Reliability-Aware Virtualized Network Function Services Provisioning in Mobile Edge Computing

Meitian Huang, Weifa Liang, *Senior Member, IEEE*, Xiaojun Shen, *Senior Member, IEEE*, Yu Ma, and Haibin Kan

Abstract—Along with Network Function Virtualization (NFV), Mobile Edge Computing (MEC) is becoming a new computing paradigm that enables accommodating innovative applications and services with stringent response delay and resource requirements, including autonomous vehicles and augmented reality. Provisioning reliable network services for users is the top priority of most network service providers, as unreliable services or severe service failures can result in tremendous losses of users, particularly for their mission-critical applications. In this paper, we study reliability-aware VNF instances provisioning in an MEC, where different users request different network services with different reliability requirements through paying their requested services with the aim to maximize the network throughput. To this end, we first formulate a novel reliability-aware VNF instance placement problem by provisioning primary and secondary VNF instances at different cloudlets in MEC for each user while meeting the specified reliability requirement of the user request. We then show that the problem is NP-hard and formulate an Integer Linear Programming (ILP) solution. Due to the NP-hardness of the problem, we instead devise an approximation algorithm with a logarithmic approximation ratio for the problem. Moreover, we also consider two special cases of the problem. For one special case where each request only requests one primary and one secondary VNF instances, the problem is still NP-hard, and we devise a constant approximation algorithm for it. For another special case where different VNFs have the same amounts of computing resource demands, we show that it is polynomial-time solvable by developing a dynamic programming solution for it. We finally evaluate the performance of the proposed algorithms through experimental simulations. Experimental results demonstrate that the proposed algorithms are promising, and the empirical results of the algorithms outperform their analytical counterparts as theoretical estimations usually are very conservative.

Index Terms—Reliability-Aware VNF instances provisioning; VNF instance placements; virtualized network function implementation; approximation algorithms; generalized assignment problems; dynamic programming; Mobile Edge Computing; combinatorial optimization problems.

1 Introduction

Mobile devices, including smartphones and tablets, have experienced exponential growth in recent years. This trend is coupled with the evolution of new mobile applications with stringent requirements such as real-time and interactive applications. However, executing computation-intensive applications on mobile devices of portable sizes is heavily constrained by their limited computing, storage, and battery capacities. Mobile edge computing (MEC) has emerged, which brings computation and storage resources, which are referred to as *cloudlets* [30], to the edge of mobile networks [11]. These cloudlets are co-located with access points (APs) in the networks and they are accessible by mobile users via wireless access. A key advantage of deployment of cloudlets is that the close physical proximity between cloudlets and users introduces shorter communication delays, thereby improving the user experience. It thus has been envisioned that MEC has many promising applications including in smart cities and smart connected vehicles [6], [26], [32], [34].

- *M. Huang, W. Liang and Y. Ma are with Research School of Computer Science, The Australian National University, Canberra, ACT 2601, Australia Emails: Meitian.Huang@anu.edu.au, wliang@cs.anu.edu.au, Yu.Ma@anu.edu.au*
- *X. Shen is with the School of Computing and Engineering, University of Missouri, Kansas City, MO 64110, USA Email: shenx@umkc.edu*
- *H. Kan is with School of Computer Science, Fudan University, Shanghai 200433, P. R. China Email: hbkan@fudan.edu.cn*

Many network service providers recently have started migrating their infrastructure to take advantage of network functions virtualization (NFV) [21], [27]. Due to the continuous advances in computing hardware, it is possible to replace resource demanding user applications, such as voice recognition, image processing, and other supporting tasks for smart cities, with software components that provide the same capability on top of commodity servers. Each network function, such as a software-based video transcoder for a smart city system, runs in a virtual machine, referred to as a virtualized network function (VNF) instance, hosted in cloudlets. NFV enables users to offload their tasks to nearby cloudlets via wireless APs for processing.

While implementing user applications as VNF instances promises significant flexibility and ease of the management of networks, it also brings more concerns on the reliability of running VNF instances. Traditional carrier-grade systems have been engineered to offer nearly 99.999% (five nines) reliability and designed to be highly fault-tolerant [14]. Achieving this level of reliability is extremely difficult for NFV in an MEC environment due to various reasons including the cloudlets hosting VNF instances are more prone to errors and failures compared to the dedicated hardware appliances, and software for implementing VNF instances may contain bugs [14] and is prone to failures. As a result, in order to provide reliable VNF services to users while meeting their requirement of the service and mitigate the risk of failures, additional VNF instances should be created at other peer cloudlets in the MEC.

In this paper, we consider reliability-aware VNF services provisioning in an MEC. We assume that each user request needs a VNF service with a specified reliability requirement. In order to meet the reliability requirement of the user, multiple VNF instances need to be placed at different cloudlets. We distinguish between the single primary VNF instance and one or multiple secondary VNF instances for each offloaded user request [17]: the former is an active VNF instance while the latter are idle ones until the primary one fails. Moreover, as the usage of network resources follows pay-as-you-go, there are multiple users competing for limited network resources, yet the resources at each cloudlet have capacity constraints. Hence, it needs to be determined which requests should be admitted, subject to the capacity constraints on cloudlets.

The novelty of the work in this paper lies in the provisioning of reliability-aware VNF instances for network function services in MEC, by formulating a novel reliability-aware VNF instances provisioning problem. Efficient approximation and exact algorithms are proposed for allocating network resources to accommodate primary and secondary VNF instances in different cloudlets to meet individual user reliability requirements. Furthermore, the formulated optimization problem may be of independent interests as the Generalized Assignment Problem (GAP) is a special case of the problem. To the best of our knowledge, there is no prior study on similarly formulated problems, and this problem, which generalizes the GAP problem, has wide potential applications in practice, particularly in the fault-tolerance domain.

The main contributions of this paper are described as follows. We study the provisioning of reliability-aware VNF instances in an MEC to meet individual user reliability requirements. We first formulate a novel optimization problem for provisioning multiple VNF instances at different cloudlets in MEC to meet the reliability requirement of each request. We then show the NP-hardness of the problem and formulate an integer linear programming (ILP) solution to the problem when its size is small, otherwise, we develop an approximation algorithm with an approximation ratio of $O(\log K)$, where K is the number of cloudlets in the MEC. Moreover, we also consider two special cases of the problem. Specifically, we devise a constant approximation algorithm for a special case of the problem where each request needs only one primary and one secondary VNF instances. We also develop a dynamic programming algorithm that delivers an exact solution to another special case of the problem where the VNF instances of different types of virtual network functions demand the same amount of computing resources. We finally evaluate the performance of the proposed algorithms through experimental simulations. Experimental results demonstrate that the proposed algorithms are promising and the empirical results outperform their analytical counterparts as the theoretical estimations usually are very conservative.

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 introduces notions, notations, and the problem definitions. Section 4 shows the NP-hardness of the problem. Section 5 provides an ILP formulation of the problem, while Section 6 devises an algorithm with a logarithmic approximation ratio for the problem. Moreover, Section 7 deals with two special cases of the problem and proposes a constant approximation algorithm and an exact solution for the two special cases, respectively. Section 8 evaluates the proposed algorithms empirically, and Section 9 concludes the paper.

2 Related Work

Several studies on the provisioning of reliable VNF services have been conducted recently. Existing methods to improve reliability of NFV are summarized by Han *et al.* [17]. For example, Beck *et al.* [4] conducted one pioneering study related to survivability of VNF instances in the context of the VNF resource allocation, by admitting a set of VNF requests with the aim of reducing the amount of network resource allocated to VNF chains, while computing resilient allocations to protect network services from both link and VNF instance failures. They however did not differentiate reliability requirements of different requests. Casazza *et al.* [8] cast a problem of assigning virtual machines for network function implementations to servers in order to guarantee the availability of virtual machines via VNF instance replications. They considered a geo-distributed data center network in which there are sufficient resources to accommodate all user requests. Their objective is to maximize the minimum availability among all requests by developing heuristics. Ceselli *et al.* [9] dealt with the design of edge cloud networks with the aim to determine where to install cloudlets among the potential AP sites by developing efficient heuristics. Ding *et al.* [12] proposed an approach to enhancing the network resilience while maximizing cost-efficiency of the network through improving the ratio of the reliability to the backup cost of VNF instances. Fan *et al.* investigated how to map service function chains to a network with high end-to-end reliability requirements, by adopting on-site and off-site VNF instance backups [14], [15]. Their work is one of the first works on this topic, and recently Li *et al.* [25] proposed onsite and off-site backup mechanisms by developing efficient scheduling algorithms for the problem through adopting the primal and dual update technique. Kang *et al.* [23] studied the tradeoff between the reliability of a network service, as measured by the probability that the service is correctly executed, and the computational load of servers. Engelmann *et al.* [13] studied the end-to-end service reliability in data center networks (DCNs). They divided large flows into several smaller flows yet provided only one backup flow for reliability guarantee. Moualla *et al.* [28] considered the placement of service chains in DCNs, and devised an algorithm for the placement on the special fat-tree topological network structure. Kong *et al.* [24] proposed a mechanism that employs both backup path protection and VNF instance replication in order to guarantee the availability of a service function chain. The proposed mechanism determines the number of VNF instance replicas required for each VNF in the SFC, and allocates the replicas to physical nodes on the primary and backup paths while taking into account the ordered dependency among VNFs. Herker *et al.* [18] introduced several VNF backup strategies for service function chains, and analyzed the impact of different data-center architectures on the service provision. They then proposed cost-per-throughput algorithms for a given reliability requirement.

Carpio and Jukan [7] investigated how to improve service reliability through the joint consideration of replications and migrations. Qu *et al.* [29] aimed to minimize the communication bandwidth usage across the network while considering the availability requirements, by developing a heuristic. Aidi *et al.* [1] recently proposed a framework to efficiently manage survivability of service function chains and the backup VNFs. They aimed to determine both the minimum number and optimal locations of backup VNFs to protect service function chains. The proposed heuristics have been empirically demonstrated to deliver near-optimal solutions.

In this paper, we consider reliability-aware VNF service provisioning in MEC. We distinguish our work from existing ones as follows. Most existing studies focused on the placement of service function chains in geo-distributed networks that consists of many powerful data centers. We provide the very first study of reliability-aware VNF placement in MEC. We assume that the required network function service of a user can be implemented by a single, consolidated VNF instance. Additionally, existing studies mainly focused on end-to-end requirements of requests specified as the probability of component failures. In this paper, we differentiate reliability requirements of different users by providing different numbers of secondary VNF instances to ensure their reliability requirements, and devised efficient algorithms with performance guarantees for the problem of concern.

3 Preliminaries

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

3.1 Network model

We consider an MEC environment for a metropolitan region. Given a wireless metropolitan area network (WMAN) $G = (V, E)$ consisting of Access Points (APs) and K cloudlets, where each cloudlet k is co-located with an AP and they are interconnected by a high-speed optical cable, yet not all APs have co-located cloudlets, i.e., the number of cloudlets in the network is far less than the number of APs. Each cloudlet k has a (residual) computing capacity C_k. E is the set of links in which a link $e \in E$ between two APs if they are interconnected by an optical cable. We assume that there are $|\mathcal{F}|$ different types of network function services offered by the network service provider. Denote by $\mathcal{F} = \{f_1, f_2, \ldots, f_{|\mathcal{F}|}\}$ the set of network functions. The computing resource demand of each VNF instance of a network function $f_i \in \mathcal{F}$ is c_i. Mobile users can access the network ubiquitously through their nearby APs. Fig. 1 is an illustrative example of such a network that consists of five APs and three cloudlets.

3.2 User requests

Denote by $\mathcal{R}$ the set of user requests. Each user request $r_j \in \mathcal{R}$ is defined by a quadruple (f_j, R_j, c_j, p_j), where f_j $(\in \mathcal{F})$ is the requested type of VNF, R_j is the reliability requirement of request r_j with $0 < R_j \le 1$, c_j is the amount of computing resource demanded for implementing the VNF

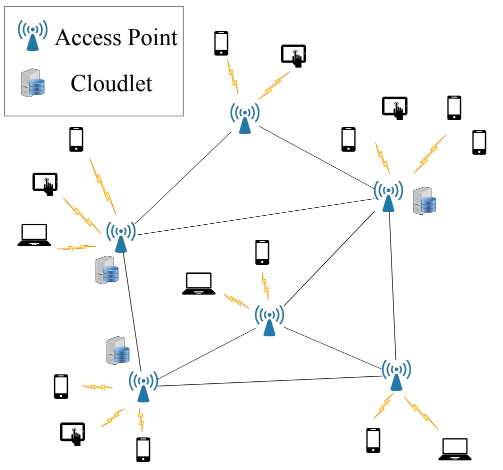


Fig. 1. An MEC consists of five APs and three cloudlets co-located with APs. User devices access the network through their nearby APs and APs are interconnected by optical links of the MEC.

instance of f_j in a cloudlet, and p_j $(\in \mathbb{R}^+)$ is the revenue collected by the service provider if request r_j is admitted. In contrast to some previous studies, we here assume that a user request can be served by a single VNF instance [22], [33], or a consolidated VNF instance [20]. This assumption is in alignment with real world deployments of VNF instances, such as Google's enterprise software-defined networks (SDN) [5], in which a single general-purpose computer implements multiple network functions, including network address translation (NAT), firewall, Intrusion Detection System (IDS), Dynamic Host Configuration Protocol (DHCP) daemon.

We assume that each request r_j needs $(n_j + 1)$ VNF instances of its network function f_j, to meet its reliable service requirement, where n_j is calculated by the reliability requirement of the request. Without loss of generality, we assume that $n_j + 1 \le K$, i.e., the reliability provided by the system through duplicating the VNF instances to different cloudlets is no more than the number of cloudlets K in the system. Among the $(n_j + 1)$ VNF instances for request r_j, one of the VNF instances will serve as *its primary VNF instance*, and the remaining n_i VNF instances will serve as *its secondary VNF instances*. Specifically, the value of $n_j \ge 0$ is determined by the reliability requirement R_j of request r_j, which is a minimum non-negative integer to meet the following inequality.

$$\begin{aligned}
&Pr(\text{at least one of the } (n_j+1) \text{ VNF instances is available})\\
&= 1 - Pr(f_j)^{n_j+1} \cdot \prod_{l=0}^{n_j} Pr(K_{j_l}) \qquad (1)\\
&= 1 - Pr(f_j)^{n_j+1} \cdot Pr(K)^{n_j+1} \qquad (2)\\
&\ge R_j, \qquad (3)
\end{aligned}$$

where $Pr(f_i)$ is the failure probability of an instance of network function f_i and $Pr(K_{j_l})$ is the failure probability of the lth instance of request r_j, and step (1) in Eq. (3) follows because, in order to be functionally identical to the primary instance, the secondary VNF instances are often running on the same software version and using the same configuration as their primary VNF instance, thereby having the same failure probabilities as the failure probability of the primary

VNF instance. Step (2) in Eq. (3) follows we assume that cloudlets in the system have the same failure probabilities. This assumption is in alignment with the common industry practice where network operators keep their equipments to a limited range so that they can standardize the process and minimize their OPEX. We also assume that failures of VNF instance are independent with each other and links (optical cables) in G are reliable.

As most network functions are stateful, the primary and secondary VNF instances of a network function need to synchronize with each other so that there is minimum interruptions when the primary VNF instance fails. It is worth noting that the internal state of a VNF instance often contain sensitive information, e.g., encryption keys. If synchronization of the internal states of primary and secondary VNF instances is via the same links as user traffic, sensitive information may be exposed. Instead, the internal state of the primary VNF instance should be periodically distributed to secondary VNF instances over a dedicated out-of-band, i.e., separate from user traffic, network. The commonly used Control Plane Network (CPN) [16], [31] is an excellent candidate for synchronization between the primary and secondary VNF instances of a network function.

3.3 Problem definition

Given an MEC $G = (V, E)$, a set $\mathcal{R}$ of requests, associated with each request $r_j \in \mathcal{R}$, there is a payment p_j, a network function $f_j \in \mathcal{F}$, and a reliability requirement represented by the number n_j $(0 \leq n_j < K)$ of secondary VNF instances with each of them being placed at a different cloudlet, *the reliability-aware VNF instances provisioning problem* is to admit as many requests in $\mathcal{R}$ as possible such that the total revenue collected by the service provider is maximized, subject to computing capacity constraint on each cloudlet and the specified reliability guarantee of each admitted request.

A request r_j is *admitted* if its $(n_j + 1)$ VNF instances are placed at $(n_j + 1)$ different cloudlets, and each cloudlet has sufficient computing resource to meet the computing resource demand c_j of its network function f_j, assuming that $\max_{1 \leq j \leq |\mathcal{R}|}\{n_j\} \leq K - 1$. There are a set $\mathcal{R}$ of requests, each request $r_j \in \mathcal{R}$ needs $n_j + 1$ replicas of its VNF f_j, and its admission will result in a revenue of p_j where each instance of f_j demands the amount c_{j_i} of computing resource from each of the $n_j + 1$ cloudlets. The problem is to admit as many requests in $\mathcal{R}$ as possible to maximize the revenue, subject to the cloudlet computing capacity constraint.

Notice that in this paper we do not explicitly consider user mobility. However, we assume the users are allowed to move in the MEC, and they can issue their requests for services through the APs in which they are located.

Table 1 lists the frequently used notations used in this paper.

TABLE 1
Table of Symbols

Notations	Descriptions
$G = (V, E)$	a WMAN
$k \in K$	a cloudlet co-located with an AP
C_k	the residual computing capacity of cloudlet k
$\mathcal{F}\ (= f_1, f_2, \ldots, f_{\lvert\mathcal{F}\rvert})$	the set of network functions offered in G
f_i	a network functions offered in G
c_i	the computing resource demand of network function f_i
$\mathcal{R}$	the set of user requests
$r_j = (f_j, R_j, c_j, p_j) \in \mathcal{R}$	a user request, where f_j $(\in \mathcal{F})$ is the requested type of VNF, R_j is the reliability requirement of request r_j with $0 < R_j \leq 1$, c_j is the amount of computing resource demanded, and p_j $(\in \mathbb{R}^+)$ is the revenue
n_j	the number of secondary VNF instances needed for request r_j to meet its reliable service requirement

4 NP-Hardness

Theorem 1. The reliability-aware VNF instances provisioning problem in an MEC $G(V, E)$ is NP-complete.

Proof: We show the claim through a reduction from the well-known NP-complete knapsack problem that is described as follows. Given n items, each item a_i has a weight w_i and a profit p_i with $1 \leq i \leq n$. There is a bin with capacity W. If an item a_j can be packed into the bin, it brings a profit of p_j. *The knapsack problem* is to pack as many items as possible such that the total profit of packed items is maximized, subject to the bin capacity W.

Given an instance of the knapsack problem, we construct an instance of the reliability-aware VNF instances provisioning problem as follows. We assume that there are two cloudlets $(K = 2)$ with each having W computing capacity. There are n requests, which correspond the n items in the knapsack problem. We further assume that each request r_j has only one secondary VNF instance, i.e., $n_j = 1$. As each request implementation has a primary VNF instance. Thus, there are two VNF instances for each request allocated to two cloudlets. The computing demand of each of the two VNF instances of r_j is w_j. If request r_j is admitted, the revenue brought by its implementation is p_j. Then, the reliability-aware VNF instances provisioning problem is to admit as many requests as possible such that the total revenue collected from admitted requests is maximized, subject the computing capacity constraints on the two cloudlets.

It can be seen that a solution to the latter in turn returns a solution to the former, while the reduction from an instance of the knapsack problem to an instance of the reliability-aware VNF instances provisioning problem takes polynomial time. Also, verifying the solution of the latter can be easily done in polynomial time. As the knapsack problem is NP-complete, the reliability-aware VNF instances provisioning problem is NP-complete, too. □

5 Integer Linear Programming

In this section, we formulate an ILP solution for the problem. We assume that x_j is a boolean variable, where $x_j = 1$ implies that request r_j is admitted; otherwise, it will not be admitted. Furthermore, if r_j is admitted, there are exactly $n_j + 1$ cloudlets with each having the amount c_j of computing resource to meet the computing resource demand by its primary and secondary VNF instances. The optimization objective thus is

$$\textit{maximize} \qquad \sum_{j=1}^{n} p_j x_j,$$

subject to the following constraints.

$$\sum_{j=1}^{|\mathcal{R}|} c_j \cdot y_{j,k} \leq C_k, \quad \forall k, 1 \leq k \leq K \tag{4}$$

$$(n_j + 1) \cdot x_j = \sum_{k=1}^{K} y_{j,k}, \quad \forall r_j \in \mathcal{R} \tag{5}$$

$$x_j \in \{0,1\}, \quad \forall r_j \in \mathcal{R} \tag{6}$$

$$y_{j,k} \in \{0,1\}, \quad \forall r_j \in \mathcal{R},\ \forall k, 1 \leq k \leq K, \tag{7}$$

where $y_{j,k}$ is a boolean variable, $y_{j,k} = 1$ if a VNF instance of request r_j is allocated to cloudlet k; $y_{j,k} = 0$ otherwise. Constraint (4) says that the total computing demand by the VNF instances of all different requests r_j in any cloudlet k is no greater than its residual computing resource capacity. Constraint (5) ensures that the number of secondary VNF instances at different cloudlets for each admitted request r_j is exactly $n_j + 1$. Constraints (6) and (7) limit the ranges of boolean variables x_j and $y_{j,k}$ to either 0 or 1.

6 Approximation Algorithm for the Reliability-Aware VNF Instances Provisioning Problem

In this section, we propose an approximation algorithm for the reliability-aware VNF instances provisioning problem. We start with cost modeling, and then provide an overview of the proposed algorithm and details of the algorithm. We finally analyze the approximation ratio and time complexity of the proposed algorithm.

6.1 Cost modeling

The approximation algorithm examines the requests in $\mathcal{R}$ one by one. When request r_j is considered, the resource availability of cloudlets affects whether r_j can be admitted. We thus denote by $C_k(j)$ the amount of residual computing resource at cloudlet k with $C_k(0) = C_k$.

The key to the approximation algorithm is that we use an exponential function to model the cost $w_k(j)$ of instantiating a VNF instance of request r_j with computing resource demand c_j at cloudlet k, which is defined as follows.

$$w_k(j) = \frac{C_k}{c_j}(\alpha^{1-\frac{C_k(j)}{C_k}} - 1), \tag{8}$$

where $\alpha > 1$ is a constant to reflect the sensitivity of the workload at each cloudlet, which will be determined later, and $1 - \frac{C_k(j)}{C_k}$ is *the utilization ratio* of cloudlet k when request r_j is considered. The rationale is that the high utilization a resource has, the higher the risk associated with the resource utilization, thus the usage of resources with high utilization should be discouraged. The proposed exponential function will guide the resource allocations.

6.2 Algorithm description

We first sort all requests by the ratio of the payment of each request to the total amount of its computing resource demand to meet its VNF instance reliability. In other words, we rank request r_j ahead of request r_i if $\frac{p_j}{(n_j+1)c_j} \geq \frac{p_i}{(n_i+1)c_i}$ for any two requests $r_i \in \mathcal{R}$ and $r_j \in \mathcal{R}$. For the sake of convenience, we assume that the sorted request sequence is $r_1, r_2, \ldots, r_{|\mathcal{R}|}$. We then examine requests in the sequence one by one to determine whether a request will be admitted immediately.

For each request r_j, we calculate *its normalized cost* if one of its VNF instances is instantiated at cloudlet k as follows.

$$\psi_k(j) = \frac{w_k(j)}{C_k} = \frac{\alpha^{1-\frac{C_k(j)}{C_k}} - 1}{c_j}. \tag{9}$$

For each request r_j, we identify top-$(n_j + 1)$ cloudlets with the lowest normalized costs, and denote by $\mathcal{K}_j$ the set of cloudlets. We here introduce *an admission control policy for request admissions*. That is, if the sum of normalized costs of the identified top-$(n_j + 1)$ cloudlets for request r_j is greater than K, i.e.,

$$\frac{\sum_{k \in \mathcal{K}_j} \psi_k(j)}{p_j} > K, \tag{10}$$

then request r_j will be rejected. Otherwise, it will be admitted, and its $(n_j + 1)$ VNF instances will be placed in the top-$(n_j + 1)$ cloudlets. The algorithm detail for the reliability-aware VNF instances provisioning problem is given in `Algorithm` 1.

Algorithm 1 Approximation algorithm for the reliability-aware VNF instances provisioning problem

```
Input: A set of K cloudlets with each having a residual computing capacity $C$, a set of requests $\mathcal{R}$ with each request $r_j = (f_j, R_j, c_j, p_j)$
Output: Admit a subset of requests in $\mathcal{R}$ that maximizes the sum of revenues of admitted requests while meeting the reliability requirement of each admitted request.
 1: $\mathcal{A} \leftarrow \emptyset$ /* the set of admitted requests */;
 2: Sort requests in $\mathcal{R}$ by the ratio of the payment of each request to the total amount of its computing resource demand to meet its VNF instance reliability;
 3: for each request $r_j$ in the sorted order do
 4:     Calculate the number $n_j$ of secondary VNF instances of its network function by Inequality (3);
 5:     Assign each cloudlet $k$ a cost by Eq. (9) if $r_j$ is admitted;
 6:     Identify top-$(n_j+1)$ cloudlets $C_{j_1}, C_{j_2}, \ldots, C_{j_{n_j}}, C_{j_{(n_j+1)}}$, with smallest costs, by a linear time selection algorithm;
 7:     if there is a cloudlet $j_k$ among the $n_j + 1$ identified cloudlets such that $C_{j_k} < c_j$ or the sum of costs of the $(n_j + 1)$ identified cloudlets is greater than $K \cdot p_j$ (by Inequality (10)) then
 8:         Reject request $r_j$;
 9:     else
10:         Allocate a VNF instance of request $r_j$ to each of the top-$(n_j + 1)$ cloudlets;
11:         Update the residual computing capacity of each of these cloudlets as $C_{j_k} \leftarrow C_{j_k} - c_j$ for all $k$ with $1 \leq k \leq n_j+1$;
12:         $\mathcal{A} \leftarrow \mathcal{A} \cup \{r_j\}$;
13:     end if
14: end for
15: return Set $\mathcal{A}$ of admitted requests and their primary and secondary VNF instance placements in cloudlets.
```

6.3 Algorithm analysis

In the following, we first provide an upper bound on the sum of costs of all cloudlets in G after a subset of requests in $\mathcal{R}$ is admitted. We then show a lower bound on the sum of costs of cloudlets that an optimal solution uses to admit a request rejected by the proposed algorithm, and we finally analyze the approximation ratio of the proposed approximation algorithm.

Lemma 1. Given an MEC $G(V,E)$ and a set of requests $\mathcal{R}$, denote by $\mathcal{A}$ the set of requests admitted by `Algorithm` 1, then, the sum of costs of all K cloudlets is

$$\sum_{k=1}^{K} w_k(j) \le 2K \cdot \log\alpha \cdot \sum_{r_{j'}\in\mathcal{A}} p_{j'},$$

where α is a constant with $2K\cdot Q_{\max}+2 \le \alpha \le 2^{C_{\min}/c_{\max}}$, $C_{\min}$ $(= \min\{C_k \mid 1\le k\le K\})$ is the minimum cloudlet computing capacity, $\mathcal{R}$ is the set of requests, and $Q_{\max}$ $(= \max\{p_{j'}\cdot c_{j'} \mid r_{j'}\in\mathcal{R}\})$ is the maximum product of the profit and computing resource demand among requests.

Please see Appendix A for the proof.

Let $\mathcal{D}$ $(\subseteq \mathcal{R})$ be the set of requests that are admitted by an optimal algorithm but rejected by the proposed approximation algorithm. We now prove a lower bound on the sum of normalized costs of all cloudlets used to admit any request in $\mathcal{D}$.

Lemma 2. Let $\mathcal{D}$ be the set of requests that are admitted by an optimal algorithm yet rejected by the approximation algorithm, and let $\mathcal{K}^{opt}_{j'}$ be the set of $(n_{j'}+1)$ cloudlets to which the optimal algorithm places VNF instances for request $r_{j'}\in\mathcal{D}$. Then, for any request $r_{j'}\in\mathcal{D}$, we have

$$\frac{\sum_{k\in\mathcal{K}^{opt}_{j'}} \psi_k(j')}{p_{j'}} \ge K, \tag{11}$$

where α is a constant with $2K\cdot Q_{\max}+2\le\alpha\le 2^{C_{\min}/c_{\max}}$, $C_{\min}$ $(= \min\{C_k \mid 1\le k\le K\})$ is the minimum cloudlet computing capacity, and $Q_{\max}$ $(= \max\{p_{j'}\cdot c_{j'} \mid r_{j'}\in\mathcal{R}\})$ is the maximum product of the profit and computing resource demand among requests.

Please see Appendix B for the proof.

Theorem 2. Given an MEC $G(V,E)$, K cloudlets in G, and a set $\mathcal{R}$ of requests, there is an approximation algorithm, `Algorithm` 1, with an approximation ratio of $O(\log K)$ for the reliability-aware VNF instances provisioning problem, which takes $O(|\mathcal{R}|\cdot\log|\mathcal{R}|+|\mathcal{R}|\cdot K)$ time.

Proof: Let $\mathbb{P}_{opt}$ and $\mathbb{P}$ be the total revenues of admitted requests by an optimal algorithm and the proposed approximation algorithm for requests in $\mathcal{R}$, respectively, we then have

$$\begin{aligned}
K(\mathbb{P}_{opt}-\mathbb{P}) &\le K\sum_{r_{j'}\in\mathcal{D}} p_{j'} = \sum_{r_{j'}\in\mathcal{D}} K\cdot p_{j'}\\
&\le \sum_{r_{j'}\in\mathcal{D}}\Big(\sum_{k\in\mathcal{K}^{opt}_{j'}}\frac{w_k(j')}{C_k}\Big), \quad \text{by Lemma 2}\\
&\le \sum_{r_{j'}\in\mathcal{D}}\Big(\sum_{k\in\mathcal{K}^{opt}_{j'}}\frac{w_k(j)}{C_k}\Big) && (12)\\
&\le \sum_{k=1}^{K} w_k(j)\sum_{r_{j'}\in\mathcal{D}}\Big(\sum_{k'\in\mathcal{K}^{opt}_{j'}}\frac{1}{C_{k'}}\Big) && (13)\\
&\le \sum_{k=1}^{K} w_k(j)\cdot 1 && (14)\\
&= \sum_{k=1}^{K} w_k(j) \le 2K\log\alpha\cdot\sum_{j'\in\mathcal{A}} p_{j'} \quad \text{by Lemma 1.} && (15)
\end{aligned}$$

Notice that Inequality (12) holds because the utilization of each resource does not decrease and consequently the cost of any cloudlet k with $1\le k\le K$ does not decrease with more request admissions. Inequality (13) holds since $\sum_{i=1}^{p}\sum_{j=1}^{q}A_iB_j \le \sum_{i=1}^{p}A_i\sum_{j=1}^{q}B_j$ for all $A_i\ge 0$ and $B_j\ge 0$. The proof of Inequality (14) proceeds as follows. All algorithms, including an optimal algorithm for the problem of concern, the total amount of allocated computing resources in any cloudlet is no more than the capacity of the cloudlet.

By Inequality (15), we have

$$\begin{aligned}
\frac{\mathbb{P}_{opt}}{\mathbb{P}} &= \frac{\mathbb{P}_{opt}-\mathbb{P}}{\mathbb{P}}+1\\
&\le \frac{2\log\alpha\cdot\sum_{r_{j'}\in\mathcal{A}}p_{j'}}{\sum_{r_{j'}\in\mathcal{A}}p_{j'}}+1 \le 2\log\alpha+1\\
&= O(\log(K\cdot Q_{\max})), \quad \text{when } \alpha = 2K\cdot Q_{\max}+2.\\
&= O(\log K+\log Q_{\max})\\
&= O(\log K), \quad \text{as } Q_{\max} \text{ usually is a constant,}
\end{aligned}$$

where $Q_{\max}$ is the maximum product of the profit and computing resource demand among requests, i.e., $Q_{\max} = \max\{p_{j'}\cdot c_{j'} \mid r_{j'}\in\mathcal{R}\}$.

The running time of `Algorithm` 1 is dominated by the time required to sort all requests by their ratio, which takes $O(|\mathcal{R}\cdot|\log|\mathcal{R}|)$ time. Then, there are $|\mathcal{R}|$ iterations, and within each iteration j, it identifies the top-(n_j+1) cloudlets with the smallest weights using a linear time selection algorithm. The algorithm thus takes $O(|\mathcal{R}|\cdot\log|\mathcal{R}|+|\mathcal{R}|\cdot K)$ time. □

7 Approximation and Exact Algorithms for Special Cases of the Reliability-Aware VNF Instances Provisioning Problem

In this section, we study two special cases of the reliability-aware VNF instances provisioning problem, and propose a constant approximation algorithm and an exact algorithm for the cases, respectively.

7.1 A constant approximation algorithm for a special reliability-aware VNF instances provisioning problem

We start dealing with a special case of the reliability-aware VNF instances provisioning problem where each request has only one primary and secondary VNF instances, i.e., $n_j = 1$ for all requests $r_j\in\mathcal{R}$. Even for this special case, it is still NP-hard, for which we develop a constant approximation algorithm by a non-trivial reduction to the well-known Generalized Assignment Problem (GAP) [10], and a solution to the latter in turn returns a feasible solution to the former.

The Generalized Assignment Problem (GAP) is defined as follows. Given a set $\mathcal{B}$ of K bins with each bin $B_k\in\mathcal{B}$

having identical capacity of C, a set $\mathcal{I}$ of N items, and for each pair of item $I_j \in \mathcal{I}$ and bin $B_k \in \mathcal{B}$, a profit $p(I_j, B_k)$ is obtained if item I_j is placed to bin B_k with size $s(I_j, B_k)$, the GAP is to pack as many as items in set $\mathcal{I}$ into the bins in $\mathcal{B}$ such that the total profit is maximized, subject to the capacity constraint on each bin.

Given an instance of this special reliability-aware VNF instances provisioning problem, we construct an instance of the GAP as follows. We first assume that the K cloudlets are indexed into $1, 2, \ldots, K$ and K is even. Otherwise, we assume that there are $K+1$ cloudlets, and one cloudlet is a *virtual one* that implies that it does not exist. We pair the K cloudlets into $K/2$ pairs, let $B'_1, B'_2, \ldots, B'_{K/2}$ be the $K/2$ *pair-bins* with each having $2C$ computing capacity as the instance of the GAP problem. Each request $r_j \in \mathcal{R}$ has a corresponding item I_j and with resource demand $2c_j$ for its primary and secondary VNF instances.

If the primary and secondary VNF instances of request $r_j \in \mathcal{R}$ will be placed into one pair of cloudlets, corresponding to one bin B'_k with size $2c_j$, then the capacity $2C$ of bin B'_k is no less than the total resource demand $2c_j$ by these two VNF instances, and this placement will result in a profit (revenue) $p(I_j, B'_k)$ that is defined as

$$p(I_j, B'_k) = p_j,$$

where $1 \le j \le |\mathcal{R}|$ and $1 \le k \le K/2$.

The size of placing item I_j into B'_k is $s(I_j, B'_k) = 2c_j$, corresponding to the resource demand $2c_j$ of VNF instances of request r_j.

The constructed GAP instance can be solved by invoking the approximation algorithm due to Cohen *et al.* [10] with an approximation ratio of $\frac{1}{2+\epsilon}$, where ϵ is a constant with $0 < \epsilon \le 1$.

The solution delivered by the approximation algorithm will be in the form of assigning a set $\mathcal{I}' \subseteq \mathcal{I}$ of requests. Specifically, each item $I_j \in \mathcal{I}'$ that corresponds to the primary and secondary VNF instances of request r_j are assigned to cloudlets B_{2k} and B_{2k+1} if its corresponding item I_j is assigned to bin B'_k with $1 \le k \le K/2$.

We then extend the solution from the even K to the odd K, for which we create *a virtual cloudlet* with the same capacity as other cloudlets. Thus, we now have $K' = K+1$ bins. They are corresponding $K'/2$ pair-bins $B'_1, B'_2, \ldots, B'_{\frac{K+1}{2}}$ with each having the computing capacity of $2C$. Assume that the virtual cloudlet is paired with the cloudlet indexed K. We then apply the approximation algorithm for the GAP problem due to Cohen *et al.* [10].

Let $\mathcal{I}'$ be the approximate solution and $\mathcal{I}'_1.\mathcal{I}'_2, \ldots, \mathcal{I}'_{K'/2}$ the sets of admitted requests in bins $B'_1, B'_2, \ldots, B'_{K'/2}$, respectively. Let $P'_1, P'_2, \ldots, P'_{K'/2}$ be the profit sum of requests admitted in their corresponding pair-bins.

Denote by $\sum_{i=1}^{K'/2} P'_i$ the total revenue collected by admitting the requests in $\mathcal{I}'$. The average profit among the $K'/2$ pair-bins thus is $\frac{\sum_{i=1}^{K'} P'_i}{K'/2}$. Let $P'_{\min} = \min_{1 \le i \le K'/2}\{P'_i\}$. If $P'_{K'/2} = P'_{\min}$, we discard the requests in $B'_{K'/2}$ from the solution, i.e., $\mathcal{I}' = \mathcal{I}' \setminus \mathcal{I}'_{K'/2}$; otherwise, let $P'_{i_0} = P'_{\min}$ and $i_0 \ne K'/2$, we swap the admitted requests between pair-bin B'_{i_0} and pair-bin $B'_{K'/2}$, i.e., $\mathcal{I}'_{i_0} = \mathcal{I}'_{K'/2}$ and $\mathcal{I}'_{K'/2} = \mathcal{I}'_{i_0}$. Then, the solution is $\mathcal{I}' = \mathcal{I}' \setminus \mathcal{I}'_{K'/2}$. The detailed algorithm is given in `Algorithm` 2.

Algorithm 2 An approximation algorithm for the special reliability-aware VNF instances provisioning problem with $n_j = 1$ for every request $r_j \in \mathcal{R}$

```
Input: A set of K cloudlets with each having a residual
    computing capacity C, a set of requests R with each request
    r_j = (f_j, R_j, c_j, p_j)
Output: Admit a subset I' of requests in R that maximizes the
    sum of revenues of admitted requests while meeting the
    reliability of each admitted request.
 1: if $K$ is even then
 2:    Construct an instance of the GAP, where each request $r_j \in \mathcal{R}$ has a corresponding item $I_j$ and each pair of cloudlets has a corresponding bin $B'_k$ with bin capacity $cap(B'_k) = 2C$ with $1 \le k \le K/2$;
 3:    Find an approximate solution $\mathcal{I}'$ to the GAP problem using the approximation algorithm due to Cohen et al. [10];
 4: else
 5:    Construct an instance of the GAP, where each request $r_j \in \mathcal{R}$ has a corresponding item $I_j$ and each pair of cloudlets has a corresponding bin $B'_k$ with bin capacity $cap(B'_k) = 2C$ with $1 \le k \le \frac{K+1}{2}$, where a virtual cloudlet (bin) is added to the system, and let $K' = K+1$, which is even;
 6:    Find an approximate solution $\mathcal{I}'$ to the GAP problem using the approximation algorithm due to Cohen et al. [10];
 7:    $P'_{\min} \leftarrow \infty$;
 8:    for $i \leftarrow 1$ to $K'/2$ do
 9:       Calculate $P'_i$ from $\mathcal{I}'_i$;
10:       if $P'_i < P'_{\min}$ then
11:          $P'_{\min} \leftarrow P'_i$; $i_0 \leftarrow i$; $\mathcal{I}'_{temp} \leftarrow \mathcal{I}'_{i_0}$;
12:       end if;
13:    end for;
14:    if $i_0 = K'/2$ then
15:       $\mathcal{I}' \leftarrow \mathcal{I}' \setminus \mathcal{I}'_{K'/2}$;
16:    else
17:       $\mathcal{I}'_{i_0} \leftarrow \mathcal{I}'_{K'/2}$; $\mathcal{I}'_{K'/2} \leftarrow \mathcal{I}'_{temp}$; $\mathcal{I}' \leftarrow \mathcal{I}' \setminus \mathcal{I}'_{K'/2}$;
18:    end if;
19: end if;
20: for each item $I_j \in \mathcal{I}'_k$ assigned to bin $B'_k$ with $1 \le k \le \lceil K/2 \rceil$ do
21:    if $k \le \lfloor K/2 \rfloor$ then
22:       Instantiate the primary and secondary VNF instances of $f_j$ for request $r_j$ in cloudlets $2k-1$ and $2k$;
23:    end if;
24: end for;
25: return the set of admitted requests and their primary and secondary VNF instance placements in cloudlets.
```

Theorem 3. Given an MEC $G(V, E)$ that contains K cloudlets with each having identical residual computing capacity C, and a set of requests $\mathcal{R}$ with each request $r_j \in \mathcal{R}$ having exactly one secondary VNF instance, there is an approximation algorithm with an approximation ratio of $\frac{1}{6+\epsilon'}$, `Algorithm` 2, for this special reliability-aware VNF instances provisioning problem. The proposed algorithm takes $O(\frac{|\mathcal{R}|\cdot K}{\epsilon'} + \frac{K}{\epsilon'^4})$ time, where ϵ' is a constant with $0 < \epsilon' \le 3$.

Proof: We first show that the solution delivered by `Algorithm` 2 is feasible. Since a pair-bin B'_k that corresponds to cloudlets $2k$ and $2k+1$ has the computing capacity of $2C$, and the size $s(I_j, B'_k) = 2c_j$ of placing item I_j (admitting request r_j by instantiating its primary and secondary VNF instances into B_{2k} and B_{2k+1}) is equal to their resource demand $2c_j$, it follows that the allocation of VNF instances by solving the GAP problem, i.e., the capacity constraint on each cloudlet will not be violated.

We then analyze the approximation ratio of `Algorithm` 2, by distinguishing it into two cases, depending on whether the value of K is even or not. If K is even, then the solution is an approximate solution, which is $\frac{1}{2+\epsilon}$ times the optimal by the approximation algorithm in [10]; otherwise (K is odd), we create a virtual cloudlet indexed as K' $(= K+1)$. Let $\mathcal{I}'$ be the approximate solution delivered by the approximation algorithm. As this solution is built upon the assumption that there are K' cloudlets, we in fact have only K cloudlets in the network. Following `Algorithm` 2, let A be the total profit of the approximate solution, the minimum profit among the $K'/2$ pairs of cloudlets thus is no greater than $\frac{A}{K'/2}$. As we will remove all admitted requests in the minimum profit pair-bin from the solution, the resulting profit by the solution thus is at least $A - \frac{A}{K'/2} = A(1 - \frac{2}{K+1}) \geq A/3$ due to the fact that $K \geq 2$. Since $A \geq \frac{OPT}{2+\epsilon}$, $\frac{A}{3} \geq \frac{OPT}{6+\epsilon'}$, where OPT is the optimal solution to the problem, and $\epsilon' = 3\epsilon$ is a constant with $0 < \epsilon' \leq 3$.

We finally analyze the running time of `Algorithm` 2. The construction of the GAP instance takes $O(|\mathcal{R}| \cdot K)$ time, while invoking the approximation algorithm due to Cohen *et al.* [10] takes $O(\frac{|\mathcal{R}| \cdot K}{\epsilon'} + \frac{K}{\epsilon'^4})$ time. The solution delivered by the proposed algorithm, `Algorithm` 2, thus is no less than $\frac{1}{6+\epsilon'}$ times the optimal one, where ϵ' is a constant with $0 < \epsilon' \leq 3$. □

7.2 A dynamic programming algorithm for another special reliability-aware VNF instances provisioning problem

We then study another special case of the reliability-aware VNF instances provisioning problem where different VNF instances of different network functions have the same amounts of computing resource demands, i.e., $c_i = c_j$ for all $f_i \in \mathcal{F}$ and $f_j \in \mathcal{F}$. For the sake of convenience, we assume that the computing resource demand by each network function is one computing unit. We propose an exact algorithm for the problem through a reduction to a profit maximization problem (defined later), and the solution to the latter in turn returns a solution to the former. The reduction is as follows.

7.2.1 A dynamic programming algorithm

Denote by n the number of requests $r_1, r_2, \ldots, r_n$ in $\mathcal{R}$, which correspond n jobs $J_1, J_2, \ldots, J_n$. There are K cloudlets in G, and each can be treated as a bin B_i with computing capacity C_i, $1 \leq i \leq K$. The reliability requirement of request r_j is implemented by placing $(n_j + 1)$ VNF instances to $n_j + 1$ cloudlets with each being allocated a computing unit at each cloudlet. The implementation of job J_j will take U_j bins and consume one computing unit in each of the chosen U_j bins. The profit (revenue) obtained by implementing job J_j is p_j. *The profit maximization problem* then is to find a subset $\mathcal{A}$ of $\mathcal{R}$ ($\mathcal{A} \subseteq \mathcal{R}$) such as the sum of profits of admitted requests in $\mathcal{A}$ is maximized, subject to the computing capacity on each cloudlet in the network.

The defined profit maximization problem can be solved, using dynamic programming as follows. Without loss of generality, we assume

$$C_1 \geq C_2 \geq \ldots \geq C_K, \tag{16}$$

and

$$U_1 \geq U_2 \geq \ldots \geq U_n. \tag{17}$$

Because each job needs at most one computing unit from any bin, we assume $n \geq C_1$, otherwise excessive capacity larger than n will be useless.

Let $W_i = |\{j \mid C_j \geq i\}|$ be the number of bins that have at least i computing units, for each and every i with $1 \leq i \leq n$. Notice that some W_i may be zero. We then have

$$W_1 \geq W_2 \geq \ldots, \geq W_n. \tag{18}$$

Definition: If a job J is selected which needs U computing units, then we assign one computing unit starting from the bin with the largest remaining capacity in an non-increasing order of remaining capacities until U computing units have been assigned. We refer to this scheduling method as *the canonical scheduling*.

Theorem 4. All jobs $J_1, J_2, \ldots, J_n$ can be selected if and only if the following condition is satisfied.

$$\sum_{j=1}^{i} U_j \leq \sum_{j=1}^{i} W_j \quad \text{for all } i \text{ with } 1 \leq i \leq n. \tag{19}$$

Proof: We prove Claim (19) by induction on the number of jobs n.

We start with induction basis. When $n = 1$, there is only one job that requires U_1 computing units. Therefore, if J_1 can be selected, we must have $U_1 \leq W_1$. On the other hand, if $U_1 \leq W_1$, then J_1 can be selected because its demanded U_1 computing units can be satisfied. We then assume that Claim (19) holds for all n with $1 \leq n \leq h$. We finally show that Claim (19) also holds when $n = h+1$ as follows.

On one hand, suppose that all $h+1$ jobs can be selected. In this case, since the first h jobs can be selected by the induction assumption, we have $\sum_{j=1}^{i} U_j \leq \sum_{j=1}^{i} W_j$ for all i with $1 \leq i \leq h$. Now, because the first $h+1$ jobs can also be selected, we then must have $\sum_{j=1}^{h+1} U_j \leq \sum_{j=1}^{h+1} W_j$ where $\sum_{j=1}^{h+1} U_j$ is the total number of computing units required by the $h+1$ jobs and $\sum_{j=1}^{h+1} W_j$ is the total number of available computing units provided by all bins, because any bin with a capacity larger than $h+1$ can only contribute at most $h+1$ computing units.

On the other hand, suppose Claim (19) holds, i.e., $\sum_{j=1}^{i} U_j \leq \sum_{j=1}^{i} W_j$ for all i with $1 \leq i \leq h+1$. We show that all $h+1$ jobs can be selected by canonically scheduling J_1 and distinguishing two cases: Case A and Case B, to deal with the remaining h jobs.

Case A: if $U_1 = W_1$, then we have $\sum_{j=2}^{i} U_j \leq \sum_{j=2}^{i} W_j$ for all i with $2 \leq i \leq h+1$. Because there are h remaining jobs, by the induction assumption, all the remaining h jobs can be selected.

Case B: if $U_1 < W_1$, then not all W_1 bins are used, and there are $W_1 - U_1$ bins not used. We relabel the h remaining jobs by labeling job J_2 as J'_1, J_3 as J'_2, …, and J_{h+1} as J'_h. Accordingly, we have $U'_1 = U_2$, $U'_2 = U_3$, …, $U'_h =$

U_{h+1}. Denote by W'_i the number of bins that have at least i remaining computing units for all i with $1 \leq i \leq n$. We further distinguish Case B into two subcases: (i) $W_2 \leq U_1$; and (ii) $W_2 > U_1$, respectively.

Subcase (i): if $W_2 \leq U_1$, then

$$\begin{aligned} W'_1 &= W_2 + W_1 - U_1, \qquad (20)\\ W'_2 &= W_3,\\ &\vdots\\ W'_h &= W_{h+1}. \end{aligned}$$

We thus have $U'_1 = U_2 \leq W_2 + W_1 - U_1 = W'_1$, because $U_1 + U_2 \leq W_1 + W_2$. In general, for all i with $2 \leq i \leq h$, we have

$$\begin{aligned} \sum_{j=1}^{i} U'_j &= \sum_{j=2}^{i+1} U_j = \sum_{j=1}^{i+1} U_j - U_1\\ &\leq \sum_{j=1}^{i+1} W_j - U_1, \quad \text{by Claim (19)}\\ &= \sum_{j=3}^{i+1} W_j + (W_2 + W_1 - U_1), \quad \text{by Eq. (20)}\\ &= \sum_{j=2}^{i} W'_j + W'_1 = \sum_{j=1}^{i} W'_j \qquad (21) \end{aligned}$$

By the induction assumption, all h remaining jobs can be selected by the bins.

Subcase (ii): Assume that there is a p such that $W_2 > U_1$, $W_3 > U_1, \ldots, W_p > U_1$ but $W_{p+1} \leq U_1$. We then have

$$\begin{aligned} W'_1 &= W_2 + (W_1 - W_2) = W_1,\\ W'_2 &= W_3 + (W_2 - W_3) = W_2,\\ &\vdots\\ W'_{p-1} &= W_p + (W_{p-1} - W_p) = W_{p-1}, \qquad (22)\\ W'_p &= W_{p+1} + (W_p - U_1), \qquad (23)\\ W'_{p+1} &= W_{p+2},\\ &\vdots\\ W'_h &= W_{h+1}. \qquad (24) \end{aligned}$$

Note that if $p = h+1$, then from Inequality (17) we have $W_j > U_j$ for all j with $1 \leq j \leq h+1$. Then, all jobs can be selected. We thus assume that $p \leq h$, and this situation is illustrated in Fig. 2. We now have $W'_1 = W_1 \geq W_2 > U_1 \geq U_2 = U'_1$.

For $i = 2, \ldots, p-1$, we have

$$\begin{aligned} \sum_{j=1}^{i} U'_j &= \sum_{j=2}^{i+1} U_j \leq \sum_{j=2}^{i+1} U_1, \quad \text{from inequality (17)}\\ &\leq \sum_{j=1}^{i} W_j, \quad \text{from the assumption of subcase (ii)}\\ &\leq \sum_{j=1}^{i} W'_j. \qquad (25) \end{aligned}$$

For $i = p, p+1, \ldots, h$, we have

$$\begin{aligned} \sum_{j=1}^{i} W'_j &= \sum_{j=1}^{p-1} W'_j + W'_p + \sum_{j=p+1}^{i} W'_j,\\ &= \sum_{j=1}^{p-1} W_j + W_{p+1} + (W_p - U_1) + \sum_{j=p+1}^{i} W'_j, \quad \text{by Eq. (23).}\\ &= \sum_{j=1}^{p+1} W_j - U_1 + \sum_{j=p+2}^{i+1} W_j, \quad \text{by Eq. (24).}\\ &= \sum_{j=1}^{i+1} W_j - U_1 \geq \sum_{j=1}^{i+1} U_j - U_1 = \sum_{j=2}^{i+1} U_j = \sum_{j=1}^{i} U'_j. \qquad (26) \end{aligned}$$

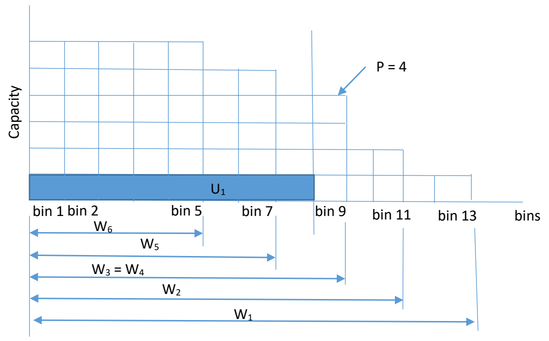


Fig. 2. An illustration of the proof of Theorem 4.

Therefore, by the induction assumption, all h remaining jobs can be selected by the bins. The theorem thus follows. □

We thus can derive the following corollary from Theorem 4.

Corollary 1. Given any subset of jobs $\{J_{j_1}, J_{j_2}, \ldots, J_{j_p}\}$ of a set of jobs $\{J_1, J_2, \ldots, J_n\}$ with $U_{j_1} \geq U_{j_2} \geq \ldots \geq U_{j_p}$, $j_l \in \{1, 2, \ldots, n\}$ and $1 \leq l \leq p$, each job J_{j_l} demands U_{j_l} ($1 \leq U_{j_l} \leq K$) bins with one computing unit per bin. Assume that there are K bins with each bin B_k having C_k computing units and $C_1 \geq C_2 \geq \ldots \geq C_K$, all the jobs in the subset is admissible by the K bins if and only if the following condition is satisfied.

$$\sum_{i=1}^{h} U_{j_i} \leq \sum_{i=1}^{h} W_i, \quad \text{for all } h \text{ with } 1 \leq h \leq p. \qquad (27)$$

Recall the admission of a job J_j leads to a profit of p_j. Let $U = \sum_{j=1}^{n} U_j$. Define $L_i = \sum_{j=1}^{i} W_j$ for all i with $1 \leq i \leq n$. Denote by $P(i, h, Y)$ the maximum profit from selecting i jobs from the first h jobs with $i \leq h \leq n$, whose total number of computing units needed is Y. Clearly $0 \leq Y \leq \sum_{j=1}^{n} U_j = U$. If there is no solution, then $P(i, h, Y) = 0$. The initiation of $P(0, h, Y)$ is 0 for any $1 \leq h \leq n$ and $0 \leq Y \leq U$. The recurrence is defined as follows.

$$P(i,h,Y) = \max \begin{cases} p_h + P(i-1, h-1, Z), \\ \quad \text{if } Y = Z + U_h \text{ and } Y \leq L_h, \\ P(i, h-1, Y), \text{ otherwise.} \end{cases} \qquad (28)$$

where p_h is the revenue collected if request r_h is admitted and $U_h = n_h + 1$ is the number of VNF instances placed for request r_h, and Z is the total number of computing units required by the first $h-1$ jobs admitted prior to the admission of job h, i.e., $Z = \sum_{i=1}^{h-1} U_{j_i}$. Note that from Corollary 1, $Y \leq L_i$ is to guarantee that the solution is valid.

Denote by $V(i, h, Y) = \{j_1, j_2, \ldots, j_i\}$ the set of indices of the selected i jobs that achieves $P(i, h, Y)$ – the maximum profit for this sub-problem. Then, $V(i,h,Y)$ is recursively defined as follows.

$$V(i,h,Y)=\begin{cases}V(i,h-1,Y) \text{ if } P(i,h,Y)=P(i,h-1,Y),\\ V(i-1,h-1,Z)\cup\{h\}.\end{cases} \tag{29}$$

The solution to the problem is $\max\{P(i,h,Y) \mid 1 \le i \le h \le n, 0 \le Y \le U\}$. Specifically, given a set $\mathcal{R}$ of requests, a set of K cloudlets with each cloudlet k accommodating C_k VNF instances, Recurrence (28) can be used to derive the maximum revenue by admitting a subset of requests in $\mathcal{R}$. An exact algorithm for the reliability-aware VNF instances provisioning problem then follows, and the detailed algorithm is given in `Algorithm` 3.

Algorithm 3 An exact algorithm for the special reliability-aware VNF instances provisioning problem

Input: K cloudlets with each cloudlet j accommodating C_j VNF instances with the same computing resource demand by different network function instances, a set of requests $\mathcal{R}$ with each request $r_j=(f_j,R_i,c_j,p_j)\in\mathcal{R}$

Output: Admit a subset of requests in $\mathcal{R}$ such that the sum of revenues of admitted requests is maximized while the reliability requirement of each admitted request is met.

```
1: for each request $r_j \in \mathcal{R}$ do
2:     Calculate the number $n_j$ of secondary VNF instances by Inequality (3);
3: end for
4: Identify a subset $\mathcal{A}$ of requests in $\mathcal{R}$ to maximize the revenue collected by solving Recurrence (28) for all $i$, $h$, and $C$, and let a subset $\mathcal{A}$ is identified such that the revenue is maximized;
5: Let $r'_1, r'_2, \ldots, r'_{|\mathcal{A}|}$ be the sequence of the ordered requests in the solution obtained in $\mathcal{A}$;
6: for $j \leftarrow 1$ to $|\mathcal{A}|$ do
7:     Perform $n_j+1$ VNF instance allocations to $n_j+1$ cloudlets, where $n_j$ VNF instances serve as secondary VNFs and one VNF instance serves as the primary VNF instance for each admitted request $r'_j \in \mathcal{A}$.
8: end for
```

The rest is to analyze the time complexity of `Algorithm` 3 by the following theorem.

Theorem 5. Given an MEC $G(V,E)$ and a set of requests $\mathcal{R}$, assume that each cloudlet k of the K cloudlets in G has a computing capacity C_k with $1\le k\le K$, there is an exact algorithm, `Algorithm` 3, for the reliability-aware VNF instances provisioning problem with different VNF instances have demanded the same amount of computing resource, which delivers an exact solution within $O(K\cdot|\mathcal{R}|^3)$ time, where K is the number of cloudlets in G.

Proof: The running time of `Algorithm` 3 is dominated by solving the recurrence in the dynamic programming and subsequently VNF instances allocations for each admitted request, both take $O(K\cdot|\mathcal{R}|^3)$ time since $0\le i\le|\mathcal{R}|$, $1\le h\le|\mathcal{R}|$. The time complexity for solving Recurrence (28) is as follows. Consider how to select i jobs from the first h jobs with $i\le h$, such that the total required computing unit is Y. Obviously, $0\le W\le\sum_{j=1}^{n}U_j\le Kn$, as each request has at most K VNF instances with each placed at one of the K cloudlets. For each fixed i, h, and W, we find a solution that has the maximum total profit. Since $1\le i\le h\le|\mathcal{R}|$, there are $O(K|\mathcal{R}|^3)$ sub-problems need to solve. □

8 Performance Evaluation

In this section, we evaluate the performance of the proposed algorithms for the reliability-aware VNF instances provisioning problem. We also investigate the impact of parameters on the performance of the proposed algorithms.

8.1 Experimental environment settings

We assume that a MEC $G=(V,E)$ consists of 100 APs, in which the number of cloudlets K is 10% of the network size, and 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 [3]. The computing capacity C_k of each cloudlet is drawn in a range from 2,000 to 4,000 MHz [19]. The network offers 20 different types of network functions, i.e., $|\mathcal{F}|=20$, where the computing resource demand c_i of a VNF instance of network function $f_i\in\mathcal{F}$ is randomly drawn between 40MHz and 400MHz [2]. The number n_j of secondary VNF instances of the network function for request r_j is randomly drawn between 0 and 4, because in the extreme case where the failure rate of each cloudlet is 10%, allocating four secondary VNF instances and one primary VNF instance to a request is able to guarantee a carrier-grade reliability of 99.999% (five nines). The running time obtained of each mentioned algorithm is based on a desktop with a 4GHz quad-core Intel i7 CPU and 16 GB RAM.

We evaluate the proposed algorithms `Algorithm` 1, `Algorithm` 2, `Algorithm` 3, and the ILP solution, which are referred to as `ALG-1`, `ALG-2`, `ALG-3`, and `ILP`, respectively. Each value in figures is the mean of the results of 30 trials.

8.2 Algorithm performance evaluation for the reliability-aware VNF instances provisioning problem

We first evaluate the proposed approximation algorithm `Algorithm` 1 for the reliability-aware VNF instances provisioning problem against a baseline algorithm `Greedy` that adopts a linear cost model. Given a request $r_j=(f_j,R_j,c_j,p_j)$ that needs (n_j+1) VNF instances, algorithm `Greedy` places its VNF instances into top-(n_j+1) cloudlets with the largest residual computing capacity. It takes $O(K)$ time to identify the top-(n_j+1) cloudlets if a linear-time selection algorithm is applied. Thus, the running of algorithm `Greedy` is $O(K\cdot|\mathcal{R}|)$, where K is the number of cloudlets and $\mathcal{R}$ is the set of requests.

Figs. 3 (a)-(b) shows the curves of the total revenues and the normalized revenues delivered by algorithms `ILP`, `ALG-1` and `Greedy`, respectively, where *the normalized revenue* refers to the average revenue of admitting a request. It can be observed from these two figures that when the number of requests is very small, algorithm `Greedy` achieves a slightly better performance than algorithm `ALG-1`. However, with more and more request arrivals over time, the total revenue delivered by algorithm `ALG-1` increases steadily while algorithm `Greedy` grows at a much slower rate. When the number of requests reaches 1,000, the total revenue of algorithm `Greedy` is approximately 70% of that of algorithm `ALG-1`. The reason behind is that algorithm `ALG-1` is more conservative: it rejects those requests with high costs by the admission control policy to alleviate overloading of

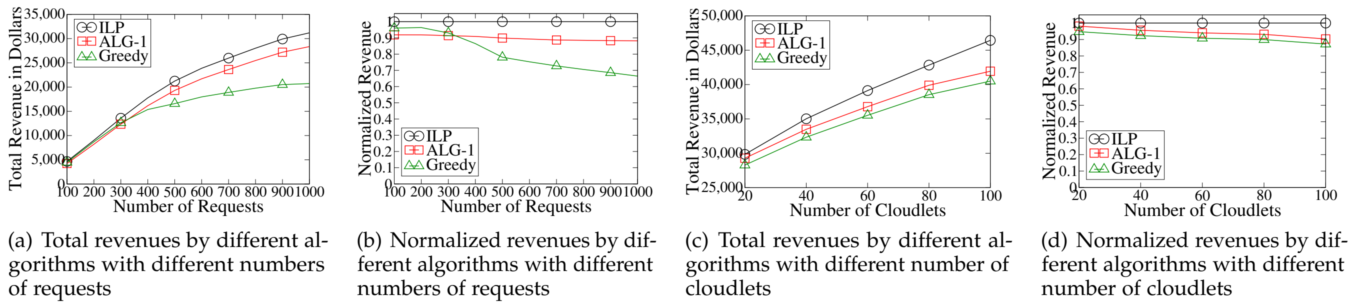
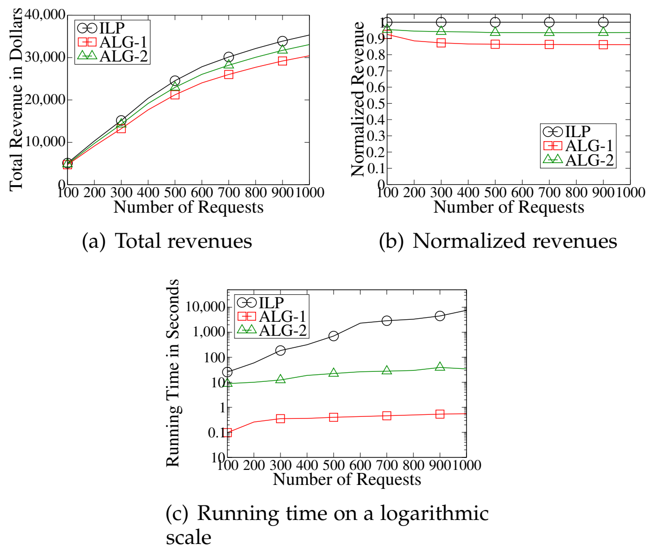

(a) Total revenues by different algorithms with different numbers of requests (b) Normalized revenues by different algorithms with different numbers of requests (c) Total revenues by different algorithms with different number of cloudlets (d) Normalized revenues by different algorithms with different number of cloudlets

Fig. 3. Performance evaluation of the approximation algorithm ALG−1 for the reliability-aware VNF instances provisioning problem

network resources, thereby achieving better performance. We also evaluate the approximation ratio of algorithm ALG−1 empirically through comparing the total revenue delivered by algorithm ALG−1 and the optimal one by the optimal algorithm ILP. It can be seen from Figs. 3 (c)-(d) that the empirical approximation ratio of algorithm ALG−1 is at most 1.17 in all cases, compared with the analytical approximation ratio of 20 according to Theorem 2 when $K = 100$. This demonstrates that the empirical performance of algorithm ALG−1 is significantly better than its analytical counterpart.

8.3 Algorithm performance evaluation for special reliability-aware VNF instances provisioning problems

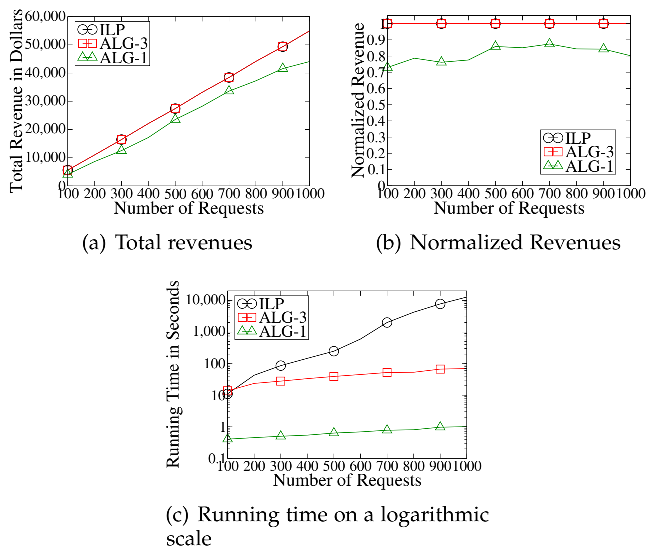

(a) Total revenues (b) Normalized revenues

(c) Running time on a logarithmic scale

Fig. 4. Performance of different algorithms for the special case of the reliability-aware VNF instances provisioning problem where each request requires exactly one secondary instance

What follows is to study the performance of the proposed approximation algorithm ALG−2 against algorithms ALG−1 and ILP for the special case of the problem where every request needs only one secondary VNF instance. It can be observed from Figs. 4 (a)-(b) that the total revenue delivered by algorithm ALG−2 is nearly close to the exact one delivered by algorithm ILP, while the solution delivered by algorithm ALG−1 is the worst. Specifically, when the number of requests is 1,000, the total revenue by algorithm ALG−2 is around 93% of the optimal solution, which is an 8% improvement over the one delivered by algorithm ALG−1. It also can be seen from Fig. 4 (c) that among the three comparison algorithms, algorithm ALG−1 runs fastest, while algorithm ILP is the slowest. This demonstrates a non-trivial tradeoff between the quality of a solution and the running time to deliver the solution.

We thirdly evaluate the performance of ALG−3 against ILP and ALG−1 for the special reliability-aware VNF instances provisioning problem where the computing demand by each network function instance is identical, by varying the number of requests from 100 to 1,000.


(a) Total revenues (b) Normalized Revenues

(c) Running time on a logarithmic scale

Fig. 5. Performance of different algorithms for the special reliability-aware VNF instances provisioning problem

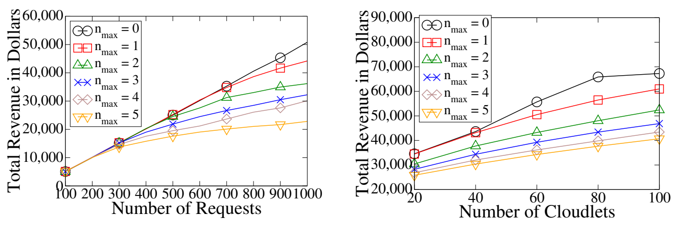
Figs. 5 (a)-(c) show the total revenue, normalized revenue, running time of the mentioned algorithms, respectively. It can be seen from Figs. 5 (a)-(b) that both algorithms ALG−3 and ILP deliver exact solutions to the problem, while the heuristic algorithm ALG−1 delivers a very good solution. The total revenue delivered by algorithm ALG−1 is no less than 80% of that by algorithms ALG−3 and ILP. Despite the optimal solutions delivered by both algorithms ALG−3 and ILP, it can be seen from Fig. 5 (c) that algorithm ILP is very time-consuming and exhibits poor scalability while algorithm ALG−3 runs significantly faster. It is observed that when there are 100 requests, the running time of algorithm ALG−3 is larger than that of algorithm ILP, due to the overhead of constructing data structures for solving the recurrence in dynamic programming. However, when the number of requests reaches 1,000, algorithm ALG−3 takes less than one minute while algorithm ILP takes more than two hours to find an optimal solution. Meanwhile, Fig. 5 (c) also indicates that the running time of algorithm `ALG-1` is only a small fraction of algorithm `ALG-3`, not to mention algorithm `ILP`.

8.4 Parameter impacts on algorithm performance

We finally study the impact of important parameters on the performance of algorithms `ALG-1`.

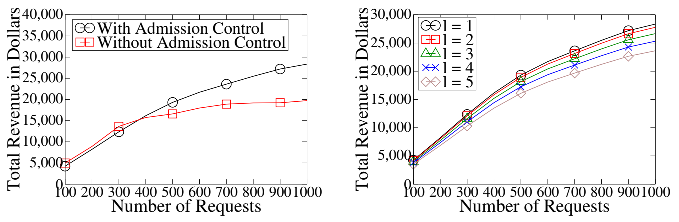
In all experiments so far we assumed that the maximum number $n_{\max}$ of secondary VNF instances of each request is set at 4. We now investigate the impact of $n_{\max}$ on the performance of algorithm `ALG-1`, by varying $n_{\max}$ while fixing the number of cloudlets at 20. Fig. 6 (a) shows the total revenue delivered by it when the number of requests grows from 100 to 1,000. Notice that when the number of requests is less than 300, the total revenues of different algorithms with different $n_{\max}$ are nearly identical, meaning the network has a relatively abundant amount of resources to accommodate different requests. However, with more requests, the larger the value of $n_{\max}$, the smaller the total revenue delivered by algorithm `ALG-1`. The reason behind this is that the amount of available resources in the network and the payments of requests do not change, yet each request demands more resources when $n_{\max}$ increases.

We also evaluate the total revenues delivered by algorithm `ALG-1` by varying the number of cloudlets K while fixing the number of requests at 1,000. The results are shown in Fig. 6 (b). When the number K of cloudlets increases, the amount of available resources increases accordingly. As a result, the larger K is, the higher the total revenue delivered by algorithm `ALG-1` is. Meanwhile, for the same K, the larger the value of $n_{\max}$, the lower the total revenue delivered by `ALG-1`, because the revenue of a request does not increase yet the resource demand of the request increases.



(a) Total revenues by varying $n_{\max}$ for different numbers of requests (b) Total revenues by varying $n_{\max}$ for different numbers of cloudlets

Fig. 6. Performance impact of parameter $n_{\max}$ on algorithm `ALG-1`



(a) Total revenues by algorithm `ALG-1` with and without the admission control policy (b) Total revenues by algorithm `ALG-1` with different $\alpha\ (= 2^l \cdot K)$

Fig. 7. Impacts of the admission control policy and the value of α on the performance of algorithm `ALG-1`

The rest is to investigate the impacts of both the admission control policy and the parameter α in Eq. (9) on the performance of algorithm `ALG-1`. Fig. 7 (a) shows the performance of algorithm `ALG-1` with and without adopting the admission control policy. It can be seen that algorithm `ALG-1` achieves a higher revenue if it does not adopt the admission control policy for the first four hundred requests. However, with more and more request arrivals, it achieves a higher revenue in the long term if the admission control policy is adopted. The rationale is that without any admission control policy, requests that consume excessive resources will be admitted if there are sufficient resources for them. Consequently, such resource allocations will heavily impact the admissions of future requests. As a result, the total revenue by algorithm `ALG-1` without the admission control policy is only two-thirds of that by itself with the admission control policy. Fig 7 (b) plots the performance curves of algorithm `ALG-1` by varying the value of α in Eq. (9) from 2^1K to 2^5K, where K is the number of cloudlets in the network. It can be seen from Fig 7 (b) that the larger the value of α, the less the total revenue delivered by `ALG-1` and vice versa. This is due to the fact that the larger the value of α, the higher the cost of using an overloaded resource will be, leading to more conservative resource usage.

9 Conclusion

In this paper, we studied reliability-aware VNF instances provisioning in MEC, by casting a novel optimization problem. We first showed that the problem is NP-hard and formulated an integer linear program solution for it. We then proposed a logarithmic-approximation algorithm for the problem. Particularly for a special case of the problem where each request needs only one secondary VNF instance, we developed a constant approximation algorithm. Moreover, we proposed an exact algorithm for another special case of the problem where resource consumptions of different VNF instances are identical. We finally evaluated the performance of the proposed algorithms through experimental simulations. Experimental results demonstrated that the proposed algorithms are promising, and the empirical results delivered by the proposed algorithms outperform their analytical counterparts as theoretical estimation usually are very conservative.

Appendix A
Proof of Lemma 1

Proof: Consider a request $r_{j'} \in \mathcal{A}$ admitted by the approximation algorithm. Then, for any cloudlet k with $1 \le k \le K$, we have

$$\begin{aligned}
& w_k(j'+1) - w_k(j') \\
=& \frac{C_k}{c_{j'+1}} \cdot (\alpha^{1-\frac{C_k(j'+1)}{C_k}} - 1) - \frac{C_k}{c_{j'}} \cdot (\alpha^{1-\frac{C_k(j')}{C_k}} - 1) \\
\le& \frac{C_k}{c_{j'}} \cdot (\alpha^{1-\frac{C_k(j'+1)}{C_k}} - 1) - \frac{C_k}{c_{j'}} \cdot (\alpha^{1-\frac{C_k(j')}{C_k}} - 1) \qquad (30) \\
=& \frac{C_k}{c_{j'}} \cdot (\alpha^{1-\frac{C_k(j'+1)}{C_k}} - \alpha^{1-\frac{C_k(j')}{C_k}}) \\
=& \frac{C_k}{c_{j'}} \cdot \alpha^{1-\frac{C_k(j')}{C_k}} (\alpha^{\frac{C_k(j')-C_k(j'+1)}{C_k}} - 1)
\end{aligned}$$

$$
\begin{aligned}
&\leq \frac{C_k}{c_{j'}} \cdot \alpha^{1-\frac{C_k(j')}{C_k}} (\alpha^{\frac{c_{j'}}{C_k}} - 1) && (31)\\
&= \frac{C_k}{c_{j'}} \cdot \alpha^{1-\frac{C_k(j')}{C_k}} (2^{\frac{c_{j'}}{C_k}\log\alpha} - 1)\\
&\leq \frac{C_k}{c_{j'}} \cdot \alpha^{1-\frac{C_k(j')}{C_k}} (c_{j'} \cdot \log\alpha / C_k) && (32)\\
&= \alpha^{1-\frac{C_k(j')}{C_k}} \cdot \log\alpha. && (33)
\end{aligned}
$$

where Inequality (30) follows because requests are sorted, Inequality (31) holds because at most $c_{j'}$ amount of computing resource is consumed at cloudlet k, and Inequality (32) holds because $2^x - 1 \leq x$ with $0 \leq x \leq 1$.

Recall that $\mathcal{K}_{j'}$ is the set of cloudlets in which the proposed approximation algorithm places a VNF instance of $r_{j'}$. We then calculate the sum of costs of cloudlets in G when admitting request $r_{j'}$. Notice that if none of the VNF instances of $r_{j'}$ is created at a cloudlet k, the cost of the cloudlet does not change after the admission of request $r_{j'}$. The difference in the cost sum of all cloudlets before and after admitting request $r_{j'}$ thus is

$$
\begin{aligned}
&\sum_{k=1}^{K}(w_k(j'+1) - w_k(j')) = \sum_{k\in\mathcal{K}_{j'}}(w_k(j'+1) - w_k(j'))\\
&\leq \sum_{k\in\mathcal{K}_{j'}}(\alpha^{1-\frac{C_k(j')}{C_k}} \cdot \log\alpha), \quad \text{by Inequality (33)}\\
&= \log\alpha \sum_{k\in\mathcal{K}_{j'}}\Big(\frac{w_k(j')}{C_k} + 1\Big)\\
&= \log\alpha \left(\sum_{k\in\mathcal{K}_{j'}}\frac{w_k(j')}{C_k} + \sum_{k\in\mathcal{K}_{j'}} 1\right)\\
&\leq \log\alpha \cdot ((p_{j'} \cdot K) + K) && (34)\\
&= \log\alpha \cdot K \cdot (p_{j'} + 1)\\
&\leq \log\alpha \cdot K \cdot (p_{j'} + p_{j'}) && (35)\\
&= 2\log\alpha \cdot K \cdot p_{j'}, && (36)
\end{aligned}
$$

where Ineq. (34) follows from the fact that $r_{j'}$ is admitted and Ineq. (10), Ineq. (35) follows because $p_{j'} \geq 1$.

The cost sum of all cloudlets after having examined last request $r_{|\mathcal{R}|}$ thus is

$$
\begin{aligned}
\sum_{k=1}^{K} w_k(|\mathcal{R}|+1) &= \sum_{j'=1}^{|\mathcal{R}|}\sum_{k=1}^{K}(w_k(j'+1) - w_k(j'))\\
&= \sum_{r_{j'}\in\mathcal{A}}\sum_{k=1}^{K}(w_k(j'+1) - w_k(j'))\\
&\leq \sum_{r_{j'}\in\mathcal{A}}(2K \cdot p_{j'} \cdot \log\alpha) \qquad (37)\\
&= 2K \cdot \log\alpha \cdot \sum_{r_{j'}\in\mathcal{A}} p_{j'},
\end{aligned}
$$

where Inequality (37) follows from Inequality (36). □

Appendix B
Proof of Lemma 2

Proof: Consider a request $r_{j'}$ that is admitted by the optimal algorithm yet rejected by the proposed approximation algorithm. Since $r_{j'}$ is admitted by the optimal algorithm, it means that the optimal algorithm is able to admit $r_{j'}$ using a set $\mathcal{K}_{j'}^{opt}$ of cloudlets. There are exactly two cases. Case 1: every cloudlet in $\mathcal{K}_{j'}^{opt}$ has sufficient resources to admit $r_{j'}$; and Case 2: at least one cloudlet in $\mathcal{K}_{j'}^{opt}$ does not have sufficient resources to meet the resource demand of $r_{j'}$.

In the following we show that Inequality (11) holds in both of the two cases.

Case 1: If every cloudlet in $\mathcal{K}_{j'}^{opt}$ has sufficient resources to admit $r_{j'}$, the proposed approximation algorithm must be able to find a set $\mathcal{K}_{j'}$ of cloudlets such that $\mathcal{K}_{j'}$ can meet the resource demand of request $r_{j'}$ and for any set including $\mathcal{K}_{j'}^{opt}$ of $n_{j'} + 1$ cloudlets. That is, $\sum_{k\in\mathcal{K}_{j'}} \psi_k(j') \leq \sum_{k\in\mathcal{K}_{j'}^{opt}} \psi_k(j')$. Since $r_{j'}$ is rejected by the proposed algorithm, the cost sum of cloudlets in $K_{j'}^{opt}$ when admitting request $r_{j'}$ is no less than the given threshold in the admission control policy (10), i.e., $K \cdot p_{j'} \leq \sum_{k\in\mathcal{K}_{j'}} \psi_k(j') \leq \sum_{k\in\mathcal{K}_{j'}^{opt}} \psi_k(j')$.

Case 2: At least one cloudlet in $\mathcal{K}_{j'}^{opt}$ does not have sufficient available resource to meet the demand of request $r_{j'}$. Thus, there must exist at least one cloudlet k' such that its residual computing capacity $C_{k'}(j')$ is less than the computing demand $c_{j'}$. Consequently, the sum of the normalized costs of cloudlets in $K_{j'}^{opt}$ is greater than $Kp_{j'}$:

$$
\begin{aligned}
&\sum_{k\in K_{j'}^{opt}} \psi_k(j') \geq \psi_{k'}(j') = \frac{\alpha^{1-\frac{C_{k'}(j')}{C_{k'}}} - 1}{c_{j'}}\\
&> \frac{\alpha^{1-\frac{c_{j'}}{C_{k'}}} - 1}{c_{j'}}, \quad \text{since } C_k(j') < c_{j'}\\
&\geq \frac{\alpha^{1-\frac{1}{\log\alpha}} - 1}{c_{j'}}, \quad \text{since } \alpha \leq 2^{C_{\min}/c_{\max}} \leq 2^{C_{k'}/c_{j'}}\\
&= \frac{\frac{\alpha}{2} - 1}{c_{j'}} \geq Kp_{j'}, \quad \text{since } \alpha \geq 2K \cdot Q_{\max} + 2 \geq c_{j'} \cdot p_{j'} + 2.
\end{aligned}
$$

□

Acknowledgments

We really appreciate the three anonymous referees and the associate editor for their expertise comments and constructive suggestions, which have helped us improve the quality and presentation of the paper greatly.

References

[1] S. Aidi, M. F. Zhani, and Yehia Elkhatib. On improving service chains survivability through efficient backup provisioning. *Proc. of International Conference on Network and Service Management (CNSM)*, IEEE, 2018.

[2] Amazon Web Services, Inc. Amazon EC2 instance configuration. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-ec2-config.html, 2018.

[3] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, Vol. 286, pp. 509–512, 1999.

[4] M. T. Beck, J. F. Botero, and K. Samelin. Resilient allocation of service function chains. *Proc. of IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, IEEE, 2017.

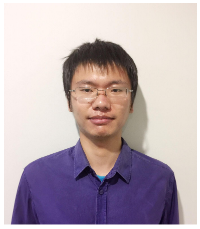
[5] J. Bailey and S. Stuart FAUCET: Deploying sdn in the enterprise. *ACM Queue*, Vol. 14, pp. 54–68, 2016.

[6] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, Sateesh Addepalli Fog computing and its role in the internet of things. *Proc. MCC workshop on Mobile cloud computing*, ACM, 2012.
[7] F. Carpio and A. Jukan Improving reliability of service function chains with combined vnf migrations and replications https://arxiv.org/abs/1711.08965, 2018.
[8] M. Casazza, P. Fouilhoux, M. Bouet, and S. Secci. Securing virtual network function placement with high availability guarantees. *Proc. of IFIP Networking*, Lecture Notes in Computer Science, 2017.
[9] A. Ceselli, M. Premoli, and S. Secci. Mobile edge cloud network design optimization. *IEEE/ACM Transactions on Networking*, Vol.25, No.3, pp.1818–1831, 2017.
[10] R. Cohen, L. Katzir, and D. Raz. An efficient approximation for the generalized assignment problem. *Information Processing Letters*, Vol. 100, pp. 162–166, Elsevier, 2006.
[11] A. V. Dastjerdi and R. Buyya. Fog computing: Helping the internet of things realize its potential *Computer*, Vol. 49, No. 8, pp. 112–116, IEEE, 2016.
[12] W. Ding, H. Yu, and S. Luo. Enhancing the reliability of services in nfv with the cost-efficient redundancy scheme. *Proc. of ICC*, IEEE, 2017.
[13] A. Engelmann and A. Jukan. A reliability study of parallelized vnf chaining. *Proc. of ICC*, IEEE, 2018.
[14] J. Fan, C. Guan, Y. Zhao, and C. Qiao. Availability-aware mapping of service function chains. *Proc. of INFOCOM'17*, IEEE, 2017.
[15] 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.
[16] R. Govindan, I. Minei, M. Kallahalla, *et al.* Evolve or die: High-availability design principles drawn from google's network infrastructure. *Proc. of SIGCOMM*, ACM, 2016.
[17] 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.
[18] S. Herker, X. An, W. Kiess, S. Beker, A. Kirstaedter. Data-center architecture impacts on virtualized network functions service chain embedding with high availability requirements. *Proc. of 2015 IEEE Globecom Workshops (GC Wkshps)*, IEEE, 2015.
[19] Hewlett-Packard Development Company. L.P. Servers for enterprise bladeSystem, rack & tower and hyperscale. http://www8.hp.com/us/en/products/servers/, 2015.
[20] M. Huang, W. Liang, Z. Xu, and S. Guo. Efficient algorithms for throughput maximization in software-defined networks with consolidated middleboxes. *IEEE Transactions on Network and Service Management*, Vol.14, No.3, pp.631–645, 2017.
[21] R. Jain and S. Paul. Network virtualization and software defined networking for cloud computing: a survey. *IEEE Communications Magazine*, Vol. 51, No. 11, pp. 24–31, IEEE, 2013.
[22] M. Jia, W. Liang, and Z. Xu. QoS-aware task offloading in distributed cloudlets with virtual network function services. *Proc of 20th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems* (MSWiM), ACM, pp.109–116, 2017.
[23] 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, pp. 1767–1770, 2017.
[24] 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 IEEE Global Communications Conference*, IEEE, 2017.
[25] J. Li, W. Liang, M. Huang, and X. Jia. Providing reliability-aware virtualized network function services for mobile edge computing. To appear in *Proc. of 39th Int'l Conf. on Distributed Computing Systems* (ICDCS'19), July, IEEE, 2019.
[26] P. G. Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere. Edge-centric computing: vision and challenges. *ACM SIGCOMM Computer Communication Review*, Vol. 45, No. 5, pp. 37–42, ACM, 2015.
[27] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba. Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys & Tutorials*, Vol. 18, No. 1, pp. 236–262, IEEE, 2016.
[28] G. Moualla, T. Turletti, and D. Saucez. An availability-aware sfc placement algorithm for fat-tree data centers. *Proc. of IEEE International Conference on Cloud Networking*, IEEE, 2018.
[29] L. Qu, C. Assi, K. Shaban, and M. J. Khabbaz. A reliability-aware network service chain provisioning with delay guarantees in nfv-enabled enterprise datacenter networks. *IEEE Transactions on Network and Service Management*, Vol 14, pp. 554–568, 2017.
[30] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing*, Vol. 8, pp. 1536–1268, 2009.
[31] A. Singh, J. Ong, A. Agarwal, et al. Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network. *Communications of the ACM*, Vol. 59, pp. 88–97, 2016.
[32] T. Taleb, S. Dutta, A. Ksentini, M. Iqbal, and H. Flinck. Mobile edge computing potential in making cities smarter. *IEEE Communications Magazine*, Vol. 55, No. 3, pp. 38–43, IEEE, 2017.
[33] Z. Xu, W. Liang, M. Jia, M. Huang, and G. Mao. Task offloading with network function services in a mobile edge-cloud network. To appear in *IEEE Transactions on Mobile Computing*, Vol.99, Acceptance date: October 19, DOI: 10.1109/TMC.2018.2877623, 2018.
[34] S. Yi, C. Li, and Q. Li A survey of fog computing: concepts, applications and issues. *Proc. of Workshop on Mobile Big Data*, ACM, 2015.

Meitian Huang received the BSc degree with the first class Honours in Computer Science at the Australian National University in 2015. He currently is studying for his PhD degree in the Research School of Computer Science at the Australian National University. His research interests include software-defined networking, algorithm design and analysis, and cloud computing.

Weifa Liang (M'99–SM'01) received the PhD degree from the Australian National University in 1998, the ME degree from the University of Science and Technology of China in 1989, and the BSc degree from Wuhan University, China in 1984, all in Computer Science. He is currently a full Professor in the Research School of Computer Science at the Australian National University. His research interests include design and analysis of energy efficient routing protocols for wireless ad hoc and sensor networks, mobile edge computing, cloud computing, Network Function Virtualization and Software-Defined Networking, design and analysis of parallel and distributed algorithms, approximation algorithms, combinatorial optimization, and graph theory. He is a senior member of the IEEE.

Xiaojun Shen (SM'02) received the B.S. degree in numerical analysis from Tsinghua University, Beijing, China, in 1968, the M.S. degree in computer science from the Nanjing University of Science and Technology, China, in 1982, and the Ph.D. degree in computer science from the University of Illinois at Urbana-Champaign in 1989. He is a full Professor in the School of Computing and Engineering, University of Missouri-Kansas City. His current research focuses on fundamental scheduling problems in wired and wireless computer networks.



Yu Ma received his BSc degree with the first class Honours in Computer Science at the Australian National University in 2015. He is currently a PhD candidate in the Research School of Computer Science at the Australian National University. His research interests include Software Defined Networking, Internet of Things (IoT), and Social Networking.

Haibin Kan (M'13) received the Ph.D. degree from Fudan University, Shanghai, China, 1999. After receiving the Ph.D. degree, he became a faculty of Fudan University. From June 2002 to January 2006, he was with the Japan Advanced Institute of Science and Technology as an Assistant Professor. He went back to Fudan University in February 2006, where he is currently a full Professor. His research topics include coding theory, complexity of computing, and information security.