budgetlimited multiarmed bandits," in *Proc. 26th AAAI Conf. Artif. Intell.*, Jul. 2012, pp. 1134–1140.
[35] X. Chen and Y. Tian, "Learning to perform local rewriting for combinatorial optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 6278–6289.
[36] S. Ruder, "An overview of gradient descent optimization algorithms," Sep. 2016. [Online]. Available: http://arxiv.org/abs/1609.04747
[37] NORDunet, "NORDUnet network statistics," [Online]. Available: http://stats.nordu.net/connections.html
[38] C. Xiaming, J. Yaohui, Q. Siwei, H. Weisheng, and J. Kaida, "Analyzing and modeling spatio-temporal dependence of cellular traffic at city scale," in *Proc. Commun. IEEE Int. Conf.*, 2015, pp. 3585–3591.
[39] S. Martello, "Knapsack problems: Algorithms and computer implementations," *Wiley-Interscience Series Discrete Math. Optim.*, 1990.
[40] S. Maher, M. Miltenberger, J. P. Pedroso, D. Rehfeldt, R. Schwarz, and F. Serrano, "PySCIPOpt: Mathematical programming in python with the SCIP optimization suite," in *Proc. Math. Softw. – ICMS 2016*. Springer International Publishing.
[41] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, "DeepCog: Cognitive network management in sliced 5G networks with deep learning," in *Proc. IEEE Int. Conf. Comput. Commun.*, Apr. 2019, pp. 280–288.
[42] Q. Zhang, F. Liu, and C. Zeng, "Adaptive interference-aware VNF placement for service-customized 5G network slices," in *Proc. IEEE Int. Conf. Comput. Commun.*, Apr. 2019, pp. 2449–2457.
[43] A. Alleg, T. Ahmed, M. Mosbah, R. Riggio, and R. Boutaba, "Delay-aware VNF placement and chaining based on a flexible resource allocation approach," in *Proc. 13th Int. Conf. Netw. Service Manag.*, Nov. 2017, pp. 1–7.
[44] K. Samdanis, X. Costa-Perez, and V. Sciancalepore, "From network sharing to multi-tenancy: The 5G network slice broker," *IEEE Commun. Mag.*, vol. 54, no. 7, pp. 32–39, Jul. 2016.
[45] B. Han, V. Sciancalepore, D. Feng, X. Costa-Perez, and H. D. Schotten, "A utility-driven multi-queue admission control solution for network slicing," in *Proc. IEEE Int. Conf. Comput. Commun.*, Apr. 2019, pp. 55–63.
[46] P. Caballero, A. Banchs, G. de Veciana, and X. Costa-Perez, "Network slicing games: Enabling customization in multi-tenant networks," in *Proc. IEEE Int. Conf. Comput. Commun.*, May 2017, pp. 1–9 .
[47] P. Caballero, A. Banchs, G. de Veciana, and X. Costa-Perez, "Multi-tenant radio access network slicing: Statistical multiplexing of spatial loads," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 3044–3058, Oct. 2017.
[48] O.-C. Granmo, B. J. Oommen, S. A. Myrer, and M. G. Olsen, "Learning automata-based solutions to the nonlinear fractional knapsack problem with applications to optimal resource allocation," *IEEE Trans. Syst., Man Cybern., Part B (Cybern.)*, vol. 37, no. 1, pp. 166–175, Feb. 2007.

Qian Xu received the B.S. degree in statistics from the Huazhong University of Science and Technology, Wuhan, China, in 2015. She is currently working toward the Ph.D. degree in computer science with the City University of Hong Kong, Hong Kong. Her current research interests include cloud computing, and network performance analysis.



Xiang Yan received the B.Sc. degree from Shanghai Jiao Tong University, Shanghai, China, where he is currently working toward the Ph.D. degree with the Department of Computer Science. His current research interests focus on algorithmic game theory and machine learning, with their applications to internet economics.

Kui Wu received the B.Sc. and M.Sc. degrees in computer science from Wuhan University, Wuhan, China, in 1990 and 1993, respectively, and the Ph.D. degree in computing science from the University of Alberta, Edmonton, AB, Canada, in 2002. In 2002, he joined the Department of Computer Science, University of Victoria, Victoria, BC, Canada, where he is currently a Professor. His current research interests include network performance analysis, online social networks, Internet of Things, and parallel and distributed algorithms.

Jianping Wang received the B.S. and M.S. degrees in computer science from Nankai University, Tianjin, China, in 1996 and 1999, respectively, and the Ph.D. degree in computer science from The University of Texas at Dallas, Richardson, TX, USA, in 2003. She is currently a Professor with the Department of Computer Science, City University of Hong Kong, Hong Kong. Her research interests include cloud computing, service oriented networking, edge computing, and network performance analysis.

Kejie Lu (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees from the Beijing University of Posts and Telecommunications, Beijing, China, in 1994 and 1997, respectively, and the Ph.D. degree in electrical engineering from the University of Texas at Dallas, Richardson, TX, USA, in 2003. In July 2005, he joined the University of Puerto Rico at Mayagüez, Mayagüez, Puerto Rico, where he is currently a Professor with the Department of Computer Science and Engineering. His research interests include computer and communication networks, cyber-physical system, and network-based computing.

Weiwei Wu received the Ph.D. degrees from the City University of Hong Kong (CityU), Hong Kong and the University of Science and Technology of China (USTC), Hefei, China, in 2011. He went to Nanyang Technological University (NTU) for Postdoctoral Research. He is currently a Professor with the School of Computer Science and Engineering, Southeast University, Nanjing, China. His research interest covers the applications of optimization method in wide range of topics, including combinatorial optimization and algorithms, game theory, reinforcement learning, multi-agent systems, cloud computing, and wireless networks.