# Reliability-Aware Service Function Chain Provisioning in Mobile Edge-Cloud Networks

Shouxu Lin
The Australian National University
Canberra, ACT 2601, Australia

Weifa Liang
The Australian National University
Canberra, ACT 2601, Australia

Jing Li
The Australian National University
Canberra, ACT 2601, Australia

***Abstract*—Mobile Edge Computing (MEC) has been envisioning as a promising technology to address limited computing and storage resources in mobile devices. The virtual services provided by the MEC platform are implemented as instances of Virtual Network Functions (VNFs). However, these VNF instances as pieces of software that run in virtual machines (VMs) are not always reliable. To provide reliable services for their users while meeting user service reliability requirements, the service providers of MEC usually adopt the replica policy that deploy a certain number of service replicas for each VNF instance. In this paper, we study reliable service provisioning in an MEC network through redundant placement of instances of VNFs. We assume that each service request consists of a Service Function Chain (SFC) requirement and a service reliability requirement. We formulate a novel reliability-aware service function chain provisioning problem with the aim to maximize the number of requests admitted, while meeting the specified reliability requirement of each admitted request. We first show that the problem is NP-hard, and formulate an ILP solution for the problem when the problem size is small. We then develop a randomized algorithm with a provable approximation ratio and high probability for the problem when the problem size is large, and the achieved approximation ratio is at the expense of moderate computing capacity and reliability constraint violations. We also devise an efficient heuristic for the problem without any resource and requirement constraint violations. We finally evaluate the performance of the proposed algorithms through experimental simulations. Experimental results demonstrate that the proposed algorithms are promising.**

## I. Introduction

With the recent development of 5G technology, more and more portable mobile devices, including various types of sensors, smartphones, tablets, and wearable sensors, have been widely deployed for business, entertainment, social networking, environmental monitoring, and automaton vehicles [3],[21],[23]. However, the portal sizes of mobile devices restrict their capability for applications that need low latency response, due to their limited computing, storage, and battery capacities. To overcome such limitations, the Mobile Edge Cloud (MEC) network as a promising networking paradigm has been introduced, which consists of cloudlets that are co-located with some Access Points (APs) within the proximity of users [20]. Specifically, MEC is capable to bring both computing and storage resources of clouds at the edge of core networks to reduce service response delays of users, thereby leveraging the capability of mobile devices through offloading their tasks to the MEC for processing [7].

Orthogonal to the MEC technology, Network Function Virtualization (NFV) [18] is another promising technology that implements various network functions by pieces of software running in virtual machines of commodity servers, enabling virtualized service provisioning in MEC more affordable, flexible, and manageable. This virtualization technology can significantly reduce the operational cost of network service providers by replacing specified hardware – the dedicated middleboxes with VNF instances [9]. It also improves the efficiency and flexibility of deploying new network functions.

However, one potential risk of deploying VNF instances is the reliability of deployed VNF instances. A mobile user usually not only requests specific VNF services but also has a certain reliability requirement [22]. Meeting user service reliability requirements is critical to network service providers. While the traditional carrier-grade system is highly fault-tolerant and can achieve a reliability with the order of 5'9s (i.e. 99.999%), it is extremely difficult for VNF – a piece of software running in VMs – to achieve this level of reliability, because software for implementing VNF instances may contain bugs and is prone to failures [8]. Moreover, the reliability issue is further exacerbated by the deployment of Service Function Chains (SFCs) because the failure of one single VNF in the chain will interrupt the end-to-end chain service [17]. Several prevention and recovery mechanisms for such SFC failures haven been developed in the past. Among them, a common and practical approach is utilizing redundancy [5]. Specifically, a single VNF instance failure can be mitigated by deploying its other VNF backup instances in the same cloudlet as its primary VNF instance for the backup purpose.

The novelty of this paper is that we study the reliability-aware SFC provisioning problem with the aim of maximizing the number of admitted service requests while satisfying the reliability requirement of each admitted request. We are the very first to consider that the number of instances of each VNF requested by a service request can be different in order to satisfy the reliability requirement. We also develop a very first randomized algorithm with provable approximation ratio for the problem.

The main contributions of this paper are given as follows. We deal with the reliable service provisioning in an MEC network. We first formulate a novel VNF instance placement problem that aims to maximize the number of admitted requests while satisfying their specified reliability requirements.

978-1-7281-6607-0/20/$31.00 ©2020 IEEE

We then show the problem is NP-hard, and formulate an Integer Linear Programming (ILP) solution to the problem when the problem size is small. We also devise a randomized approximation algorithm with a provable approximation ratio and high probability for it when the problem size is large. The approximation ratio is achieved at the expense of moderate computing capacity and reliability constraint violations. We also develop an efficient heuristic for the problem without any constraint violations. We finally evaluate the performance of the proposed algorithms through experimental simulations. Experimental results demonstrate that the proposed algorithms are promising.

The rest of the paper is organized as follows. Section II reviews the related work. Section III introduces notions, notations, and problem definition. Section IV shows the NP-hardness of the problem. Section V provides an ILP formulation of the problem. Section VI proposes a randomized algorithm for the problem. Section VII devises a heuristic for the problem. Section VIII evaluates the performance of the proposed algorithms. Section IX concludes the paper.

## II. Related Work

Several studies on reliability-aware SFC service provisioning have been conducted recently. For example, Han *et al.* [10] surveyed several existing models for VNF reliability improvement. Huang *et al.* [12] investigated the reliability-aware VNF instance provisioning in an MEC network with the aim to maximize the network throughput. They devised approximation algorithms with provable approximation ratios for the problem under different assumptions. However, their work is based on an assumption that each SFC contains only one network function which may not be realistic. Kanizo *et al.* [13] showed how to plan and deploy backup VNF instances to ensure the high-level survivability by proposing a heuristic algorithm for the problem. However, their work is based on an assumption that there is only one backup VNF instance for each primary VNF instance, and they did not consider that for each primary VNF instance, multiple backup VNF instances are needed to meet the reliability requirement of each request. Beck *et al.* [4] conducted a study of VNF instance survivability in the context of VNF resource allocation, with the aim to minimize the resource consumption. They however did not consider that different requests may have different reliability requirements. Providing all services with the same reliability for different service requests will increase unnecessary backup VNF instances and thus consume unnecessary resource consumptions. Li *et al* [15], [16] studied the problem of reliable VNF service provisioning with the aim to maximize the revenue collected by the service providers. They proposed an on-line algorithm with a provable competitive ratio for the problem at the expense of moderate resource violations. Fan *et al.* [9] proposed an approach to allocate the minimum amount of backup resource for a single SFC request in a data-center network, and then developed an online algorithm for multiple SFC request admissions with the aim of admitting as many requests as possible. Their work however is not applicable to the MEC network, as both computing and bandwidth resources in MEC are limited.

Unlike the aforementioned studies, in this paper we consider the reliable SFC placement without before-mentioned restrictions on existing studies, e.g., each SFC contains only one network function, there is only one backup instance for each primary VNF instance, the reliability requirements of different requests are the same, the VNF instances of all network functions must be placed into one server, or there are unlimited computing resource for VNF placement. In addition, we aim to maximize the number of requests admitted by the network.

## III. Preliminaries

In this section, we first introduce the system model, notions and notations, and then define the problem precisely.

### A. Network system model

We consider a Mobile Edge-Cloud (MEC) network that consists of Access Points (APs), cloudlets and links between APs. The MEC network is represented by an undirected graph $G = (V, E)$, where $V$ is the set of APs, and some of the APs are co-located with cloudlets. $E$ is the set of links between APs.

Denote by $F = \{f_1, f_2, \ldots, f_n\}$ the set of different types of Virtualized Network Functions (VNFs) in the system. Without loss of generality, we assume that the amount of computing resource required to implement a single instance of a specific type of VNF, $f_i \in F$, is measured by the number of computing units, denoted by $c_i$. In addition, we assume that different types of VNFs have different reliabilities, and we use $\rho_i$ to denote the reliability of VNF $f_i \in F$ with $0 < \rho_i \leq 1$. If a VNF $f_i \in F$ has $l$ VNF instances, the reliability of $f_i$ as the probability that at least one of its instances is available can be calculated as $1 - (1 - \rho_i)^l$.

Denote by $C = \{cl_1, cl_2, ..., cl_m\}$ the set of cloudlets in $G$ with $m \leq |V|$. The computing capacity of cloudlet $cl_j \in C$ is denoted by $cap_j$. For a specific type of VNF, $f_i \in F$, a cloudlet can implement it as a piece of the software component in a virtual machine with the computing resource demanded, i.e., $c_i$. We assume the failure probability of any cloudlet is zero and hence, the reliabilities of different instances of the same VNF implemented at different cloudlets are the same.

### B. Request model

Let $R$ be the set of user requests. Each user request $r_k \in R$ is represented by a tuple $(SFC_k, \eta_k)$, where $SFC_k \subseteq F$ is the Service Function Chain (SFC) requested by $r_k$, and $\eta_k$ is the reliability requirement. For simplicity, we use $c_{k,i}$ and $\rho_{k,i}$ to denote the computing resource demand and reliability of VNF $f_i \in SFC_k$, respectively. We define the reliability of an SFC as the probability that at least one VNF instance of each function in $SFC_k$ is available, which can be calculated as $\prod_{f_i \in SFC_k} (1 - (1 - \rho_{k,i})^{l_{k,i}})$, where $l_{k,i}$ is the number of VNF instances of function $f_i \in SFC_k$.

### C. Problem definition

Given an MEC network $G = (V, E)$, a set $R$ of user requests with each having an SFC and a given reliability requirement, *the reliability-aware SFC provisioning problem* in $G$ is to admit as many as requests in $R$ while the specified reliability requirement of each admitted request must be met, subject to the computing capacity of each cloudlet in $G$.

*A request* $r_k \in R$ *is admitted* if at least one instance of each VNF $f_i \in SFC_k$ has been placed, its specified reliability requirement is met, and the cloudlets have enough computing resource to implement all the instances of each VNF $f_i \in SFC_k$. To meet the reliability requirement of each request, backups are adopted. Specifically, we deploy multiple instances of each VNF $f_i \in SFC_k$ such that the reliability of $SFC_k$ is no less than $\eta_k$. We further assume that for a specific request $r_k$, all the instances of a specific VNF $f_i \in SFC_k$ must be placed to the same cloudlet.

## IV. NP-hardness of the problem

In this section, we show that the decision version of the reliability-aware SFC provisioning problem is NP-hard by the following theorem.

**Theorem 1.** *Given an MEC network $G = (V, E)$ and a set $R$ of requests with SFC requirements, the reliability-aware SFC provisioning problem is NP-hard.*

*Proof:* To show that the problem is NP-hard, we consider a simplified version **P1** of the problem where the number of instances of each network function of an SFC is given (in our problem, this number needs to be determined). It can be seen that the reliability-aware SFC provisioning problem is as hard as Problem **P1**. In the following, we show that Problem **P1** is NP-hard, by a reduction from the maximum profit generalized assignment problem (GAP) that is NP-hard. We then can conclude the problem in this paper is NP-hard, too.

The maximum profit GAP is defined as follows. Given $n$ items and $m$ bins, each item $i$ has size of $s_i$ and each bin $j$ has a capacity $B_j$. If item $i$ can be packed into bin $j$, then it brings a profit $p_{ij}$, the problem is to pack as many as items to the $m$ bins to maximize the total profit of the packed items, subject to the capacity on each bin.

We reduce the maximum profit GAP to Problem **P1** as follows. Let $p_{max} = \max\{p_{ij} \mid 1 \leq i \leq n,\ 1 \leq j \leq m\}$ be the maximum profit among the $n$ items. Each item $i$ corresponds a request $i$ with an SFC that consists of only a single VNF, and there are $|V|$ bins with each bin $v \in V$ having a capacity $C'_v$. We further assume that the cloudlets are indexed by $1, 2, \ldots, m$ with $m = |V|$ and each item can be packed to any of the cloudlets. Thus, if the given number of instances of the VNF of request $i$ (item $i$) can be placed to cloudlet $j$, it will bring a profit $\frac{p_{ij}}{p_{max}}$ which is the reliability of request $i$ achieved by redundant instance placements of its VNF, and the total computing resource consumption for this redundant VNF instance placement is $s_i$, which is the total computing resource demanded by the VNF instances. Problem **P1** thus is to maximize the number of requests admitted while maximizing the total profit of admitted requests, subject to the capacity on each cloudlet. It can be seen that a solution to Problem **P1** will return a solution to the maximum profit GAP. It is well known that the maximum profit GAP is NP-hard [6], thus Problem **P1** is NP-hard. So is the reliability-aware SFC provisioning problem. ■

## V. Integer Linear Programming

In this section, we formulate an Integer Linear Programming (ILP) solution to the *Reliability-aware SFC provisioning problem.* Let $y_k$ be a decision variable with value 1 if request $r_k$ is admitted and 0 otherwise. If the variable $x_{k,i,l,j}$ is set to 1, then it means that the VNF $f_i \in SFC_k$ has $l$ instances and all of these instances are implemented at cloudlet $cl_j$. We assume that at most $\omega$ instances of each VNF $f_i$ are needed for any request $r_k \in R$ to ensure the specified reliability requirement. Let $W = \{1, 2, ..., \omega\}$. We use $c_{k,i,l}$ to denote the total amount of computing resource demanded by the $l$ VNF instances of $f_i \in SFC_k$, which is $c_{k,i} \cdot l$, where $c_{k,i}$ is the amount of computing resource demand of one VNF instance of $f_i$. Let $\rho_{k,i,l} = 1 - (1 - \rho_{k,i})^l$ be the reliability of VNF $f_i \in SFC_k$ with $l$ instances, where $\rho_{k,i}$ is the reliability of $f_i$. Define constants $p_{k,i,l} = -\log \rho_{k,i,l}$ and $p_k = -\log \eta_k$, where $\eta_k$ is the reliability requirement of request $r_k$. The ILP is described as follows.

$$Maximize \quad \sum_{r_k \in R} y_k,$$

subject to:

$$\sum_{r_k \in R} \sum_{f_i \in SFC_k} \sum_{l \in W} x_{k,i,l,j} \cdot c_{k,i,l} \leq cap_j, \qquad \forall cl_j \in \mathcal{C} \quad (1)$$

$$\sum_{f_i \in SFC_k} \sum_{l \in W} \sum_{cl_j \in C} x_{k,i,l,j} \cdot p_{k,i,l} \leq p_k, \qquad \forall r_k \in R \quad (2)$$

$$\sum_{l \in W} \sum_{cl_j \in C} x_{k,i,l,j} = y_k, \qquad \forall r_k \in R, f_i \in SFC_k \quad (3)$$

$$y_k \in \{0, 1\}, \qquad \forall r_k \in \mathcal{R} \quad (4)$$

$$x_{k,i,l,j} \in \{0, 1\} \quad \forall r_k \in R, f_i \in SFC_k, l \in W, cl_j \in C \quad (5)$$

- Constraint (1) ensures that the total computing resource demand of all VNF instances in any cloudlet $cl_j$ does not exceed its computing capacity.
- Constraint (2) guarantees that the reliability requirement of each request must be met if it is admitted.
- Constraint (3) says that if any request $r_k \in R$ is admitted, then at least one instance of each VNF $f_i \in SFC_k$ should be implemented, and all the instances of the same VNF should be implemented at the same cloudlet.

## VI. Randomized Algorithm

Although the ILP can provide an exact solution, its time complexity is prohibitively high when the problem size is large. In this section, we instead propose a randomized algorithm for the problem when the problem size is large, and the solution delivered by the algorithm is scalable.

## *A. Algorithm*

Let $\{y_k^*, x_{k,i,l,j}^*\}$ denote the optimal solution of the Linear Programming (LP) relaxation of the ILP. The value of the optimal solution of the LP is $\mu^*$. We now seek to use the optimal LP solution to obtain an integer solution for the ILP through adopting the randomized rounding technique [19]. Let $\{\hat{y}_k, \hat{x}_{k,i,l,j}\}$ and $\hat{\mu}$ denote the integer solution and the value of the integer solution respectively. Each variable $\hat{y}_k$ will be set to 1 with probability $y_k^*$. If a variable $\hat{y}_k$ has been set to 1, thus for each $i, \forall f_i \in SFC_k$, set $\hat{x}_{k,i,l,j}$ to 1 with probability $\frac{x_{k,i,l,j}^*}{y_k^*}$. The choice is done in an exclusive manner, with Constraint (3): given $i$, for each pair of $l$ and $j, \forall l \in W, cl_j \in C$, exactly one of the variables $\hat{x}_{k,i,l,j}$ is set to 1 and the rest are set to 0. This random choice is made independently for all $i$. The detailed description of the randomized algorithm is given in `Algorithm 1`.

**Algorithm 1** Randomized Algorithm

**Input:** An MEC network $G = (V, E)$, a list of requests $R$.
**Output:** A solution that maximizes the number of requests admitted.

```
1: $y_k^*, x_{k,i,l,j}^* \leftarrow$ the optimal LP solution;
2: $\hat{y}_k, \hat{x}_{k,i,l,j} \leftarrow 0$;
3: for all $r_k \in R$ do
4:     $g_y \leftarrow$ a randomly generated number $\in [0, 1]$;
5:     if $y_k^* \geq g_y$ then
6:         $\hat{y}_k \leftarrow 1$;
7:         for all $f_i \in SFC_k$ do
8:             $g_x \leftarrow$ a randomly generated number $\in [0, y_k^*]$;
9:             for all $l^* \in W$ do
10:                for all $cl_{j^*} \in C$ do
11:                    if $\sum_{l=1}^{l^*} \sum_{j=1}^{j^*} x_{k,i,l,j}^* \geq g_x$ then
12:                        $\hat{x}_{k,i,l^*,j^*} \leftarrow 1$;
13:                        Jump to the for loop at line 7;
14:    else
15:        $\hat{y}_k \leftarrow 0$;
16: return $\hat{y}_k, \hat{x}_{k,i,l,j}$;
```

## *B. Analysis of the randomized algorithm*

We now analyze the performance of the proposed randomized algorithm, `Algorithm 1` as follows.

We define a positive constant $\lambda$ as follows.

$$\lambda = \min\{\frac{p_k}{p_{k,i,l}}; \frac{cap_j}{c_{k,i,l}}, \forall r_k \in R, f_i \in SFC_k, l \in W, cl_j \in C\}, \tag{6}$$

where $cap_j$ is the capacity of cloudlet $cl_j$, and $c_{k,i,l}, p_k, p_{k,i,l}$ are constants defined in the ILP.

**Lemma 1.** *The violation of the computing capacity constraint (i.e. Constraint (1)) of each cloudlet $cl_j \in C$ is bounded by $1 + \sigma$ with high probability $1 - \frac{1}{|C|}$, where $\sigma = \frac{\log|C| + \sqrt{\log^2|C| + 4\lambda \log|C|}}{\lambda}$. Recall that $|C|$ is the number of cloudlets and $\lambda$ is a positive constant defined in Eq. (6).*

*Proof:* Let a generalized Bernoulli variable $z_{k,i,l}^j$ denote the amount of computing resource of cloudlet $cl_j$ occupied by all the $l$ instances of $f_i$ in request $r_k$, which is defined as follows.

$$z_{k,i,l}^j = \begin{cases} c_{k,i,l}, & \text{with probability } x_{k,i,l,j}^* \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

where $c_{k,i,l}$ is the computing resource demand of all the $l$ instances of $f_i \in SFC_k$. We also define the random variable $\psi_{k,i,l}^j$ as follows.

$$\psi_{k,i,l}^j = \frac{z_{k,i,l}^j \cdot \lambda}{cap_j}, \tag{8}$$

with $0 \leq \psi_{k,i,l}^j \leq 1$ since $0 \leq \psi_{k,i,l}^j = \frac{z_{k,i,l}^j \cdot \lambda}{cap_j} \leq \frac{c_{k,i,l} \cdot \lambda}{cap_j} \leq 1$.

Thus, for a cloudlet $cl_j$, the probability that its computing capacity violation exceeds $\sigma$, is calculated as follows.

$$\mathbf{Pr}\Big[\sum_{r_k \in R, f_i \in SFC_k, l \in W} z_{k,i,l}^j > (1+\sigma) \cdot cap_j\Big]$$
$$= \mathbf{Pr}\Big[\sum_{r_k \in R, f_i \in SFC_k, l \in W} \psi_{k,i,l}^j > (1+\sigma)\lambda\Big] \tag{9}$$

By applying the well-known Chernoff bound, we have

$$\mathbf{Pr}\Big[\sum_{r_k \in R, f_i \in SFC_k, l \in W} \psi_{k,i,l}^j > (1+\sigma)\lambda\Big] < e^{-\frac{\sigma^2\lambda}{2+\sigma}} \tag{10}$$

Note that the Chernoff bound is applicable because

$$\mathbb{E}\Big[\sum_{r_k \in R, f_i \in SFC_k, l \in W} \psi_{k,i,l}^j\Big] = \mathbb{E}\Big[\sum_{r_k \in R, f_i \in SFC_k, l \in W} \frac{z_{k,i,l}^j \cdot \lambda}{cap_j}\Big]$$
$$= \frac{\lambda}{cap_j} \sum_{r_k \in R, f_i \in SFC_k, l \in W} \mathbb{E}[z_{k,i,l}^j]$$
$$= \frac{\lambda}{cap_j} \sum_{r_k \in R, f_i \in SFC_k, l \in W} (c_{k,i,l} \cdot x_{k,i,l,j}^*)$$
$$\leq \frac{\lambda}{cap_j} \cdot cap_j \tag{11}$$
$$= \lambda \tag{12}$$

Inequality (11) holds, because the feasible solution $x_{k,i,l,j}^*$ meets Constraint (1).

We then set the value of $\sigma$ according to the following equation

$$e^{-\frac{\sigma^2\lambda}{2+\sigma}} = \frac{1}{|C|^2} \tag{13}$$

Thus, the solution of Eq. (13) can be expressed as

$$\sigma = \frac{\log|C| + \sqrt{\log^2|C| + 4\lambda \log|C|}}{\lambda} \tag{14}$$

Combining Equation (9), Inequality (10) and Equation (13), we have

$$\mathbf{Pr}\Big[\sum_{r_k \in R, f_i \in SFC_k, l \in W} z_{k,i,l}^j > (1+\sigma)cap_j\Big] < \frac{1}{|C|^2} \tag{15}$$

Hence, the probability that the computing capacity constraint of each cloudlet $cl_j \in C$ will not exceed $\sigma$ can be expressed as follows.

$$\mathbf{Pr}\Big[\bigwedge_{cl_j \in C}\Big(\sum_{r_k \in R, f_i \in SFC_k, l \in W} z_{k,i,l}^j \leq (1+\sigma) \cdot cap_j\Big)\Big]$$
$$= 1 - \mathbf{Pr}\Big[\bigvee_{cl_j \in C}\Big(\sum_{r_k \in R, f_i \in SFC_k, l \in W} z_{k,i,l}^j > (1+\sigma) \cdot cap_j\Big)\Big]$$

$$\geq 1 - \sum_{cl_j \in C} \mathbf{Pr}\Big[ \sum_{r_k \in R, f_i \in SFC_k, l \in W} z^j_{k,i,l} > (1+\sigma) \cdot cap_j \Big] \tag{16}$$

$$> 1 - |C| \cdot \frac{1}{|C|^2} = 1 - \frac{1}{|C|} \tag{17}$$

Note that Inequality (16) holds due to the Union Bound Inequality. ■

**Lemma 2.** *The violation of the reliability constraint (i.e. Constraint (2)) of each request $r_k \in R$ is bounded by $1+\varepsilon$ with high probability $1 - \frac{1}{|R|}$, where $\varepsilon = \frac{\log |R| + \sqrt{\log^2 |R| + 4\lambda \log |R|}}{\lambda}$. Recall that $|R|$ is the number of requests and $\lambda$ is a positive constant defined in Eq. (6).*

*Proof:* We define a generalized Bernoulli variable $z^j_{k,i,l}$, denoting the negative logarithmic of the reliability of $f_i$ when all its $l$ instances are placed at $cl_j$, and $z^k_{i,l,j}$ is defined as follows.

$$z^k_{i,l,j} = \begin{cases} p_{k,i,l}, & \text{with probability } \tilde{x}_{k,i,l,j} \\ 0, & \text{otherwise} \end{cases} \tag{18}$$

where $p_{k,i,l} = -\log(\rho_{k,i,l})$, and $\rho_{k,i,l}$ is the reliability of $f_i \in SFC_k$ with $l$ instances. We also define the random variable $\psi^k_{i,l,j}$ as follows.

$$\psi^k_{i,l,j} = \frac{z^k_{i,l,j} \cdot \lambda}{p_k} \tag{19}$$

with $0 \leq \psi^k_{i,l,j} \leq 1$ since $0 \leq \psi^k_{i,l,j} = \frac{z^k_{i,l,j} \cdot \lambda}{p_k} \leq \frac{p_{k,i,l} \cdot \lambda}{p_k} \leq 1$.

Thus, for a request $r_k$, the probability that its reliability violation exceeds $\varepsilon$, is calculated as follows.

$$\mathbf{Pr}\Big[ \sum_{f_i \in SFC_k, l \in W, cl_j \in C} z^k_{i,l,j} > (1+\varepsilon) \cdot p_k \Big]$$
$$= \mathbf{Pr}\Big[ \sum_{f_i \in SFC_k, l \in W, cl_j \in C} \psi^k_{i,l,j} > (1+\varepsilon) \cdot \lambda \Big] \tag{20}$$

By applying the well-known Chernoff bound, we have

$$\mathbf{Pr}\Big[ \sum_{f_i \in SFC_k, l \in W, cl_j \in C} \psi^k_{i,l,j} > (1+\varepsilon) \cdot \lambda \Big] < e^{-\frac{\varepsilon^2 \lambda}{2+\varepsilon}} \tag{21}$$

Note that the Chernoff bound is applicable because

$$\mathbb{E}[ \sum_{f_i \in SFC_k, l \in W, cl_j \in C} \psi^k_{i,l,j}]$$
$$= \mathbb{E}[ \sum_{f_i \in SFC_k, l \in W, cl_j \in C} \frac{z^k_{i,l,j} \cdot \lambda}{p_k}]$$
$$= \frac{\lambda}{p_k} \cdot \mathbb{E}[ \sum_{f_i \in SFC_k, l \in W, cl_j \in C} z^k_{i,l,j}]$$
$$= \frac{\lambda}{p_k} \cdot \sum_{f_i \in SFC_k, l \in W, cl_j \in C} \mathbb{E}[z^k_{i,l,j}]$$
$$= \frac{\lambda}{p_k} \cdot \sum_{f_i \in SFC_k, l \in W, cl_j \in C} p_{k,i,l} \cdot \tilde{x}_{k,i,l,j}$$
$$\leq \frac{\lambda}{p_k} \cdot p_k \tag{22}$$
$$= \lambda \tag{23}$$

Inequality (22) holds because the feasible solution $x^*_{k,i,l,j}$ meets Constraint (2).

We then set the value of $\varepsilon$ according to the following equation

$$e^{-\frac{\varepsilon^2 \lambda}{2+\varepsilon}} = \frac{1}{|R|^2} \tag{24}$$

The solution of Eq. (24) can be expressed as $\varepsilon = \frac{\log |R|^3 + \sqrt{\log^2 |R|^3 + 8\lambda \log |R|^3}}{2\lambda}$.

Combining Equation (20), Inequality (21), and Equation (24), we have

$$\mathbf{Pr}\Big[ \sum_{f_i \in SFC_k, l \in W, cl_j \in C} z^k_{i,l,j} > (1+\varepsilon) \cdot p_k \Big] < \frac{1}{|R|^2} \tag{25}$$

Hence, the probability that the reliability constraint of each request $r_k \in R$ will not exceed $\varepsilon$ can be expressed as follows.

$$\mathbf{Pr}\Big[ \bigwedge_{r_k \in R} \Big( \sum_{f_i \in SFC_k, l \in W, cl_j \in C} z^k_{i,l,j} < (1+\varepsilon) \cdot p_k \Big) \Big]$$
$$= 1 - \mathbf{Pr}\Big[ \bigvee_{r_k \in R} \Big( \sum_{f_i \in SFC_k, l \in W, cl_j \in C} z^k_{i,l,j} \geq (1+\varepsilon) \cdot p_k \Big) \Big]$$
$$\geq 1 - \sum_{r_k \in R} \mathbf{Pr}\Big[ \sum_{f_i \in SFC_k, l \in W, cl_j \in C} z^k_{i,l,j} \geq (1+\varepsilon) \cdot p_k \Big] \tag{26}$$

$$> 1 - |R| * \frac{1}{|R|^2} = 1 - \frac{1}{|R|} \tag{27}$$

Note that Inequality (26) holds due to the Union Bound Inequality. ■

**Lemma 3.** *The number of admitted requests will be at least $(1-\alpha) \cdot \mu$ with high probability $1 - \frac{1}{|R|}$, where $\alpha = \sqrt{\frac{2 \log |R|}{\mu^*}}$, $\mu$ and $\mu^*$ are the value of the optimal solution of the ILP and the LP, respectively.*

*Proof:* Let a Bernoulli variable $z_k$ denote whether or not request $r_k$ is admitted, which is defined as follows.

$$z_k = \begin{cases} 1, & \text{with probability } y^*_k \\ 0, & \text{otherwise} \end{cases} \tag{28}$$

Thus, the probability that the number of admitted requests is less than $(1-\alpha) \cdot \mu^*$, can be calculated as follows, by the well-known Chernoff bound.

$$\mathbf{Pr}\Big[ \sum_{r_k \in R} z_k < (1-\alpha) \cdot \mu^* \Big] < e^{-\frac{\alpha^2 \mu^*}{2}} \tag{29}$$

Note that the Chernoff bound is applicable because

$$\mathbb{E}\Big[ \sum_{r_k \in R} z_k \Big] = \sum_{r_k \in R} \mathbb{E}[z_k] = \sum_{r_k \in R} y^*_k = \mu^* \tag{30}$$

We then set the value of $\alpha$ according to the equation

$$e^{-\frac{\alpha^2 \cdot \mu^*}{2}} = \frac{1}{|R|} \tag{31}$$

where $|R|$ is the number of requests. The solution of Equation (31) can be expressed as

$$\alpha = \sqrt{\frac{2\log|R|}{\mu^*}} \tag{32}$$

Notice that it requires $\sqrt{\frac{2\log|R|}{\mu^*}} < 1$. Thus, $\mu^* > 2\cdot\log|R|$. Combining Inequality (29) and Equation (31), we have

$$\mathbf{Pr}\Big[\sum_{r_k\in R} z_k < (1-\alpha)\cdot\mu^*\Big] < \frac{1}{|R|} \tag{33}$$

Hence, the probability that the number of admitted requests is at least $(1-\alpha)\cdot\mu$ can be calculated as follows.

$$\mathbf{Pr}\Big[\sum_{r_k\in R} z_k \geq (1-\alpha)\cdot\mu\Big] = 1-\mathbf{Pr}\Big[\sum_{r_k\in R} z_k < (1-\alpha)\cdot\mu\Big]$$

$$\geq 1-\mathbf{Pr}\Big[\sum_{r_k\in R} z_k < (1-\alpha)\cdot\mu^*\Big] \tag{34}$$

$$> 1-\frac{1}{|R|} \tag{35}$$

Note that $\mu \leq \mu^*$ since for a maximization problem, the optimal solution of the LP is no less than the optimal solution of the ILP. Thus, Inequality (34) holds. ■

**Theorem 2.** *Given an MEC network $G(V,E)$ and a set $R$ of user requests, there is a randomized algorithm with high probability $min\{1-\frac{1}{|C|}, 1-\frac{1}{|R|}\}$, for the reliability-aware SFC provisioning problem, which achieves an approximation ratio $\alpha = \sqrt{\frac{2\log|R|}{\mu^*}}$ while the computing capacity violation at any cloudlet is bounded by $1+\sigma = 1+\frac{\log|C|+\sqrt{\log^2|C|+4\lambda\log|C|}}{\lambda}$ and the reliability constraint violation of any request is bounded by $1+\varepsilon = 1+\frac{\log|R|+\sqrt{\log^2|R|+4\lambda\log|R|}}{\lambda}$, where $|R|$ and $|C|$is the number of requests and cloudlets respectively, $\mu^*$ is the optimal solution of the LP, and constant $\lambda$ is defined in Equation (6).*

*Proof:* Following Lemma 1, 2, and 3, the proof of the theorem is straightforward. ■

## VII. Heuristic algorithm

In this section, we devise an efficient heuristic algorithm for the problem. Unlike the randomized algorithm in the previous section, the solution delivered by it is feasible, and there are no any computing capacity or reliability constraint violations.

### A. Algorithm

We start with determining the number of instances of each VNF $f_i \in SFC_k$ so that the reliability requirement of request $r_k$ can be satisfied.

Let $\theta_k = \prod_{f_i\in SFC_k}\varphi_{k,i}$ be the current reliability of $SFC_k$ where $\varphi_{k,i}$ is the current reliability of $f_i \in SFC_k$. If one more VNF instance of some $f_{i^*}$ is added, the reliability of $SFC_k$, denoted by $\theta'_k$, will be $(\prod_{f_i\in SFC_k\setminus\{f_{i^*}\}}\varphi_{k,i})\cdot\varphi'_{k,i^*} = \frac{\theta_k}{\varphi_{k,i^*}}\cdot\varphi'_{k,i^*}$, where $\varphi_{k,i^*}$ is the current reliability of $f_{i^*}$ and $\varphi'_{k,i^*}$ is the reliability of $f_{i^*}$ with one more VNF instance. Thus, the ratio $\frac{\theta'_k-\theta_k}{c_{k,i}}$ is the marginal increase in the reliability of $SFC_k$ per unit computing resource for $f_i$, where $c_{k,i}$ is the amount of computing resource demanded by one VNF instance of $f_i \in SFC_k$. The strategy adopted here is to repeatedly add one VNF instance of a function $f_i$ with the maximum ratio of $\frac{\theta'_k-\theta_k}{c_{k,i}}$ until $\theta_k \geq \eta_k$.

We then sort requests in $R$ in the non-decreasing order of the total computing resource demand of each of them, which is $\sum_{f_i\in SFC_k}\gamma_{k,i}$, where $\gamma_{k,i}$ is the total computing resource demand of all VNF instances of $f_i$. Next, for each request $r_k \in R$, we determine whether it can be admitted, depending on whether each VNF can be matched. Note that a VNF $f_i$ can be matched to a cloudlet $cl_j \in C$ only if $cl_j$ has sufficient residual computing resource to accommodate all its VNF instances, i.e., $cap'_j \geq \gamma_{k,i}$ where $cap'_j$ is the residual computing resource of cloudlet $cl_j$.

**Algorithm 2** Heuristic

**Input:** An MEC network $G=(V,E)$, a set of requests $R$.
**Output:** A solution that maximizes the number of requests admitted, while meeting the reliability requirement of each admitted request.

```
1:  $A \leftarrow \emptyset$; /* the set of admitted requests */
2:  for all $r_k \in R$ do
3:      $\gamma_{k,i} \leftarrow c_{k,i}, \forall f_i \in SFC_k$;
4:      while $\theta_k < \eta_k$ do
5:          $i^* \leftarrow \underset{f_i\in SFC_k}{\texttt{argmax}}\ (\frac{\theta'_k-\theta_k}{\gamma_{k,i}})$;
6:          $\gamma_{k,i^*} \leftarrow \gamma_{k,i^*} + c_{k,i^*}$;
7:  Sort $|R|$ in non-decreasing order of the total computing resource demand
        of each request;
8:  $cap'_j \leftarrow cap_j, \forall cl_j \in C$;
9:  for all $r_k \in R$ do
10:     $L \leftarrow SFC_k$;
11:     $cap^*_j \leftarrow cap'_j, \forall cl_j \in C$;
12:     while $L \neq \emptyset$ do
13:         Construct an auxiliary bipartite graph $G' \leftarrow (L\cup C, E')$ where
                $E' \leftarrow \{(f_i, cl_j) \mid cap'_j \geq \gamma_{k,i}\}$;
14:         if $E' = \emptyset$ then
15:             $cap'_j \leftarrow cap^*_j, \forall cl_j \in C$;
16:             Reject $r_k$;
17:         else
18:             Find a maximum matching $P$ in $G'$ by the Hopcroft– Karp
                    algorithm where $P \leftarrow \{(f_i, cl_j) \mid f_i$ is matched to $cl_j\}$;
19:             $cap'_j \leftarrow cap'_j - \gamma_{k,i},\ \forall (f_i, cl_j) \in P$;
20:             $L \leftarrow L \setminus \{f_i \mid \exists cl_j, (f_i, cl_j) \in P\}$;
21:     Admit $r_k$ and $A \leftarrow A \cup \{r_k\}$;
22: return Set $A$ of admitted requests and their VNF instance placements;
```

The algorithm proceeds iteratively. Within each iteration, we construct an auxiliary bipartite graph $G' = (V', E')$, where $V'$ is the set of nodes containing all VNFs $f_i \in SFC_k$ which have not been matched and all the cloudlets $cl_j \in C$, and $E'$ is the set of edges. There is an edge $e \in E'$ between a cloudlet $cl_j \in V'$ and a VNF $f_i \in V'$ if $cap'_j \geq \gamma_{k,i}$. The problem then is to match as many VNFs as possible by finding a maximum matching in $G'$. This procedure continues until either all VNFs $f_i \in SFC_k$ are matched, or $E'$ becomes empty which means no further VNF placement is possible. The detailed description of the algorithm is given in Algorithm 2.

## *B. Analysis of the heuristic algorithm*

**Theorem 3.** *Given an MEC network $G(V, E)$ and a set $R$ of requests, there is a heuristic algorithm,* `Algorithm 2` *for the reliability-aware SFC provisioning problem, which takes $\mathcal{O}(|R| \cdot |C| \cdot L^2 \cdot \sqrt{L + |C|})$ time, where |R| and |C| are the number of requests and cloudlets respectively, and $L$ is the maximum length of $SFC_k, \forall r_k \in R$.*

*Proof:* The solution delivered by `Algorithm 2` is feasible, because the number of instances of each VNF $f_i \in SFC_k$ that needs to be placed in order to meet its reliability requirement has already been determined, prior to the admission of request $r_k \in R$. In addition, the computing capacity constraint of each cloudlet cannot be violated since no VNF $f_i$ would be matched to a cloudlet if the residual computing resource of the cloudlet is insufficient to place its instances.

The rest is to analyze the time complexity of the proposed heuristic algorithm.

It can be seen that it takes $\mathcal{O}(L \cdot \omega \cdot |R|)$ time from Step 1 to Step 7. For each request, the number of VNF instances required is at most $L \cdot \omega$, where $L$ is the maximum length of any $SFC_k, \forall r_k \in R$, and $\omega$ is the maximum number of instances for each VNF. Hence, it takes $\mathcal{O}(L \cdot \omega \cdot |R|)$ time to find the number of VNF instances required for all the requests. Step 8 takes $\mathcal{O}|R| \cdot \log(|R|)$ time for sorting the requests in $R$, and it takes $\mathcal{O}(|R| \cdot |C| \cdot L^2 \cdot \sqrt{L + |C|})$ time from Step 9 to Step 22. For each request $r_k \in R$, it appears in at most $L$ maximum matching, and it takes at most $\mathcal{O}(\sqrt{L + |C|} \cdot L \cdot |C|)$ time for finding a maximum matching by the Hopcroft – Karp algorithm, since the number of nodes $|V'|$ is at most $L + |C|$ and the number of edges $|E'|$ is at most $L \cdot |C|$. Thus, the algorithm takes $\mathcal{O}(|R| \cdot |C| \cdot L^2 \cdot \sqrt{L + |C|})$ time to finding a series of maximum matching between each VNF and cloudlets for all requests. ■

## VIII. Performance Evaluation

In this section, we evaluate the performance of the proposed algorithms through experimental simulations. We also investigate the impact of important parameters on the performance of the proposed algorithms.

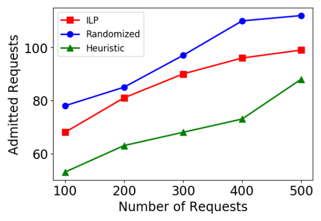
### *A. Experimental environment settings*

We consider an MEC $G = (V, E)$ network that consists of 200 APs, and the number of cloudlets $|C|$ is 10% of network size. The cloudlets are randomly co-located with some of the APs. Each network topology is generated, using the widely adopted approach due to Barabási and Albert [2]. The computing capacity $cap_j$ of each cloudlet is randomly drawn from 2,000 to 4,000 MHz [11]. The network offers 20 different types of VNFs, i.e., $|F| = 20$, where the computing resource demand of a VNF $f_i \in F$, $c_i$, is randomly drawn from 40 MHz to 400 MHz [1]. The reliability of one instance of VNF $f_i$ is randomly drawn in a range from 0.9 to 0.9999 [14]. For each request $r_k \in R$, the VNFs in its SFC are randomly selected from $F$ with the length of the SFC being from 3 to 5, and the reliability requirement is drawn in the range between 0.85 and 0.9. The running time of each mentioned algorithm is based on a desktop with a 3.4GHz 8-core Intel i7 CPU and 16 GB RAM. Each value in figures is the mean of the results of 30 trials.

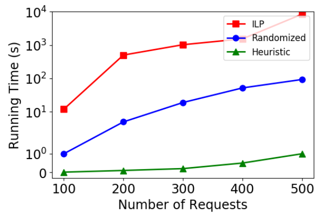
### *B. Performance evaluation of different algorithms*

We first study the performance of different algorithms including the ILP, the randomized algorithm, and the heuristic algorithm, by varying the number of requests $|R|$.

Fig. 1(a) plots the performance of the three comparison algorithms. It can be observed that when $|R| = 500$, approximately 12% more requests are admitted by the randomized algorithm than the ILP, while the number of requests admitted by the heuristic is about 10% less than the ILP. Note that the super performance of the randomized algorithm is achieved at the expense of the computing capacity and reliability constraint violations, which will be dealt with later in this section. It can also be seen that the running times of both the randomized algorithm and the heuristic are significantly less than that of the ILP when $|R|$ becomes large, as shown in Fig. 1(b).



(a) Numbers of admitted requests



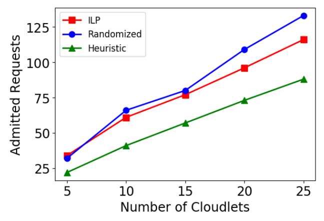
(b) The running time of algorithms (in seconds)

Fig. 1. Performance of different algorithms, by varying $|R|$ from 100 to 500.

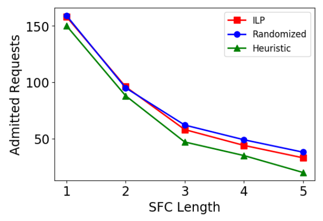
### *C. Impacts of parameters on algorithm performance*

We then evaluate the impact of important parameters such as the number of cloudlets $|C|$ and the length of SFC $L$ on the algorithm performance. On one hand, it can be seen from Fig. 2(a) that the randomized algorithm achieves a better performance with the increase of $|C|$. Specifically, the three algorithms exhibit the similar behaviors in terms of performance when $|C| = 5$. However, it is noted that the randomized algorithm outperforms both the ILP and the heuristic by nearly 15% and 40% respectively, when $|C| = 25$. The reason behind is that each $y_k^*, \forall r_k \in R$ in the LP solution has a larger probability to be rounded to 1, since there is more computing resource available. It can also be observed that the performance gap between the ILP and the heuristic is kept around 20.

On the other hand, Fig. 2(b) implies that both the randomized algorithm and the ILP have the similar performance, while the performance of the heuristic drastically degrades with the increase in the SFC length. The rationale behind is that more VNF instances will be placed with the growth of $L$, and thus consuming more computing resource. Thus, fewer numbers of VNF instances can be assigned to cloudlets.



(a) Numbers of admitted requests by varying $|C|$ from 5 to 25, with $|R| =$ 400 and $L$=3.



(b) Numbers of admitted requests by varying $L$ from 1 to 5, with $|R| =$ 200 and $|C|$=10.

Fig. 2. Impact of parameters $|C|$ and $L$ on the performance of different algorithms.

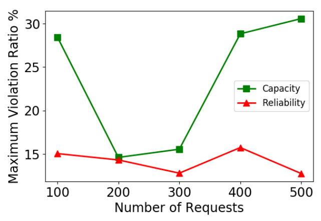
## D. *Constraint violation analysis of the randomized algorithm*

We finally investigate the violation of computing capacity and the reliability constraints by the randomized algorithm.

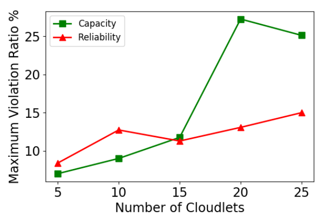
Fig. 3(a) shows that the maximum violation ratio of the computing capacity and reliability constraints respectively. It can be observed that with different numbers of requests in $R$, the violation ratios of computing capacity and reliability constraints are around 30% and 15%, respectively. Moreover, it is noted that the violations correspond to the performance gap between the randomized algorithm and the ILP in Fig. 1(a). For example, when $|R| = 200$, there is very little difference between the number of requests admitted, and the computing capacity and reliability constraint violations are both around 15%. However, when $|R| = 400$, the randomized algorithm admits 15 more requests than the ILP and the violations become 30% and 17%, respectively. The rationale is that the more requests are admitted by the randomized algorithm than the ILP, the more constraints must be violated. A similar pattern can be found in Fig. 3(b). Fig. 3(c) illustrates the violation on both computing capacity and reliability constraints, by varying the length $L$ of SFCs, from which it can be seen that both computing capacity and reliability constraint violations increase with the increase of $L$. The reason is that each $f_i \in SFC_k$ requires more instances in order to meet the reliability requirement of request $r_k$ with a larger $L$. Hence, each cloudlet has a larger probability to be overloaded, and each SFC has a larger probability to be less reliable as the probability that some VNF instances required to be placed but not in fact increases.

## IX. Conclusion

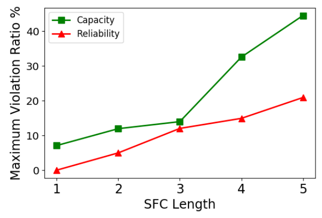
In this paper, we studied the reliability-aware SFC provisioning problem in MEC with the aim to maximize the number of requests admitted while satisfying individual request's reliability requirement. We first showed that the problem is NP-hard and formulated an ILP solution for it. We then developed a randomized algorithm with a provable approximation ratio with high probability, at the expense of moderate computing capacity and the specified reliability constraint violations. Also, we devised an efficient heuristic for the problem without violating computing capacity and reliability constraints. We finally evaluated the performance of the proposed algorithms through experimental simulations. Experimental results demonstrated that the proposed algorithms are promising.



(a) The violation ratio by varying $|R|$.



(b) The violation ratio by varying $|C|$.



(c) The violation ratio by varying $L$.

Fig. 3. Capacity and reliability violations by the randomized algorithm.

## Acknowledgements

It is acknowledged that the work by Shouxu Lin, Weifa Liang and Jing Li was supported by the Australian Research Council Discovery Project No. DP200101985.

## References

[1] Amazon Web Services, Inc.Amazon EC2 instance configuration. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-ec2-config.html, 2018.
[2] A.L. Barabási, and R. Albert. Emergence of scaling in random networks. *Science*, vol.286, pp.509–512, 1999.
[3] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. Fog computing and its role in the internet of things. *Proc. of Workshop on Mobile cloud Computing*, ACM, 2012.
[4] M.Y. Beck, J.F. Botero, and K. Samelin. Resilient allocation of service function chains. *Proc. of the Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pp.128-133, IEEE, 2017.
[5] H.D. Chantre, and N.L. da Fonseca. Redundant placement of virtualized network functions for lte evolved multimedia broadcast multicast services. *Proc. of ICC'17*, IEEE, 2017.
[6] R. Cohen, L. Katzir, and D. Raz. An efficient approximation for the generalized assignment problem. *Information Processing Letters*, vol.100, pp.162–166, 2006.
[7] A.V. Dastjerdi, and R. Buyya. Fog computing: Helping the Internet of Things realize its potential. *Computer*, vol.49, no.8, pp.112 – 116, 2016.
[8] J. Fan, C. Guan, Y. Zhao, and C. Qiao. Availability-aware mapping of service function chains. *Proc. of INFOCOM'17*, IEEE, 2017.
[9] J. Fan, M. Jiang, and C. Qiao. Carrier-grade availability-aware mapping of service function chains with on-site backups. *Proc. of IWQoS'17*, IEEE, 2017.
[10] B. Han, V. Gopalakrishnan, G. Kathirvel, and A. Shaikh. On the resiliency of virtual network functions. *IEEE Communications Magazine*, vol.55, pp.152-157, 2017.
[11] Hewlett-Packard Development Company. L.P. Servers for enterprise bladeSystem, rack & tower and hyperscale. http://www8.hp.com/us/en/products/servers/, 2015.
[12] 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*, to be published, doi: 10.1109/TMC.2019.2927214, 2019.
[13] Y. Kanizo, O. Rottenstreich, I. Segall, and J. Yallouz. Optimizing virtual backup allocation for middleboxes. *IEEE/ACM Transactions on Networking*, vol.25, no.5, pp.2759–2772, 2017.
[14] J. Kong, I. Kim, X. Wang, Q. Zhang, H.C. Cankaya, W. Xie, T. Ikeuchi, and J.P. Jue. Guaranteed-availability network function virtualization with network protection and vnf replication. *Proc. of GLOBECOM'17*, pp.1–6, IEEE, 2017.

[15] J. Li, W. Liang, M. Huang, and X. Jia. Providing reliability-aware virtualized network function services for mobile edge computing. *Proc. of ICDCS'19*, IEEE, 2019.

[16] J. Li, W. Liang, M. Huang, and X. Jia. Reliability-aware network service provisioning in mobile edge-cloud networks. *IEEE Transactions on Parallel and Distributed Systems*, to be published, doi: 10.1109/TPDS.2020.2970048, 2020.

[17] S. Mehraghdam, M. Keller, and H. Karl. Specifying and placing chains of virtual network functions. *Proc. of CloudNet*, pp.7–13, IEEE, 2014.

[18] J. Pan, and J. McElhannon. Future edge cloud and edge computing for internet of things applications. *IEEE Internet of Things Journal*, vol.5, pp.439–449, 2017.

[19] P. Raghavan and C. D. Thompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, vol.7, No.4, pp.365–374, 1987.

[20] M. Satyanarayanan, V. Bahl, R. Caceres, and N. Davies. The case for vm-based cloudlets in mobile computing. *IEEE pervasive Computing*, vol.8, pp.1536–1268, 2009.

[21] F. Wang, J. Xu, X. Wang, and S. Cui. Joint offloading and computing optimization in wireless powered mobile-edge computing systems. *IEEE Transactions on Wireless Communications*, vol.17, no.3, pp.1784–1797, 2017.

[22] Y. Sang, B. Ji, G. Gupta, X. Du, and L. Ye. Provably efficient algorithms for joint placement and allocation of virtual network functions. *Proc of GLOBECOM'17*, IEEE, 2017.

[23] S. Yi, C. Li, and Q. Li. A survey of fog computing: concepts, applications and issues. *Proc. of the Workshop on Mobile Big Data*, ACM, 2015.