Availability Aware Online Virtual Network Function Backup in Edge Environments

Yu Liu, Xiaojun Shang, Yingling Mao, Zhenhua Liu, and Yuanyuan Yang, *Fellow, IEEE*

Abstract—With the rapid advancement of edge computing and network function virtualization, it is promising to provide flexible and low-latency network services at the edge. However, due to the vulnerability of edge services and the volatility of edge computing system states, i.e., service request rates, failure rates, and resource prices, it is challenging to minimize the online service cost while providing the availability guarantee. This paper considers the problem of online virtual network function backup under availability constraints (OVBAC) for cost minimization in edge environments. We formulate the problem based on the characteristics of the volatility system states derived from real-world data and show the hardness of the formulated problem. We use an online backup deployment scheme named Drift-Plus-Penalty (DPP) with provable near-optimal performance for the OVBAC problem. In particular, DPP needs to solve an integer programming problem at the beginning of each time slot. We propose a dynamic programming-based algorithm that can optimally solve the problem in pseudo-polynomial time. Extensive real-world data-driven simulations demonstrate that DPP significantly outperforms popular baselines used in practice.

Index Terms—Virtual Network Function Backup, Edge Computing, Online Cost Minimization, Reliability Constraints

1 Introduction

Network Function Virtualization (NFV), which migrates traditional network functions, e.g., firewalls, load balancers, proxies, from dedicated hardware to **Virtual Network Functions** (VNFs) running on general-purpose commercial servers, has drawn tremendous attention recently. NFV has introduced great flexibility, scalability, and elasticity to the deployment and management of network services [1], [2], [3]. Recently, there has been a rise in applications requiring low latency, including augmented reality, real-time motion estimation and compensation in videos, the internet of things, and self-driving [4]. To support such applications, deploying network services supported by VNFs at the network edge is promising [1], [5], [6].

Although deploying VNFs at the network edge has numerous potential benefits, it does raise new challenges. In particular, edge servers are not as reliable as servers in the cloud, making the network services more vulnerable to failures [7]. While deploying backup VNF instances is one of the ubiquitous paradigms to deal with such failures, it significantly increases the system cost, which is undesirable due to highly limited energy and resources at the network edge. In other words, there is a trade-off between the reliability of VNFs and the cost incurred.

The trade-off between cost and reliability is dynamic due to the volatility of system states in edge environments, i.e., resource price, failure probability, and request rates of VNFs [8], [9]. Therefore, finding the sweet spot of the trade-off is challenging for the following reasons. First, the cost of deploying a VNF backup or primary instance depends on the time-varying prices of edge resources, e.g., central processing unit (CPU), graphics processing unit (GPU), and random-access memory (RAM) [8]. In addition, the failure probability of each VNF instance is time-varying [9], [10]. That is, given the same number of backups, the reliability of a VNF can change over time. Moreover, network functions have different request rates at different time slots due to the high mobility of users in edge environments [11], [12]. A failure during a time slot with high request rate results in greater damage compared to a time slot with a low request rate. The relationship between cost and reliability in the system is complex due to the interplay of the three time-varying edge system states, namely, resource price, failure probability, and request rate of VNFs. As a result, a static VNF backup scheme is unable to achieve the optimal cost while satisfying reliability requirements. The deployment of VNF backup for high reliability has received substantial research attention, resulting in the proposal of several methods with varying objectives, such as in [13], [14], [15], [16]. However, existing methods are inadequate for edge scenarios as they lack the ability to adapt to real-time changes in the edge system state. In this paper, we aim to address these limitations by jointly considering the volatility of system states, minimum availability requirements at each time slot, and time-average availability requirements of each VNF.

For more realistic problem formulation, we make use of system state characteristics observed from real-world data. There are often peak periods and off-peak periods for mobile Internet requests each day [12], [17], where peak periods are when more people are likely to request network services and off-peak periods are generally when most users are inactive such as midnight and early afternoon. The real-world data about the number of requests over time of a video [18] are shown in Figure 1. It is evident that the request numbers follow a periodic underlying trend.

- *Y. Liu, X. Shang, Y. Mao, and Y. Yang are with the Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY, 11794. Z. Liu is with the Department of Applied Mathematics and Statistics, Stony Brook University, Stony Brook, NY, 11794. E-mail: {yu.liu.3, xiaojun.shang, zhenhua.liu, yuanyuan.yang}@stonybrook.edu .*

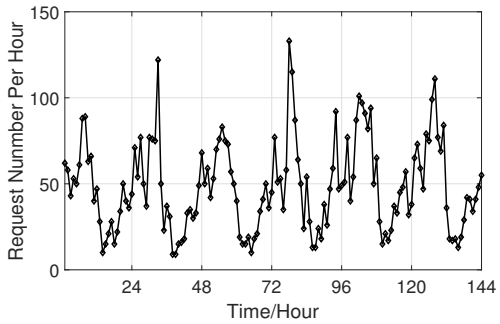


Fig. 1: Real-world request rates of a video over time.

It is neither independent and identically distributed (iid) over time [19], [20] nor fully arbitrary without any pattern [7]. Motivated by the observation, we generalize existing work by assuming the request rate of each VNF consists of a periodic trend and some random noise. The failure probabilities of VNF instances and prices of edge resources have similar structures, and detailed information about these edge system states can be found in Section 3.3.

In this paper, we study the problem of online VNF backup under availability constraints and time-varying edge system states with the goal of minimizing the overall system cost (OVBAC). There is a certain number of heterogeneous types of VNFs in the system. The cost of each VNF during a time slot is the cost of edge resources required by its primary instance and all backup instances during that time slot. There are two types of reliability constraints for each VNF. The first constraint is that the availability, the probability of having at least one available instance, is no less than a given threshold for each time slot. The second constraint requires that the time-average availability of the VNF weighted by the request rate is no less than another threshold, which essentially limits the average availability for each unit of the request of each time slot. With this constraint, VNFs with higher request rates tend to have more backups to prevent serious damage caused by their failures. These two thresholds may vary for different VNFs.

We design an online algorithm based on the Drift-Plus-Penalty (DPP) scheme for the proposed problem in Section 4. DPP is designed for systems with statiionary states [21]. The system states of the proposed problem are not iid, and we prove the performance of DPP is near-optimal for OVBAC in Section 4.2. Our main contributions in this paper are summarized as follows.

- We formulated the problem of online VNF backup under availability constraints to minimize the edge system cost in Section 3 and proved the offline version of the proposed problem is NP-hard.
- We developed an algorithmic scheme named DPP and prove its performance in Section 4.1. Specifically, DPP makes decisions in an online manner, which needs to solve an integer programming problem (P2) at the beginning of each time slot. We proved that DPP achieves near optimal cost.
- We proposed a dynamic programming-based algorithm to solve P2 in Section 4.2. The proposed algorithm can solve P2 in pseudo-polynomial time.
- Our proposed scheme is evaluated by extensive real-world trace-driven simulations under a wide range of settings. The results highlight that the proposed algorithm significantly outperforms popular baseline algorithms used in practice.

The remainder of this paper is organized as follows. Section 2 discusses related works. Section 3 formulates the online VNF backup problem under availability constraints for cost minimization. Section 4 proposes the DPP scheme and provides its theoretical performance guarantee. Section 5 presents performance evaluation results. Section 6 concludes this paper.

2 RELATED WORK

There are a lot of studies with various purposes, applications, and approaches for NFV.

VNF placement and chaining with various purposes are well-studied, e.g., [13], [14], [15], [16], [27], [28]. In [13], Sang *et al.* studied the problem of minimizing the total number of VNF instances, and the authors proposed two algorithms with performance guarantees. In [14], [15], the authors investigated the VNF placement and chaining problem for cost minimization. In [16], Liu *et al.* explored the service function chain deployment and resource management jointly to minimize the system latency, and a game-theoretic approach with a constant approximation ratio is proposed. [29], [30] investigated the VNF placement problem in Edge environments, and [31], [32] study VNF in data center environments. To solve the problems proposed above, different approaches are introduced, e.g., integer linear programming [15], game theoretic-based methods [16], and machine learning algorithms [33], [34].

Several previous studies have investigated the reliability of virtual network functions [35], [36], [37], [38]. In [35], Fan *et al.* addressed the problem of reducing resource consumption while satisfying reliability constraints and proposed an algorithm with polynomial time complexity. In [36], Zhang *et al.* aimed to minimize backup resource consumption while maintaining overall availability demands. They presented an algorithm based on differential evolution to solve the NP-hard problem they formulated. In [37], Taleb *et al.* investigate a system that deploys VNF backups in a reactive manner by exploiting early failure detection. In [38], Jing *et al.* addressed a budget-aware service provisioning problem with the goal of minimizing costs. They developed an approximation algorithm with a provable approximation ratio for the NP-hard problem they proposed. However, none of the previous studies took into account the volatility of the system.

Additionally, there are some pioneering works about online VNF backup as listed in Table 1. In [7], Shang *et al.* explored the problem of maximizing the sum of VNF availabilities while subject to a cost constraint in the presence of time-varying failure probability rates, and an algorithm based on competitive online optimization is designed. In addition, in [22], Shang *et al.* investigated the offline problem of VNF backup over the edge and cloud for cost minimization and proposed an online algorithm to balance the load of servers. In [23], Jing *et al.* investigated the SFC-enabled service provisioning problem, and the authors proposed an offline

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3282156

Paper	Goal	Method	Online	Miminum Availability Guarantee	Multiple Resource Types	Performance Guarantee
[7]	Maximize overall availability under time average budget	Competitive Online Optimization	✓	✗	✗	$O(\log(\theta))$
[22]	Minimize maximum load over edge servers	Competitive Online Optimization	✓	✗	✗	$O(\log(\|V\|))$
[23]	Maximize dynamic service admission rate	Primal-Dual Dynamic Updating Technique	✓	✓	✗	$1+\psi_{max}$
[24]	Maximize the backup hit reward	Online Bandit	✓	✗	✗	Bounded Regret
[25]	Maximize a comprehensive backup reward	Online Bandit	✓	✗	✗	
[26]	Minimize the upper-bound latency	Offline Binary Optimization	✗	✓	✗	$\epsilon^{-2}\|V\|\log(b_{\max})$
This Paper	Minimize overall cost under availability constraints	Drift-Plus-Penalty	✓	✓	✓	Near Optimal

TABLE 1: Papers Related to VNF Backup

approximation algorithm and a digital twin-assisted online algorithm based on the offline approximate solution. The above papers [7], [22], [23], with diverse problem formulations and objectives, disregarded real-world observations when assuming system states, resulting in relatively high competitive ratios. On the other hand, in this paper, we model the system states based on real-world data, yielding a provable performance guarantee that is close to optimal. In [39], [40], Zichuan *et al.* investigated the online social welfare maximization problem to maximize the overall revenue, and the authors presented a multi-armed bandits-based online learning algorithm. In [24], Chen *et al.* studied the online VNF backup problem with stationary system states to maximize the backup hit reward, and an online bandit learning-based algorithm was proposed. In [25], Chen *et al.* explored the service function chain backup problem with stationary system states to maximize system latency, and the authors proposed an algorithm leveraging the online bandit learning technique. In [24], [25], [39], [40], system states are from stationary iid distributions and learned by the online bandit learning technique. Given that real-world data reveals that system states are non-iid, this paper considers a scenario that strikes a balance between arbitrary system states and stationary system states. Specifically, motivated by real-world data, the system states in this study are modeled as periodic baselines with added random noise.

The online competitive optimization algorithms proposed in [7], [22], [23] have been found to have large competitive ratios. On the other hand, papers in [24], [25], [39], [40] are based on online bandit learning algorithms, which typically assume iid stationary system states. As such, they are not suitable for handling the non-stationary system states motivated by real-world data, which is a key contribution of our paper. We apply the DPP scheme to the formulated problem and prove the scheme is near-optimal for our system with non-iid system states.

3 System Model and Problem Formulation

We consider a VNF backup problem with time-varying edge computing system states, i.e., request rates, failure probabilities, and resource prices, under availability constraints for VNFs. Some important notations are listed in Table 2.

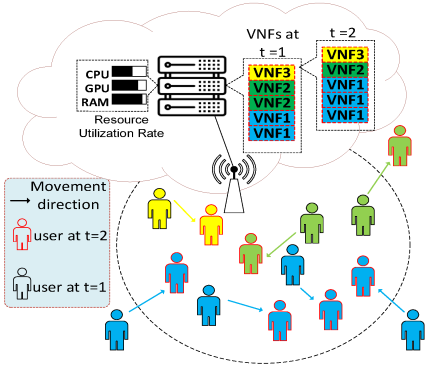


Fig. 2: A simple system diagram for $t = 1$ and 2.

3.1 System Model

We consider a system provider providing VNF services to users over slotted time, i.e., $t \in \{1, 2, \cdots, T\}$, where T is the number of operating time slots. At the beginning of each time slot t, the service provider decides the number of backups for each VNF. The system cost at time slot t is a function of the number of backups of VNFs and the prices of edge resources at the current time slot. Besides, the availability of each type of VNF is a function of the number of backups and the failure probability of the corresponding VNF instance at the current time slot. The goal of the system is to minimize the time average cost of the system with availability constraints for the VNFs. A simple system diagram is shown in Figure 2. The system shown in Figure 2 have three VNFs represented by different colors. VNF 1 and VNF 2 have backups at $t = 1$, while VNF 3 only has a primary instance. The dashed ellipse is the area covered by the edge computing system. Users are labeled by the colors of their requesting VNFs. Users' moving is one of the reasons that cause request rates for VNFs to be time-varying. The backup number of VNF 1 increased to 2 at $t = 2$ partially because VNF 1 has higher request rates.

VNFs: There are V different types of VNFs, and we use $\mathcal{V} = \{1, 2, \cdots, V\} \triangleq [V]$ to denote the set of VNFs[1]. Deploying an instance of VNF v requires different edge resources, e.g., CPU, GPU, and RAM. Assume

1. $[x] \triangleq \{k \in \mathbb{Z}^+ : k \leq x\}$ where $\mathbb{Z}^+$ is the set of positive integers.

there are I types of resources, where I is a relatively small fixed number. Let $s_{v,i}$ be the size of the i^{th} type resource required for deploying an instance of VNF v. Let $\mathbf{s}_v = (s_{v,1}, s_{v,2}, \cdots, s_{v,I})^T$ be the collection of resource sizes required by an instance of VNF v. There is a primary VNF instance has to be deployed for each VNF $v \in \mathcal{V}$ as required by [7]. Let x_v^t represent the number of backup instances of VNF v deployed at time slot t. The feasible space of x_v^t is $\mathcal{X}_v \triangleq \{0, 1, 2, \cdots, X_v\}$, which is motivated by the fact that the number of backups of VNF v cannot be fractional. X_v is the maximum number of backup instances of VNF v at each time slot. Similar to [41] considering edge task offloading, the edge server has limited amount of resources. Let S_i be the total amount of i^{th} resource, and $\mathbf{S}$ represents $(S_1, S_2, \cdots, S_I)^T$. Therefore, there is a constraint on the total resources used at each time slot as follows:

$$\sum_{v \in \mathcal{V}} x_v^t \mathbf{s}_v \leq \mathbf{S} \text{ for } t \in \mathcal{T}. \tag{1}$$

Moreover, we use $\mathbf{x}^t$ to denote the collection of $x_v^t, t \in \mathcal{T}$, i.e., $\mathbf{x}^t = \{x_v^t | v \in \mathcal{V}\} \in \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_V$. $\mathbf{x}^t$ is the online decision of the system that needs to be made at the beginning of time slot t. For each VNF $v \in \mathcal{V}$, its request rate is time-varying because users move in/out of the system and behave differently at different time slots. We use r_v^t to denote the request rate of VNF $v \in \mathcal{V}$ at time slot t. In addition, we use $\mathbf{r}^t$ to denote the collection of $r_v^t, v \in \mathcal{V}$, i.e., $\mathbf{r}^t = \{r_v^t | v \in \mathcal{V}\}$.

Resource Price and System Cost: The edge resources for deploying primary and backup VNF instances have time-varying prices. We use p_i^t to denote the unit price per time slot of the i^{th} type resource at time slot t. $\mathbf{p}^t = (p_1^t, p_2^t, \cdots, p_I^t)$ is the collection of prices of edge resources at time slot t. The system cost at time slot t is the cost of resources consumed for deploying backups. Note that, $\mathbf{p}^t$ is a row vector, $\mathbf{s}^t$ is a column vector, and $\mathbf{p}^t \mathbf{s}_v$ is the cost for deploying a backup instance of VNF v. The system cost at time slot t, denoted by $c(t)$, is as follows:

$$c(t) = \sum_{v \in \mathcal{V}} x_v^t \mathbf{p}^t \mathbf{s}_v. \tag{2}$$

The goal of the system is to minimize the expectation of time average system cost as follows:

$$\min_{\mathbf{x}^t, t \in \mathcal{T}} \quad \frac{1}{T} \sum_{t=1}^{T} c(t) \tag{3}$$

Availability Constraints: For each primary and backup instance of VNF $v \in \mathcal{V}$, it has a failure probability of f_v^t at time slot t. If some VNF instances are unable to meet the QoS requirements due to various reasons, such as server crashes and connection issues, they are considered failures. For each VNF $v \in \mathcal{V}$, f_v^t is time-varying as assumed in [7]. Moreover, we use $\mathbf{f}^t$ to denote the collection of $f_v^t, t \in \mathcal{T}$, i.e., $\mathbf{f}^t = \{f_v^t | v \in \mathcal{V}\}$. The failure probability of VNF v at time slot t, denoted by F_v^t, is the probability of having no available instance of VNF v at time slot t as follows:

$$F_v^t = (f_v^t)^{(1+x_v^t)}. \tag{4}$$

Then, we define the availability of a VNF as the probability of having at least one VNF instance available. The availability of VNF v at time slot t, denoted by a_v^t, is equal to $1 - F_v^t$, i.e.,

$$a_v^t = 1 - F_v^t = 1 - \left(f_v^t\right)^{\left(1+x_v^t\right)}.$$

For each $v \in \mathcal{V}$ and $t \in \mathcal{T}$, there is a constraint for the availability of VNF v at time slot t as follow:

$$a_v^t \geq A_v, \forall v \in \mathcal{V}, \forall t \in \mathcal{T}. \tag{5}$$

We refer to A_v as the minimum availability requirement of VNF v. We use $\mathcal{X}(\mathbf{f}^t)$ to denote the set of $\{\mathbf{x}^t \in \mathcal{X} | a_v^t \geq A_v, \forall v \in \mathcal{V}$ and $\sum_{v \in \mathcal{V}} x_v^t \mathbf{s}_v \leq \mathbf{S}\}$. $\mathcal{X}(\mathbf{f}^t)$ is the set of $\mathbf{x}^t \in \mathcal{X}$ satisfying the minimum availability constraint in (5) and the total resource constraint in (1). $\mathcal{X}(\mathbf{f}^t)$ is a function of $\mathbf{f}^t$, because for different failure probability f_v^t, the minimum $x_v^t \in \mathcal{X}_v$ satisfying the minimum availability constraint in (5) is different.

Moreover, there is a weighted time average availability constraint for each VNF $v \in \mathcal{V}$ as follows:

$$\frac{1}{T} \sum_{t=1}^{T} r_v^t a_v^t \geq \bar{A}_v \bar{r}_v. \tag{6}$$

The above constraint sets a limit for the time average of $r_v^t \cdot a_v^t$ rather than a_v^t. This is motivated by the fact that the VNF with a high request rate is more crucial than that with a relatively low request rate, and the constraint essentially limits the average availability for each unit of the request of each time slot. $\bar{r}_v$ is the average request rate of VNF V over the time horizon, i.e., $\bar{r}_v = \frac{1}{T} \sum_{t=1}^{T} r_v^t$, and $\bar{r}_v$ is

$\mathcal{T} = \{1, 2, \cdots, T\}$	Set of operating time slots
$\mathcal{V} = \{1, 2, \cdots, V\}$	Set of different types of VNFs
$\mathbf{s}_v = (s_{v,1}, \cdots, s_{v,I})$	Sizes of resources of VNF v
$\mathbf{S} = (S_1, S_2, \cdots, S_I)$	Total amount of resources in the system
x_v^t	Number of backup instances of VNF v
$\mathcal{X}_v = \{1, 2, \cdots, X_v\}$	Space for $x_v^t, t \in \mathcal{T}$
$\mathbf{x}^t = \{x_v^t \| v \in \mathcal{V}\}$	Decision at time slot t
$\mathcal{X} \triangleq \mathcal{X}_1 \times \cdots \times \mathcal{X}_V$	Feasible space for $\mathbf{x}^t, t \in \mathcal{T}$
$r_v^t = \bar{r}_v^t + z_{r_v^t}$	Request rate of VNF v at time slot t
$\bar{r}_v^t, t \in \mathcal{T}$	Periodic underlying trend of p_i^t
$z_{r_v^t}, t \in \mathcal{T}$	I.I.D random variable
$\mathbf{r}^t = \{r_v^t \| v \in \mathcal{V}\}$	Collection of $r_v^t, v \in \mathcal{V}$
$p_i^t = \bar{p}_i^t + z_{p_i^t}$	Price of the i^{th} type resource at time slot t
$\bar{p}_i^t, t \in \mathcal{T}$	Periodic underlying trend of p_i^t
$z_{p_i^t}, t \in \mathcal{T}$	I.I.D random variable
$\mathbf{p}^t = (p_1^t, \cdots, p_I^t)$	Prices of resources at time slot t
$f_v^t = \bar{f}_v^t + z_{f_v^t}$	Failure probability of VNF v at time slot t
$\bar{f}_v^t, t \in \mathcal{T}$	Periodic underlying trend of f_v^t
$z_{f_v^t}, t \in \mathcal{T}$	I.I.D random variable
$\mathbf{f}^t = \{f_v^t \| v \in \mathcal{V}\}$	Collection of $f_v^t, v \in \mathcal{V}$
$\mathbf{y}^t = (\mathbf{r}^t, \mathbf{p}^t, \mathbf{f}^t)$	Collection of system state at time slot t
a_v^t	Availability of VNF v at time slot t
A_v	Minimum availability of VNF v
$\bar{A}_v$	Minimum average availability of VNF v
$c(t)$	System cost at time slot t
$\mathcal{X}(\mathbf{p}^t) = \mathcal{X}(\mathbf{y}^t)$	Set of $\{\mathbf{x}^t \in \mathcal{X} \| a_v^t \geq A_v, \forall v \in \mathcal{V}\}$

TABLE 2: Important Notations

known in advance. $\bar{A}_v \cdot \bar{r}_v$ in the above inequality is the minimum availability requirement for VNF v, which is specified and known in advance. We refer to $\bar{A}_v$ as the minimum weighted time average availability requirement of VNF v. Different VNFs have different reliability requirements. That is, $A_v, v \in \mathcal{V}$ and $\bar{A}_v, v \in \mathcal{V}$ can be specified to different values based on the different availability requirements of the VNFs.

3.2 Problem Formulation

Next, we state the problem we formulated above as an online optimization problem, and we refer to the problem as OVBAC, which is short for Online VNF Backup problem under Availability Constraints. OVBAC is as follows:

$$\begin{aligned} \min_{\mathbf{x}^t, t\in\mathcal{T}} \quad & \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[c(t)] \\ \text{s.t.} \quad & a_v^t \geq A_v, \forall v \in \mathcal{V}, \forall t \in \mathcal{T}, \\ & \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[r_v^t a_v^t] \geq \bar{A}_v \bar{r}_v, \forall v \in \mathcal{V}, \\ & \mathbf{x}^t \in \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_V, \forall t \in \mathcal{T}, \\ & \sum_{v\in\mathcal{V}} x_v^t \mathbf{s}_v \leq \mathbf{S} \ \text{ for } t \in \mathcal{T}. \end{aligned} \tag{P}$$

The goal of the problem is to minimize the average cost of the system over time. The first constraint of OVBAC sets a limit to the worst availability for each VNF at each time slot. The second constraint of OVBAC sets a limit to the worst weighted time average availability for each VNF. The third constraint of OVBAC limits the maximum number of backups that can be deployed for each VNF $v \in \mathcal{V}$. The fourth constraint ensures that the utilized resources do not exceed the available resources.

Note that $\mathcal{X}(\mathbf{f}^t) \triangleq \{\mathbf{x}^t \in \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_V | a_v^t \geq A_v, \forall v \in \mathcal{V} \text{ and } \sum_{v\in\mathcal{V}} x_v^t \mathbf{s}_v \leq \mathbf{S}\}$ is the set of $\mathbf{x}^t$ satisfying $\mathbf{x}^t \in \mathcal{X}$ and the minimum availability constraint in (5). Therefore, we can replace the first, third, and last constraints in P by $\mathbf{x}^t \in \mathcal{X}(\mathbf{f}^t)$. We use $\mathbf{y}^t$ to denote the collection of time-varying system states at time slot t, i.e., $\mathbf{y}^t = (\mathbf{r}^t, \mathbf{p}^t, \mathbf{f}^t)$. Therefore, let $\mathcal{X}(\mathbf{y}^t)$ be equivalent to $\mathcal{X}(\mathbf{f}^t)$. For each VNF $v \in \mathcal{V}$, we define a function $H(\mathbf{x}^t, \mathbf{y}^t)$ as follows:

$$h_v(t) = H_v(\mathbf{x}^t, \mathbf{y}^t) = \bar{A}_v \bar{r}_v - r_v^t a_v^t. \tag{7}$$

In addition, we define $C(\mathbf{x}^t, \mathbf{y}^t)$ as follows:

$$c(t) = C(\mathbf{x}^t, \mathbf{y}^t) = \sum_{v\in\mathcal{V}} x_v^t \mathbf{p}^t \mathbf{s}_v. \tag{8}$$

With $\mathcal{X}(\mathbf{f}^t)$, (7), (8), and P, the formulated problem is equivalent to

$$\begin{aligned} \min_{\mathbf{x}^t, t\in\mathcal{T}} \quad & \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[C(\mathbf{x}^t, \mathbf{y}^t)] \\ \text{s.t.} \quad & \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[H_v(\mathbf{x}^t, \mathbf{y}^t)] \leq 0, \forall v \in \mathcal{V}, \\ & \mathbf{x}^t \in \mathcal{X}(\mathbf{y}^t), \forall t \in \mathcal{T}. \end{aligned} \tag{OVBAC}$$

At the beginning of each time slot $t \in \mathcal{T}$, the service provider observes $\mathbf{y}^t$, and chooses a VNF backup decision $\mathbf{x}^t$, where $\mathbf{x}^t \in \mathcal{X}(\mathbf{y}^t)$. **Use ρ^* to denote the optimal objective value of OVBAC**.

3.3 Assumptions

Periodic System States: $\mathbf{y}^t$ is the collection of the edge system states at time slot t. From Figure 1, the request rate has a underlying periodic trend. Since the failure probability of a VNF instance increases as the request rate increases, we assume the failure probabilities of VNF instances are also equal periodic trends plus iid random variables [10]. Moreover, the prices of resources are functions of the utilization ratios of the edge resources which is periodic [42]. The prices of resources also have underlying periodic trends, e.g., lower price during nighttime service hours [43], and we assume the prices equal to periodic trends plus iid random variables. Therefore, for each system state, we assume it equals a periodic trend plus a random variable as follows.

For each $r_v^t \in \mathbf{r}^t$, we assume $r_v^t = \bar{r}_v^t + z_{r_v^t}$. Let $\bar{\mathbf{r}}^t = (\bar{r}_1^t, \bar{r}_2^t, \cdots, \bar{r}_V^t)$ and $\mathbf{z}_{\mathbf{r}^t} = (z_{r_1^t}, z_{r_2^t}, \cdots, z_{r_V^t})$. Assume $p_i^t = \bar{p}_i^t + E_{p_i^t}$ for $p_i^t \in \mathbf{p}^t$. Let $\bar{\mathbf{p}}^t = (\bar{p}_1^t, \bar{p}_2^t, \cdots, \bar{p}_I^t)$ and $\mathbf{z}_{\mathbf{p}^t} = (z_{p_1^t}, z_{p_2^t}, \cdots, z_{p_I^t})$. Assume $f_v^t = \bar{f}_v^t + E_{f_v^t}$ for $f_v^t \in \mathbf{f}^t$. Let $\bar{\mathbf{f}}^t = (\bar{f}_1^t, \bar{f}_2^t, \cdots, \bar{f}_V^t)$ and $\mathbf{z}_{\mathbf{f}^t} = (z_{f_1^t}, z_{f_2^t}, \cdots, z_{f_V^t})$. Moreover, let $\bar{\mathbf{y}}^t = (\bar{\mathbf{r}}^t, \bar{\mathbf{p}}^t, \bar{\mathbf{f}}^t)$ and $\mathbf{z}_{\mathbf{y}^t} = (\mathbf{z}_{\mathbf{r}^t}, \mathbf{z}_{\mathbf{p}^t}, \mathbf{z}_{\mathbf{f}^t})$, and we have $\mathbf{y}^t = \bar{\mathbf{y}}^t + \mathbf{z}_{\mathbf{y}^t}$. We assume $\bar{r}_v^t$, $\bar{f}_v^t$, and $\bar{p}_i^t$ are known in advance and periodic with period of D, and $z_{r_v^t}$, $z_{p_i^t}$ and $z_{f_v^t}$ are independent and identically distributed. We formally state the assumption for system states in Assumption 1.

Assumption 1. System states $\mathbf{y}^t = \bar{\mathbf{y}}^t + \mathbf{z}_{\mathbf{y}^t}, t \in \mathcal{T}$ satisfy

- $\bar{\mathbf{y}}^t$ is periodic with a period of D, and T is an integer multiple of D.
- $\mathbf{z}_{\mathbf{y}^t}$ is independent and identically distributed.

We do not require any information about the distributions of the iid random variables, i.e., variables in $\mathbf{z}_{\mathbf{y}^t}$, beforehand. In addition, when $\bar{\mathbf{y}}^t$ is time-invariant, $\mathbf{y}^t$ is simply independent and identically distributed. That is, our system incorporates the case that some (possibly all) of the system states are iid.

Feasibility of OVBAC: In what follows, we make an assumption to ensure the OVBAC problem is feasible.

Assumption 2. Let $\mathbf{x}_{opt}^t$ be the optimal decision made by an optimal algorithm at each slot t. We assume there exists $\epsilon > 0$ such that

- $\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[H_v(\mathbf{x}_{opt}^t, \mathbf{y}^t)] \leq -\epsilon$ for $v \in \mathcal{V}$.
- $\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[C(\mathbf{x}_{opt}^t, \mathbf{y}^t)] = \rho^*$.

Assumption 2 ensures the OVBAC problem is feasible, and Assumption 2 is also required by [21]. In cases where the minimum availability constraints cannot be satisfied due to extreme scenarios, the problem becomes infeasible, and the system must resort to cloud servers, which results in increased latency.

Boundedness of $h_v(t)$: We assume the request rates of VNFs are bounded. Since $a_v^t \in [0, 1]$ and r_v^t is bounded, we have $h_v(t) = \bar{A}_v \bar{r}_v - r_v^t a_v^t$ is also bounded. Then, we have $\frac{1}{2}\sum_{v\in\mathcal{V}} h_v^2(t)$ is bounded. Let B be an upper bound of $\frac{1}{2}\sum_{v\in\mathcal{V}} h_v^2(t)$ as follows.

Algorithm 1: HEDP-based DPP

```
Choose a parameter μ for DPP;
while t ∈ 𝒯 do
    Call HEDP to get x^t, the optimal solution of P2;
    Set the online decision at time lot t as x^t;
    Update Q_v(t), v ∈ 𝒱 using (9);
end
```

Assumption 3. $|h_v(t)|$ is bounded by h_v, and $\frac{1}{2}\sum_{v\in\mathcal{V}} h_v^2(t)$ is upper bounded by B.

Assumption 3 is used for proving the performance of the proposed algorithm.

3.4 NP-hardness

Next, we show the NP-hardness of the offline version of the proposed problem, where T and $\mathbf{y}^t, t \in [T]$ are known in advance.

Theorem 1. *The offline version of OVBAC is NP-hard.*

Proof. We consider a degenerated version of the problem where there is only one type of VNF, i.e., $\mathcal{V} = \{1\}$ and $\{x_1^t\}_{t=1}^T$ are the decision variables. Moreover, we assume $x_1^t \in \mathcal{X}_1 = \{0, 1\}$, and there is no minimum availability constraint for each VNF i.e., no constraint (5). Moreover, we assume there is only one type of resource, i.e., $I = 1$. Let $\hat{a}_0^t$ be the availability at time slot t under $x_1^t = 0$, and let $\hat{a}_0^t$ be the availability at time slot t under $x_1^t = 1$. We have $\hat{a}_0^t = r_1^t\left(1 - (f_1^t)^1\right)$ and $\hat{a}_1^t = r_1^t\left(1 - (f_1^t)^2\right)$. Then, the degenerated version of the offline OVBAC problem can be stated as follows:

$$\begin{aligned} \min_{x_1^t, t\in\mathcal{T}} \quad & \frac{1}{T}\sum_{t=1}^{T} p_1^t s_1 x^t \\ \text{s.t.} \quad & \frac{1}{T}\sum_{t\in\mathcal{T}} (\hat{a}_1^t - \hat{a}_0^t) x_1^t \geq A_1' \triangleq A_1 - \sum_{t\in\mathcal{T}} \hat{a}_0^t \\ & x_1^t \in \{0,1\}, \forall t \in \mathcal{T}. \end{aligned} \tag{P1}$$

If T and $\mathbf{y}^t, t \in [T]$ are known in advance, P1 is the classic minimum knapsack problem, and is NP-hard [44]. That is a degenerated version of offline OVBAC is NP-hard. Therefore, the offline version of OVBAC is NP-hard. □

Since the offline version of OVBAC is NP-hard, it is even more difficult to solve OVBAC in an online manner, where $\mathbf{y}^\tau, \tau > t$ are not known at time slot t.

4 Algorithm Design For OVBAC

In this section, we design an algorithm for OVBAC and analyze its performance. In Section 4.1, a drift plus penalty scheme is presented for OVBAC, and the scheme needs to solve a optimization problem (P2) at each time slot. The theoretical performance guarantee of the proposed scheme is provided in Section 4.1. In Section4.2, an algorithm named HEDP for P2 is given.

4.1 The Drift Plus Penalty Scheme

In this section, we present the Drift Plus Penalty scheme (DPP) for OVBAC.

Assume there are V virtual queues that correspond to the V types of VNFs. Let $Q_v(t)$ be the backlog of the virtual queue corresponding to VNF v at time slot t. The $Q_v(t+1), v \in \mathcal{V}$ are updated as follow:

$$Q_v(t+1) = [Q_v(t) + h_v(t)]^+, \tag{9}$$

where $[x]^+ \triangleq \max\{x, 0\}$ for $x \in \mathbb{R}$.

The physical meaning of $Q_v(t)$ is the gap between the weighted time average availability of VNF v before time slot t and the minimum weighted time average availability requirement of VNF v. A large $Q_v(t)$ means a large $\sum_{\tau=1}^{t-1} h_v(\tau) = \sum_{\tau=1}^{t-1} \bar{A}_v\bar{r}_v - r_v^\tau a_v^\tau$, and a small $Q_v(t)$ means a small $\sum_{\tau=1}^{t-1} h_v(\tau) = \sum_{\tau=1}^{t-1} \bar{A}_v\bar{r}_v - r_v^\tau a_v^\tau$.

DPP has a single tunable parameter $\mu > 0$. At each time slot t, DPP chooses online decision $\mathbf{x}^t$ by solving the problem, denoted by $P2$, as follows:

$$\begin{aligned} \min_{\mathbf{x}^t} \quad & \mu \cdot c(t) + \sum_{v\in\mathcal{V}} Q_v(t) h_v(t) \\ \text{s.t.} \quad & \mathbf{x}^t \in \mathcal{X}(\mathbf{y}^t). \end{aligned} \tag{P2}$$

P2 at time slot t moves the current weighted availabilities of VNFs from constraints to the objective function, and the coefficient for the weighted availability of VNF v is $Q_v(t)$. The main idea of DPP is as follows. When $Q_v(t)$ is larger, the weighted time average availability of VNF v before time slot t is low, and P2 will yield a $\mathbf{x}^t$ with large a_v^t to compensate the weighted time average availability. On the contrast, when $Q_v(t)$ is small, the availability of VNF v before time slot t is relatively high, and P2 will yield a $\mathbf{x}^t$ with low cost $c(t)$ to minimize the system cost. Eventually, $Q_v(t)$ will be stable at a certain sweet point which can balance the cost and the availabilities. We can set $Q_v(1), \forall v \in \mathcal{V}$ to their corresponding stable values learned from historical data. A detailed method for learning $Q_v(1), \forall v \in \mathcal{V}$ is provided later in Algorithm 2.

An algorithm named Heuristic Enhanced Dynamic Programming (HEDP) for solving P2 is proposed in Section 4.2. HEDP can find the optimal solution of P2 in polynomial time. HEDP-based DPP sets the decision at time slot t as the optimal solution of P2 given by HEDP. We formally state HEDP-based DPP in Algorithm 1.

Next, we prove the performance of HEDP-based DPP. We first define a policy named $\mathbf{y}$-only policy as follows. A policy for OVBAC is called to be a $\mathbf{y}$-only policy if decision $\mathbf{x}^t$ under the policy is a pure function (possibly randomized) of $\mathbf{y}^t$ and the pure functions of a $\mathbf{y}$-only policy for different slots are identical. We then show there exist a $\mathbf{y}$-only policy which is optimal for OVBAC as follows.

Lemma 1. There exist a $\mathbf{y}$-only policy and $\epsilon > 0$ such that for any $\tau \in [T - D]$, we have

$$\begin{aligned} &\frac{1}{D}\sum_{t=\tau+1}^{\tau+D} E[C(\bar{\mathbf{x}}^t, \mathbf{y}^t)] = \rho^* \\ &\frac{1}{D}\sum_{t=\tau+1}^{\tau+D} E[H_v(\bar{\mathbf{x}}^t, \mathbf{y}^t)] \leq -\epsilon, \forall v \in \mathcal{V} \end{aligned} \tag{10}$$

where $\bar{\mathbf{x}}^t$ is the $\mathbf{y}$-only policy's decision at time slot t.

The proof of Lemma 1 is found in our technical report in [45], which is motivated by [21]. The main result of this paper is as follows.

Theorem 2. *If algorithm HEDP for P2 is optimal, under HEDP-based DPP for OVBAC, we have*

- $\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[c(t)]\leq\rho^*+\frac{BD}{\mu}$
- $\lim_{T\to\infty}\sum_{t=1}^{T}\frac{1}{T}E[h_v(t)]\leq 0, \forall v\in\mathcal{V}$.

Proof. Let a $\mathbf{y}$-only policy be a policy for OVBAC which chooses decision based on the current system state only. Different form DPP, a $\mathbf{y}$-only policy makes decisions based on the current system state only. A $\mathbf{y}$-only policy makes decisions independent of $Q_v(t), v\in\mathcal{V}$. From Lemma 1, there exist a $\mathbf{y}$-only policy such that

$$\begin{aligned}&\frac{1}{D}\sum_{t=\tau+1}^{\tau+D}E[C(\bar{\mathbf{x}}^t,\mathbf{y}^t)]=\rho^*\\&\frac{1}{D}\sum_{t=\tau+1}^{\tau+D}E[H_v(\bar{\mathbf{x}}^t,\mathbf{y}^t)]\leq-\epsilon,\forall v\in\mathcal{V}\end{aligned}\tag{11}$$

for any τ, where $\bar{\mathbf{x}}^t$ is the decision for slot t under the $\mathbf{y}$-only policy. $\bar{\mathbf{x}}^t$ is used for proving the performance of DPP and not been implemented. Note that $\mathbf{x}^t$ be the decision for slot t under HEDP-based DPP.

Define $L(t)$ as $L(t)\triangleq\frac{1}{2}\sum_{v\in\mathcal{V}}Q_v^2(t)$, which measures the virtual queue backlogs of the system at slot t. Moreover, define $\Delta L(t)$ as

$$\Delta L(t)\triangleq L(t+1)-L(t).$$

Since B is an upper bound of $\frac{1}{2}\sum_{v\in\mathcal{V}}h_v^2(t)$, we have

$$\begin{aligned}\Delta L(t)=&\frac{1}{2}\sum_{v\in\mathcal{V}}Q_v^2(t+1)-\frac{1}{2}\sum_{v\in\mathcal{V}}Q_v^2(t)\\=&\frac{1}{2}\sum_{v\in\mathcal{V}}\big(Q_v^2(t)+h_v(t)\big)^2-\frac{1}{2}\sum_{v\in\mathcal{V}}Q_v^2(t)\\=&\frac{1}{2}\sum_{v\in\mathcal{V}}h_v^2(t)+\sum_{v\in\mathcal{V}}Q_v(t)h_v(t)\\\leq&B+\sum_{v\in\mathcal{V}}Q_v(t)h_v(t).\end{aligned}\tag{12}$$

From (12), we have

$$\begin{aligned}&\Delta L(t)+\mu c(t)\leq B+\mu c(t)+\sum_{v\in\mathcal{V}}Q_v(t)h_v(t)\\=&B+\mu C(\mathbf{x}^t,\mathbf{y}^t)+\sum_{v\in\mathcal{V}}Q_v(t)H_v(\mathbf{x}^t,\mathbf{y}^t)\\\overset{(a)}{\leq}&B+\mu C(\bar{\mathbf{x}}^t,\mathbf{y}^t)+\sum_{v\in\mathcal{V}}Q_v(t)H_v(\bar{\mathbf{x}}^t,\mathbf{y}^t).\end{aligned}\tag{13}$$

(a) of (13) holds, because $\mathbf{x}^t$ given by HEDP is the optimal solution of P2. Summing (13) up over a period from t_0 to t_0+D-1, we have

$$\begin{aligned}&\sum_{t=t_0}^{t_0+D-1}\Delta L(t)+\mu\sum_{t=t_0}^{t_0+D-1}c(t)\\\leq&BD+\mu\sum_{t=t_0}^{t_0+D-1}C(\bar{\mathbf{x}}^t,\mathbf{y}^t)\\&+\sum_{t=t_0}^{t_0+D-1}\sum_{v\in\mathcal{V}}Q_v(t)H_v(\bar{\mathbf{x}}^t,\mathbf{y}^t).\end{aligned}\tag{14}$$

Since $|h_v(t)|$ is bounded by h_v, we have $Q_v(t)\leq Q_v(t_0)+(t-t_0)h_v$. Taking expectation of (14), for $t>t_0$ we have

$$\begin{aligned}&\sum_{t=t_0}^{t_0+D-1}\mathbb{E}[\Delta L(t)+\mu c(t)]\\\leq&BD+\sum_{t=t_0}^{t_0+D-1}\mu\mathbb{E}[C(\bar{\mathbf{x}}^t,\mathbf{y}^t)]\\&+\sum_{t=t_0}^{t_0+D-1}\sum_{v\in\mathcal{V}}\mathbb{E}[Q_v(t)H_v(\bar{\mathbf{x}}^t,\mathbf{y}^t)]\\\overset{(a)}{\leq}&BD+\mu\sum_{t=t_0}^{t_0+D-1}\mathbb{E}[C(\bar{\mathbf{x}}^t,\mathbf{y}^t)]\\&+\sum_{t=t_0}^{t_0+D-1}\sum_{v\in\mathcal{V}}(\mathbb{E}[Q_v(t_0)]+(t-t_0)h_v)\mathbb{E}[H_v(\bar{\mathbf{x}}^t,\mathbf{y}^t)]\\\overset{(b)}{\leq}&BD+\sum_{v\in\mathcal{V}}h_v^2\frac{D(D-1)}{2}+\mu D\rho^*\leq D^2B+\mu D\rho^*.\end{aligned}\tag{15}$$

(a) of (15) holds because $\bar{\mathbf{x}}^t$ is given by a $\mathbf{y}$-only policy and is independent of $Q_v(t)$, and (b) of (15) holds because of the properties of the $\mathbf{y}$-only policy in (11). Let n is the integer satisfying $nD=T$, i.e., n is the number of periods of the system's operation horizon. Summing (15) up over $t_0\in\{1,D+1,2D+1,\cdots,(nD-D+1)\}$, we have

$$\begin{aligned}&(BD+\mu\rho^*)nD\geq\sum_{\tau\in[nD]}\mathbb{E}[\Delta L(\tau)+\mu c(\tau)]\\=&\mathbb{E}[L(nD)]-\mathbb{E}[L(1)]+\mu\sum_{\tau\in[nD]}\mathbb{E}[c(\tau)]\\\overset{(a)}{\geq}&\mu\sum_{\tau\in[nD]}\mathbb{E}[c(\tau)]-\mathbb{E}[L(1)].\end{aligned}\tag{16}$$

(a) of (16) holds because $L(t)$ and $L(1)$ are defined to be non-negative. Dividing both sides of (16) by μnD, we have

$$\begin{aligned}&\frac{1}{nD}\sum_{\tau\in[nD]}\mathbb{E}[c(\tau)]=\frac{1}{T}\sum_{\tau\in[T]}\mathbb{E}[c(\tau)]\\\leq&\pi\rho^*+\frac{BD}{\mu}+\frac{L(1)}{\mu T}.\end{aligned}\tag{17}$$

Letting $T\to\infty$ proves the first part of the theorem, i.e.,

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[c(t)]\leq\rho^*+\frac{BD}{\mu}.$$

Next, we focus on the last part of the theorem, i.e., $\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[h_v(t)]\leq 0$ for $v\in\mathcal{V}$. From (16), we have

$$\mathbb{E}[L(nD)]\leq(BD+\mu\rho^*)nD.\tag{18}$$

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3282156

(18) is equivalent to (19) as follows:

$$\frac{1}{2}\sum_{v\in\mathcal{V}}\mathbb{E}[Q_v^2(nD)] \leq (BD+\mu\rho^*)nD. \tag{19}$$

Since $\mathbb{E}[Q_v^2(nD)] \geq \mathbb{E}^2[Q_v(nD)]$, from (19), we have

$$\mathbb{E}[Q_v(nD)] \leq \sqrt{2(BD+\mu\rho^*)nD}, \forall v \in \mathcal{V}. \tag{20}$$

Dividing (20) by nT and letting $n \to \infty$, we have

$$\lim_{n\to\infty}\frac{\mathbb{E}[Q_v(nD)]}{nD} \leq \frac{\sqrt{2(BD+\mu\rho^*)nD}}{nD}, \forall v \in \mathcal{V}. \tag{21}$$

For any $t = nD+\tau, \tau \in [D]$, from 21, we have

$$\lim_{t\to\infty}\frac{\mathbb{E}[Q_v(t)]}{t} \leq \lim_{n\to\infty}\frac{\mathbb{E}[Q_v(nD)]+h_vD}{nD} = 0. \tag{22}$$

From [21], [46], (22) implies

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}E[h_v(t)] \leq 0, \forall v \in \mathcal{V}$$

which proves the last part of the theorem. □

Note that Theorem 2 holds is irrelevant to the setting of $Q_v(1), v \in \mathcal{V}$. Since ρ^* is the optimal solution of OVBAC and μ is a parameter of DPP, $\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[c(t)] \leq \rho^* + \frac{BD}{\mu}$ in Theorem 2 means that if T is large, the objective value of OVBAC under HEDP-based DPP can be arbitrarily close to ρ^* by setting a relatively large μ. In addition, for any finite T, we have

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[c(t)] \leq \rho^* + \frac{BD}{\mu}, \text{ if } Q_v(1) = 0, \forall v \in \mathcal{V}.$$

The proof of the above statement is similar to the proof of Theorem 2, so we omit it. On the other hand, $\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}E[h_v(t)] \leq 0, \forall v \in \mathcal{V}$ in Theorem 2 means that the weighted time average availability constraints of VNFs will be satisfied under HEDP-based DPP if T is large.

When we implement HEDP-based DPP, $Q_v(t)$ will be stable at a sweet point after a certain number of time slots. After $Q_v(t)$ being stable, the cost will be near-optimal and the weighted time average availability constraints will be satisfied. By presetting $Q_v(1), v \in \mathcal{V}$ to its stable backlogs, simulation results show the weighted time average availability constraints will be satisfied and $\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[c(t)] \leq \rho^* + \frac{BD}{\mu}$ will hold for finite T (see Section 5).

Next, we provide a way to learning stable virtual queue backlogs at the beginning. Assume we have historical data of the system states of length $T' = nD$ time slots, where $n \in \mathbb{N}^+$. If $\frac{T'}{D} \notin \mathbb{N}^+$, we only use the latest $\lfloor\frac{T'}{D}\rfloor D$ slots' data. Let $\mathbf{y}^j$ be the j^{th} slot's system state of the historical data. We set $Q_v(1) = 0, \forall v \in \mathcal{V}$. We apply DPP to the problem and repetitively feed the previous data to the system. Let J be the length of data that has been fed to the system. The learning algorithm terminates when $\sum_{j=J-D+1}^{J}Q_v(j) \leq \frac{1}{K}\sum_{j=J-KD+1}^{J}Q_v(j)$, where $K \in \mathbb{N}^+$ is a parameter. Then, we set the initial queue backlogs to be $\frac{1}{D}\sum_{j=J-D+1}^{J}Q_v(j), v \in \mathcal{V}$. We formally state the algorithm for leaning the stable queue backlogs in Algorithm 2. Note that, Algorithm 2 is executed before the start of the system; thus, it will not affect the decision time at each time slot.

Algorithm 2: Leaning stable queue backlogs

```
Set J = 1;
while ∃v ∈ V such that
  Σ_{j=J-D+1}^{J} Q_v(j) ≥ (1/K) Σ_{j=J-KD+1}^{J} Q_v(j) do
    Set J = J + 1;
    Let y^{J%N} be the current system states;
    Call HEDP to get the optimal solution of P2;
    Update Q_v(J+1), v ∈ V using (9);
end
J is the number of learning iterations;
Setting Queue backlog of VNF v at the beginning as
  (1/D) Σ_{j=J-D+1}^{J} Q_v(j) for v ∈ V.
```

4.2 Algorithm Design for P2

In this section, we design an algorithm named Heuristic Enhanced Dynamic Programming (HEDP) algorithm for P2. By substituting the expressions of $h_v(t)$, $c(t)$, and $\mathcal{X}(\mathbf{y}^t)$ into P2, P2 can be rewritten as follows:

$$\begin{aligned}
\min_{\mathbf{x}^t} \quad & \mu\sum_{v\in\mathcal{V}}x_v^t\mathbf{p}^t\mathbf{s}_v + \sum_{v\in\mathcal{V}}Q_v(t)\left(\bar{A}_v\bar{r}_v - r_v^t\left(1-\left(f_v^t\right)^{1+x_v^t}\right)\right)\\
\text{s.t.} \quad & x_v^t \in \{0,1,\cdots,X_v\}, \forall v \in \mathcal{V}\\
& \left(1-\left(f_v^t\right)^{1+x_v^t}\right) \geq A_v, \forall v \in \mathcal{V},\\
& \sum_{v\in\mathcal{V}}x_v^t\mathbf{s}_v \leq \mathbf{S}.
\end{aligned} \tag{P3}$$

Define $f_v\left(x_v^t\right)$ as follows:

$$\begin{aligned}
f_v\left(x_v^t\right) = \mu\mathbf{p}^t\mathbf{s}_v x_v^t + & Q_v(t)r_v^t\left(f_v^t\right)^{1+x_v^t}\\
& + Q_v(t)(\bar{A}_v\bar{r}_v - r_v^t).
\end{aligned} \tag{23}$$

The objective function of P3, denoted by O_{P3}, is equivalent to

$$O_{P3} = \sum_{v\in\mathcal{V}}f_v\left(x_v^t\right). \tag{24}$$

For each VNF $v \in \mathcal{V}$, let $\underline{X}_v^t$ be the minimum x_v^t such that the current availability is no less than the minimum availability A_v, i.e.,

$$\underline{X}_v^t \triangleq \min\left\{x_v^t \in \mathbb{N} | \left(1-\left(f_v^t\right)^{1+x_v^t}\right) \geq A_v\right\}. \tag{25}$$

From the monotonicity of availability a_v^t, the minimum availability constraints will be satisfied for any $x_v^t \geq \underline{X}_v^t$. Therefore, the first two constraints of P3 is equivalent to

$$x_v^t \in \{\underline{X}_v^t, \underline{X}_v^t+1, \cdots, X_v\}, \forall v \in \mathcal{V}. \tag{26}$$

P3 is equivalent to minimize O_{P3} in (24) subjecting to $\sum_{v\in\mathcal{V}}x_v^t\mathbf{s}_v \leq \mathbf{S}$ and the constraints in (26). We then design a dynamic programming-based algorithm for P3. We can prove that the problem is NP-hard by showing an induction form knapsack problem. We assume S_i and s_i are integers. If S_i and s_i are not integers, we can change the units of resources to make S_i and s_i integers. Let

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3282156

Algorithm 3: HEDP for P2

```
Compute x̂^t by (29);
if Σ_{v∈V} x̂_v^t s_v ≤ S then
    Return x̂^t;
else
    for s_i ∈ {0,1,··· S_i}, i ∈ [I] do
        Update h(1,s) by (27);
    end
    for k = {2,3,··· ,V} do
        for s_i ∈ {0,1,··· S_i}, i ∈ [I] do
            Update h(k,s) by (28);
        end
    end
    Get optimal x^t by tracking back matrix h(k,s);
    Return the optimal x^t;
end
```

$h(k,\mathbf{s}) = h(k,s_1,\cdots,s_I), k \in [V], \mathbf{s} \in \{\mathbb{Z}^I | 0 \le \mathbf{s} \le \mathbf{S}\}$ be the optimum of

$$\begin{aligned} \min_{\mathbf{x}^t} \quad & \sum_{v=1}^{k} f_v(x_v^t) \\ \text{s.t.} \quad & x_v^t \in \{\underline{X}_v, \cdots, X_v\}, \forall v \in [k] \\ & \sum_{v=1}^{k} x_v^t \mathbf{s}_v \le \mathbf{s}. \end{aligned} \tag{P4}(k,\mathbf{s})$$

Define $h(k,\mathbf{s}) = \infty$ if there exists $s_i < 0$, where $h(k,\mathbf{s})$ represents the optimum of P3 if there are only the first k types of VNFs and the total amount of resource is $\mathbf{s} = (s_1, s_2, \cdots, s_I)$. When $k = 1$, for $s_i \in \{0,1,\cdots,S_i\}, i \in [I]$, we have

$$h(1,\mathbf{s}) = \begin{cases} \text{optimum of P4}(1,\mathbf{s}), & \text{if P4}(1,\mathbf{s}) \text{ is feasible} \\ \infty, & \text{otherwise.} \end{cases} \tag{27}$$

When $k > 1$, for $s_i \in \{0,1,\cdots,S_i\}, i \in [I]$, we have

$$h(k,\mathbf{s}) = \min_{x_k^t \in \{\underline{X}_v^t, \cdots, X_v\}} \left\{ h(k-1, s - x_k^t \mathbf{s}_k) + f_k(x_k) \right\}. \tag{28}$$

Then we have the optimum of P3 is $h(V,\mathbf{S})$ by the definition of $h(k,\mathbf{s})$, and we can get the optimal solution by tracking back the calculations already performed.

If constraint $\sum_{v\in\mathcal{V}} x_v^t \mathbf{s}_v \le \mathbf{S}$ is omitted, P3 will be much more easier to solve. We provide a heuristic enhancement to the above dynamic programming-based algorithm in what follows. We first derive the optimal solution of P3 under the assumption that constraint $\sum_{v\in\mathcal{V}} x_v^t \mathbf{s}_v \le \mathbf{S}$ is omitted. Since each variable in (24) and (26) is not coupled with others, the optimal solution of P3 is equivalent to the collection of optimal solutions of minimizing $f_v(x_v^t)$ subjecting to $x_v^t \in \{\underline{X}_v^t, \underline{X}_v^t + 1, \cdots, X_v\}$ for $v \in \mathcal{V}$. Let $\hat{x}_v^t$ be the optimal solution of P5 as follows:

$$\begin{aligned} \min_{x_v^t} \quad & f_v(x_v^t) \\ \text{s.t.} \quad & x_v^t \in \{\underline{X}_v^t, \underline{X}_v^t + 1, \cdots, X_v\}. \end{aligned} \tag{P5}$$

We have $\hat{x}_v^t$, the optimal solution of P5, is equal to

$$\hat{x}_v^t \triangleq \arg\min\{f_v(x) | x \in \{\underline{X}_v^t, \underline{X}_v^t + 1, \cdots, X_v\}\}. \tag{29}$$

Let $\hat{\mathbf{x}}^t \triangleq (\hat{x}_1^t, \hat{x}_2^t, \cdots, \hat{x}_V^t)$. Then, at the beginning of each slot t, we first compute $\hat{\mathbf{x}}^t$. If $\sum_{v\in\mathcal{V}} \hat{x}_v^t \mathbf{s}_v \le \mathbf{S}$, $\hat{\mathbf{x}}^t$ is the optimal solution of P3; Otherwise, call the dynamic programming-based algorithm proposed above. We formally state the heuristic-enhanced dynamic programming-based algorithm in Algorithm 3.

Next, we consider the time complexity of the dynamic programming subroutine. For the case that $I > 1$, HEDP maintains a $I+1$ dimensional matrix $h(k, s_1, \cdots, s_I)$. Usually, I is a small fixed number, e.g., $I = 3$ when CPU, GPU, and RAM are the bottleneck resources. Since the algorithm runs in an online manner, the time complexity for computing an online decision at each time slot needs to be low. The time complexity for computing each $h(k, s_1, \cdots, s_I)$ is upper bounded by $\bar{X} = \max\{X_v | v \in \mathcal{V}\}$. When $I = 1$, the time complexity of HEDP is $O(VS\bar{X})$. For the case that $I > 1$, we have the time complexity of the dynamic programming method is $O(V\bar{S}^I\bar{X})$, where $\bar{S}$ is the mean of $S_i, i \in \{1,2,\cdots,I\}$ and $\bar{S}^I \ge \prod_{i=1}^{I} S_i$. In addition, since the time complexity for computing $\hat{\mathbf{x}}^t$ is $O(\bar{X}V)$, we have the time complexity of HEDP is $O(V\bar{S}^I\bar{X}) + O(\bar{X}V) = O(V\bar{S}^I\bar{X})$ as shown in the following theorem.

Theorem 3. *HEDP can get the optimal solution of P3 in $O(V\bar{S}^I\bar{X})$ time complexity.*

The optimality of HEDP comes from the definition and dynamics of $h(k,\mathbf{s})$, and the complexity comes from the size of $h(k,\mathbf{s})$. The time complexity for the HEDP-based DPP to computing an online decision at each time slot is pseudo-polynomial, which is efficient.

5 Simulation

In this section, we evaluate the performance of the proposed algorithm over a wide range of settings. In this paper, we implement our simulations using MATLAB R2017b in an HP Pavilion 15-cb0xx laptop with 16GB RAM and i7-7700HQ CPU running of Windows 10 OS.

5.1 Simulation Setup

For the simulation setup, we consider a system with 20 types of VNFs, i.e., $V = 20$. Moreover, let the maximum number of backups of each VNF be five, i.e., $X_v = 5, \forall v \in \mathcal{V}$, as did by [7]. Let each time slot represent an hour. Set $D = 24$, which means the underlying trends of system states are periodic with a period of a day. We set $T = 120$. That is, the system lasts for 5 days. We first get requests rates of length 24 time slots by taking the average of the 6-period real-world values; then, we smooth the 24-slots values by taking an average with its closest 5 values; finally, we get the waveform of average real-world 24-hour request rates as shown in Figure 3. Let r be a periodic waveform where each period of r is equal to the waveform in Figure 3. We generate $\bar{r}_v^t, v \in [\mathcal{V}]$ based on the pattern in Figure 3 as follows. We let $\bar{r}_v^t = B_v + C_v \cdot r(t - \phi_v)$, where B_v and C_v are real numbers and $\phi_v \in [0,1,\cdots,D-1]$ is a randomly generated phase shift. Motivated by the fact that different VNFs have different peak hours, ϕ_v is different for different VNF $v \in \mathcal{V}$. There is a positive correlation between $\bar{\mathbf{f}}^t$ and $\bar{\mathbf{r}}^t$ since the failure probability increases as the requests rate

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3282156

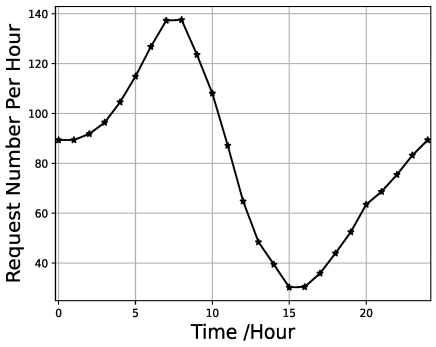


Fig. 3: Waveform of average real-world 24-hour request rates.

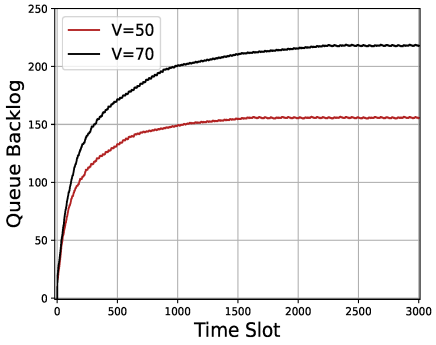


Fig. 4: Queue backlog of VNF 1 v.s. time slots under $\mu = \{50, 70\}$.

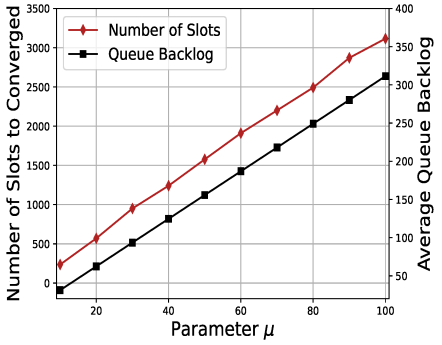


Fig. 5: Number of iterations to converge and queue backlog v.s. parameter μ.

increases [10]. Therefore, we assume the periodic trend of failure probability of VNF v is a function of the request rate r_v^t. To be more specific, let $\bar{f}_v^t = D_v + E_v \cdot r_v^t$ where D_v, E_v be two real numbers such that $\bar{f}_v^t = D_v + E_v \cdot r_v^t$ is in the range of $[0.1, 0.15]$. We assume $z_{f_v^t}$ is randomly drawn from $[0, 0.05]$, and the failure probability of each VNF instance is in the range of $[0.10, 0.20]$, as assumed by [3]. We set the minimum availability for each VNF to be 0.95, i.e., $A_v = 0.95, \forall v \in \mathcal{V}$. The price of each VNF instance, i.e., $\mathbf{p}_v^t \cdot s_v$, is randomly drawn from $[1, 2]$ dollar for $v \in \mathcal{V}$. The unit of cost in the following simulations is a dollar if it is not specified in particular. Similar to [24], we set $I = 1$, and s_v is randomly drawn from integers ranging between two and four units. Additionally, the total capacity of the system, S, is set to 200 units. Moreover, the weighted time average availabilities for VNFs are in the range of $[0.990, 0.998]$, i.e., $\bar{A}_v$ is in the range of $[0.990, 0.998]$. The settings of other parameters, e.g., μ, $z_{r_v^t}$, B_v, $\bar{A}_v$, and C_v vary in different simulations and will be illustrated in detail in the following sections.

5.2 Baselines

We compare the performance of HEDP-based DPP with three types of baselines as follows.

The first baseline is the BSPS algorithm proposed in [24]. In particular, at the beginning of each time slot t, the BSPS algorithm estimates the average system states using online bandit learning, and then solves the online problem by assuming the total time slots, T, equals the current slot number t. As our system is aware of the system states at the beginning of each time slot, we assume that the BSPS algorithm utilizes the actual system states rather than the estimated average system states at each time slot. Furthermore, as the BSPS algorithm requires solving an integer optimization problem at the start of each time slot, we assume that it utilizes the commercially available GUROBI solver [47] to resolve the issue at each time slot.

The second baseline is named Relaxation and Rounding (RR). Relaxation and Rounding has been widely used for VNF placement and backup problems with integer variables [15], [48], [49]. RR is an offline algorithm where system states $\mathbf{y}^t, t \in \mathcal{T}$ are known in advance. Since HEDP-based DPP runs in an online manner where future system states are unknown, RR needs a stronger assumption compared to our algorithm. RR first relaxes the integer constraints of $x_v^t, \forall v \in \mathcal{V}, t \in \mathcal{T}$. Without the integer constraints, OVBAC becomes a convex optimization problem, and we apply a convex optimization solver, e.g. cvx [50], to get the optimal solution of the relaxed OVBAC. Since the optimal variables of the relaxed OVBAC are not integer, we then round the optimal variables to integers. We use threshold rounding method for it, where there is a threshold named $\pi \in (0, 1)$. Under the threshold rounding, available x is rounded to $\lceil x \rceil$ if the fractional part of x is larger than the given threshed. Also, the minimum availability constraint for each VNF has to be satisfied. If the minimum availability constraint is violated, we directly round the corresponding variable to the least integer greater than it. Moreover, by tuning parameter π, the weighted time average availability will be satisfied. To be more specific, we choose π as the minimum value in $\{0, 0.1, 0.2, \cdots, 1\}$ such that the weighted time average availability constraint is satisfied.

The third type of baseline, named Single Slot scheme (SS), breaks down the weighted time average constraints into constraints for each time slot. In other words, SS breaks down the weighted time average constraints to constraints for each time slot, which can be regarded as a static algorithm. We consider two different algorithms, SS1 and SS2, with the same basic idea of SS. SS1 assumes time slots are independent, and it chooses the decision at each time slot by minimizing the cost at the current slot subjecting to $r_v^t a_v^t \geq \bar{r}_v \bar{A}_v^t, v \in \mathcal{V}$. If there is no x_v^t can satisfy $r_v a_v^t \geq \bar{r}_v \bar{A}_v^t$, SS1 sets x_v^t to 5, the maximum allowed number of backups. SS2 assumes time slots are independent, and it chooses the decision at each time slot by minimizing the cost at the current slot subjecting to $a_v^t \geq \bar{A}_v^t$,. If there is no x_v^t can satisfy $a_v^t \geq \bar{A}_v^t$, SS2 sets x_v^t to x_v, the maximum allowed number of backups.

5.3 Presetting Queue Backlogs

For the following simulations, we automatically learn stable values of $Q_v(1), \forall v \in \mathcal{V}$ using Algorithm 2 with $K = 10$, and preset $Q_v(1), v \in \mathcal{V}$ to the corresponding stable values. We randomly generate $\mathbf{y}^t$ of length $10D$ as the previous data. First, we set $\mu = \{50, 70\}$. Set A_v, B_v to the values such that $\bar{r}_v^t$ is in the range of $[40, 48]$ and $z_{r_v^t}$ is randomly drawn from $[-4, 4]$. Figure 4 shows the virtual queue backlog of VNF 1 versus the number of learning iterations. As we can

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3282156

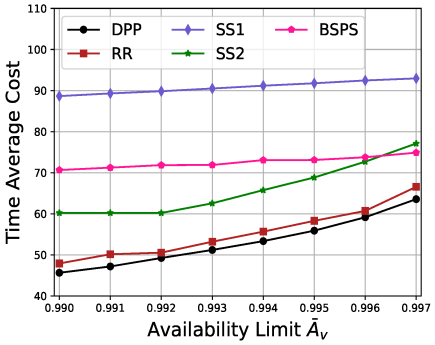


Fig. 6: Time average cost of the system v.s. time average availability limit.

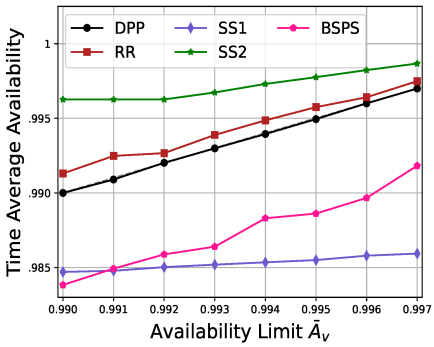


Fig. 7: Time average availability v.s. time average availability limit.

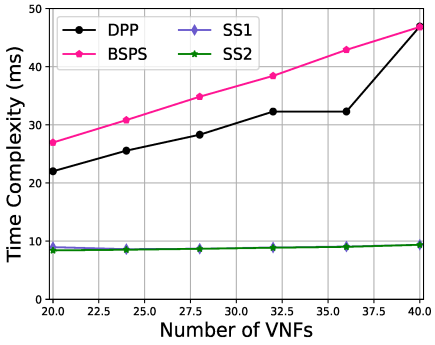


Fig. 8: Time complexity of online algorithms v.s. number of VNFs.

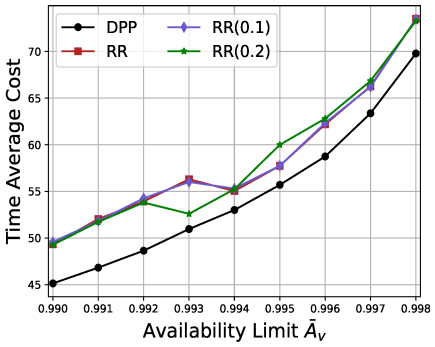


Fig. 9: Time complexity of online algorithms v.s. number of VNFs.

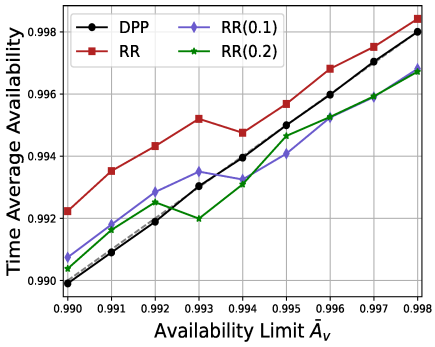


Fig. 10: Time complexity of online algorithms v.s. number of VNFs.

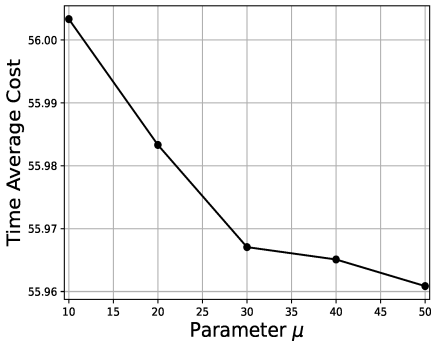


Fig. 11: Time Average Cost of the DPP Algorithm v.s. Parameter μ.

see from Figure 4, the queue backlog of VNF 1 will be stable after a given number of learning iterations. Also, we show the relationship between the number of iterations needed for the queue of VNF 1 to be stable and the parameter of HEDP-based DPP, i.e., μ. As we can see from the black line in Figure 5, the number of iterations needed for the queue of VNF 1 to be stable is linear to parameter μ. In addition, the black line in Figure 5 shows that the stable queue backlog is linear to parameter μ. Since we have $\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[c(t)] \leq \rho^* + \frac{BD}{\mu}$ from Theorem 2, there is a trade-off between the cost of the system and the number of learning iterations. As μ increases, the performance bound of HEDP-based DPP becomes better, and the number of learning iterations will increase.

5.4 Simulation Results

In this section, we evaluate the performance of the proposed algorithm. We first compare the performance of HEDP-based DPP, BSPS, RR, SS1 and SS2 under different settings as follows. We set $A_v, v \in \mathcal{V}$ to be 0.9 and set B_v, C_v to the values such that $\bar{r}_v^t$ is in the range of $[40, 48]$. $z_{r_v^t}$ is randomly drawn from $[-4, 4]$. In addition, ϕ_v is randomly draw from $\{0, 1, \cdots, D-1\}$ for $v \in \mathcal{V}$. The iid random components of systems states are generated as follows. $z_{r_v^t}, v \in \mathcal{V}$ are randomly drawn from $[0, 8]$. Figure 6 shows the cost of HEDP-based DPP, BSPS, RR, SS1, and SS2 under $\bar{A}_v = \{0.990, 0.991, \cdots, 0.998\}, \forall v \in \mathcal{V}$ and $\mu = 50$. As we can see from Figure 6, the time average cost of HEDP-based DPP is less than that of the classic offline algorithm RR and significantly less than that of online algorithms SS1 and SS2. In addition, HEDP-based DPP outperforms BSPS. BSPS assumes $T = t$ at each time slot t, and the time average constraint has to be satisfied for T equals each t, which is similar to the SS scheme. To be more specific, on average, the cost of HEDP-based DPP is 6%, 19%, 28%, and 42% less than that of RR, SS1, BSPS, and SS2, respectively. Since RR is a widely accepted offline algorithm where future system states are given and our algorithm is an online algorithm, our algorithm has a satisfactory cost. The costs under DPP, RR, SS1, and SS2 increase as $\bar{A}_v$ increases, which reflects the trade-off between the system cost and availability. In addition, Figure 7 shows the weighted time average availability of VNF 1 under HEDP-based DPP with $\mu = 50$, BSPS, RR, SS1, and SS2 v.s. the weighted time average availability limit $\bar{A}_v = \{0.990, 0.991, \cdots, 0.998\}$. From Figure 7, the weighted time average availability constraints are satisfied under our algorithm, RR, and SS2. Figure 7 can validate that the weighted time average availability constraints under finite T can be satisfied by DPP by presetting queue backlogs to corresponding stable values. In addition, we compare the time complexity of HEDP-based DPP with $\mu = 50$, BSPS, SS1, and SS2 under different numbers of VNFs. In particular, we set the capacity of the system, S, to be six times the number of VNFs. The reason why we omit RR is that RR is not an online algorithm. As shown in Figure 8, The time complexity of HEDP-based DPP is comparable to that of BSPS using the commercial GUROBI solver, both of which take tens of milliseconds, and they are considered efficient.

The SS1 and SS2 algorithms have low time complexity due to their heuristic nature, but their performance is relatively poor. The time complexity of HEDP-based DPP and BSPS increases as the scale of the system increases.

From Figure 6, the performance of the offline algorithm RR is comparable to HEDP-based DPP, where RR knows future system states precisely. Motivated by the fact that it is impossible to have a perfect system states predictor, we assume the offline algorithm RR makes decisions based on future states with prediction errors. In particular, let RR(0.1) be RR with a prediction error of 0.1 where each future system state is generated by multiplicating the corresponding system state and a factor randomly drawn from $[0.9, 1.1]$. Similarly, RR(0.2) is RR with a prediction error of 0.2 where each future system state is generated by multiplicating the corresponding system state and a factor randomly drawn from $[0.8, 1.2]$. As we can see from Figure 9, the costs of RR, RR(0.1), and RR(0.2) are higher than the cost of DPP. Figure 10 shows that RR(0.1) and RR(0.2) may violate the weighted time average availability constraints. That is, RR can not meet the weighted time average availability constraints in the case that future systems have prediction errors. Next, we set $\bar{A}_v$ to be 0.995, and other parameters are the same as above. We compare the time average cost of HEDP-based DPP under $\mu = \{10, 20, 30, 40, 50\}$. As shown in Figure 11, the time average cost of HEDP-based DPP decreases as μ increases, which matches the trade-off between cost and parameter μ stated in Theorem 2, i.e., $\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[c(t)] \leq \rho^* + \frac{BD}{\mu}$.

6 Conclusion

In this paper, we have studied the online VNF backup problem under reliability constraints in edge environments to minimize the time average system cost. We made use of system state characteristics observed from real-world data to facilitate the problem formulation. We have proved the offline version of the formulated problem is NP-hard. We proposed an online stochastic scheme named DPP for the problem and proved the proposed scheme for the problem is near-optimal. In particular, the system cost under the proposed scheme can be arbitrarily close to the optimal system cost of the problem if the time horizon goes to infinity. At each time slot, the proposed scheme needs to solve a combinatorial problem. We proposed a dynamic programming-based algorithm that can solve the optimization problem optimally in pseudo-polynomial time. Simulation results based on real-world data have shown that the proposed scheme significantly outperforms the baselines. In particular, the proposed online scheme beats a classic offline algorithm used in practice. In this paper, we modeled the system states as periodical baselines plus iid random noises. One interesting future work is investigating a more general scenario where the random noises are non-iid.

Acknowledgments

We thank Prof. Yaodong Huang for providing the data used in this paper. This work was supported in part by the National Science Foundation under grant numbers CCF-1730291, CCF-2046444, CNS-2146909, CNS-2106027, and CNS-2214980.

References

[1] W. Liang, Y. Ma, W. Xu, Z. Xu, X. Jia, and W. Zhou, "Request reliability augmentation with service function chain requirements in mobile edge computing," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2021.

[2] M. Huang, W. Liang, X. Shen, Y. Ma, and H. Kan, "Reliability-aware virtualized network function services provisioning in mobile edge computing," *IEEE Transactions on Mobile Computing*, vol. 19, no. 11, pp. 2699–2713, 2020.

[3] J. Zhang, Z. Wang, C. Peng, L. Zhang, T. Huang, and Y. Liu, "Raba: Resource-aware backup allocation for a chain of virtual network functions," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 2019, pp. 1918–1926.

[4] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[5] S. Yang, F. Li, S. Trajanovski, X. Chen, Y. Wang, and X. Fu, "Delay-aware virtual network function placement and routing in edge clouds," *IEEE Transactions on Mobile Computing*, vol. 20, no. 2, pp. 445–459, 2021.

[6] S. Gamal, Z. Zheng, and B. Ji, "Placement and allocation of virtual network functions: Multi-dimensional case," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2022.

[7] X. Shang, Y. Liu, Y. Mao, Z. Liu, and Y. Yang, "Greening reliability of virtual network functions via online optimization," in *2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS)*, 2020, pp. 1–10.

[8] S. Jošilo and G. Dán, "Joint wireless and edge computing resource management with dynamic network slice selection," *arXiv preprint arXiv:2001.07964*, 2020.

[9] G. Wang, L. Zhang, and W. Xu, "What can we learn from four years of data center hardware failures?" in *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2017, pp. 25–36.

[10] J. Kang, O. Simeone, and J. Kang, "On the trade-off between computational load and reliability for network function virtualization," *IEEE Communications Letters*, vol. 21, no. 8, pp. 1767–1770, 2017.

[11] J. Comden, S. Yao, N. Chen, H. Xing, and Z. Liu, "Online optimization in cloud resource provisioning: Predictions, regrets, and algorithms," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 3, no. 1, Mar. 2019. [Online]. Available: https://doi.org/10.1145/3322205.3311087

[12] T. Lawrence, "Evening internet 'rush-hour' affects broadband users - news - gadgets & tech," in *The Independent*, 2013.

[13] Y. Sang, B. Ji, G. R. Gupta, X. Du, and L. Ye, "Provably efficient algorithms for joint placement and allocation of virtual network functions," in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE, 2017, pp. 1–9.

[14] D. B. Oljira, K. Grinnemo, J. Taheri, and A. Brunstrom, "A model for qos-aware vnf placement and provisioning," in *IEEE NFV-SDN 2017 - IEEE Conference on Network Function Virtualization and Software Defined Networks*. IEEE, 2017, pp. 1–7.

[15] J. Zhang, W. Wu, and J. C. S. Lui, "On the theory of function placement and chaining for network function virtualization," in *Proceedings of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, ser. Mobihoc '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 91–100. [Online]. Available: https://doi.org/10.1145/3209582.3209592

[16] Y. Liu, X. Shang, and Y. Yang, "Joint sfc deployment and resource management in heterogeneous edge for latency minimization," *IEEE Transactions on Parallel and Distributed Systems*, 2021.

[17] Y. Liu, Y. Mao, X. Shang, Z. Liu, and Y. Yang, "Distributed cooperative caching in unreliable edge environments," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, 2022, pp. 1049–1058.

[18] L. K. Sullivan, "Shoes the full version," Website, 2021, https://www.youtube.com/watch?v=wCF3ywukQYA/.

[19] J. Li, W. Shi, Q. Ye, W. Zhuang, X. Shen, and X. Li, "Online joint vnf chain composition and embedding for 5g networks," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–6.

[20] X. Wang, C. Wu, F. Le, and F. C. M. Lau, "Online learning-assisted vnf service chain scaling with network uncertainties," in *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, 2017, pp. 205–213.

[21] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.

[22] X. Shang, Y. Huang, Z. Liu, and Y. Yang, "Reducing the service function chain backup cost over the edge and cloud by a self-adapting scheme," *IEEE Transactions on Mobile Computing*, vol. 21, no. 8, pp. 2994–3008, 2022.

[23] J. Li, S. Guo, W. Liang, Q. Chen, Z. Xu, W. Xu, and A. Y. Zomaya, "Digital twin-assisted, sfc-enabled service provisioning in mobile edge computing," *IEEE Transactions on Mobile Computing*, pp. 1–16, 2022.

[24] C. Wang, Q. Hu, D. Yu, and X. Cheng, "Proactive deployment of chain-based vnf backup at the edge using online bandit learning," in *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*, 2021, pp. 740–750.

[25] ——, "Online learning for failure-aware edge backup of service function chains with the minimum latency," 2022. [Online]. Available: https://arxiv.org/abs/2201.06884

[26] Y. Chen and J. Wu, "Latency-efficient vnf deployment and path routing for reliable service chain," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 1, pp. 651–661, 2021.

[27] Y. Mao, X. Shang, and Y. Yang, "Provably efficient algorithms for traffic-sensitive sfc placement and flow routing," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, 2022, pp. 950–959.

[28] ——, "Joint resource management and flow scheduling for sfc deployment in hybrid edge-and-cloud network," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, 2022, pp. 170–179.

[29] R. Cziva, C. Anagnostopoulos, and D. P. Pezaros, "Dynamic, latency-optimal vnf placement at the network edge," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*. IEEE, 2018, pp. 693–701.

[30] Z. Xu, Z. Zhang, W. Liang, Q. Xia, O. Rana, and G. Wu, "Qos-aware vnf placement and service chaining for iot applications in multi-tier mobile edge networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 16, no. 3, pp. 1–27, 2020.

[31] D. Li, P. Hong, K. Xue *et al.*, "Virtual network function placement considering resource optimization and sfc requests in cloud datacenter," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 7, pp. 1664–1677, 2018.

[32] V. Nikam, J. Gross, and A. Rostami, "Vnf service chaining in optical data center networks," in *IEEE NFV-SDN 2017 - IEEE Conference on Network Function Virtualization and Software Defined Networks*. IEEE, 2017, pp. 1–7.

[33] M. Bunyakitanon, X. Vasilakos, R. Nejabati, and D. Simeonidou, "End-to-end performance-based autonomous vnf placement with adopted reinforcement learning," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 2, pp. 534–547, 2020.

[34] D. M. Manias, M. Jammal, H. Hawilo, A. Shami, P. Heidari, A. Larabi, and R. Brunner, "Machine learning for performance-aware virtual network function placement," in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[35] J. Fan, C. Guan, Y. Zhao, and C. Qiao, "Availability-aware mapping of service function chains," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, 2017, pp. 1–9.

[36] J. Zhang, Z. Wang, C. Peng, L. Zhang, T. Huang, and Y. Liu, "Raba: Resource-aware backup allocation for a chain of virtual network functions," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 1918–1926.

[37] T. Taleb, A. Ksentini, and B. Sericola, "On service resilience in cloud-native 5g mobile systems," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 3, pp. 483–496, 2016.

[38] J. Li, W. Liang, W. Xu, Z. Xu, X. Jia, A. Y. Zomaya, and S. Guo, "Budget-aware user satisfaction maximization on service provisioning in mobile edge computing," *IEEE Transactions on Mobile Computing*, pp. 1–13, 2022.

[39] Z. Xu, H. Ren, W. Liang, Q. Xia, W. Zhou, G. Wu, and P. Zhou, "Near optimal and dynamic mechanisms towards a stable nfv market in multi-tier cloud networks," in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, 2021, pp. 1–10.

[40] Z. Xu, H. Ren, W. Liang, Q. Xia, W. Zhou, P. Zhou, W. Xu, G. Wu, and M. Li, "Near optimal learning-driven mechanisms for stable nfv markets in multitier cloud networks," *IEEE/ACM Transactions on Networking*, vol. 30, no. 6, pp. 2601–2615, 2022.

[41] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Transactions on Mobile Computing*, vol. 19, no. 11, pp. 2581–2593, 2020.

[42] S. Vimal, M. Khari, N. Dey, R. G. Crespo, and Y. Harold Robinson, "Enhanced resource allocation in mobile edge computing using reinforcement learning based moaco algorithm for iiot," *Computer Communications*, vol. 151, pp. 355–364, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366419319255

[43] NYSEG, "Electric pricing," Website, 2021, https://www.nyseg.com/wps/portal/nyseg/account/understandyourbill/electricpricing/day-nightrate/.

[44] J. Csirik, "Heuristics for the 0-1 min-knapsack problem," *Acta Cybernetica*, vol. 10, no. 1-2, pp. 15–20, 1991.

[45] Y. Liu, X. Shang, Z. Liu, and Y. Yang, "Technical report," [Online]. Available: https://drive.google.com/drive/folders/1tGDTLkvd9va_14LaBOVncI49aq66aUa-?usp=share_link.

[46] M. J. Neely, "Stability and capacity regions or discrete time queueing networks," *CoRR*, vol. abs/1003.3396, 2010. [Online]. Available: http://arxiv.org/abs/1003.3396

[47] L. Gurobi Optimization, "Gurobi optimizer reference manual," 2021.

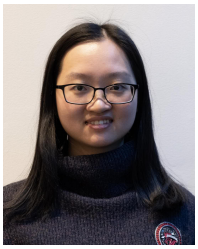
[48] X. Shang, Z. Li, and Y. Yang, "Partial rerouting for high-availability and low-cost service function chain," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–6.

[49] Z. Tan and J. Zhang, "Network function placement for service chains with server maintenance cost," in *2020 IEEE Wireless Communications and Networking Conference Workshops*, 2020, pp. 1–6.

[50] M. Grant, S. Boyd, and Y. Ye, "Cvx users' guide," *online: http://www. stanford. edu/˜ boyd/software. html*, 2009.

Yu Liu received his B. Eng. degree with honor in Telecommunication Engineering from Xidian University, Xi'an, China. He is now pursuing his Ph.D. degree in Computer Engineering at Stony Brook University. His research interests are in online algorithms, distributed storage, cloud computing, edge computing and data center networks, with focus on placement and resource management of virtual network functions and reliability of service function chains.

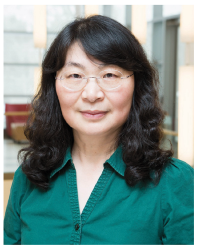
Xiaojun Shang received his B. Eng. degree in Information Science and Electronic Engineering from Zhejiang University, Hangzhou, China, and M.S. degree in Electronic Engineering from Columbia University, New York, USA. He is now pursuing his Ph.D. degree in Computer Engineering at Stony Brook University. His research interests are in cloud computing and data center networks, with focus on placement and routing of virtual network functions and resilience of service function chains.



Yingling Mao received the B.S. degree in Mathematics and Applied Mathematics in lZhiyuan College from Shanghai Jiao Tong University, Shanghai, China, in 2018. She is currently working toward the Ph.D degree in the Department of Electrical and Computer Engineering, Stony Brook University. Her research interests include network function virtualization, software-defined network, cloud computing.

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3282156

Zhenhua Liu is currently assistant professor in the Department of Applied Mathematics and Statistics, also affiliated with Department of Computer Science and Smart Energy Technology Cluster, since August 2014. During the year 2014-2015, he is on leave for the ITRI-Rosenfeld Fellowship in the Energy and Environmental Technology Division at Lawrence Berkeley National Laboratory. Dr. Liu received his Ph.D. degree in Computer Science at the California Institute of Technology, where he was co-advised by Prof. Adam Wierman and Prof. Steven Low. Before Caltech, he received an M.S. degree of Computer Science Technology in 2009 and a B.E. degree of Measurement control in 2006, both from Tsinghua University with honor, as well as a B.S. degree of Economics from Peking University in 2009.



Yuanyuan Yang received the BEng and MS degrees in computer science and engineering from Tsinghua University, Beijing, China, and the MSE and Ph.D. degrees in computer science from Johns Hopkins University, Baltimore, Maryland. She is a SUNY Distinguished Professor of computer engineering and computer science at Stony Brook University, New York, and is currently on leave at the National Science Foundation as a Program Director. Her research interests include edge computing, data center networks, cloud computing and wireless networks. She has published more than 460 papers in major journals and refereed conference proceedings and holds seven US patents in these areas. She is currently the Editor-in-Chief for IEEE Transactions on Cloud Computing and an Associate Editor for IEEE Transactions on Parallel and Distributed Systems and ACM Computing Surveys. She has served as an Associate Editor-in-Chief for IEEE Transactions on Cloud Computing, Associate Editor-in-Chief and Associated Editor for IEEE Transactions on Computers, and Associate Editor for IEEE Transactions on Parallel and Distributed Systems. She has also served as a general chair, program chair, or vice chair for several major conferences and a program committee member for numerous conferences. She is an IEEE Fellow.