2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)

Providing Reliability-Aware Virtualized Network Function Services for Mobile Edge Computing

Jing Li[†], Weifa Liang[†], Meitian Huang[†], and Xiaohua Jia[‡]
[†]Research School of Computer Science, The Australian National University, Canberra, Australia
[‡]Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong
Emails: u6013763@anu.edu.au, wliang@cs.anu.edu.au, meitian.huang@anu.edu.au, csjia@cityu.edu.hk

***Abstract*—Mobile Edge Computing (MEC) has emerged as a promising paradigm to address the conflict between increasing computing-intensive applications and resource-constrained mobile Internet-of-Thing (IoT) devices with portable size and storage. In MEC environments, Virtualized Network Functions (VNFs) are deployed for provisioning network services to users to reduce the service cost on top of dedicated hardware infrastructures. However, VNFs may suffer from failures and malfunctions while network service providers have to guarantee continuously reliable services to their users to meet ever-growing service demands of the users. In this paper, we focus on reliable VNF service provisioning in MECs, by provisioning primary and backup VNF instances in order to meet the reliability requirements of mobile users. We first formulate a novel VNF service reliability problem with the aim to maximize the revenue collected by admitting as many as user requests while meeting individual user service reliability requirements. We then develop two efficient on-line scheduling algorithms for the problem under two different backup schemes: on-site (local) and off-site (remote) schemes, by adopting the primal and dual updating technique. Particularly for the on-site scheme, the proposed on-line algorithm achieves a provable competitive ratio with bounded moderate resource violations. We finally evaluate the proposed algorithms through experimental simulations. The experimental results demonstrate that the proposed algorithms are promising, compared with existing baseline algorithms.**

I. Introduction

The recent advancement of Internet of the Things (IoTs) has further flourished new applications of mobile IoT devices [1]. However, due to their portable sizes and limited computing capacities, mobile IoT devices are unable to meet the demands of both computing and storage resources of these applications [2]. To meet ever-growing mobile users' resource demands, MEC as a promising paradigm has been introduced, which extends cloud computing services to the edge of mobile networks by utilizing cloudlets (servers, or clusters of servers) co-located with Access Points (APs) within the proximity of mobile users. In addition, Network Functions Virtualization (NFV) is a promising technique that contributes to applications of MEC [10], and the instances of NFVs are employed to replace dedicated hardware equipment due to not only the cost reduction but also the advantage of adjusting services with agility to cope with rapid-changing user demands without the hassle of deployment and maintenance of physical infrastructures [3], where each network function is implemented in a virtual machine as a piece of software that is referred to as an instance of the Virtualized Network Function (VNF). Meeting user service reliability requirements is critical to any network service provider [4]. Reliability of a network is defined as the ability of the network to provide stable services that ensure an agreed level of operational performance facing the risk of failures of underlying network components [5]. There are several prevention and recovering mechanisms to deal with such failures. In reality, a mobile user not only requests specific VNF services but also with a certain reliability requirement, and a common and practical approach to meet the reliability requirement is utilizing redundancy [6]. In this paper, VNF replicas are deployed as backups to meet the requested reliability requirements of mobile users. This poses a challenge to maximize the revenue of network service providers by admitting as many as user requests through developing efficient on-line algorithms for their request admissions.

The risk of a single VNF instance failure could be mitigated by deploying the other backup instances of the VNF in the same cloudlet [7]. We refer to this recovering scheme as *the on-site scheme* that utilizes local redundancy to meet the required reliability, which guarantees low switching latency from primary VNF instances to their backup VNF instances in the same cloudlet. However, the VNF reliability under the on-site scheme is restricted by the reliability of the cloudlet at which the VNF instances located, because all VNF instances in a cloudlet will not be able to function when the cloudlet fails. Instead, another recovering scheme - *the off-site scheme*, is proposed, which can mitigate this restriction. The off-site scheme instantiates backup VNF instances at cloudlets that are physically separated from the cloudlet hosting the primary VNF instances [8]. Although the off-site scheme can improve VNF reliabilities, it suffers drawbacks. Since it utilizes geographic redundancy, the recovery time will be slightly longer, and will incur extra costs of network traffic between the cloudlets hosting primary and backup VNF instances [9].

The novelty of this paper is that we study the *VNF service reliability problem* for IoT applications under both on-site and off-site schemes in MEC environments. To the best of our knowledge, we are the first to consider the provisioning of reliable VNF services in MEC to meet both resource demands and service requirements of different users so that the revenue collected from the admitted requests is maximized, by proposing efficient on-line algorithms.

The main contributions of this paper are as follows. We first formulate a VNF service reliability problem with the

2575-8411/19/$31.00 ©2019 IEEE
DOI 10.1109/ICDCS.2019.00077

aim to maximize the revenue by admitting as many as user requests while meeting their individual reliability requirements under both on-site and off-site schemes. We then propose an on-line algorithm for the problem under the on-site scheme, which achieves a provable competitive ratio with moderate resource capacity violations. We also develop a heuristic algorithm for the problem under the off-site scheme by adopting the primal-dual dynamic updating technique. We finally evaluate the performance of the proposed algorithms through experiments. The experimental results demonstrate that the proposed algorithms are promising, and outperform baseline algorithms.

The remainder of this paper is organized as follows. Section II reviews the related work. Section III defines the system model and formulates the problem under both on-site and off-site schemes. Sections IV and V propose on-line algorithms for the problem under both on-site and off-site schemes. Section VI evaluates the performance of the proposed algorithms by experimental simulation, and Section VII concludes the paper.

II. Related Work

Several existing methods on the improvement of VNF instance reliability are summarized in [9]. They include state management, VNF migration, and rollback recovery. The authors in [11] addressed the problem of optimizing the allocation of VNF backups. They proposed a survivability architecture and adopted the maximum matching method. However, they only considered one backup VNF instance for each primary VNF instance, and did not consider multiple backup VNF instances to meet reliability requirements. Especially, infrastructures, where VNF instances run, could also suffer from failures and malfunctions. Therefore, the awareness of infrastructure reliability poses challenges on reliability provisioning for VNF instance deployments. [15] proposed a coordinated protection mechanism that employs both network-layer protection and VNF replicas. They focused on minimizing the total computing resource consumption. The authors in [7] studied the problem of how to enhance the reliability of Service Function Chains (SFCs) more efficiently, and proposed the Cost-aware Importance Measure (CIM) in selecting the qualified backups. [16] considered reliabilities of both VNFs and dedicated infrastructures, and discussed the resource cost saving by sharing the resource among all backup VNF instances.[12] introduced an on-site pooling mechanism which improves the resource utilization and thus reduces resource consumption. [13] studied the VNF placement problem under the off-site scheme to provision reliable Service Chains (SCs). Both on-site or off-site schemes are practical strategies to guarantee the safeness of data and robustness of the provided services. In this paper, we study the VNF service reliability problem for IoT applications in MEC environments, by jointly considering reliabilities of both VNF instances and cloudlets that accommodate the VNF instances, with the aim of maximizing the revenue of the service provider. In addition, we admit user requests while meeting their individual reliability requirements under both on-site and off-site schemes.

III. Preliminaries

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

A. Network System Model

We consider a Mobile Edge Computing (MEC) network consisting of APs, cloudlets, and links that connect APs. Each cloudlet is co-located with an AP while an AP may or may not be co-located with a cloudlet. The MEC network is represented by an undirected graph $G = (V, E)$ where V is the set of APs, cloudlets are co-located with APs, and E is the set of links between APs. Mobile users issue their requests with reliability requirements through their nearby APs.

Let $F = \{f_1, f_2, \ldots, f_n\}$ be a set of different types of Virtualized Network Functions (VNFs), e.g., Virtualized Load Balancers, Firewalls, and Intrusion-Detection Systems. We assume that different VNFs need different amounts of computing resource for their implementations. Without loss of generality, we assume that the required amount of computing resource for a specific type of VNF, $f_i \in F$, is measured by computing units, denoted by $c(f_i)$. Furthermore, different VNFs have different reliabilities, denoted by $r(f_i)$ as the reliability of f_i with $0 < r(f_i) < 1$.

Let $C = \{c_1, c_2, \ldots, c_m\}$ be the set of cloudlets in G with $m \leq |V|$. Each cloudlet has a limited amount of computing resource. We assume that for each cloudlet $c_i \in C$, its computing capacity is cap_i. For a specific type of VNF, $f_i \in F$, the cloudlets can implement f_i as a piece of the software component in a virtual machine with the computing resource demanded, i.e., $c(f_i)$. In addition, different cloudlets have different reliabilities, and each cloudlet $c_i \in C$ is associated with a reliability $r(c_i)$, with $0 < r(c_i) < 1$. When a cloudlet c_i fails, all VNF instances in c_i will not be able to offer services to mobile users anymore and thus become unavailable.

B. User request scheduling

We consider that the MEC network is in a discrete-time fashion and assume that the given monitoring period is slotted into equal time slots, each of which is a time unit. And the set of time slots is denoted by $\mathbb{T} = \{1, 2, \ldots, T\}$. Consequently, the arrival time of a request can be represented by a time slot label, while the processing duration of the request in MEC is expressed by the number of time slots needed.

We assume that each mobile user requests only one VNF instance service per request, and the request ρ_i of the user is defined by a tuple $(f_i, R_i, a_i, d_i, pay_i)$, where $f_i \in F$ is the requested VNF instance, R_i is the reliability requirement of request ρ_i with $0 < R_i < 1$, $a_i \in \mathbb{T}$ is the arrival time slot of the request, d_i is the requested duration and pay_i is the payment. For simplicity, we assume that d_i is a positive integer and pay_i is the payment received for the implementation of request ρ_i.

We assume that a request arrives at the beginning of each time slot, and only consider the set of requests that are ending at the specified time horizon. Denote by $\mathbb{R}$ the set of all requests within the given time horizon T, i.e., for a request ρ_i, we

consider it only in the event that $a_i + d_i - 1 \in \mathbb{T}$. For a better presentation of both the arrival time and the requested execution duration, we introduce V_i, which is a binary vector with the length of T, and generated by both a_i and d_i. In a certain time slot t, $V_i[t] = 1$ represents that the requested execution duration of request covers time slot t, and $V_i[t] = 0$ otherwise, e.g., in the case $T = 3, a_i = 1, d_i = 2$, then we let $V_i = [1, 1, 0]$.

Suppose that the admission scheduling of requests is on-line, and there is a hypervisor to deal with incoming requests, where the requests arrive one by one without the knowledge of future request arrivals. If the hypervisor accepts request ρ_i, it will allocate the resource from a cloudlet to accommodate the primary and backup VNF instances for the request and find a routing path for the data traffic of the request.

Let $X_i \in \{0, 1\}$ denote whether request ρ_i is admitted. Namely, $X_i = 1$ if the request is admitted; and $X_i = 0$ otherwise. If request ρ_i is admitted, then the probability of that at least one of its primary and backup VNF instances of f_i is available is no less than its reliability requirement R_i must be guaranteed. We denote this event as A_i, i.e.,

$$P(A_i) \geq R_i, \quad \forall \rho_i \in \mathbb{R}. \tag{1}$$

Let $Y_{ij} \in \{0, 1\}$ denote whether or not a primary or backup VNF instance of request ρ_i is placed in cloudlet $c_j \in C$, that is, $Y_{ij} = 1$ if a primary or backup VNF instance of ρ_i is placed in cloudlet c_j, and $Y_{ij} = 0$ otherwise. Associated with each request ρ_i, we assume that its payment and reliability requirement are given too.

C. Problem definitions

In this paper, we deal with the VNF service reliability problem under both the on-site and off-site schemes, respectively. We first consider the offline version of the problem, which will be used as the benchmark for the performance evaluation of the proposed on-line algorithms for the on-line version of the problem under two different schemes defined as follows.

1) The VNF service reliability problem under the on-site scheme: Under this scheme, all primary and backup VNF instances of each request are hosted by a single cloudlet. Let N_{ij} be the minimum number of primary and backup VNF instances needed to be placed in cloudlet $c_j \in C$ for request ρ_i in order to meet its reliability requirement R_i. Then,

$$P(A_i) = r(c_j) * (1 - (1 - r(f_i))^{N_{ij}}) \geq R_i, \forall \rho_i \in \mathbb{R}, \tag{2}$$

where $(1-r(f_i))^{N_{ij}}$ is the failure probability of all primary and backup VNF instances of request ρ_i, while $1-(1-r(f_i))^{N_{ij}}$ is the probability of at least one VNF instance survival. Because all VNF instances of ρ_i are in a single cloudlet, we have $P(A_i) = r(c_j) * (1 - (1 - r(f_i))^{N_{ij}})$.

If $r(c_j) > R_i$, N_{ij} can be calculated as follows.

$$N_{ij} = \lceil \log_{1-r(f_i)}(1 - \frac{R_i}{r(c_j)}) \rceil. \tag{3}$$

Otherwise ($r(c_j) \leq R_i$), the reliability requirement of ρ_i cannot be met, and the VNF instances of ρ_j should not be placed in cloudlet c_j. Notice that both $r(c_j)$ and R_i are given constants, N_{ij} thus is constant. For the sake of simplicity, in the rest of discussions, we assume that $r(c_j) > R_i$, $\forall c_j \in C, \forall \rho_i \in \mathbb{R}$, under the on-site scheme.

To meet the capacity constraint on each cloudlet, we have

$$\sum_{\rho_i \in \mathbb{R}} V_i[t] * N_{ij} * c(f_i) * Y_{ij} \leq cap_j, \qquad \forall t \in \mathbb{T}, \forall c_j \in C. \tag{4}$$

Constraint (4) ensures that in each time slot, the computing capacity of each cloudlet will not be violated.

To ensure only one cloudlet is chosen to accommodate all VNF instances of request ρ_i, we have

$$\sum_{c_j \in C} Y_{ij} = X_i, \qquad \forall \rho_i \in \mathbb{R}. \tag{5}$$

The *VNF service reliability problem under the on-site scheme* can be formulated as an Integer Linear Programming (ILP) with the optimization objective to

$$maximize \sum_{\rho_i \in \mathbb{R}} X_i * pay_i, \tag{6}$$

subject to

$$\begin{aligned} &(4), \quad (5), \\ &X_i \in \{0, 1\}, \qquad \forall \rho_i \in \mathbb{R}, &(7)\\ &Y_{ij} \in \{0, 1\}, \qquad \forall \rho_i \in \mathbb{R}, \forall c_j \in C. &(8) \end{aligned}$$

2) The VNF service reliability problem under the off-site scheme: In this scheme, the VNF instances of a request can be placed at different cloudlets. As one cloudlet $c \in C$ fails, all VNF instances in cloudlet c will fail. Therefore, we assume that only one VNF instance of the request will be placed in each cloudlet under this off-site scheme.

To admit a request ρ_i among all cloudlets $c_j \in C$, we need to identify which of the cloudlets will accommodate the primary and backup VNF instances of ρ_i while the capacity constraints on identified cloudlets are met.

$$\sum_{\rho_i \in \mathbb{R}} V_i[t] * c(f_i) * Y_{ij} \leq cap_j, \qquad \forall t \in \mathbb{T}, \forall c_j \in C. \tag{9}$$

Constraint (9) ensures that the computing capacity of each cloudlet at each time slot will not be violated.

The reliability constraint on each admitted request ρ_i is as follows.

$$P(A_i) = 1 - \prod_{c_j \in C} (1 - r(f_i) * r(c_j) * Y_{ij}) \geq R_i. \tag{10}$$

If $Y_{ij} = 1$, the failure probability of a VNF instance f_i at c_j is $1-r(f_i)*r(c_j)*Y_{ij} = 1-r(f_i)*r(c_j)$. Otherwise ($Y_{ij} = 0$), $1-r(f_i)*r(c_j)*Y_{ij} = 1$. Thus, $\prod_{c_j \in C}(1-r(f_i)*r(c_j)*Y_{ij})$ is the probability that both primary and backup VNF instances of ρ_i fail, while $1 - \prod_{c_j \in C}(1 - r(f_i) * r(c_j) * Y_{ij})$ is the success reliability of ρ_i by a scheduling. However, if ρ_i is rejected, for each cloudlet $c_j \in C$, Y_{ij} should be 0, i.e., no VNF instance will be placed in any cloudlet c_j and Inequality

(10) does not hold. We now modify Inequality (10) through a transformation, using the following technique, $\forall \rho_i \in \mathbb{R}$,

$$R_i * X_i \leq 1 - \prod_{c_j \in C} (1 - r(f_i) * r(c_j) * Y_{ij}) \leq X_i. \quad (11)$$

Lemma 1. *Inequality* (11) *meets the reliability requirement of request* ρ_i *if the request is admitted.*

Proof: If $X_i = 1$, request ρ_i is admitted, Inequality (11) is shown as follows.

$$R_i \leq 1 - \prod_{c_j \in C} (1 - r(f_i) * r(c_j) * Y_{ij}) \leq 1. \quad (12)$$

As $0 < 1 - r(f_i) * r(c_j) * Y_{ij} \leq 1$, it can be seen that Inequality (12) and Inequality (10) are equivalent, satisfying the reliability requirement of request ρ_i.

Otherwise ($X_i = 0$), request ρ_i will be rejected, and

$$1 \leq \prod_{c_j \in C} (1 - r(f_i) * r(c_j) * Y_{ij}) \leq 1. \quad (13)$$

It can be seen that we ensure that $\prod_{c_j \in C}(1 - r(f_i) * r(c_j) * Y_{ij}) = 1$, since $0 < r(f_i) < 1$ and $0 < r(c_j) < 1$, we have $Y_{ij} = 0$ for each cloudlet $c_j \in C$, i.e., if request ρ_i is rejected, none of its VNF instances will be placed in any cloudlet. The lemma thus follows. ∎

The *VNF service reliability problem under the off-site scheme* can be formulated as an Integer Non-linear Programming (INP) problem as follows.

The optimization objective is to

$$maximize \sum_{\rho_i \in \mathbb{R}} X_i * pay_i, \quad (14)$$

subject to

$$\sum_{\rho_i \in \mathbb{R}} V_i[t] * c(f_i) * Y_{ij} \leq cap_j, \qquad \forall t \in \mathbb{T}, \forall c_j \in C, \quad (15)$$

$$1 - \prod_{c_j \in C} (1 - r(f_i) * r(c_j) * Y_{ij}) \geq R_i * X_i, \ \forall \rho_i \in \mathbb{R}, \quad (16)$$

$$1 - \prod_{c_j \in C} (1 - r(f_i) * r(c_j) * Y_{ij}) \leq X_i, \qquad \forall \rho_i \in \mathbb{R}, \quad (17)$$

$$X_i \in \{0, 1\}, \qquad \forall \rho_i \in \mathbb{R}, \quad (18)$$

$$Y_{ij} \in \{0, 1\}, \qquad \forall \rho_i \in \mathbb{R}, \forall c_j \in C, \quad (19)$$

where Inequality (15) is the cloudlet capacity constraint, Inequalities (16) and (17) are the request reliability requirements.

D. The competitive ratio of an on-line algorithm

When considering an on-line maximization problem $\mathbb{P}$, we say an on-line algorithm with competitive ratio α for the problem if the solution delivered by the on-line algorithm is no less than $OPT(\mathbb{P})_{off}/\alpha$, where $OPT(\mathbb{P})_{off}$ is the optimal solution of the offline version of problem $\mathbb{P}$ with $\alpha > 1$.

IV. Algorithm for the VNF service reliability problem under the on-site scheme

In this section, we deal with the VNF service reliability problem under the on-site scheme, and denote by this problem **P1**. The general strategy for **P1** is as follows. We first solve the Linear Programming (LP) relaxation of **P1**, and denote by this relaxation **P2**, and denote by **P3** the dual of **P2**. A feasible solution to **P3** will return a feasible solution to **P1** ultimately.

To adopt the primal and dual dynamic updating technique [17], we perform the LP relaxation on **P1** and we have

$$maximize \sum_{\rho_i \in \mathbb{R}} X_i * pay_i, \quad (20)$$

subject to

(5), (9),

$$X_i \leq 1, \qquad \forall \rho_i \in \mathbb{R}, \quad (21)$$

$$X_i \geq 0, Y_{ij} \geq 0, \qquad \forall \rho_i \in \mathbb{R}, \forall c_j \in C. \quad (22)$$

Lemma 2. *Any feasible solution for* **P1** *is a feasible solution for* **P2**. *Let* OPT_1 *and* OPT_2 *be the optimal solutions of* **P1** *and* **P2**, *then* $OPT_1 \leq OPT_2$.

As **P3** is the dual of **P2**, to solve **P2**, we now solve **P3**, by adopting the primal-dual dynamic updating technique [17] as follows.

$$minimize \sum_{t \in \mathbb{T}, c_j \in C} cap_j * \lambda_{tj} + \sum_{\rho_i \in \mathbb{R}} \delta_i, \quad (23)$$

subject to

$$\beta_i + \delta_i - pay_i \geq 0, \qquad \forall \rho_i \in \mathbb{R}, \quad (24)$$

$$\sum_{t \in \mathbb{T}} V_i[t] * N_{ij} * c(f_i) * \lambda_{tj} - \beta_i \geq 0, \ \forall \rho_i \in \mathbb{R}, \forall c_j \in C, \quad (25)$$

$$\lambda_{tj} \geq 0, \ \beta_i \geq 0, \ \delta_i \geq 0, \ \forall \rho_i \in \mathbb{R}, \forall c_j \in C, \forall t \in \mathbb{T}, \quad (26)$$

where λ_{tj}, β_i, and δ_i are dual variables, corresponding to constraints (9), (5) and (21) respectively in **P2**.

We then rewrite (24) and (25) as follows.

$$\beta_i \geq pay_i - \delta_i, \qquad \forall \rho_i \in \mathbb{R}, \quad (27)$$

$$\beta_i \leq \sum_{t \in \mathbb{T}} V_i[t] * N_{ij} * c(f_i) * \lambda_{tj}, \ \forall \rho_i \in \mathbb{R}, \forall c_j \in C. \quad (28)$$

From Inequality (28), we have

$$\beta_i \leq \min_{c_j \in C} \sum_{t \in \mathbb{T}} V_i[t] * N_{ij} * c(f_i) * \lambda_{tj}, \quad \forall \rho_i \in \mathbb{R}. \quad (29)$$

Combining Inequalities (27) and (29), we have

$$pay_i - \delta_i \leq \min_{c_j \in C} \sum_{t \in \mathbb{T}} V_i[t] * N_{ij} * c(f_i) * \lambda_{tj}, \quad \forall \rho_i \in \mathbb{R}. \quad (30)$$

Lemma 3. *If Inequality* (30) *holds, there always exists a feasible* β_i *to satisfy constraints* (25) *and* (26).

Proof: Because $V_i[t] \geq 0, \ N_{ij} > 0, \ c(f_i) > 0, \ \lambda_{tj} \geq 0, \ \forall c_j \in C, \ \forall t \in \mathbb{T}$, it can be seen that $\min_{c_j \in C}\{\sum_{t \in \mathbb{T}} V_i[t] *$

$N_{ij} * c(f_i) * \lambda_{tj}\} \geq 0$. Since $\beta_i \geq 0$, if Inequality (30) holds, there exists a β_i such that Inequality (31) holds.

$$pay_i - \delta_i \leq \beta_i \leq \min_{c_j \in C} \sum_{t \in \mathbb{T}} V_i[t] * N_{ij} * c(f_i) * \lambda_{tj}. \quad (31)$$

Then, Inequalities (27) and (28) hold. Constraints (25) and (26) hold. The lemma then follows. ■

Following Lemma 26, we now eliminate dual variable β_i, and then only consider dual variables δ_i and λ_{tj} to guarantee Inequality (30) to be satisfied. We perform a transformation on Inequality (30) as follows, $\forall \rho_i \in \mathbb{R}$,

$$\delta_i \geq pay_i - \min_{c_j \in C} \sum_{t \in \mathbb{T}} V_i[t] * N_{ij} * c(f_i) * \lambda_{tj}. \quad (32)$$

The basic idea of the primal-dual dynamic updating technique is to update the variables in both primal and dual problems simultaneously such that they can form a primal-dual pair. For request ρ_i, the set of execution time slots is denoted as $\mathbb{T}'_i \subset \mathbb{T}$, and its duration is $d_i = |\mathbb{T}'_i|$. Dual variables λ_{tj} and β_i are 0 initially. Upon the arrival of request ρ_i, the dual variables are required to be set properly to satisfy Inequality (32). To this end, we first calculate N_{ij} for each cloudlet $c_j \in C$. We then calculate the value of $\sum_{t \in \mathbb{T}} V_i[t] * N_{ij} * c(f_i) * \lambda_{tj}$ associated with each cloudlet $c_j \in C$. We finally identify the cloudlet c_j with $\min_{c_j \in C}\{\sum_{t \in \mathbb{T}} V_i[t] * N_{ij} * c(f_i) * \lambda_{tj}\}$. If $pay_i - \min_{c_j \in C}\{\sum_{t \in \mathbb{T}} V_i[t] * N_{ij} * c(f_i) * \lambda_{tj}\} \leq 0$, request ρ_i will be rejected. Otherwise, it will be admitted and all its VNFs will be accommodated in cloudlet c_j, and the value of δ_i is updated as follows.

$$\delta_i := pay_i - \min_{c_j \in C}\{\sum_{t \in \mathbb{T}} V_i[t] * N_{ij} * c(f_i) * \lambda_{tj}\}. \quad (33)$$

Then, λ_{tj} associated with the selected cloudlet $c_j \in C_i$ and $t \in \mathbb{T}'_i$ is updated as follows.

$$\lambda_{tj} := \lambda_{tj} * (1 + \frac{N_{ij} * c(f_i)}{cap_j}) + \frac{N_{ij} * c(f_i) * pay_i}{d_i * cap_j}. \quad (34)$$

The proposed on-line algorithm for the VNF service reliability problem under the on-site scheme is given in `Algorithm` 1.

Let $a_{ij} = N_{ij} * c(f_i), \forall \rho_i \in \mathbb{R}, \forall c_j \in C$, $a_{max} = \max_{\rho_i \in \mathbb{R}, c_j \in C}\{a_{ij}\}$, and $a_{min} = \min_{\rho_i \in \mathbb{R}, c_j \in C}\{a_{ij}\}$.

Lemma 4. *Let P and D be the values of the solutions delivered by the proposed algorithm of* **P2** *and* **P3***, respectively, Then $(1 + a_{max}) * P \geq D$, where $a_{max} = \max_{\rho_i \in \mathbb{R}, c_j \in C}\{N_{ij} * c(f_i)\}$.*

Proof: It can be seen that $P = D = 0$ initially, thus, we prove the claim by showing that when request ρ_i arrives, $(1 + a_{max}) * \Delta P \geq \Delta D$, where ΔP and ΔD are the value differences of objective functions before and after the request arrives.

If request ρ_i is rejected, $\Delta P = \Delta D = 0$, and $(1 + a_{max}) * \Delta P \geq \Delta D$. Otherwise, $\Delta P = pay_i$ and $\Delta D = \sum_{t \in \mathbb{T}'_i} cap_j * \Delta\lambda_{tj} + \delta_i$, where $\Delta\lambda_{tj}$ is the difference before and after the update to λ_{tj}, which is associated with the selected cloudlet c_j. By the update function (34) of λ_{tj}, we have

$$\begin{aligned}
\Delta D &= \sum_{t \in \mathbb{T}'_i} cap_j * \Delta\lambda_{tj} + \delta_i \\
&= \sum_{t \in \mathbb{T}'_i} cap_j * (\frac{N_{ij} * c(f_i)}{cap_j} * \lambda_{tj} + \frac{N_{ij} * c(f_i) * pay_i}{d_i * cap_j}) + \delta_i \\
&= \sum_{t \in \mathbb{T}'_i} (N_{ij} * c(f_i) * \lambda_{tj}) + \sum_{t \in \mathbb{T}'_i} (\frac{N_{ij} * c(f_i) * pay_i}{d_i}) + \delta_i \\
&= \sum_{t \in \mathbb{T}} V_i[t] * N_{ij} * c(f_i) * \lambda_{tj} + \delta_i + N_{ij} * c(f_i) * pay_i \\
&= pay_i + N_{ij} * c(f_i) * pay_i \quad (35) \\
&= (1 + N_{ij} * c(f_i)) * pay_i \\
&\leq (1 + a_{max}) * pay_i \\
&= (1 + a_{max}) * \Delta P. \quad (36)
\end{aligned}$$

Notice that Inequality (35) holds, because from the update function (33), λ_{tj} is the value derived by identifying a cloudlet c_j with $\min \sum_{t \in \mathbb{T}} V_i[t] * N_{ij} * c(f_i) * \lambda_{tj}$, we have $pay_i = \sum_{t \in \mathbb{T}} V_i[t] * N_{ij} * c(f_i) * \lambda_{tj} + \delta_i$. Hence, the lemma follows. ■

Algorithm 1 On-line Scheduling Algorithm for the VNF Service Reliability Problem under the on-site Scheme

Input: An MEC network $G = (V, E)$ and incoming requests.
Output: An on-line scheduling of incoming requests.
Initialize X_i, Y_{ij}, λ_{tj}, β_i =0, $\forall \rho_i \in \mathbb{R}$, $\forall c_j \in C$, $\forall t \in \mathbb{T}$;
while Upon arrival of the request ρ_i **do**
  for cloudlet $c_j \in C$ **do**
    Calculate N_{ij} and $\sum_{t \in \mathbb{T}} V_i[t] * N_{ij} * c(f_i) * \lambda_{tj}$ for c_j;
  end for;
  select cloudlet $c_{j'}$ such that $\sum_{t \in \mathbb{T}} V_i[t] * N_{ij'} * c(f_i) * \lambda_{tj'} = \min_{c_j \in C}\{\sum_{t \in \mathbb{T}} V_i[t] * N_{ij} * c(f_i) * \lambda_{tj}\}$;
  if $pay_i - \sum_{t \in \mathbb{T}} V_i[t] * N_{ij'} * c(f_i) * \lambda_{tj'} > 0$ **then**
    Admit request ρ_i, update $X_i \leftarrow 1$ and $Y_{ij'} \leftarrow 1$;
    Update $\delta_i \leftarrow pay_i - \sum_{t \in \mathbb{T}} V_i[t] * N_{ij'} * c(f_i) * \lambda_{tj'}$;
    According to the selected cloudlet $c_{j'}$, and execution timeslot $t \in \mathbb{T}'_i$,
    $\lambda_{tj'} \leftarrow \lambda_{tj'} * (1 + \frac{N_{ij'} * c(f_i)}{cap_{j'})}) + \frac{N_{ij'} * c(f_i) * pay_i}{d_i * cap_{j'}}$;
  else
    Reject request ρ_i;
  end if;
end while;

Lemma 5. *Algorithm 1 delivers a feasible solution for* **P3***.*

Proof: Following `Algorithm` 1, Constraint (32) is satisfied by the update rule of δ_i when request arrives. Notice that the update function (34) of λ_{tj} is non-decreasing, $pay_i - \min_{c_j \in C}\{\sum_{t \in \mathbb{T}} V_i[t] * N_{ij} * c(f_i) * \lambda_{tj}\}$ thus is non-increasing. Constraint (32) still holds when updating the value of λ_{tj}. The lemma then follows. ■

Denote by pay_{max} and pay_{min} the maximum and minimum payments, d_{max} and d_{min} the maximum and minimum execution durations among requests, and cap_{max} and cap_{min} the maximum and minimum capacities of cloudlets, respectively.

Lemma 6.

$$\lambda_{tj} \geq \frac{pay_{min}}{d_{max}} * ((1 + \frac{a_{min}}{cap_{max}})^{\sum_{\rho_i \in \mathbb{R}} V_i[t] * Y_{ij}} - 1). \quad (37)$$

Proof: We prove the claim by induction. Clearly, the RHS of Inequality (37) is 0 before the arrival of the first request. Thus, this induction hypothesis holds when $\lambda_{tj} = 0$. Let $\lambda_{tj}(start)$ and $\lambda_{tj}(end)$ be the values of variable λ_{tj} before and after request $\rho_{i'}$ arrives. We then perform the induction based on whether or not the value of λ_{tj} is updated.

Case (i). λ_{tj} is not updated. This happens when the request is either rejected, or admitted but its VNF instances are not placed in cloudlet c_j, or its execution time slots do not include time slot t, i.e., $V_{i'}[t] * Y_{i'j} = 0$. In this case, there is no update on λ_{tj}, i.e., $\lambda_{tj}(end) = \lambda_{tj}(start)$. We then have

$$\begin{aligned}
\lambda_{tj}(end) =& \lambda_{tj}(start) \\
\geq& \frac{pay_{min}}{d_{max}} * ((1 + \frac{a_{min}}{cap_{max}})^{\sum_{\rho_i \in \mathbb{R} \setminus i'} V_i[t] * Y_{ij}} - 1) \\
\geq& \frac{pay_{min}}{d_{max}} * ((1 + \frac{a_{min}}{cap_{max}})^{\sum_{\rho_i \in \mathbb{R} \setminus i'} V_i[t] * Y_{ij}} * \\
& (1 + \frac{a_{min}}{cap_{max}})^{V_{i'}[t] * Y_{i'j}} - 1), \text{ by } V_{i'}[t] * Y_{i'j} = 0 \\
=& \frac{pay_{min}}{d_{max}} * ((1 + \frac{a_{min}}{cap_{max}})^{\sum_{\rho_i \in \mathbb{R}} V_i[t] * Y_{ij}} - 1).
\end{aligned} \quad (38)$$

Case (ii). λ_{tj} is updated. This happens when the request is admitted and all its VNF instances are placed in cloudlet c_j in time slot t, i.e., $V_{i'}[t] * Y_{i'j} = 1$. We then have

$$\begin{aligned}
\lambda_{tj}(end) =& \lambda_{tj}(start) * (1 + \frac{N_{ij} * c(f_i)}{cap_j}) + \frac{N_{ij} * c(f_i) * pay_i}{d_i * cap_j} \\
\geq& \lambda_{tj}(start) * (1 + \frac{a_{min}}{cap_{max}}) + \frac{a_{min} * pay_{min}}{d_{max} * cap_{max}}.
\end{aligned}$$

Apply hypothesis (37), we have

$$\begin{aligned}
\lambda_{tj}(end) \geq& \frac{pay_{min}}{d_{max}} * ((1 + \frac{a_{min}}{cap_{max}})^{\sum_{\rho_i \in \mathbb{R} \setminus i'} V_i[t] * Y_{ij}} - 1) * \\
& (1 + \frac{a_{min}}{cap_{max}}) + \frac{a_{min} * pay_{min}}{d_{max} * cap_{max}} \\
=& \frac{pay_{min}}{d_{max}} * ((1 + \frac{a_{min}}{cap_{max}})^{\sum_{\rho_i \in \mathbb{R} \setminus i'} V_i[t] * Y_{ij}} * \\
& (1 + \frac{a_{min}}{cap_{max}}) - 1) \\
\geq& \frac{pay_{min}}{d_{max}} * ((1 + \frac{a_{min}}{cap_{max}})^{\sum_{\rho_i \in \mathbb{R} \setminus i'} V_i[t] * Y_{ij}} * \\
& (1 + \frac{a_{min}}{cap_{max}})^{V_{i'}[t] * Y_{i'j}} - 1), \text{ by } V_{i'}[t] * Y_{i'j} = 1 \\
=& \frac{pay_{min}}{d_{max}} * ((1 + \frac{a_{min}}{cap_{max}})^{\sum_{\rho_i \in \mathbb{R}} V_i[t] * Y_{ij}} - 1).
\end{aligned}$$

Hence, the lemma follows. ■

Lemma 7. $\lambda_{tj} < \frac{pay_{max}}{a_{min}} * (1 + \frac{a_{max}}{cap_{min}}) + \frac{a_{max} * pay_{max}}{d_{min} * cap_{min}}$.
.

Proof: Recall that the value of λ_{tj} is initialized 0. We only admit a new request ρ_i if $pay_i - \min_{c_j \in C} \sum_{t \in \mathbb{T}} V_i[t] * N_{ij} * c(f_i) * \lambda_{tj} > 0$ and update λ_{tj} associated with the cloudlet with $\min_{c_j \in C} \sum_{t \in \mathbb{T}} V_i[t] * N_{ij} * c(f_i) * \lambda_{tj}$. Then, for the selected cloudlet c_j, we have

$$\begin{aligned}
& pay_i - \sum_{t \in \mathbb{T}} V_i[t] * N_{ij} * c(f_i) * \lambda_{tj} > 0 \\
\Rightarrow & \sum_{t \in \mathbb{T}} V_i[t] * \lambda_{tj} < \frac{pay_i}{N_{ij} * c(f_i)} \\
\Rightarrow & \sum_{t \in \mathbb{T}} V_i[t] * \lambda_{tj} < \frac{pay_{max}}{a_{min}} \\
\Rightarrow & \lambda_{tj} < \frac{pay_{max}}{a_{min}}.
\end{aligned} \quad (39)$$

As a result, λ_{tj} will not be updated if $\lambda_{tj} \geq \frac{pay_{max}}{a_{min}}$, with the update function (34) of λ_{tj}, we thus have

$$\begin{aligned}
\lambda_{tj} <& \frac{pay_{max}}{a_{min}} * (1 + \frac{N_{ij} * c(f_i)}{cap_j}) + \frac{N_{ij} * c(f_i) * pay_i}{d_i * cap_j} \\
<& \frac{pay_{max}}{a_{min}} * (1 + \frac{a_{max}}{cap_{min}}) + \frac{a_{max} * pay_{max}}{d_{min} * cap_{min}}.
\end{aligned} \quad (40)$$

The lemma then follows. ■

Lemma 8. *The violation of the capacity constraint of each cloudlet for* **P2** *in the solution delivered by* `Algorithm 1` *is bounded by* ξ, *where* $\xi = \frac{a_{max}}{cap_{min} * \log_2(1 + \frac{a_{min}}{cap_{max}})} * \log_2(\frac{pay_{max} * d_{max}}{pay_{min}} * (\frac{1}{a_{min}} + \frac{a_{max}}{a_{min} * cap_{min}} + \frac{a_{max}}{d_{min} * cap_{min}}) + 1)$.

Proof: Combining lemmas 6 and 7, we have

$$\begin{aligned}
& \frac{pay_{min}}{d_{max}} * ((1 + \frac{a_{min}}{cap_{max}})^{\sum_{\rho_i \in \mathbb{R}} V_i[t] * Y_{ij}} - 1) \\
< & \frac{pay_{max}}{a_{min}} * (1 + \frac{a_{max}}{cap_{min}}) + \frac{a_{max} * pay_{max}}{d_{min} * cap_{min}}
\end{aligned}$$

Then, we have

$$\begin{aligned}
& \sum_{\rho_i \in \mathbb{R}} V_i[t] * Y_{ij} \\
< & \frac{\log_2(\frac{pay_{max} * d_{max}}{pay_{min}} * (\frac{1}{a_{min}} + \frac{a_{max}}{a_{min} * cap_{min}} + \frac{a_{max}}{d_{min} * cap_{min}}) + 1)}{\log_2(1 + \frac{a_{min}}{cap_{max}})}.
\end{aligned} \quad (41)$$

To calculate the capacity violation, we have

$$\begin{aligned}
& \sum_{\rho_i \in \mathbb{R}} V_i[t] * N_{ij} * c(f_i) * Y_{ij} \\
\leq & \sum_{\rho_i \in \mathbb{R}} V_i[t] * a_{max} * Y_{ij} \\
\leq & \frac{a_{max}}{\log_2(1 + \frac{a_{min}}{cap_{max}})} * \log_2(\frac{pay_{max} * d_{max}}{pay_{min}} * \\
& (\frac{1}{a_{min}} + \frac{a_{max}}{a_{min} * cap_{min}} + \frac{a_{max}}{d_{min} * cap_{min}}) + 1), \text{by (41)}.
\end{aligned}$$

Considering the capacity constraint (9), we have the capacity violation bounded by ξ. Hence, the lemma follows. ■

Theorem 1. *Given the MEC network* $G = (V, E)$, *there is an on-line algorithm* `Algorithm 1` *for the VNF service*

reliability problem under the on-site scheme, with a $(1+a_{max})$*-competitive ratio while the violation of the computing capacity at any cloudlet is bounded by* ξ*, where* $a_{max} = \max_{\rho_i \in \mathbb{R}, c_j \in C}\{N_{ij} * c(f_i)\}$ *and* ξ *is defined in Lemma 8.*

Proof: Let OPT_3 be the optimal solution of **P3**. Following Lemma 4, we have $P \geq \frac{D}{1+a_{max}}$. Following Lemma 5, D is a feasible solution to **P3**, thus, $D \geq OPT_3$ as **P3** is a minimization problem. Following the weak duality and Lemma 2, we have $OPT_3 \geq OPT_2$ and $OPT_2 \geq OPT_1$. We then have

$$P \geq \frac{D}{1+a_{max}} \geq \frac{OPT_3}{1+a_{max}} \geq \frac{OPT_2}{1+a_{max}} \geq \frac{OPT_1}{1+a_{max}}.$$

Following Lemma 4, P is the value of objective function to **P2** which is the LP relaxation of **P1**. In Algorithm 1, we always update the decision variables as 0 or 1. So the solution delivered by Algorithm 1 is also the solution to **P1** and P is the value of the objective function of **P1** as well. Combining Lemma 8, the violation of the computing capacity at cloudlets is bounded by ξ.

The time complexity analysis is omitted due to space limitation. ■

V. Algorithm for the VNF service reliability problem under the off-site scheme

In this section, we deal with the VNF service reliability problem under the off-site scheme. We denote this problem as **P4**. The strategy for this problem is similar to the one for **P1**. That is, we first consider the LP relaxation of **P4** and denote this relaxation as **P5**. We then solve **P6** which is the dual of **P5**. A feasible solution to **P6** ultimately returns a feasible solution to **P4**. As **P4** is an INP problem in Section III, the primal-dual dynamic updating technique cannot be applied to this INP problem directly due to the fact that both Inequalities (16) and (17) in its INP formulation are nonlinear. Fortunately, through a non-trivial equivalent transformation, we can convert the INP formulation of **P4** into an equivalent ILP formulation, and solve the problem. As Inequalities (16) and (17) are reliability constraints that are driven from Inequality (10), we focus on converting Inequality (10) into a linear one, we thus have

$$\prod_{c_j \in C} (1 - r(f_i) * r(c_j) * Y_{ij}) \leq 1 - R_i, \qquad \forall \rho_i \in \mathbb{R}. \quad (42)$$

Property 1. *Function* $\ln(x)$ *is monotonically increasing with the growth of the value of* $x > 0$*, i.e., when* $0 < x_1 \leq x_2$*,* $\ln(x_1) \leq \ln(x_2)$.

Since $0 < r(c_i) < 1$, $0 < r(c_j) < 1$, $0 < R_i < 1$ and $Y_{ij} \in \{0, 1\}$, it is clear that $\prod_{c_j \in C}(1 - r(f_i) * r(c_j) * Y_{ij}) > 0$ and $1 - R_i > 0$. With regard to Property 1, we show that Inequality (42) is equivalent to the following Inequality.

$$\begin{aligned} &\ln \prod_{c_j \in C} (1 - r(f_i) * r(c_j) * Y_{ij}) \leq \ln(1 - R_i), \\ &\Rightarrow \sum_{c_j \in C} \ln(1 - r(f_i) * r(c_j) * Y_{ij}) \leq \ln(1 - R_i). \end{aligned} \quad (43)$$

Lemma 9.

$\ln(1 - r(f_i) * r(c_j) * Y_{ij}) = (\ln(1 - r(f_i) * r(c_j))) * Y_{ij}$, $Y_{ij} \in \{0, 1\}, \forall \rho_i \in \mathbb{R}, \forall c_j \in C$.

Due to space limitation, the proof is omitted.

Following Lemma 9 and Inequality (43), we have $\forall \rho_i \in \mathbb{R}$,

$$\sum_{c_j \in C} (\ln(1 - r(f_i) * r(c_j))) * Y_{ij} \leq \ln(1 - R_i). \quad (44)$$

Inequality (43) can be rewritten as follows,

$$L * X_i \leq \sum_{c_j \in C} (\ln(1 - r(f_i) * r(c_j))) * Y_{ij} \leq \ln(1 - R_i) * X_i, \quad (45)$$

where $L = \min_{\rho_i \in \mathcal{R}}\{\sum_{c_j \in C} \ln(1 - r(f_i) * r(c_j))\}$ is a lower bound of $\sum_{c_j \in C}(\ln(1 - r(f_i) * r(c_j))) * Y_{ij}$. L is constant as both $r(f_i)$ and $r(c_j)$ are constants.

Lemma 10. *Inequality* (45) *meets the reliability requirement when a request* ρ_i *is admitted, and no VNF instance will be placed in cloudlets when the request is rejected.*

Proof: If $X_i = 1$, request ρ_i is admitted, Inequality (45) is shown as follows.

$$L \leq \sum_{c_j \in C} (\ln(1 - r(f_i) * r(c_j))) * Y_{ij} \leq \ln(1 - R_i), \ \forall \rho_i \in \mathbb{R}. \quad (46)$$

It can be seen that $L \leq \sum_{c_j \in C}(\ln(1 - r(f_i) * r(c_j))) * Y_{ij}$ always holds, while $\sum_{c_j \in C}(\ln(1 - r(f_i) * r(c_j))) * Y_{ij} \leq \ln(1 - R_i)$ by Inequality (44). Otherwise ($X_i = 0$), request ρ_i is rejected. We have

$$\begin{aligned} 0 \leq &\sum_{c_j \in C} (\ln(1 - r(f_i) * r(c_j))) * Y_{ij} \leq 0 \\ \Rightarrow &\sum_{c_j \in C} (\ln(1 - r(f_i) * r(c_j))) * Y_{ij} = 0. \end{aligned} \quad (47)$$

Since $0 < r(f_i) < 1$, and $0 < r(c_j) < 1$, we have $Y_{ij} = 0$ for each cloudlet $c_j \in C$, i.e., no VNF instance will be placed in any cloudlet. The lemma then follows. ■

P4 now can be reformulated as an equivalent ILP as follows.

$$maximize \sum_{\rho_i \in \mathbb{R}} X_i * pay_i, \quad (48)$$

subject to:

$$\sum_{\rho_i \in \mathbb{R}} V_i[t] * c(f_i) * Y_{ij} \leq cap_j, \qquad \forall t \in \mathbb{T}, \forall c_j \in C, \quad (49)$$

$$\sum_{c_j \in C} (\ln(1 - r(f_i) * r(c_j))) * Y_{ij} \leq (\ln(1 - R_i)) * X_i, \forall \rho_i \in \mathbb{R}, \quad (50)$$

$$\sum_{c_j \in C} (\ln(1 - r(f_i) * r(c_j))) * Y_{ij} \geq L * X_i, \ \forall \rho_i \in \mathbb{R}, \quad (51)$$

$$X_i \in \{0, 1\}, \qquad \forall \rho_i \in \mathbb{R}, \quad (52)$$

$$Y_{ij} \in \{0, 1\}, \qquad \forall \rho_i \in \mathbb{R}, \forall c_j \in C. \quad (53)$$

We then perform the LP relaxation of the defined ILP, and thus **P5** can be reformulated equivalently as follows.

$$maximize \sum_{\rho_i \in \mathbb{R}} X_i * pay_i, \tag{54}$$

subject to

$$(49),\ (50),\ (51),$$
$$X_i \leq 1, \qquad \forall \rho_i \in \mathbb{R}, \tag{55}$$
$$X_i \geq 0,\ Y_{ij} \geq 0, \qquad \forall \rho_i \in \mathbb{R},\ \forall c_j \in C. \tag{56}$$

Lemma 11. *Any feasible solution for* **P4** *is a feasible solution for* **P5**. *Let* OPT_4 *and* OPT_5 *be the optimal solutions to* **P4** *and* **P5**, *respectively. Then, we have* $OPT_4 \leq OPT_5$.

We then have **P6**, which is the dual of **P5** as follows,

$$minimize \sum_{t \in \mathbb{T}, c_j \in C} cap_j * \lambda_{tj} + \sum_{\rho_i \in \mathbb{R}} \delta_i, \tag{57}$$

subject to

$$\ln(1 - R_i) * \beta_i + L * \mu_i + \delta_i \geq pay_i, \forall \rho_i \in \mathbb{R}, \tag{58}$$
$$\sum_{t \in \mathbb{T}} V_i[t] * c(f_i) * \lambda_{tj} + \ln(1 - r(f_i) * r(c_j)) * (\beta_i - \mu_i) \geq 0,$$
$$\forall \rho_i \in \mathbb{R}, \forall c_j \in C, \tag{59}$$
$$\lambda_{tj} \geq 0,\ \beta_i \geq 0,\ \mu_i \geq 0,\ \delta_i \geq 0,\ \forall \rho_i \in \mathbb{R}, \forall c_j \in C, \forall t \in \mathbb{T}, \tag{60}$$

where $\lambda_{tj},\ \beta_i,\ \mu_i$ and δ_i are dual variables corresponding to constraints (49), (50), (51) and (55) respectively.

Let us first examine Constraint (58). Because L is a lower bound of $\sum_{c_j \in C}(\ln(1 - r(f_i) * r(c_j))) * Y_{ij}$, it can be treated as $-\infty$ in an extreme case. To satisfy this constraint, with $\mu_i \geq 0,\ \forall \rho_i \in \mathbb{R}$, we can set $\mu_i = 0$ and it can be eliminated. Then, constraints (58) and (59) can be rewritten as follows.

$$\beta_i \geq \quad \frac{pay_i - \delta_i}{-\ln(1 - R_i)}, \qquad \forall \rho_i \in \mathbb{R}, \tag{61}$$
$$\beta_i \leq \quad \frac{\sum_{t \in \mathbb{T}} V_i[t] * c(f_i) * \lambda_{tj}}{-\ln(1 - r(f_i) * r(c_j))}, \quad \forall \rho_i \in \mathbb{R}, \forall c_j \in C. \tag{62}$$

From Inequality (62), we have

$$\beta_i \leq \min_{c_j \in C} \frac{\sum_{t \in \mathbb{T}} V_i[t] * c(f_i) * \lambda_{tj}}{-\ln(1 - r(f_i) * r(c_j))}, \quad \forall \rho_i \in \mathbb{R}. \tag{63}$$

Combining Inequalities (61) and (63), we have $\forall \rho_i \in \mathbb{R}$,

$$\frac{pay_i - \delta_i}{-\ln(1 - R_i)} \leq \min_{c_j \in C} \frac{\sum_{t \in \mathbb{T}} V_i[t] * c(f_i) * \lambda_{tj}}{-\ln(1 - r(f_i) * r(c_j))}. \tag{64}$$

Lemma 12. *If Inequality* (64) *holds,* $\forall \rho_i \in \mathbb{R}$, *there always exists a feasible* β_i *to satisfy constraints* (58) *and* (59).

Proof: Because $V_i[t] \geq 0,\ c(f_i) > 0,\ \lambda_{tj} \geq 0,\ 0 < r(f_i) < 1$, and $0 < r(c_j) < 1,\ \forall \rho_i \in \mathbb{R},\ \forall c_j \in C,\ \forall t \in \mathbb{T}$, then $\min_{c_j \in C} \frac{\sum_{t \in \mathbb{T}} V_i[t] * c(f_i) * \lambda_{tj}}{-\ln(1 - r(f_i) * r(c_j))} \geq 0$. As $\beta_i \geq 0$, if Inequality (64) holds, there always exists a certain β_i to make the following inequality hold.

$$\frac{pay_i - \delta_i}{-\ln(1 - R_i)} \leq \beta_i \leq \min_{c_j \in C} \frac{\sum_{t \in \mathbb{T}} V_i[t] * c(f_i) * \lambda_{tj}}{-\ln(1 - r(f_i) * r(c_j))}. \tag{65}$$

Algorithm 2 An on-line algorithm for the VNF service reliability problem under the off-site Scheme

Input: An MEC network $G = (V, E)$ and incoming requests.
Output: An on-line scheduling of incoming requests.
1: Initialize λ_{tj} =0, $\forall c_j \in C,\ \forall t \in \mathbb{T}$;
2: **Upon arrival of the request** ρ_i
3: **for** cloudlet $c_j \in C$ **do**
4: Calculate $\frac{\sum_{t \in \mathbb{T}} V_i[t] * \lambda_{tj}}{-\ln(1 - r(f_i) * r(c_j))}$;
5: **if** $pay_i + \ln(1 - R_i) * c(f_i) * \frac{\sum_{t \in \mathbb{T}} V_i[t] * \lambda_{tj}}{-\ln(1 - r(f_i) * r(c_j))} \leq 0$ **then**
6: no VNF instance is placed at cloudlet c_j;
7: **end if**;
8: **end for**;
9: Sort cloudlets in non-decreasing order of $\frac{\sum_{t \in \mathbb{T}} V_i[t] * \lambda_{tj}}{-\ln(1 - r(f_i) * r(c_j))}$;
10: **while** consider cloudlet c_j in the sorted cloudlet sequence **do**
11: **if** (c_j has enough residual resource during the execution timeslots of request ρ_i) **then**
12: $S(i) \leftarrow S(i) \cup \{c_j\}$ /* $S(i)$ is set of cloudlets for hosting its VNF instances of request ρ_i */
13: **if** the reliability requirement of ρ_i is met with $S(i)$ **then**
14: **break** ;
15: **end if**;
16: **end if**;
17: **end while**;
18: **if** the reliability requirement of ρ_i is met with $S(i)$ **then**
19: Admit request ρ_i by putting one VNF instance in each of the selected cloudlets;
20: Update λ_{tj} by formula (67);
21: **else**
22: Reject request ρ_i;
23: **end if**;

Inequalities (61) and (62) then hold. Thus, constraints (58) and (59) hold, and the lemma follows. ■

Following Lemma 12, the dual variable β_i now can be eliminated. Inequality (64) can be rewritten as follows, $\forall \rho_i \in \mathbb{R}$,

$$\delta_i \geq pay_i + \ln(1 - R_i) * c(f_i) * \min_{c_j \in C} \frac{\sum_{t \in \mathbb{T}} V_i[t] * \lambda_{tj}}{-\ln(1 - r(f_i) * r(c_j))}. \tag{66}$$

Clearly, we now only need to consider constraint (66), and two dual variables $\lambda_{tj} \geq 0$ and $\delta_i \geq 0,\ \forall \rho_i \in \mathbb{R},\ \forall c_j \in C,\ \forall t \in \mathbb{T}$ to solve the problem. We devise an on-line algorithm `Algorithm 2` for the *VNF service reliability problem* under the off-site scheme. The dual variable λ_{tj} is 0 initially. We then calculate $\frac{\sum_{t \in \mathbb{T}} V_i[t] * \lambda_{tj}}{-\ln(1 - r(f_i) * r(c_j))}$ for each cloudlet c_j. If $pay_i + \ln(1 - R_i) * c(f_i) * \frac{\sum_{t \in \mathbb{T}} V_i[t] * \lambda_{tj}}{-\ln(1 - r(f_i) * r(c_j))} \leq 0$, no VNF instance will be placed in cloudlet c_j. What followed is to sort cloudlets in non-decreasing order of $\frac{\sum_{t \in \mathbb{T}} V_i[t] * \lambda_{tj}}{-\ln(1 - r(f_i) * r(c_j))}$. We finally identify a set of cloudlets for hosting VNF instances of the request in their sorted order until it exists a scheduling such that the capacity of each chosen cloudlet and the reliability requirement of the request is met. If such scheduling can not be found, the request is rejected. Otherwise, the request is admitted by putting one VNF instance in each of the selected cloudlets and the value of δ_i is updated as follows.

$$\lambda_{tj} := \lambda_{tj}\left(1 + \frac{\ln(1 - R_i) * c(f_i)}{\ln(1 - r(f_i) * r(c_j)) * cap_j}\right) + \frac{\ln(1 - R_i) * c(f_i) * pay_i}{\ln(1 - r(f_i) * r(c_j)) * d_i * cap_j}, \tag{67}$$

where d_i is the execution duration.

Theorem 2. *Given an MEC network $G = (V, E)$, there is an on-line algorithm,* `Algorithm` *2, for the VNF service reliability problem under the off-site scheme, which runs in polynomial time.*

Proof: Although `Algorithm` 2 aims to solve **P5** that is a LP relaxation of **P4**, we always select a cloudlet instead of a fraction of a cloudlet in `Algorithm` 2. Furthermore, the solution delivered meets the capacity constraints on the selected cloudlets until the reliability requirement of each admitted request is met. If the reliability requirement of a request cannot be met, the request will be rejected. Thus, no capacity constraints on cloudlets are violated. Due to space limitation, the time complexity analysis of the proposed algorithm is omitted. ■

VI. Performance Evaluation

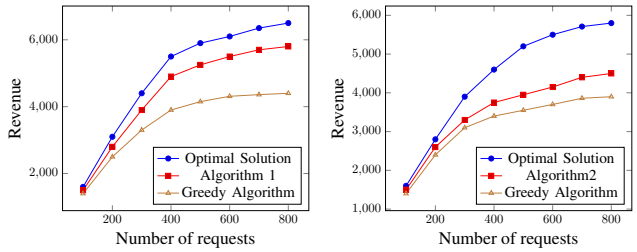
In this section, we study the performance of the proposed algorithms for the VNF service reliability problem under both on-site and off-site schemes.

A. Environment settings

To evaluate the performance of the proposed algorithm `Algorithm` 1 under the on-site scheme, we do not directly implement the proposed algorithm, because the solution delivered by the algorithm may violate resource capacity constraints on cloudlets and links. Thus, for the on-site case, we adopt the scaling approach [14] to avoid the resource capacity violation. We augment the required amount of resource for all users, i.e., we scale the amount of resource required for implementing the requested VNF instances such that no actual capacity constraint violation occurs. We assume that there are 10 types of VNFs with reliabilities between 0.9 and 0.9999 and the amounts of computing resource demanded for their implementations are ranged from 1 to 3 computing units [15]. We adopt real network topologies in [18] while requests along with the reliability and capacity of each cloudlet are randomly generated, using the data sets in [19]. We introduce two benchmarks that are used to evaluate the proposed algorithm performance. The first one is a greedy algorithm which always tries to admit all coming requests by preferring to place VNF instances in cloudlets with high reliabilities. Then, as we formulate the problem as an ILP under the on-site case while we reduce the problem under the off-site scheme from an INP to an ILP, we are able to obtain optimal results of the problems in its off-line setting by utilizing CPLEX Optimizer. To investigate the impact of variation of request payment rates, we denote the payment rate of each request ρ_i as pr_i with $pr_i = \frac{pay_i}{d_i * c(f_i) * R_i}$, where pay_i, d_i, $c(f_i)$ and R_i are the payment, execution duration, resource consumption of the number of requested VNF instances, and the reliability requirement of request ρ_i, respectively. The maximum and minimum payment rate are denoted by pr_{max} and pr_{min}, respectively. The ratio H of pr_{max} to pr_{max} is defined as the variation of request payment rates, i.e., $H = \frac{pr_{max}}{pr_{min}}$. Denote by rc_{max} and rc_{min} the maximum and minimum cloudlet reliabilities respectively. The ratio of rc_{max} to rc_{min} is defined as the variation of cloudlet reliabilities which is denoted as K, i.e., $K = \frac{rc_{max}}{rc_{min}}$.

B. Performance evaluation of the proposed algorithms

As the number of requests is an essential factor in evaluating on-line algorithms. When the number of requests is large, it can be seen that not all requests can be admitted due to the limited network resource available. So we need a strategy to maximize the revenue collected by admitting requests without the knowledge of future request arrivals. We keep the other settings and parameters the same such as the cloudlet capacities, and only increase the number of requests in the given time horizon. In addition, the requirements and payments of requests are randomly generated but in the same specific ranges respectively. Figure 1 demonstrates the performance of different algorithms by varying the number of requests considered.



(a) Performance of `Algorithm` 1 (b) Performance of `Algorithm` 2

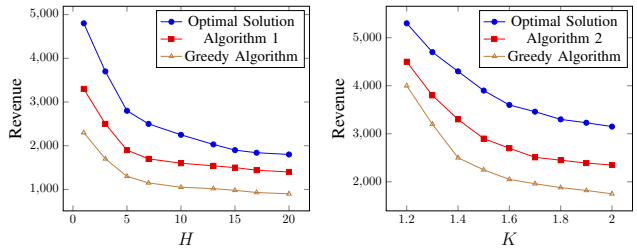
Fig. 1. Performance of the proposed algorithms by varying numbers of requests.

We first evaluate the performance of `Algorithm` 1 by varying the number of requests in a given finite horizon. Figure 1(a) shows that Algorithm 1 always achieves better performance than that by the greedy algorithm. Notice that when the number of requests is small, both `Algorithm` 1 and the greedy algorithm can achieve nearly optimal performance. This can be justified that the network has abundant resource to meet the resource demands of all request. However, with the increase in the number of requests, `Algorithm` 1 outperforms the greedy algorithm significantly by 31.8% when the number of requests reaches 800. We then evaluate the performance of `Algorithm` 2 by varying the number of requests in a given finite horizon. It can be seen from Figure 1(b) that Algorithm 2 outperforms the greedy algorithm by 15.4% in terms of the performance.

C. Impact of parameters on the performance of the proposed algorithms.

To exploit the impact of the variation of request payment rates, we conduct experiments with different values of H by fixing pr_{max} while varying pr_{min}. Note that among all requests, the payment rates of requests follow a uniform distribution over a value interval $[pr_{min}, pr_{max}]$. Fig 2(a) shows the performance results of different algorithms.

It can be seen from Figure 2(a) that the revenue decreases with the growth of the value of H, because we reduce pr_{min} to



(a) Impact of H on `Algorithm 1` (b) Impact of K on `Algorithm 2`

Fig. 2. Impact of parameters on the performance of the proposed algorithms

increase H, i.e., the users tend to pay less but want to consume more resources. As the total amount of resource is fixed, we can only collect less and less revenue with the increase in the value of H. We also notice that the impact of the value of H is significant when H increases from 1 to 5, but its impact then diminishes and becomes negligible. This is because the network tends to reject requests with low payment rates.

The intention of placing VNF instances in reliable cloudlets seems to consume less resource because of fewer required backup VNF instances. However, when the number of requests is large, an optimal strategy should fully utilize the limited cloud computing resources at different cloudlets.

To investigate the impact of the variation of request payment rates, we conduct experiments with different K by fixing rc_{max} while varying rc_{min}. Note that among all cloudlets, cloudlet reliability follows a uniform distribution over the interval $[rc_{min}, rc_{max}]$. Fig 2(b) shows the performance results. The experimental results show that Algorithm 2 always achieves better performance than that of the greedy algorithm by varying the value of K. It can be seen from Figure 2(b) that the revenue decreases with the growth of K. It is because we fix rc_{max} and reduce rc_{min} to increase K, i.e., the cloudlets tend to be less reliable. Albeit the same total amount of available resource, implementations of requests need to consume more resource because more backup VNF instances are required to be established. We also notice that the greedy algorithm performs poorly, especially when K is large. The justification is that in the off-site scheme, backup VNF instances are placed in each of chosen cloudlets to meet the reliability requirements of requests, i.e., to admit a request, multiple cloudlets with enough residual resource are usually needed while only one cloudlet is required in the on-site scheme. The greedy algorithm always puts VNF instances in reliable cloudlets while the proposed algorithm and the optimal solution are able to fully utilize all cloudlets, but the greedy algorithm does not. In the extreme case where all resource in reliable cloudlets is exhausted, the greedy algorithm fails to admit any incoming requests in spite of existing lots of failure-prone cloudlets in the MEC.

VII. Conclusion

In this paper, we studied the reliable VNF service provisioning for IoT applications in an MEC environment to meet service reliability requirements of mobile users. We first formulated a novel VNF service reliability problem with the aim to maximize the revenue collected by user request admissions. We then developed an on-line algorithm with a provable competitive ratio for the problem under the on-site scheme with bounded moderate computing resource violations. We also devised an efficient on-line heuristic for the problem under the off-site scheme via adopting the primal-dual dynamic updating technique. We finally evaluate the proposed algorithms through experimental simulations. The experimental results demonstrated that the proposed algorithms are promising, and outperform the mentioned benchmark.

References

[1] F. Wang, J. Xu, X. Wang and S. Cui, Joint Offloading and Computing Optimization in Wireless Powered Mobile-edge Computing Systems, *Proc. of IEEE ICC'17*, 2017.

[2] C. Wang, J. Kuo. D. Yang and W. Chen, Green Software-Defined Internet of Things for Big Data Processing in Mobile Edge Networks, *Proc. of IEEE ICC'18*, 2018.

[3] J. Zhang, D. Zeng, L. Gu, H. Yao and M. Xiong, Joint Optimization of Virtual Function Migration and Rule Update in Software Defined NFV Networks, *Proc of IEEE Globecom'17*, 2017.

[4] 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 IEEE Globecom'17*, 2017.

[5] W. Ding, H. Yu and S. Luo, Enhancing the Reliability of Services in NFV with the Cost-efficient Redundancy Scheme, *2017 IEEE International Conference on Communications (ICC)*, 2017.

[6] H. Chantre and N. Fonseca, Redundant Placement of Virtualized Network Functions for LTE Evolved Multimedia Broadcast Multicast Services, *2017 IEEE International Conference on Communications (ICC)*, 2017.

[7] W. Ding, H. Yu and S. Luo, Enhancing the Reliability of Services in NFV with the Cost-efficient Redundancy Scheme, *Proc. of IEEE ICC'17*, 2017.

[8] W. Chang and P. Wang, An Adaptable Replication Scheme in Mobile Online System for Mobile-edge Cloud Computing, *Proc. INFOCOM WKSHPS*, 2017.

[9] B. Han, V. Gopalakrishnan, G. Kathirvel and A. Shaikh, On the Resiliency of Virtual Network Functions, *IEEE Communications Magazine*, vol. 55, no. 7, pp. 152-157, 2017.

[10] J. Pan and J. McElhannon, Future Edge Cloud and Edge Computing for Internet of Things Applications, *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 439-449, 2018.

[11] 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.

[12] J. Fan, M. Jiang and C. Qiao, Carrier-grade Availability-aware Mapping of Service Function Chains with On-sites, *2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS)*, 2017.

[13] A. Hmaity, M. Savi, F. Musumeci, M. Tornatore and A. Pattavina, Virtual Network Function Placement for Resilient Service Chain Provisioning, *2016 8th International Workshop on Resilient Networks Design and Modeling (RNDM)*, 2016.

[14] Q. Fan and N. Ansari, Cost Aware Cloudlet Placement for Big Data Processing at the Edge, *Proc. of IEEE ICC'17*, 2017.

[15] J. Kong, I. Kim, X. Wang, Q. Zhang, H. Cankaya, W. Xie, T. Ikeuchi and J. Jue, Guaranteed-availability Network Function Virtualization with Network Protection and VNF Replication, *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, 2017.

[16] M. Beck, J. Botero and K. Samelin, Resilient Allocation of Service Function Chains, *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2016.

[17] B. Yin, Y. Cheng, L. Cai and X. Cao, Online SLA-Aware Multi-Resource Allocation for Deadline Sensitive Jobs in Edge-Clouds, *Proc of IEEE Globecom'17*, 2017.

[18] S. Knight et al. The internet topology zoo. *IEEE Journal on Selected Areas in Communications*, Vol 29, pp. 17651775, 2011.

[19] Google Cluster Data, J. L. Hellerstein, 2010. Available: https://ai.googleblog.com/2010/01/google-cluster-data.html. [Accessed: 15-May- 2018]