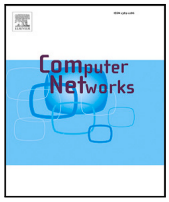
Computer Networks 233 (2023) 109882

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



Reveal: Robustness-aware VNF placement and request scheduling in edge clouds

Jin Fang [a,b], Gongming Zhao [a,b,*], Hongli Xu [a,b,*], Huaqing Tu [a,b], Haibo Wang [c]

[a] School of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui, 230027, China
[b] Suzhou Institute for Advanced Research, University of Science and Technology of China, Suzhou, Jiangsu, 215123, China
[c] Department of Computer and Information Science and Engineering, University of Florida, Gainesville, FL, 32611, USA

ARTICLE INFO

Keywords:
Edge cloud
VNF placement
Request scheduling
Robustness

ABSTRACT

In the edge cloud network, service providers place virtual network functions (VNFs) in edge clouds to serve users' requests. Thus, it is essential to consider VNF placement and request scheduling in edge clouds. Existing works often focus on minimizing request completion time or maximizing network throughput to utilize network resources and ensure users' QoS efficiently. However, they ignore two practical factors: **malicious users** and **failed VNFs**, leading to poor network robustness. To this end, this paper studies robustness-aware VNF placement and request scheduling, named Reveal. Specifically, we limit the number of VNFs each user can access and the number of users each VNF can serve to control the influence scope of malicious users and VNF failures. Since placing VNFs is time-consuming and requests arrive dynamically, we solve this problem through two phases: **robust VNF placement and assignment**, and **online request scheduling**. For the first phase, we design an efficient knapsack-based rounding algorithm with bounded approximation factors. For online request scheduling, we propose a primal–dual based algorithm with a competitive ratio of $\left[1-\epsilon, O(\log 1/\epsilon)\right]$ where $\epsilon \in (0,1)$. Experiment and simulation results show that Reveal can achieve better performance and robustness than other alternatives.

1. Introduction

Due to the development of the Internet of Things and 5G, the computing paradigm is shifting from centralized cloud computing to distributed cloud computing such as edge clouds [1,2]. Compared with centralized cloud computing, edge cloud pushes computing capabilities to the network edge, saving backhaul transmission bandwidth and reducing transmission delay. Therefore, it is suitable for real-time applications (*e.g.*, online gaming [3], stream video [4] and object recognition [5]).

A typical edge cloud comprises an edge server and a base station. Service providers supply services by placing VNFs (*e.g.*, firewall, load balancer) in edge servers [6], while users obtain services through forwarding requests to corresponding VNFs [7]. Since the resource capacity of the edge server is limited, it is essential to consider VNF placement and request scheduling in edge clouds. Existing works on VNF placement and request scheduling mainly focus on minimizing request completion time [8,9] or maximizing network throughput [10,11]. For example, the authors in [8] consider the scenario of a dense edge cloud, where each user can reach multiple edge nodes simultaneously. They propose an online algorithm to minimize the computation latency with energy constraints. Work [11] considers the problem of VNF placement and request scheduling to maximize the network throughput.

However, the above works ignore the following two factors, which will degrade the robustness of the edge cloud. The first factor is the **malicious users**. There are massive users in edge clouds, and the diversity of users brings more probability of malicious users [12–14]. For example, one malicious user can observe the location information of other legitimate users by monitoring the traffic of VNFs [15]. The malicious user needs to access as many VNFs to achieve high locating accuracy and even track the movements of other users. The second factor is the **failed VNFs**. Other than being attacked by malicious users, VNF failures are also widespread. According to [16], the median time between two consecutive firewall and load balancer failures is 7.5 h and 5.2 h, respectively. When a VNF is unavailable, the served requests will be rescheduled to other available VNFs, which causes a long rescheduling delay and decreases users' QoS [17].

Although the robustness of the edge cloud can be enhanced by installing security services (*e.g.*, IDS [12]) or placing redundant VNFs [17], these solutions consume additional resources. It is an unrealistic

* Corresponding authors.
E-mail addresses: fangjin98@mail.ustc.edu.cn (J. Fang), gmzhao@ustc.edu.cn (G. Zhao), xuhongli@ustc.edu.cn (H. Xu).

https://doi.org/10.1016/j.comnet.2023.109882
Received 4 April 2023; Received in revised form 11 June 2023; Accepted 15 June 2023
Available online 22 June 2023

expectation to eliminate the negative influence caused by malicious users and failed VNFs without other help. Instead, we may relax the requirement and try to **improve** the robustness of the edge cloud, such that the negative influence is limited. In this paper, we propose robust VNF placement and request scheduling, which effectively alleviates the negative influence when encountering malicious users or failed VNFs without consuming additional resources. Moreover, our proposed method can coexist with existing solutions instead of trying to substitute them, *e.g.*, [12,17].

Specifically, we consider two robustness constraints when placing VNFs and scheduling requests: **(1) User constraint:** Unlike legitimate users, malicious users may try to access as many VNFs as possible to expand the influence scope of the attacks. For example, a malicious user may use malicious programs like Bolt [18] to collect user information of the accessed VNFs, using this information to launch DoS attacks against these accessed VNFs with a high success rate. Damage caused by the malicious user increases as more VNFs is accessed. Thus, to limit the influence scope of the malicious user, each user's requests will be scheduled to a limited number of VNFs. This way, we can control the number of affected VNFs when encountering the malicious user. **(2) VNF constraint:** the failure of VNF will result in an outage to the users who have requests being served by it. The more users a VNF serves, the more significant the negative impact caused by its failure (*e.g.*, degrading the QoS of served user). As a result, to control the effect of the VNF failure on users, each VNF will serve a limited number of users. This way, we can limit the affected number of users when encountering the VNF failure. The main contributions of this paper are as follows:

1. We present Reveal and define the problem of robustness-aware VNF placement and request scheduling in edge clouds. We solve this problem through robust VNF placement and assignment, and online request scheduling to fit the practical scenario where these two phases are performed in different time scales.
2. For robust VNF placement and assignment, we propose an efficient knapsack-based rounding algorithm with the approximation factor of $O(\log|\mathcal{F}|)$, where $|\mathcal{F}|$ is the number of VNFs in the edge cloud.
3. For online request scheduling, we present a primal–dual based online algorithm with a competitive ratio of $\left[1-\epsilon, O(\log\frac{1}{\epsilon})\right]$ where $\epsilon \in (0,1)$.
4. We conduct a small-scale testbed based on Nvidia Jetson TX2s and Raspberry Pis and a large-scale simulation based on real-world datasets [19]. The results of experiments and simulations show that our algorithms achieve better performance compared with existing solutions.

The rest of this paper is organized as follows. Section 2 presents the problem statement and algorithm workflow of Reveal. In Section 3, we illustrate how to determine VNF placement and assignment. Section 4 describes how to determine request scheduling. The experimental and simulation results are presented in Section 5. The related works are presented in Section 6. We conclude this paper in Section 7.

2. Preliminary

This section first presents an application scenario, which motivates our research. Then we give the definition of network model and the problem statement of Reveal. At last, we present the algorithm workflow.

2.1. Application scenario and motivation

The degradation of user QoS may come from other users and the VNF itself. Specifically, malicious users intend to access as many VNFs to increase their influence [20]. When a VNF fails, it will affect all the users it serves. Therefore, it is crucial to consider both aspects to guarantee the robustness of edge clouds. In the following, we give an application scenario to show the superiority of limiting the impact of malicious users/failed VNFs.

An edge cloud can host many third-party services, where the user data may be leaked to malicious users or untrusted services. On one hand, 34% attacks in edge clouds are launched by authenticated users [21]. One malicious authenticated user can hijack attacked VNFs [22], monitor user information [21] or inject poisonous data [23]. On the other hand, some untrusted service providers may fail a VNF by injecting malicious program. These failed VNFs can process the request of legitimate users but also track and analyze their data for service providers [15]. Since these attacks are usually performed by authenticated users or service providers, the traditional protection methods may not classify them. We can weaken the privacy leakage by limiting the number of VNFs/users each malicious user/failed VNF can access. To this end, we design Reveal to improve the robustness of edge clouds in this paper.

2.2. System model

Infrastructure model. Let $N = \{n_1, n_2, \ldots, n_{|N|}\}$ denote the set of edge clouds. Each edge cloud $n \in N$ comprises an edge server with a processing capacity $P(n)$ and a base station covering a specified area. One user can only access services on edge clouds if only it is covered by the corresponding base stations. Moreover, a central controller is hosted in the remote cloud, managing the whole network.

Multi-user model. The service provider can place VNF instances in edge clouds, and users generate requests to access VNF instances for serving. We use B to represent the set of VNF types. For each type $b \in B$, the service provider may create multiple VNF instances to satisfy user demands. We use $\mathcal{F}_b$ to denote the set of VNFs with type b. The total set of VNFs in the network can be represented as $\mathcal{F} = \bigcup_{b\in B}\mathcal{F}_b$. For each VNF $f \in \mathcal{F}$, its processing capacity is denoted as $p(f)$ and placement cost, *e.g.*, bandwidth consumption, is denoted as $c(f)$. Let $U = \{u_1, u_2, \ldots, u_{|U|}\}$ denote the set of users. Each user generates requests for computation offloading and obtains services by sending requests to corresponding VNFs. Considering that the amount of user traffic can be forecasted by the history data [24], we use $t(u)$ to denote the estimated traffic amount of user u. The architecture is depicted in Fig. 1.

2.3. Problem statement

Our goal is to minimize the VNF placement cost while maximizing the network throughput. On one hand, we can avoid over-provision VNFs and save network resources by reducing the VNF placement cost. On the other hand, we attempt to maximize the network throughput to satisfy the needs of as many users as possible.

Furthermore, we mainly consider the following two constraints to improve the network robustness. *(1) User constraint:* Since malicious users will try to access as many VNFs as possible to expand the attack scope, it is necessary to limit the number of VNFs a malicious user can access. Thus, each user's request will be scheduled to no more than k VNFs. By limiting the number of VNFs each user assigns, we can control the impact of a malicious user. *(2) VNF constraint:* Since failed VNFs will influence the QoS of served users, it is necessary to limit the number of users a failed VNF will influence. As a result, each VNF will serve no more than p users. By limiting the number of users each VNF will serve, we can control the impact of the VNF failure. The value of k and p can be determined by the network controller adopting methods such as network measurement [25–27] and forecast [28–30].

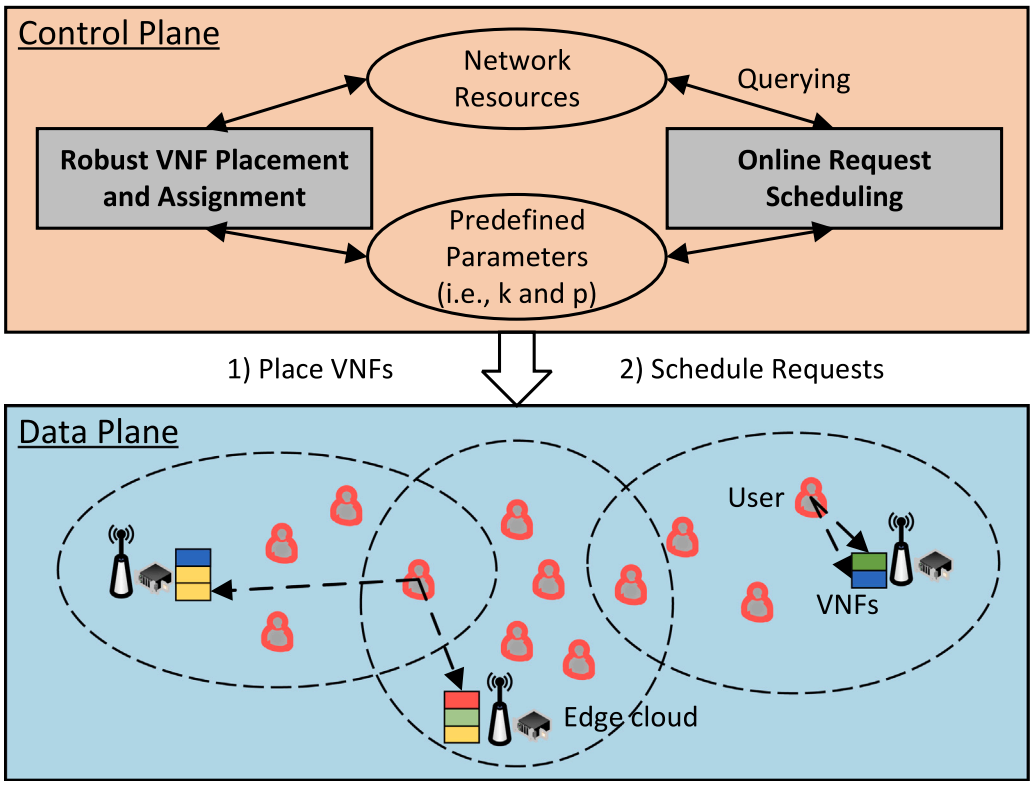


Fig. 1. System architecture and workflow of Reveal. In the first phase, Reveal determines VNF placement and assignment in a long-term intervals. In the second phase, Reveal schedules the arrival requests to VNFs.

2.4. Algorithm workflow

We find that VNF placement and request scheduling often operate on different time scales. On one hand, once a VNF is placed, it usually performs its function on a *long-term* scale. For each edge cloud, frequent creation and updating VNFs may result in enormous bandwidth consumption. Therefore, we should determine the VNF placement according to user demands at a longer interval. On the other hand, even though the total traffic of each user may be relatively stable in the long term [31], the user's requests may be generated randomly, so we should schedule the requests as soon as possible to ensure the user's QoS. This motivates us to design Reveal through two phases.

The first phase determines VNF placement based on user demands at long-term intervals (*e.g.*, hours). At the same time, we select a feasible user set for each VNF with robustness constraints, *i.e.*, VNF assignment. In this way, we ensure that the request scheduling of each user does not violate the robustness constraints. The objective of the phase is to minimize the VNF placement cost. The algorithm will be described in Section 3.3. The second phase is triggered by accidents, *e.g.*, arrival requests. At this phase, we decide how to schedule users' requests within the assigned VNFs. The objective of this phase is to maximize the network throughput. Since requests are generated dynamically, we design an online algorithm to schedule requests in Section 4.2.

3. Robust VNF placement and assignment

This section illustrates how to determine VNF placement and assignment on a long-term scale. We first formulate the Robust VNF Placement and Assignment (RVPA) problem and analyze its complexity. Then we propose a knapsack-based rounding algorithm. At last, we analyze the approximation performance of the algorithm.

Table 1
Table of notations.

Notations	Semantics
N	The set of edge clouds
B	The set of VNF types
$\mathcal{F}_b$	The set of VNFs with type b
U	The set of users
$\mathcal{F}_{u,b}$	The set of assigned VNFs with type b of user u
$\Gamma_{u,b}$	The set of requests of user u for VNFs with type b
k	The maximum number of VNFs with the same type each user can access
p	The maximum number of users each VNF can serve
$P(n)$	The processing capacity of edge cloud n
$p(f)$	The processing capacity of VNF f
$c(f)$	The placement cost of VNF f
$t(u)$	The estimated traffic amount of user u
$t(\gamma)$	The traffic amount of request γ
x_f^n	Whether edge cloud n places VNF f or not
$y_{u,f}^n$	Whether VNF f placed in edge cloud n is assigned to user u or not
$r_{u,f}^n$	The traffic proportion of user u served by VNF f placed in edge cloud n
$z_{u,\gamma}^f$	Whether request γ of user u is scheduled to VNF f or not

3.1. Problem formulation

Let $x_f^n \in \{0,1\}$ denote whether the edge cloud n places the VNF f or not. For each user u, let $y_{u,f}^n \in \{0,1\}$ denote whether the VNF f placed in edge cloud n is assigned to the user u or not. We use $r_{u,f}^n \in [0,1]$ to denote the traffic proportion of user u served by VNF f placed in edge cloud n. The notations are summaried in Table Table 1. A robust VNF placement and assignment scheme should satisfy the following constraints.

1. *Placement Constraint:* User u can obtain the service of VNF f from edge cloud n if and only if VNF f has been placed in edge cloud n. That is, $y_{u,f}^n \le x_f^n, \forall u \in U, f \in \mathcal{F}, n \in N$.

2. *Service Constraint:* The traffic of user u can be processed by VNF f if and only if it is assigned to user u, that is $r^n_{u,f} \le y^n_{u,f}, \forall u \in U, f \in \mathcal{F}, n \in N$.
3. *Access Constraint:* Similar to works [15,32–34], this paper focuses on allowing users to access services on edge computing nodes that are not within their one-hop communication range to provide real-time services. We use $\phi_u \subseteq N$ to denote the unavailable edge cloud set of user u. That means, $\sum_{n\in\phi_u} y^n_{u,f} = 0, \forall f \in \mathcal{F}, u \in U$.
4. *Instance Constraint:* Each VNF is an instance (*e.g.*, VM), and can only be placed in at most one edge cloud. *i.e.*, $\sum_{n\in N} x^n_f \le 1, \forall f \in \mathcal{F}$.
5. *User Constraint:* To reduce the impact of malicious users on the network, each user's requests with the same service requirement will be processed by k VNFs at most. That is, $\sum_{n\in N}\sum_{f\in\mathcal{F}_b} y^n_{u,f} \le k, \forall u \in U, b \in B$.
6. *VNF Constraint:* To reduce the impact of the VNF failure on the network, each VNF can serve no more than p users. That means, $\sum_{n\in N}\sum_{u\in U} y^n_{u,f} \le p, \forall f \in \mathcal{F}$.
7. *Traffic Constraint:* All traffic of each user must be served, which means $\sum_{n\in N}\sum_{f\in\mathcal{F}} r^n_{u,f} = 1, \forall u \in U$.
8. *Capacity Constraints:* All edge clouds and VNFs have capacity constraints. On one hand, the placement of the VNF needs to occupy the physical resources of the corresponding edge cloud. Thus, we must satisfy the physical resource constraint of each edge cloud, *i.e.*, $\sum_{f\in\mathcal{F}} x^n_f \cdot p(f) \le P(n), \forall n \in N$. On the other hand, to satisfy the VNF processing constraint, each VNF can only serve limited user traffic, which is $\sum_{n\in N}\sum_{u\in U} r^n_{u,f} \cdot t(u) \le p(f), \forall f \in \mathcal{F}$.

With the above constraints, the RVPA problem can be formulated as follows.

$$\min \sum_{n\in N}\sum_{f\in\mathcal{F}} x^n_f \cdot c(f)$$

$$S.t.\begin{cases} y^n_{u,f} \le x^n_f, & \forall n \in N, f \in \mathcal{F}, u \in U \\ r^n_{u,f} \le y^n_{u,f}, & \forall n \in N, f \in \mathcal{F}, u \in U \\ \sum_{n\in\phi_u} y^n_{u,f} = 0, & \forall f \in \mathcal{F}, u \in U \\ \sum_{n\in N} x^n_f \le 1, & \forall f \in \mathcal{F} \\ \sum_{n\in N}\sum_{f\in\mathcal{F}_b} y^n_{u,f} \le k, & \forall u \in U, b \in B \\ \sum_{n\in N}\sum_{u\in U} y^n_{u,f} \le p, & \forall f \in \mathcal{F} \\ \sum_{n\in N}\sum_{f\in\mathcal{F}} r^n_{u,f} = 1, & \forall u \in U \\ \sum_{f\in\mathcal{F}} x^n_f \cdot p(f) \le P(n), & \forall n \in N \\ \sum_{n\in N}\sum_{u\in U} r^n_{u,f} \cdot t(u) \le p(f), & \forall f \in \mathcal{F} \\ x^n_f, y^n_{u,f} \in \{0,1\}, & \forall n \in N, f \in \mathcal{F}, u \in U \\ r^n_{u,f} \in [0,1], & \forall n \in N, f \in \mathcal{F}, u \in U \end{cases} \tag{1}$$

The first set of inequalities means that each user can obtain the service of VNF from the edge cloud only if the corresponding VNF has been placed in the edge cloud. The second set of inequalities indicates that a user's traffic can be processed by a VNF only if it is assigned to the corresponding user. The third set of equations represents that each user can only access the services via the edge clouds close to him. The fourth set of inequalities means that each VNF can be placed in at most one edge cloud. The fifth and sixth sets of inequalities denote the user and VNF constraints as mentioned in Section 2.3, respectively. The seventh set of equations denotes that each user's traffic must be served. The eighth and ninth sets of inequalities represent the capacity constraints of edge clouds and VNFs, respectively. Our goal is to minimize the VNF placement cost.

3.2. Problem complexity analysis

The RVPA problem contains VNF placement and VNF assignment. We first consider the part of VNF placement and compare it with the generalized assignment problem [35].

Definition 1 (*Generalized Assignment Problem*)**.** Given J jobs and M capacity constrained machines. Each job has a cost when assigned to a machine. We need to assign each job to one machine, while total costs are minimized.

$$\min \sum_{n\in N}\sum_{f\in\mathcal{F}} x^n_f \cdot c(f)$$

$$S.t.\begin{cases} \sum_{n\in N} x^n_f \le 1, & \forall f \in \mathcal{F} \\ \sum_{f\in\mathcal{F}} x^n_f \cdot p(f) \le P(n), & \forall n \in N \\ x^n_f, \in \{0,1\}, & \forall n \in N, f \in \mathcal{F} \end{cases} \tag{2}$$

Compared with GAP. By removing the constraints of the VNF assignment of RVPA, we can obtain Eq. (2). We can regard VNF instances and edge clouds as jobs and machines. The processing capacity of edge clouds is like the resource constraint of machines, and the VNF placement cost is like the assignment cost of jobs. Therefore, we can say that the generalized assignment problem is a special case of the RVPA problem.

We then add both robust constraints of the VNF assignment (*i.e.*, the *user constraint* and *VNF constraint*) to Eq. (2), and show how this problem includes exceptional cases such as k-splittable routing [36] and data distribution [37] problems, demonstrating the complexity of RVPA problem.

$$\min \sum_{n\in N}\sum_{f\in\mathcal{F}} x^n_f \cdot c(f)$$

$$S.t.\begin{cases} y^n_{u,f} \le x^n_f, & \forall n \in N, f \in \mathcal{F}, u \in U \\ \sum_{n\in N} x^n_f \le 1, & \forall f \in \mathcal{F} \\ \sum_{n\in N}\sum_{f\in\mathcal{F}_b} y^n_{u,f} \le k, & \forall u \in U, b \in B \\ \sum_{n\in N}\sum_{u\in U} y^n_{u,f} \le p, & \forall f \in \mathcal{F} \\ \sum_{f\in\mathcal{F}} x^n_f \cdot p(f) \le P(n), & \forall n \in N \\ x^n_f, y^n_{u,f} \in \{0,1\}, & \forall n \in N, f \in \mathcal{F}, u \in U \end{cases} \tag{3}$$

Definition 2 (*K-Splittable Routing (KSR) Problem*)**.** Given a network with a set of flows $R = \{r_1, r_2, \ldots, r_{|R|}\}$, each of which has a traffic size $f(r)$. For each flow r, we need to determine a set of feasible paths P. Specifically, we will choose at most k paths for each path set P to minimize the maximum load factor of all links.

Compared with KSR problem. According to [36], there exists a complex rounding-based algorithm with the approximation factor of $O(\log N)$, where N is the number of links in the network. We can regard each user as a flow with traffic size $t(u)$, and choose at most k paths (placed VNFs) for each user. If we ignore the *VNF constraint*, the problem can be seen as the KSR problem.

Definition 3 (*Data Distribution (DD) Problem*)**.** Given a set of server $M = \{m_1, m_2, \ldots, m_{|M|}\}$, each with a memory size $S(m)$ and a set of document $N = \{n_1, n_2, \ldots, n_{|N|}\}$, each with a document size $s(n)$ and an access cost $c(n)$. We need to select servers for each document under the server memory size constraint with the objective of minimizing the maximum access cost among all servers.

Compared with DD problem. According to [37], there exists an algorithm achieving the optimal solution while exceeding both the access cost and memory size by at most a factor $2 \cdot (1 + \frac{1}{\phi})$, where ϕ is the maximum number of documents saved by a server. In this case, we can regard each user as a document with one unit size and access cost $t(u)$ and each VNF as a server that can only serve p documents. By ignoring the *user constraint*, we can say that the DD problem is a special case of the RVPA problem.

Theorem 1. *The RVPA problem is NP-hard.*

Proof. Since the special cases of the RVPA problem can be seen as some well-known NP-hard problems, our RVPA problem is NP-hard as well. □

The preceding analysis shows that the RVPA problem is much more difficult than KSP and DD. Thus, It is far from trivial to design an algorithm with bounded approximation factors to solve the RVPA problem.

3.3. Algorithm design

By using the traditional randomized rounding algorithm [38] to solve RVPA, we can achieve approximation factors of $(O(\log|\mathcal{F}|), O(\log|\mathcal{F}|), O(\log|N|), O(\log|\mathcal{F}|))$, which represent the maximum exceeded factors of user constraint, VNF constraint, edge cloud capacity constraints, and VNF capacity constraints, respectively. To improve the approximation performance, we present a Knapsack-based Rounding for VNF Placement and Assignment (KVPA) algorithm to solve RVPA in polynomial time. It can achieve approximation factors of $(1, O(\log|\mathcal{F}|), O(\log|N|), O(\log|\mathcal{F}|))$. *Under proper assumptions, the bounds can be tightened to* $(1, 4, 2, 2)$. It means KVPA can strictly satisfy the user constraint and exceed the VNF constraint, edge cloud capacity constraints and VNF capacity constraints at most by a factor of 4, 2 and 2, respectively, in most practical scenarios, which will be proved in Section 3.4.

Our algorithm consists of three steps. The first step relaxes Eq. (1) to a linear program by replacing $\{x_f^n\}, \{y_{u,f}^n\}$ with their fractional versions. We can solve it with a linear program solver (*e.g.*, PULP [39]) and the optimal solutions are denoted as $\{\widetilde{x}_f^n\}$, $\{\widetilde{y}_{u,f}^n\}$ and $\{\widetilde{r}_{u,f}^n\}$. After that, KVPA places VNF in edge clouds based on the optimal solutions. Each VNF f is placed in edge cloud n with probability $\widetilde{x}_f^n$. Let n_f represent the edge cloud where VNF f is placed. In the third step, KVPA assigns VNFs to users. For each user u, we first calculate $k(u,b) = \left\lfloor \sum_{f\in\mathcal{F}_b} \widetilde{y}_{u,f}^{n_f} \right\rfloor$, which is the required number of VNFs with type b. Then KVPA puts variables $\widetilde{y}_{u,f}^{n_f}$ $(\forall f \in \mathcal{F}_b)$ into $k(u,b)$ knapsacks with min–max sum. For each knapsack a, VNF f will be assigned to user u with probability $\frac{\widetilde{y}_{u,f}^{n_f}}{S_a \cdot \widetilde{x}_f^{n_f}}$, where S_a is the sum of $\widetilde{y}_{u,f}^{n_f}$ in knapsack a. After KVPA ends, VNFs are placed in edge clouds and assigned to users. The KVPA algorithm is summarized in Alg. 1.

Note that, after we put variables into knapsacks with the objective of min–max sum, S_a usually equals approximately 1. According to the fourth and fifth constraints in Eq. (1), we can see that $\widetilde{y}_{u,f}^{n_f}$ is generally much smaller than $\widetilde{x}_f^{n_f}$. Thus, we believe $\frac{\widetilde{y}_{u,f}^{n_f}}{S_a \cdot \widetilde{x}_f^{n_f}} \le 1$.

Algorithm 1 KVPA: Knapsack-based Rounding for VNF Placement and Assignment

Step 1: Solving the Relaxed Problem
Construct a LP by replacing with $x_f^n, y_{u,f}^n \in [0,1]$.
Obtain the optimal solutions $\{\widetilde{x}_f^n\}$, $\{\widetilde{y}_{u,f}^n\}$ and $\{\widetilde{r}_{u,f}^n\}$.
Step 2: VNF Placement
for each VNF $f \in \mathcal{F}$ **do**
  Choose edge cloud n to place VNF f with probability $\widetilde{x}_f^n$
  Let n_f denote the edge cloud which places VNF f
Step 3: VNF Assignment
for each user $u \in U$ **do**
  for each VNF type $b \in B$ **do**
    Let $k(u,b) = \left\lfloor \sum_{f\in\mathcal{F}_b} \widetilde{y}_{u,f}^{n_f} \right\rfloor$
    Put $\widetilde{y}_{u,f}^{n_f}$ where $f \in \mathcal{F}_b$ into $k(u,b)$ knapsacks with min–max sum
    for each knapsack a **do**
      Let $\mathbb{A}$ denote the variables in knapsack a
      Calculate $S_a = \sum_{\widetilde{y}_{u,f}^{n_f}\in\mathbb{A}} \widetilde{y}_{u,f}^{n_f}$
      Choose f for $\widetilde{y}_{u,f}^{n_f} \in \mathbb{A}$ with probability $\frac{\widetilde{y}_{u,f}^{n_f}}{S_a \cdot \widetilde{x}_f^{n_f}}$ and set $\widehat{y}_{u,f}^{n_f} = 1$ for chosen VNF f.
      Set the traffic proportion of user u processed by VNF f to $\widehat{r}_{u,f}^{n_f} = \frac{\widetilde{r}_{u,f}^{n_f} \cdot S_a \cdot \widetilde{x}_f^{n_f}}{\widetilde{y}_{u,f}^{n_f}}$
      Assign the VNFs with $\widehat{y}_{u,f}^{n_f} = 1$ for user u

3.4. Performance analysis

Theorem 2. *KVPA can strictly guarantee that each user $u \in U$ will be assigned at most k VNFs, i.e., we can strictly guarantee the user constraint.*

Proof. Each user $u \in U$ is assigned with VNFs from $k(u,b)$ knapsacks, and only one VNF is selected in each knapsack. Thus, there are $k(u,b)$ VNFs selected in total. According to user constraint in Eq. (1) and the definition of $k(u,b)$, we have:

$$k(u,b) = \left\lfloor \sum_{f\in\mathcal{F}_b} \widetilde{y}_{u,f}^{n_f} \right\rfloor \le \sum_{n\in N} \sum_{f\in\mathcal{F}_b} \widetilde{y}_{u,f}^{n} \le k \tag{4}$$

It shows that $k(u,b) \le k$, *i.e.*, the user constraint is strictly guaranteed. □

Lemma 3. *For each knapsack a, the lower bound of S_a is greater than 0.5.*

Proof. We first prove that the lower bound of S_a is greater than 0.5. By the definition of $k(u,b)$, we have:

$$\sum_{f\in\mathcal{F}_b} \widetilde{y}_{u,f}^{n_f} = k(u,b) + \varepsilon, \forall u \in U, b \in B \tag{5}$$

Then, we define two sets as follows:

$$\begin{cases} \mathcal{Y}_1 = \left\{ \widetilde{y}_{u,f}^{n_f} | 0.5 < \widetilde{y}_{u,f}^{n_f} < 1, f \in \mathcal{F}_b \right\} \\ \mathcal{Y}_2 = \left\{ \widetilde{y}_{u,f}^{n_f} | 0 < \widetilde{y}_{u,f}^{n_f} < 0.5, f \in \mathcal{F}_b \right\} \end{cases} \tag{6}$$

Supposing that we select two variables denoted as $y_{u,1}^{n_1}$ and $y_{u,2}^{n_2}$ from $\mathcal{Y}_2$ randomly. The value of $y_{u,3}^{n_3} = y_{u,1}^{n_1} + y_{u,2}^{n_2}$ is either greater than 0.5 or less than 0.5. If $y_{u,3}^{n_3} > 0.5$, then $\mathcal{Y}_1 = \mathcal{Y}_1 + y_{u,3}^{n_3}$ and $\mathcal{Y}_2 = \mathcal{Y}_2 - \left\{ y_{u,1}^{n_1}, y_{u,2}^{n_2} \right\}$. Otherwise, $\mathcal{Y}_2 = \mathcal{Y}_2 - \left\{ y_{u,1}^{n_1}, y_{u,2}^{n_2} \right\} + y_{u,3}^{n_3}$. We repeat the above operations until $|\mathcal{Y}_2| \le 1$. Supposing that there is one variable in $\mathcal{Y}_2$, there are at most $k(u,b) - 1$ variables in $\mathcal{Y}_1$. According to the definition of $\mathcal{Y}_1$, the

value of variables in $\mathcal{Y}_1$ is all less than 1. Thus, we have:

$$\sum_{\tilde{y}^{n_f}_{u,f}\in\mathcal{Y}_1} \tilde{y}^{n_f}_{u,f} < k(u,b) - 1 \tag{7}$$

$$\sum_{\tilde{y}^{n_f}_{u,f}\in\mathcal{Y}_2} \tilde{y}^{n_f}_{u,f} < 0.5 \tag{8}$$

Combining Eqs. (7) and (8), we have:

$$\sum_{f\in F_b} \tilde{y}^{n_f}_{u,f} < k(u,b) - 0.5 \tag{9}$$

However, Eq. (9) contradicts Eq. (5). Thus, at least $k(u,b)$ variables are in $\mathcal{Y}_1$. Since we put variables to knapsacks with min–max sum, the sum $\mathcal{S}_a$ of knapsack a must be greater than 0.5. □

In order to facilitate the description of approximation analysis, we now give one famous lemma for analysis.

Lemma 4 (*Chernoff Bound [40]*). *Given n independent variables: $y_1, y_2, \ldots, y_n, \forall y_i \in [0,1]$. Let $\tau = \mathbb{E}\left[\sum_{i=1}^{n} y_i\right]$. Then,* $\Pr\left[\sum_{i=1}^{n} y_i \geq (1+\varrho)\tau\right] \leq e^{\frac{-\varrho^2\tau}{2+\varrho}}$, *where ϱ is an arbitrary positive value.*

Theorem 5. *KVPA guarantees that the number of users each VNF can serve will not exceed p by a factor of $O(\log|\mathcal{F}|)$, where $|\mathcal{F}|$ is the number of VNFs in the network. Under the proper assumption, the bound can be tightened to 4.*

Proof. VNF $f \in \mathcal{F}_b$ can provide service to user $u \in U$ if and only if it is placed in an edge cloud and assigned to user u. Since each VNF $f \in \mathcal{F}_b$ is placed in edge cloud n_f with probability $\frac{\tilde{x}^{n_f}_f}{z_a}$. Each user $u \in U$ will be assigned with VNF f with probability $\frac{\tilde{y}^{n_f}_{u,f}}{\mathcal{S}_a \cdot \tilde{x}^{n_f}_f}$. Thus we have $\mathbb{E}\left[\hat{y}^{n_f}_{u,f}\right] = \frac{\tilde{y}^{n_f}_{u,f}}{\mathcal{S}_a}$. According to Lemma 3, we have:

$$\mathbb{E}\left[\hat{y}^{n_f}_{u,f}\right] = \frac{\tilde{y}^{n_f}_{u,f}}{\mathcal{S}_a} \leq 2 \cdot \tilde{y}^{n_f}_{u,f} \tag{10}$$

We define a constant value $\vartheta = p$, which means the most number of users that each VNF can serve. According to the algorithm, each VNF f will be assigned to user u ($y^{n_f}_{u,f} = 1$), or not ($y^{n_f}_{u,f} = 0$), so we define $w^{n_f}_{u,f} \in \{0,1\}$, where $w^{n_f}_{u,f} = 1$ with probability of $\frac{\tilde{y}^{n_f}_{u,f}}{\mathcal{S}_a}$. It is obviously that variables $w^{n_f}_{u,f}$ are independent. The expected number of users that VNF f provide service to satisfies:

$$\mathbb{E}\left[\sum_{u\in U} w^{n_f}_{u,f}\right] = \sum_{u\in U} \frac{\tilde{y}^{n_f}_{u,f}}{\mathcal{S}_a} \leq 2 \cdot \sum_{u\in U} \tilde{y}^{n_f}_{u,f} \leq 2 \cdot p \tag{11}$$

Combining Eq. (11) and the definition of ϑ, we have:

$$\begin{cases} \frac{w^{n_f}_{u,f}\cdot\vartheta}{2\cdot p} \in [0,1] \\ \mathbb{E}\left[\sum_{u\in U} \frac{w^{n_f}_{u,f}\cdot\vartheta}{2\cdot p}\right] \leq \vartheta \end{cases} \tag{12}$$

By applying Lemma 4, we have:

$$\Pr\left[\sum_{u\in U} \frac{w^{n_f}_{u,f}\cdot\vartheta}{2\cdot p} \geq (1+\varrho)\cdot\vartheta\right] \leq e^{\frac{-\varrho^2\vartheta}{2+\varrho}}$$

$$\Rightarrow \Pr\left[\sum_{u\in U} \frac{w^{n_f}_{u,f}}{2\cdot p} \geq (1+\varrho)\right] \leq e^{\frac{-\varrho^2\vartheta}{2+\varrho}} \tag{13}$$

where ϱ is an arbitrary positive value.

Now, we assume that:

$$\Pr\left[\sum_{u\in U} \frac{w^{n_f}_{u,f}}{2\cdot p} \geq (1+\varrho)\right] \leq e^{\frac{-\varrho^2\vartheta}{2+\varrho}} \leq \frac{1}{|\mathcal{F}|} \tag{14}$$

When the network grows, $\frac{1}{|\mathcal{F}|}$ approaches to zero. By solving Eq. (14), we have:

$$\varrho \geq \frac{\log|\mathcal{F}| + \sqrt{\log^2|\mathcal{F}| + 8\vartheta\log|\mathcal{F}|}}{2\vartheta}, (|\mathcal{F}| \geq 2)$$

$$\Rightarrow \varrho \geq \frac{\log|\mathcal{F}|}{\vartheta} + 2, (|\mathcal{F}| \geq 2) \tag{15}$$

In practice, there exist thousands of VNFs in the edge cloud, according to the definition of ϑ, we can assume that $\vartheta \geq 3\cdot\log|\mathcal{F}|$. Under this assumption, we have:

$$\varrho \geq \frac{\log|\mathcal{F}| + \sqrt{\log^2|\mathcal{F}| + 8\vartheta\log|\mathcal{F}|}}{2\vartheta}$$

$$\Rightarrow \varrho \geq \frac{\log|\mathcal{F}| + \sqrt{(2\vartheta - \log|\mathcal{F}|)^2}}{2\vartheta} \Rightarrow \varrho \geq 1 \tag{16}$$

Thus, the approximate factor of VNF constraint is $2\cdot(\varrho+1) = \frac{2\cdot\log|\mathcal{F}|}{\vartheta} + 6 = O(\log|\mathcal{F}|)$. Under the proper assumption (*i.e.*, $\alpha \geq 3\cdot\log|\mathcal{F}|$), the bound can be tightened to $2\cdot(\varrho+1) = 4$. □

Theorem 6. *KVPA will not exceed the edge cloud capacity constraint and VNF capacity constraint by approximation factors of $O(\log|N|)$ and $O(\log|\mathcal{F}|)$, respectively. Under proper assumptions, the bound can all be tightened to 2.*

Proof. Since we place VNF f in edge cloud n with probability of $\frac{\tilde{x}^n_f}{z_a}$, we have $\mathbb{E}\left[\hat{x}^n_f\right] = \frac{\tilde{x}^n_f}{z_a}$. We use φ^n_f to denote the process resource required by VNF f in edge cloud n. So the expected process resource needed by edge cloud n is

$$\begin{aligned} \mathbb{E}\left[\sum_{f\in\mathcal{F}} \varphi^n_f\right] &= \mathbb{E}\left[\sum_{f\in\mathcal{F}} \hat{x}^n_f \cdot p(f)\right] = \sum_{f\in\mathcal{F}} \mathbb{E}\left[\hat{x}^n_f \cdot p(f)\right] \\ &= \sum_{f\in\mathcal{F}} \frac{\tilde{x}^n_f \cdot p(f)}{z_a} \leq 2\cdot P(n) \end{aligned} \tag{17}$$

We define a constant value ν as follows:

$$\nu = \min\{\frac{2\cdot P^{\min}_n}{p(f)}, f \in \mathcal{F}\} \tag{18}$$

similar to Theorem 6, $P^{\min}_n$ denotes the minimum capacity among edge cloud $n \in N$.

Combining the definition of ν, we have:

$$\begin{cases} \frac{\varphi^n_f\cdot\nu}{2\cdot P(n)} \in [0,1] \\ \mathbb{E}\left[\sum_{f\in\mathcal{F}} \frac{\varphi^n_f\cdot\nu}{2\cdot P(n)}\right] \leq \nu \end{cases} \tag{19}$$

Similar to the proof of Theorem 5, by using Lemma 4, we have:

$$\varrho \geq 2\cdot\frac{\log|N|}{\nu} + 2, |N| \geq 2 \tag{20}$$

where $|N|$ is the number of edge clouds in the network.

As a result, the approximate factor for *edge cloud capacity constraint* is $\varrho + 1 = \frac{\log|N|}{\nu} + 3 = O(\log|N|)$.

For each VNF, we compute $\hat{r}^{n_f}_{u,f} = \frac{\tilde{r}^{n_f}_{u,f}\cdot\mathcal{S}_a}{\tilde{y}^{n_f}_{u,f}}$ with probability of $\frac{\tilde{y}^{n_f}_{u,f}}{\mathcal{S}_a}$, we have $\mathbb{E}\left[\hat{r}^n_{u,f}\right] = \tilde{r}^n_{u,f}$.

We use $\phi^f_{u,b}$ to denote the traffic amount of user $u \in U$ handled by VNF $f \in \mathcal{F}_b$. The expectation of traffic load of VNF f is:

$$\begin{aligned} &\mathbb{E}\left[\sum_{u\in U} \phi^n_{u,f}\right] = \sum_{u\in U} \mathbb{E}\left[\phi^n_{u,f}\right] \\ &= \sum_{u\in U} \tilde{r}^n_{u,f} \cdot t(u,b) \leq p(f) \end{aligned} \tag{21}$$

The equation above shows that our algorithm can guarantee that the expectation of traffic load on VNF f will not exceed its process capacity $p(f)$. We define a constant value ν as follows:

$$\nu = \min\{\frac{p_f^{\min}}{t(u,b)}, u \in U\} \tag{22}$$

where $p_f^{\min}$ denotes the minimum capacity among VNF $f \in \mathcal{F}_b$.

Similar to the proof of Theorem 5, by using Lemma 4, we have:

$$\varrho \geq \frac{\log|\mathcal{F}|}{\nu} + 2, |\mathcal{F}| \geq 2 \tag{23}$$

where $|\mathcal{F}|$ is the number of VNF in the network.

Thus, we can conclude that the approximate factor for *VNF capacity constraint* is $\varrho + 1 = \frac{\log|\mathcal{F}|}{\nu} + 3 = O(\log|\mathcal{F}|)$. □

Theorem 7. *After rounding, the VNF placement cost will not exceed the fractional solution by a factor of $O(\log|\mathcal{F}|)$. Under the proper assumption, we can tighten the bound to 2.*

Proof. According to Alg. 1, we place VNF $f \in \mathcal{F}$ in edge cloud $n \in N$ with probability of $\tilde{x}_f^n$, thus we have $\mathbb{E}\left[\hat{x}_f^n\right] = \tilde{x}_f^n$. The expected VNF placement cost can be calculated as:

$$\mathbb{E}\left[\sum_{n\in N}\sum_{f\in F}\hat{x}_f^n \cdot c(f)\right] = \sum_{n\in N}\sum_{f\in F}\tilde{x}_f^n \cdot c(f)$$

Since the following proof is similar to Theorem 5, we omit the detailed proof here. □

4. Online request scheduling

In this section, we first give the formulation of the Online Request Scheduling (ORS) problem. Then we propose an online algorithm based on primal–dual method. At last, we analyze its approximation performance and present the competitive ratio of the proposed algorithm.

4.1. Problem formulation

Considering VNFs with type b, we use $\mathcal{F}_{u,b} \subseteq \mathcal{F}_b$ to represent the assigned VNF set of user u, which is determined in Section 3. Let $\Gamma_u^b = \{\gamma_{u,1}^b, \gamma_{u,2}^b, \ldots, \gamma_{u,d}^b\}$ denote the set of requests generated by user u that need to be served by VNFs of type b and $t(\gamma)$ represent the traffic amount associated with request γ. We use variable $z_{u,\gamma}^f \in \{0,1\}$ to denote whether the request γ of user u is scheduled to VNF f or not. With these notations, we formulate ORS as follows:

$$\max \sum_{u\in U}\sum_{\gamma\in\Gamma_u^b}\sum_{f\in\mathcal{F}_b} I \cdot z_{u,\gamma}^f \cdot t(\gamma)$$

$$S.t.\begin{cases}\sum_{f\in\mathcal{F}_b} I z_{u,\gamma}^f \leq 1, & \forall u\in U, \gamma\in\Gamma_u^b \\ \sum_{u\in U}\sum_{\gamma\in\Gamma_u^b} I z_{u,\gamma}^f t(\gamma) \leq p(f), & \forall f\in\mathcal{F}_b \\ z_{u,\gamma}^f \in \{0,1\}, & \forall u\in U, \gamma\in\Gamma_u^b, f\in\mathcal{F}_b\end{cases} \tag{24}$$

where $I \in \{0,1\}$ is the abbreviation of $I(f,u)$.

I is a constant indicating that whether VNF f is assigned to user u or not. The first set of inequalities denotes that at most one VNF is selected for each request. The second set of inequalities indicates that the workload of VNF f should not exceed its process capacity. Our goal is to maximize the network throughput.

Algorithm 2 PDRS: Primal–Dual based Request Scheduling

1: **Step 1: Variable Initialization**
2: Initialize all the dual variables
3: $\alpha_{u,\gamma} \leftarrow 0, \forall u \in U, \gamma \in \Gamma_u^b$
4: $\beta_f \leftarrow 0, \forall f \in \mathcal{F}_b$
5: **Step 2: Online Request Scheduling**
6: **for** each arrival request γ of user u **do**
7: Calculate the cost of all candidate VNFs in $\mathcal{F}_{u,b}$
8: $f^* \leftarrow \arg\min K_f$
9: **if** $K_{f^*} < 1$ **then**
10: Schedule request γ to VNF f^*
11: Update dual variables according to Eq. (27)
12: **else**
13: Reject request γ and set $\alpha_{u,\gamma} \leftarrow 0$

4.2. Algorithm design

To solve the ORS problem, we design an online algorithm called Primal–Dual based Request Scheduling (PDRS). We first construct the dual problem for the linear relaxation of Eq. (24). Let $\alpha_{u,\gamma}$, β_f represent the dual variables of the first and second sets of inequalities, respectively. Note that all dual variables are non-negative. The dual version of the ORS problem can be formulated as:

$$\min \sum_{u\in U}\sum_{\gamma\in\Gamma_u^b}\alpha_{u,\gamma} + \sum_{f\in\mathcal{F}_b}\beta_f \cdot p(f)$$

$$S.t.\begin{cases} I\alpha_{u,\gamma} + I\beta_f t(\gamma) \geq I t(\gamma), & \forall u\in U, \gamma\in\Gamma_u^b, f\in\mathcal{F}_b \\ \alpha_{u,\gamma}, \beta_f \geq 0, & \forall u\in U, \gamma\in\Gamma_u^b, f\in\mathcal{F}_b\end{cases} \tag{25}$$

We can rewrite the first set of inequalities of Eq. (25) as:

$$I \cdot \alpha_{u,\gamma} \geq I \cdot t(\gamma)(1-\beta_f), \forall u\in U, \gamma\in\Gamma_u^b, f\in\mathcal{F}_b \tag{26}$$

For each arrival request of user u, the algorithm needs to select a VNF $f \in \mathcal{F}_{u,b}$ to handle it. In this way, the selected VNF f satisfies $I = 1$. When a request arrives, the algorithm calculates the cost of each candidate VNF $f \in \mathcal{F}_{u,b}$, which is defined as $K_f = \beta_f$. PDRS selects the VNF f^* with the lowest cost, denoted by K_{f^*}, for the arrival request. Since $I = 1$, if $K_{f^*} > 1$, then $1 - K_{f^*} < 0$ violating the second set of constraints in Eq. (25). Thus, the arrival request will be rejected and the dual variable β_f will be set as 0 when $K_{f^*} \geq 1$. Otherwise, the arrival request will be accepted and scheduled to VNF f^*. The algorithm updates the dual variables as follows:

$$\begin{cases}\alpha_{u,\gamma} = t(\gamma)(1 - K_{f^*}) \\ \beta_f = \beta_f(1 + \frac{t(\gamma)}{p(f)}) + \frac{t(\gamma)}{\frac{1}{\epsilon}\cdot p(f)}\end{cases} \tag{27}$$

The PDRS algorithm is described in Alg. 2.

4.3. Performance analysis

Lemma 8. *When PDRS ends, the constraints of the dual problem, i.e., Eq. (25) will not be violated.*

Proof. We first consider the positivity constraint (*i.e.*, the second set of inequalities) of Eq. (25). In the beginning, we set all dual variables as 0, which satisfies the positivity constraint. For variables β_f, they will keep increasing. For variables $\alpha_{u,\gamma}$, according to Line 9 of Alg. 2, they will be updated only if $K_{f^*} < 1$, which means $\alpha_{u,\gamma} = t(\gamma)(1 - K_{f^*}) > 0$. Thus, all dual variables will not violate the positivity constraint after the algorithm ends. Then, let us consider the first set of constraints in

Eq. (25). For each arrival request, we will select the VNF f with the lowest cost K_{f^*}. If $K_{f^*} \geq 1$, the right side of Eq. (26) will be non-positive. Since we have proved that $\alpha_{u,\gamma}$ is non-negative, Eq. (26) is satisfied. If the arrival request is scheduled to VNF f^*, according to the update rules in Eq. (27) and the definition of K_{f^*}, we have:

$$\alpha_{u,\gamma} = t(\gamma)(1 - K_{f^*}) \geq t(\gamma)(1 - \beta_f) \tag{28}$$

The inequality above is the same as the first set of constraints in Eq. (25). Thus, the first set of constraints in Eq. (25) will not be violated after updating of dual variable $\alpha_{u,\gamma}$. Moreover, the update of dual variables β_f is accumulated, making the right side of the inequality above smaller, which satisfies the constraint. As a result, the algorithm will not violate the constraints in Eq. (25). □

To evaluate the performance of the proposed algorithm, we give the definition of competitive ratio as follows [41].

Definition 4. An online algorithm is $[\zeta, \eta]$ competitive if it achieves at least $\zeta \cdot OPT$, where OPT is the result of the optimal solution, and the constraints are violated by a factor of η at most.

In the following, we will prove that PDRS has a competitive ratio of $\left[1 - \epsilon, O(\log \frac{1}{\epsilon})\right]$.

Lemma 9. *PDRS can reach a throughput of at least $(1 - \epsilon) \cdot OPT$, where OPT is the result of the optimal solution.*

Proof. When request γ is accepted, the objective value of Eq. (24) increases by $t(\gamma)$. However, for the dual program in Eq. (25), its objective value increases by Δ:

$$\begin{aligned}\Delta =& t(\gamma)(1 - K_{f^*}) + (\beta_f \cdot \frac{t(\gamma)}{p(f)} + \frac{\epsilon \cdot t(\gamma)}{p(f)}) \cdot p(f) \\ =& (1 + \epsilon) \cdot t(\gamma)\end{aligned} \tag{29}$$

That is, PDRS increases the objective value of the dual program by $(1+\epsilon) \cdot t(\gamma)$. Therefore, the overall objective value of the dual program is at least $1/(1 + \epsilon) \geq (1 - \epsilon)$ times as that of the optimal solution, *i.e.*, the throughput of our primal–dual algorithm is at least $(1 - \epsilon) \cdot OPT$. □

Then, we will prove that the violation of the capacity constraint on each VNF will not exceed by a factor of $O(\log \frac{1}{\epsilon})$. To prove this, we use $L(f, k)$ to denote the load on VNF f after request γ_k has been scheduled to f.

Lemma 10. *For each request γ_k, we have:*

$$\beta_{f,k} \geq \epsilon \cdot (\exp(L(f, k)/p(f)) - 1) \tag{30}$$

where $\beta_{f,k}$ is the value of β_f after request γ_k is accepted.

Proof. We prove the lemma by the induction of request $\gamma_k (k = 1, 2, \ldots, |\Gamma_u^b| - 1)$. Apparently, for each VNF $f \in \mathcal{F}_b$, $\beta_{f,k}$ and $L(f, 0)$ are zero at the beginning. Therefore, the inequality above holds. We first consider the situation where request γ is rejected. When request γ is rejected, VNF's workload will not increase, so $L(f, k) = L(f, k - 1)$. According to Alg. 2, $\beta_{f,k} = \beta_{f,k-1}$ either. The lemma holds. Then we consider the situation where request γ_k is accepted. For $L(f, k)$, it will be updated as $L(f, k) = L(f, k - 1) + t(\gamma)$. According to Alg. 2, $\beta_{f,k}$ will be updated as:

$$\beta_{f,k} = \beta_{f,k-1}(1 + \frac{t(\gamma)}{p(f)}) + \frac{\epsilon \cdot t(\gamma)}{p(f)} \tag{31}$$

By induction hypothesis, we apply inequality $\beta_{f,k-1} \geq \epsilon \cdot (\exp(\frac{L(f,k-1)}{p(f)}) - 1)$ to Eq. (31) as follows:

$$\begin{aligned}\beta_{f,k} &= \epsilon \cdot \left[\exp(L(f, k - 1)/p(f)) \cdot (1 + t(\gamma)/p(f)) - 1\right] \\ &\approx \epsilon \cdot \left[\exp(L(f, k - 1)/p(f)) \cdot \exp(t(\gamma)/p(f)) - 1\right] \\ &= \epsilon \cdot (\exp(L(f, k)/p(f)) - 1)\end{aligned} \tag{32}$$

Here we apply the first order approximation $exp(x) \approx 1 + x$ for a small positive value x. Strict inequality can be established by a more complicated update rule and leads to unnecessary complexity. As a result, the lemma holds. □

Lemma 11. *PDRS algorithm will not violate the capacity constraint by a factor of $O(\log \frac{1}{\epsilon})$ on each VNF.*

Proof. Without loss of generality, we consider the capacity constraint violation for each VNF $f \in \mathcal{F}_b$. According to Alg. 2 and Eq. (27), the value of β_f will be changed only if $K_{f^*} < 1$. Since $K_{f^*} = \beta_f$, we have $\beta_f < 1$ before the last update. Next, we discuss the last update of β_f. From the second set of equations in Eq. (27), we have $\beta_f \leq 1 + \frac{t(\gamma)}{p(f)} + \frac{\epsilon \cdot t(\gamma)}{p(f)} \leq 3$. By applying Eq. (30) in Lemma 10, we can obtain:

$$\frac{L(f, k)}{p(f)} \leq \log(\frac{3}{\epsilon} + 1) = O(\log \frac{1}{\epsilon}) \tag{33}$$

That means the load on VNF f will exceed its capacity by a factor of $O(\log \frac{1}{\epsilon})$ at most once a request is rejected. □

By combining Lemmas 9 and 11, we present the competitive ratio of Alg. 2 as follows.

Theorem 12. *The proposed algorithm has a competitive ratio of $\left[1 - \epsilon, O(\log \frac{1}{\epsilon})\right]$ with $\epsilon \in (0, 1)$.*

5. Performance evaluation

In this section, we compare Reveal with state-of-the-art solutions. We first give the metrics and benchmarks for performance comparison. Then, we construct a small-scale testbed with Nvidia Jetson Tx2s and Raspberry Pis to test the efficiency of the proposed algorithms. Finally, to complement the small-scale testbed experiments, we perform simulations to show the theoretical performance of Reveal in large-scale scenarios.

5.1. Performance metrics and benchmarks

Metrics. We adopt the following performance metrics in evaluations. The first metric is the maximum number of affected VNFs. We calculate the number of VNFs each user assigns and record the largest value as the maximum number of affected VNFs when encountering the malicious user. The second metric is the maximum number of affected users. We calculate the number of users each VNF serves and record the largest value as the maximum number of affected users when encountering the failed VNF. The third and fourth metrics are packet loss rates and round-trip time (RTT), which can be measured by the typing tool [42]. The fifth and sixth metrics are the request completion time and the 99%tile request completion time, which can be measured by the Iperf tool [43]. The seventh metric is the average CPU utilization of each edge cloud. We run command *top* on each edge cloud to monitor the CPU utilization during the experiments. The eighth metric is the network throughput, *i.e.*, the total traffic of accepted requests.

Benchmarks. We compare Reveal with three benchmarks. The first benchmark is a two-phase solution for joint VNF placement and request scheduling, called GSP-SS [44]. GSP-SS first decides the VNF placement based on the greedy heuristic with shadow request scheduling. Then it decides the request scheduling based on the maximum flow algorithm. Experiments show that it can achieve near-optimal performance. The second benchmark is a randomized-rounding based algorithm called SPR3 [11]. The objective of SPR3 is to maximize the network throughput. Unlike our algorithm, SPR3 considers the VNF placement and request scheduling simultaneously and can achieve close-to-optimal

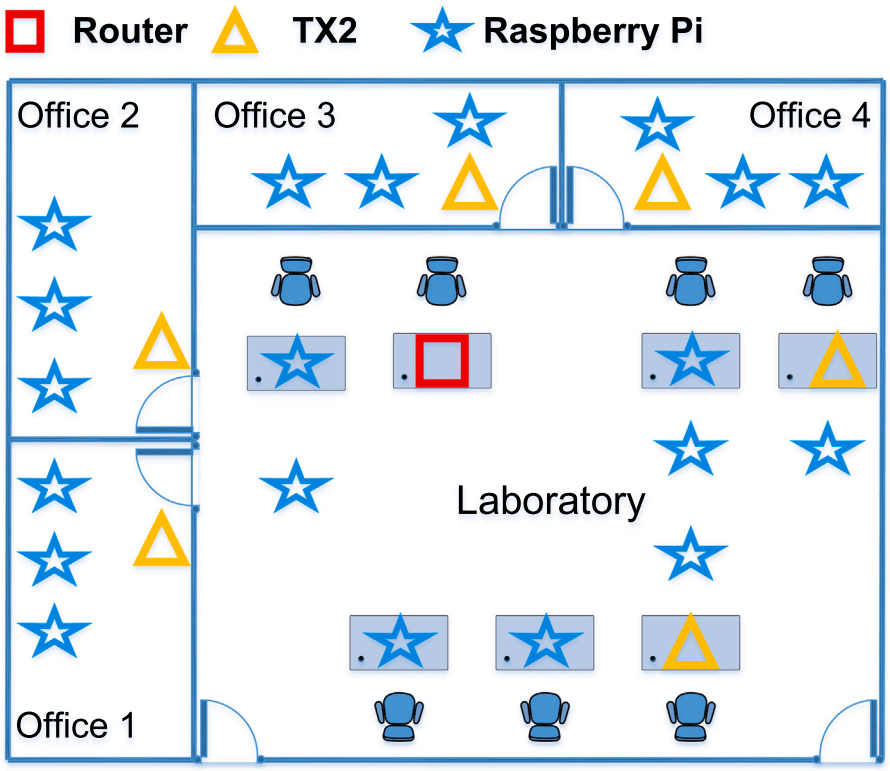


Fig. 2. The topology of testbed.

performance. The last benchmark is a reliability-aware adaptive VNF placement scheme called RAD [45], which guarantees the availability of VNFs by deciding the backup scheme of VNFs while minimize the backup costs. Specifically, RAD determines the placement of VNFs and additional backups for the given user request set. For fair evaluation, we schedule the user requests to VNFs with minimum loads, which can achieve near-optimal performance for the traditional VNF placement problem [46].

5.2. Testbed evaluation

Settings. We use 6 Nvidia Jetson Tx2s equipped with ARM Cortex-A57 and 20 Raspberry Pis equipped with ARM Cortex-A72 to build a small-scale testbed. Specifically, Nvidia Jetson Tx2s act as the edge clouds to place VNFs, and Raspberry Pis act as the users to send requests. The experimental network is established via a router, where Nvidia Jetson Tx2s are directly connected to the router by Ethernet while Raspberry Pis are accessed via wireless link. The network topology is depicted as Fig. 2 We choose video cache as the VNF to provide services. Each user obtains different videos from VNFs. If they are not cached in the VNFs, they will be downloaded from the cloud and sent to the users. We implement our tests with requests from Google cluster data [19]. We use Iperf tool [43] to generate requests and send them to corresponding VNFs. The default robustness constraints are set to $k = 6$, and $p = 6$.

Overall performance comparison. This set of experiments evaluates the influence scope (*i.e.*, the maximum number of affected VNFs and users) of abnormal situations of different algorithms. Figs. 3–4 show that as the number of requests increases, Reveal obtains the least influence scope compared with other benchmarks. For example, in Fig. 3, Reveal reduces the maximum number of affected VNFs by 57% on average compared with others. The reason is that Reveal takes the robustness constraints into consideration, which can reduce the influence scope of abnormal events.

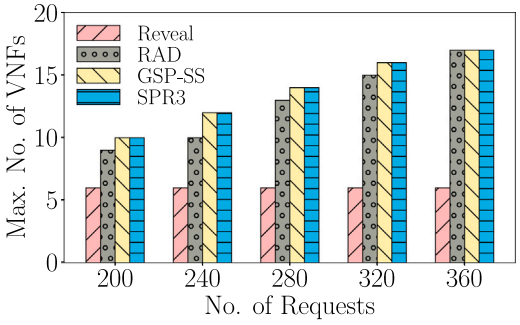


Fig. 3. Max. no. of affected VNFs vs. no. of requests.

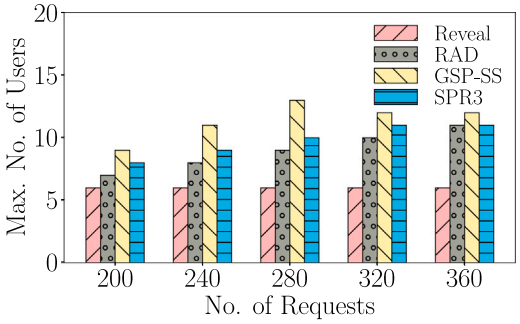


Fig. 4. Max. no. of affected users vs. no. of requests.

Comparison with the malicious user event. In this set of experiments, we conduct the experiments under the scenario when encountering a malicious user to fully illustrate that limiting the number

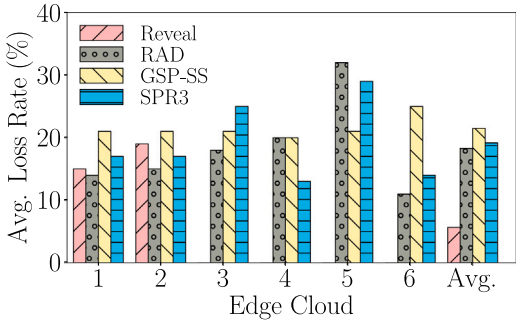


Fig. 5. Average packet loss rate of each edge cloud.

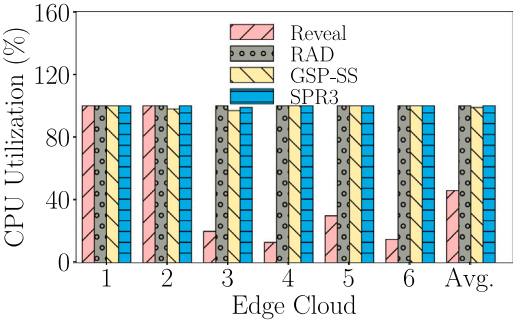


Fig. 8. Average CPU utilization (%) of each edge cloud.

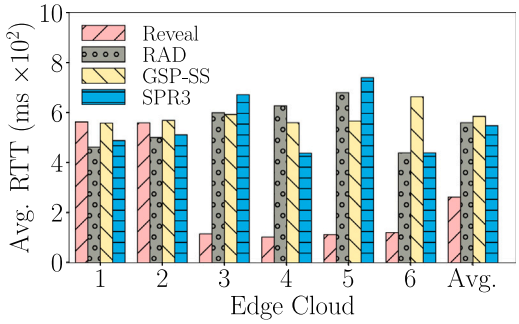


Fig. 6. Average RTT of each edge cloud.

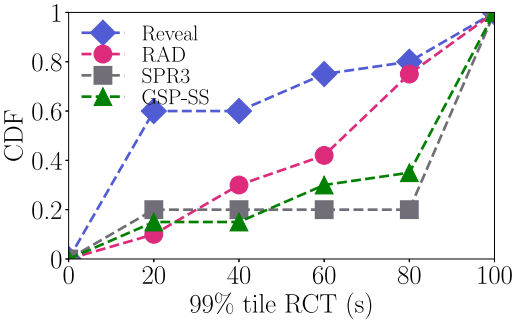


Fig. 9. CDF of %99 tile RCT (s).

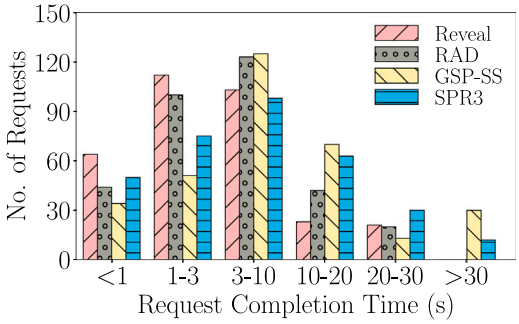


Fig. 7. No. of requests vs. request completion time.

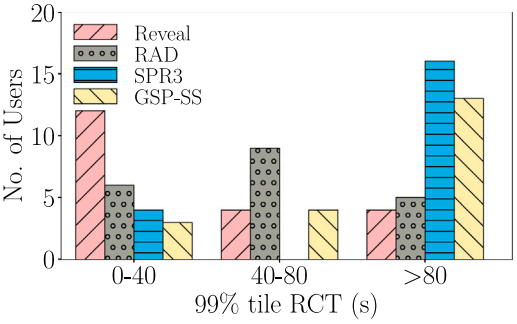


Fig. 10. %99 tile RCT of users.

of VNFs assigned to each user can enhance the network robustness. Specifically, we randomly select a user as the malicious user to attack its assigned VNFs. The malicious user adopts the hping3 [47] to exhaust the network bandwidth and uses the stress tool [48] to simulate malicious applications that preempt computing resources. We then measure packet loss rate, RTT, request completion time and CPU utilization for performance comparison. From Figs. 5–8, we can see that Reveal always achieves the best performance compared to three benchmarks. For example, in Fig. 6, Reveal reduces the average RTT by 55%, 52% and 53% compared with GSP-SS, SPR3 and RAD, respectively. Fig. 7 shows that 90% of requests have a completion time of less than 10 s in Reveal while only 70%, 65% and 82% in GSP-SS, SPR3 and RAD, respectively. The reason is that Reveal limits the number of VNFs each user can assign, therefore controlling the influence of the malicious user.

Comparison with the VNF failure event. In this set of experiments, we conduct the experiments under the scenario of VNF failure to illustrate that limiting the number of users served by each VNF can enhance the network robustness. The experiment results are shown in Figs. 9–10. We randomly select a VNF to fail and all its served requests will be transferred to another VNF. Since they need to be migrated to another VNF, these requests will take a longer completion time. Specifically, in Fig. 9, the frequency of users with 99% tile request completion time less than the 40 s is 60% by Reveal, while the frequency of that is 15%, 20% and 42% by GSP-SS, SPR3 and RAD. Fig. 10 shows that the number of affected users is 4 in Reveal, while in RAD, GSP-SS and SPR3 are 5, 13 and 16, respectively. Since RAD backups additional VNFs, it achieves similar performance with Reveal. The results show that by limiting the number of users each VNF will serve, Reveal controls the number of users influenced by the VNF failure.

Summary. Through the above experiments, we can draw some conclusions. First, Figs. 3–4 show that, by applying robustness constraints, Reveal can reduce negative effects when encountering abnormal situations. Second, from Figs. 5–8, we can see that the performance of Reveal in terms of packet loss rates, average RTT, request completion time and CPU utilization is much better than that of the comparison algorithms. By limiting the number of VNFs each user can assign, Reveal can reduce the impact of accidents such as encountering the malicious user. At last, as shown in Figs. 9–10, Reveal obtains the least number of users affected by the VNF failure, which means we can guarantee QoS for more users when encountering the VNF failure.

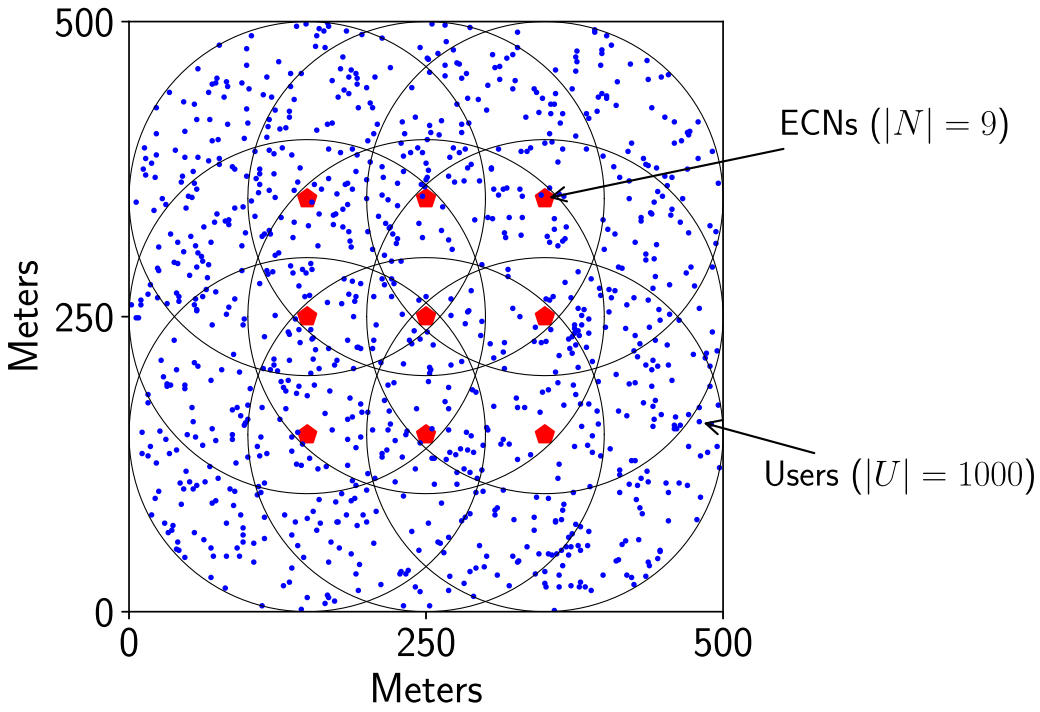


Fig. 11. The large-scale network topology.

5.3. Simulation evaluation

Settings. We conduct a similar simulation as [8,11]. The network contains $|N| = 9$ edge clouds regularly distributed inside a 500×500 m^2 area. To evaluate the influence of the number of users in the network, we adopt two scales of topologies. The first is a small-scale topology containing $|U| = 300$ users, and the second has a larger scale containing $|U| = 1000$ users, as shown in Fig. 11. The users randomly distribute in the base station coverage area. Similar to [11], we set the processing capacity of each edge cloud within $[20, 30]$ GHz. The number of VNFs in the network is $|\mathcal{F}| = 500$. The processing capacity of each VNF is within $[0.375, 0.5]$ GHz. The traffic of each user is conducted by real-world datasets collected from Google [19]. We set the robustness constraint to $k = 6$, $p = 50$ in the small topology, and $k = 10$, $p = 100$ in the large topology by default.

The simulations are performed under two scenarios. In the first scenario, we do not modify all benchmarks, so they ignore the robustness constraints when scheduling requests. The purpose of this scenario is to evaluate the maximum number of affected VNFs if the malicious user launches DoS attacks and the maximum number of affected users if the VNF failure encounters. The scenario is denoted by (a). In the second scenario, we modify all benchmarks by applying robustness constraints. By limiting the number of VNFs each user can assign and the number of users each VNF can serve, we mainly test the throughput for three algorithms. This scenario is denoted by (b).

Performance comparison in scenario (a). The simulation results are shown in Figs. 12–14. Fig. 12 shows that compared with all benchmarks, our algorithm always acquires the minimum number of affected VNFs. For example, giving 3600 requests in the small topology, Reveal decreases the maximum number of affected VNFs by 50%, 55% and 58% compared with RAD, SPR3 and GSP-SS, respectively. In Fig. 13, Reveal always obtains the minimum number of affected users compared with other algorithms. Note that the maximum number of affected users occasionally decreases when the number of requests increases. The reason is that more VNFs are placed as the number of requests increases, while the number of users is constant, so the number of users allocated to each VNF decreases. Fig. 14 shows that Reveal achieves similar throughput compared with RAD, SPR3 and GSP-SS, while Reveal considers robustness constraints.

From Figs. 12–14, we can conclude that without robustness constraints, RAD, SPR3 and GSP-SS will suffer more influence when encountering the malicious user or the failed VNF compared with Reveal. Moreover, Reveal can achieve similar network throughput compared with other benchmarks while considering robustness constraints.

Performance comparison in scenario (b). In this set of simulations, we add robustness constraints to other algorithms. When they schedule requests, they will reject requests when the robustness constraints are violated. Figs. 15–17 illustrate the influence of robustness constraints on the network throughput. When the robustness constraints are relaxed, the throughput of all algorithms increases. For example, in Fig. 15(a), when k is relaxed from 2 to 6, the throughput of Reveal, RAD, GSP-SS and SPR3 increases by 30%, 179%, 190% and 61%, respectively. When we relax the robustness constraints, the network throughput increases since the solution space expands. This denotes that k and p should be carefully determined to achieve high network performance. In addition, our algorithm always achieves the highest network throughput compared with other algorithms. As shown in Fig. 17(a), when there are 3.6×10^3 requests in the network, Reveal improves the throughput by 210.9%, 56.7% and 26.2% compared with SPR3, GSP-SS and RAD, respectively.

Summary. From the simulation results, we can draw some conclusions. First, Reveal can significantly reduce the influence of the malicious user and the failed VNF in the network. Second, Figs. 15–17 indicate that Reveal can consistently achieve the highest throughput compared with other algorithms, which apply the robustness constraints in request scheduling. The reason is that we consider robustness constraints when designing Reveal, so even if we limit the number of VNFs each user can assign and the number of users each VNF can serve, we can schedule more requests compared with other algorithms.

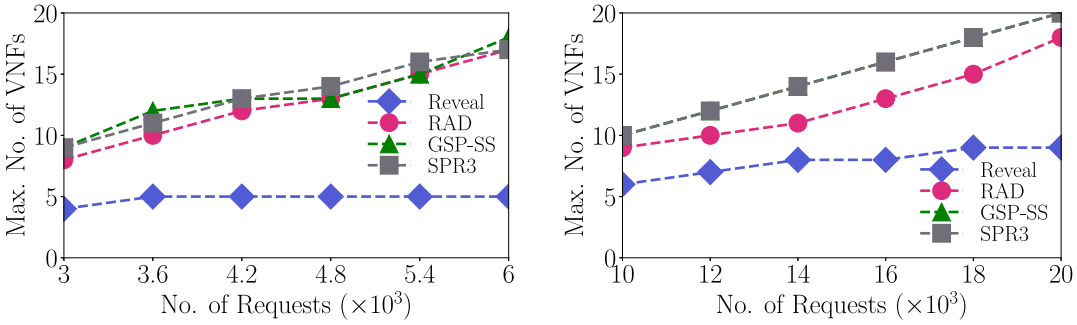


Fig. 12. Max. no. of affected VNFs vs. no. of requests in (a).

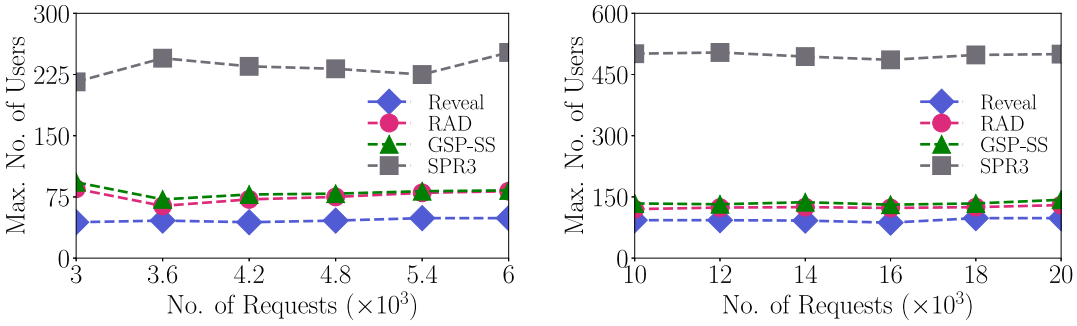


Fig. 13. Max. no. of affected Users vs. no. of requests in (a).

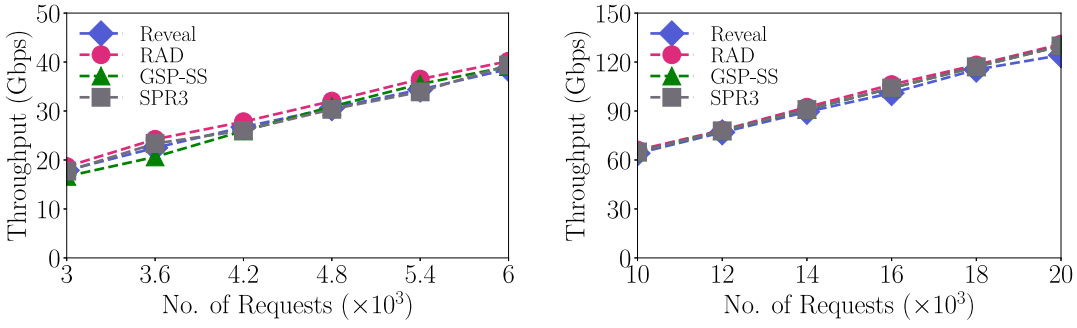


Fig. 14. Network throughput vs. no. of requests in (a).

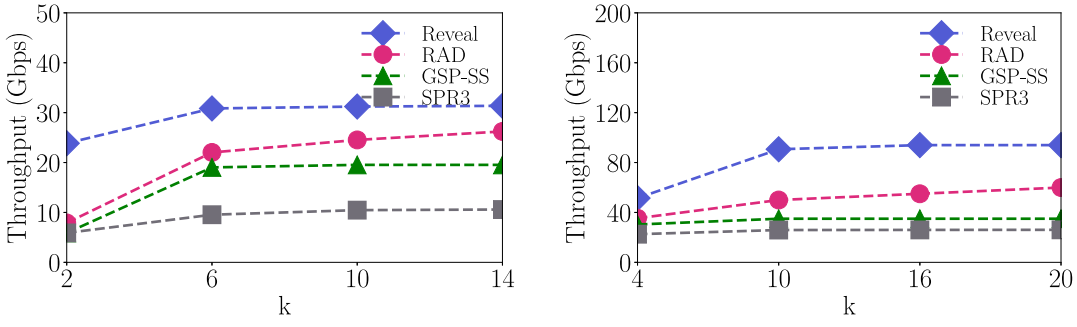


Fig. 15. Network throughput vs. k in scenario (b).

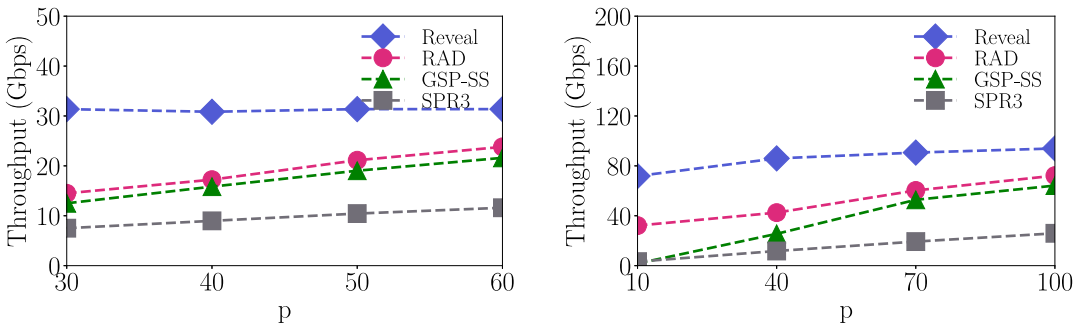


Fig. 16. Network throughput vs. p in scenario (b).

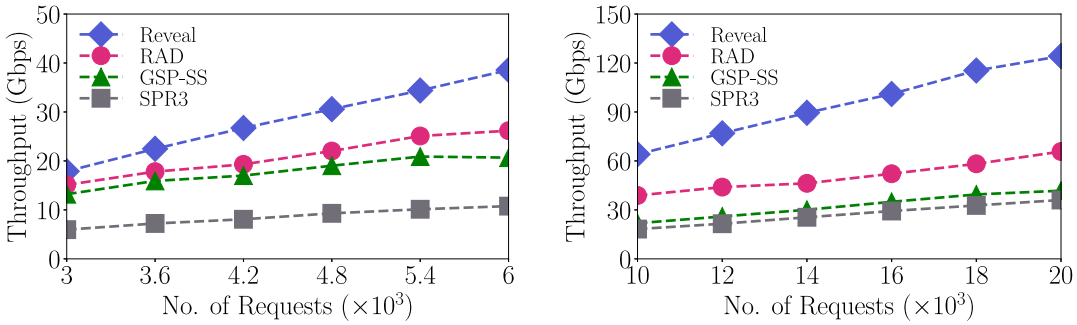


Fig. 17. Network throughput vs. no. of requests in (b).

Table 2
Comparison table of existing solutions.

Categories	Goals
VNF placement	Minimizing the latency [51–53] Maximizing the throughput [54,55]
Joint VNF placement and request scheduling	Maximizing the throughput [11,44,49] Provisioning deadline guarantee [56] Maximizing social welfare [57]
SFC routing	Guaranteeing SFC reliability [58–62]
Traffic analysis	Preserving user privacy [21,63,64]
VNF migration	Reducing migration cost [45,65,66]

6. Related works

As the scale of mobile applications increases, the edge cloud [49,50] is proposed to serve user requests with less latency than remote central clouds. Specifically, service providers place VNFs in edge clouds, while users obtain services through forwarding requests to corresponding VNFs. To provide stable services, we should consider the problem of placing which VNFs in each edge cloud, and which user requests are served by each VNF. In this section, we provide an overview of the state-of-the-art VM placement and request scheduling approaches, as well as existing works for improving the robustness of edge clouds. We summary the existing works in Table 2.

Existing works on VNF placement and request scheduling mainly focus on improving the performance of edge clouds by minimizing access latency [51–53], maximizing throughput [11,44,49,54,55], guaranteeing deadline [56] and maximizing social welfare [57]. For example, work [51] jointly considers minimizing access latency and maximizing service availability in edge clouds. It formulates the problem into an integer linear programming problem and proposes a genetic algorithm. The authors in work [11] propose SPR3, which studies the joint optimization of service placement and request routing in dense edge clouds and achieves close-to-optimal performance with a randomized rounding method. The authors in work [57] consider the problem of pricing for distributed training tasks in edge clouds and propose an auction-based online framework to solve it.

Besides improving the performance of edge clouds, it is also critical to enhance the robustness of edge clouds. Some works ranging from service function chain (SFC) routing [58–62], user traffic analysis [21,63,64] to VNF migration [45,65,66] are proposed for DDoS attack prevention [67], privacy protection [68,69] and federated leanring security [23,70]. For example, The authors in [58] present a reliability oriented SFC construction and backup method to solve the SFC backup problem. The experiments show that the proposed backup method can reduce the consumption of bandwidth by 11.7% compared with other alternatives. The authors in work [67] adopt remote clouds to filter the potentially large volume of DDoS traffic, which cannot be prevented entirely in the edge cloud due to its limited computing resources. However, this work introduces new drawbacks such as privacy violation and latency, since the user traffic will be redirected and processed by remote clouds. To prevent leakage of location information, work [69] provides a chaff-based approach minimizing the malicious tracking accuracy. This approach is implemented as a dedicated VNF, which needs additional resources to place at edge clouds. Work [71] provides a VNF replication scheme to guarantee users' QoS when some VNFs fail. Experimental results demonstrate that the proposed algorithms can reduce the VNF backup costs compared with baseline algorithms. Specifically, one user request may be served by multiple VNFs, and the user waits for the fastest completion of requests among replicas. However, in edge clouds, resources such as computing capacity are limited. If we schedule one request to 5 edge nodes, the system throughput will be significantly degraded by 1/5. From the above discussion, we can find that previous works mainly focus on improving edge clouds' robustness by preventing malicious attacks or recovery of VNF failures with additional resources (*e.g.*, dedicated security VNFs, backup VNFs). Considering that edge clouds have limited resource, these methods may

degrade the total system throughput. Unlike the previous studies, in this paper, we focus on how to limit the impact scope caused by malicious users or VNF failures to alleviate negative impacts to edge clouds, which complements previous works.

7. Conclusion

In this paper, we focus on the problem of robust VNF placement and request scheduling in edge clouds and present Reveal. We split the problem into two sub-problems: *robust VNF placement and assignment*, and *online request scheduling*. Two efficient algorithms are proposed to solve these sub-problems. Experimental and simulation results show that our algorithms can achieve high performance while guaranteeing robustness.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

The corresponding author of this paper is Gongming Zhao. This work was supported in part by the National Science Foundation of China (NSFC) under Grants 62102392, 62132019, 61936015, in part by the National Science Foundation of Jiangsu Province under Grant BK20210121, in part by the Hefei Municipal Natural Science Foundation under Grant 2022013, and in part by the Youth Innovation Promotion Association of Chinese Academy of Sciences under Grant 2023481.

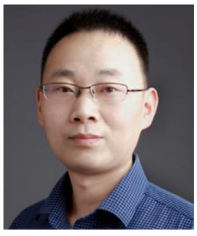
References

[1] L. Tong, Y. Li, W. Gao, A hierarchical edge cloud architecture for mobile computing, in: IEEE INFOCOM 2016, pp. 1–9.
[2] J. Pan, J. McElhannon, Future edge cloud and edge computing for internet of things applications, IEEE Internet Things J. 5 (1) (2017) 439–449.
[3] X. Lyu, H. Tian, W. Ni, Y. Zhang, P. Zhang, R.P. Liu, Energy-efficient admission of delay-sensitive tasks for mobile edge computing, IEEE Trans. Commun. 66 (6) (2018) 2603–2616.
[4] Y. Wang, Y. Zhang, M. Sheng, K. Guo, On the interaction of video caching and retrieving in multi-server mobile-edge computing systems, IEEE Wirel. Commun. Lett. 8 (5) (2019) 1444–1447.
[5] L. Liu, M. Gruteser, EdgeSharing: Edge assisted real-time localization and object sharing in urban streets, in: IEEE INFOCOM 2021.
[6] M.E. Computing, Deployment of Mobile Edge Computing in an NFV Environment, Vol. 17, ETSI Group Report MEC, 2018, p. V1.
[7] T. Ouyang, R. Li, X. Chen, Z. Zhou, X. Tang, Adaptive user-managed service placement for mobile edge computing: An online learning approach, in: IEEE INFOCOM 2019, pp. 1468–1476.
[8] J. Xu, L. Chen, P. Zhou, Joint service caching and task offloading for mobile edge computing in dense networks, in: IEEE INFOCOM 2018, pp. 207–215.
[9] S. Agarwal, F. Malandrino, C.-F. Chiasserini, S. De, Joint VNF placement and CPU allocation in 5G, in: IEEE INFOCOM 2018, pp. 1943–1951.
[10] L. Gu, D. Zeng, J. Hu, B. Li, H. Jin, Layer aware microservice placement and request scheduling at the edge, in: IEEE INFOCOM 2021 - IEEE Conference on Computer Communications, 2021, pp. 1–9, http://dx.doi.org/10.1109/INFOCOM42981.2021.9488779.
[11] K. Poularakis, J. Llorca, A.M. Tulino, I. Taylor, L. Tassiulas, Service placement and request routing in MEC networks with storage, computation, and communication constraints, IEEE/ACM Trans. Netw. 28 (3) (2020) 1047–1060.
[12] H. Li, L. Wang, Online orchestration of cooperative defense against DDoS attacks for 5G MEC, in: 2018 IEEE Wireless Communications and Networking Conference, WCNC.
[13] N.-N. Dao, T.V. Phan, U. Sa'ad, J. Kim, T. Bauschert, D.-T. Do, S. Cho, Securing heterogeneous iot with intelligent ddos attack behavior learning, IEEE Syst. J. (2021).
[14] J. Hou, P. Fu, Z. Cao, A. Xu, Machine learning based ddos detection through NetFlow analysis, in: MILCOM 2018 - 2018 IEEE Military Communications Conference, MILCOM, pp. 1–6, http://dx.doi.org/10.1109/MILCOM.2018.8599738.
[15] T. He, E.N. Ciftcioglu, S. Wang, K.S. Chan, Location privacy in mobile edge clouds: A chaff-based approach, IEEE J. Sel. Areas Commun. 35 (11) (2017) 2625–2636, http://dx.doi.org/10.1109/JSAC.2017.2760179.
[16] R. Potharaju, N. Jain, Demystifying the dark side of the middle: A field study of middlebox failures in datacenters, in: Proceedings of the 2013 Conference on Internet Measurement Conference, 2013, pp. 9–22.
[17] Jia, C. Yang, Reliability-aware dynamic service chain scheduling in 5G networks based on reinforcement learning, in: IEEE INFOCOM 2021.
[18] C. Delimitrou, C. Kozyrakis, Bolt: I know what you did last summer... in the cloud, ACM SIGARCH Comput. Archit. News 45 (1) (2017) 599–613.
[19] J. Wilkes, Google Cluster-Usage Traces v3, Google Inc, Mountain View, CA, USA, 2020.
[20] M. Masdari, M. Jalali, A survey and taxonomy of DoS attacks in cloud computing, Secur. Commun. Netw. 9 (16) 3724–3751.
[21] A.S. Sohal, R. Sandhu, S.K. Sood, V. Chang, A cybersecurity framework to identify malicious edge device in fog computing and cloud-of-things environments, Comput. Secur. 74 (2018) 340–354.
[22] A. Randazzo, I. Tinnirello, Kata containers: An emerging architecture for enabling MEC services in fast and secure way, in: 2019 Sixth International Conference on Internet of Things: Systems, Management and Security, IOTSMS, pp. 209–214.
[23] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, V. Shmatikov, How to backdoor federated learning, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2020, pp. 2938–2948.
[24] S. Ntalampiras, M. Fiore, Forecasting mobile service demands for anticipatory MEC, in: 2018 IEEE 19th International Symposium on" a World of Wireless, Mobile and Multimedia Networks"(WoWMoM), pp. 14–19.
[25] I. Pelle, F. Paolucci, B. Sonkoly, F. Cugini, Latency-sensitive edge/cloud serverless dynamic deployment over telemetry-based packet-optical network, IEEE J. Sel. Areas Commun. 39 (9) (2021) 2849–2863.
[26] S. Guo, Y. Dai, S. Xu, X. Qiu, F. Qi, Trusted cloud-edge network resource management: DRL-driven service function chain orchestration for IoT, IEEE Internet Things J. 7 (7) (2019) 6010–6022.
[27] T.L. Duc, R.G. Leiva, P. Casari, P.-O. Östberg, Machine learning methods for reliable resource provisioning in edge-cloud computing: A survey, ACM Comput. Surv. 52 (5) (2019) 1–39.
[28] L. Toka, G. Dobreff, B. Fodor, B. Sonkoly, Machine learning-based scaling management for kubernetes edge clusters, IEEE Trans. Netw. Serv. Manag. 18 (1) (2021) 958–972.
[29] X. Xu, Z. Fang, L. Qi, X. Zhang, Q. He, X. Zhou, Tripres: Traffic flow prediction driven resource reservation for multimedia iov with edge computing, ACM Trans. Multimedia Comput. Commun. Appl. (TOMM) 17 (2) (2021) 1–21.
[30] M. Chen, Y. Miao, H. Gharavi, L. Hu, I. Humar, Intelligent traffic adaptive resource allocation for edge computing-based 5G networks, IEEE Trans. Cogn. Commun. Netw. 6 (2) (2019) 499–508.
[31] A. Furno, D. Naboulsi, R. Stanica, M. Fiore, Mobile demand profiling for cellular cognitive networking, IEEE Trans. Mob. Comput. 16 (3) (2016) 772–786.
[32] K. Ha, Y. Abe, Z. Chen, W. Hu, B. Amos, P. Pillai, M. Satyanarayanan, Adaptive VM Handoff Across Cloudlets, Technical Report CMU-CS-15-113, Computer Science Department, Carnegie Mellon University, 2015.
[33] Z. Xu, W. Liang, W. Xu, M. Jia, S. Guo, Efficient algorithms for capacitated cloudlet placements, IEEE Trans. Parallel Distrib. Syst. 27 (10) (2015) 2866–2880.
[34] A. Ceselli, M. Premoli, S. Secci, Mobile edge cloud network design optimization, IEEE/ACM Trans. Netw. 25 (3) (2017) 1818–1831.
[35] D.B. Shmoys, É. Tardos, An approximation algorithm for the generalized assignment problem, Math. Program. 62 (1) (1993) 461–474.
[36] H. Xu, X.-Y. Li, L. Huang, H. Deng, H. Huang, H. Wang, Incremental deployment and throughput maximization routing for a hybrid SDN, IEEE/ACM Trans. Netw. 25 (3) (2017) 1861–1875.
[37] L.-C. Chen, H.-A. Choi, Approximation algorithms for data distribution with load balancing of web servers, in: Cluster, Vol. 1, Citeseer, 2001, p. 274.
[38] M. Mitzenmacher, E. Upfal, Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis, Cambridge University Press, 2017.
[39] PuLP, 2022, URL https://pypi.org/project/PuLP/. (Accessed 14 February 2022).
[40] G. Zhao, H. Xu, S. Chen, L. Huang, P. Wang, Joint optimization of flow table and group table for default paths in SDNs, IEEE/ACM Trans. Netw. (2018).
[41] L. Guo, J. Pang, A. Walid, Joint placement and routing of network function chains in data centers, in: IEEE INFOCOM 2018, pp. 612–620.
[42] tcping, 2022, URL https://www.elifulkerson.com/projects/tcping.php. (Accessed 14 February 2022).
[43] iperf, 2022, URL https://iperf.fr/. (Accessed 14 February 2022).
[44] V. Farhadi, F. Mehmeti, T. He, T.F. La Porta, H. Khamfroush, S. Wang, K.S. Chan, K. Poularakis, Service placement and request scheduling for data-intensive applications in edge clouds, IEEE/ACM Trans. Netw. 29 (2) (2021) 779–792.

[45] X. Shang, Y. Huang, Z. Liu, Y. Yang, Reducing the service function chain backup cost over the edge and cloud by a self-adapting scheme, IEEE Trans. Mob. Comput. 21 (8) (2021) 2994–3008.
[46] K. Shanmugam, N. Golrezaei, A.G. Dimakis, A.F. Molisch, G. Caire, Femtocaching: Wireless content delivery through distributed caching helpers, IEEE Trans. Inform. Theory 59 (12) (2013) 8402–8413.
[47] hping, 2022, URL http://hping.org/. (Accessed 14 February 2022).
[48] stress, 2022, URL https://linux.die.net/man/1/stress. (Accessed 14 February 2022).
[49] T. He, H. Khamfroush, S. Wang, T. La Porta, S. Stein, It's hard to share: Joint service placement and request scheduling in edge clouds with sharable and non-sharable resources, in: 2018 IEEE 38th International Conference on Distributed Computing Systems, ICDCS, IEEE, 2018, pp. 365–375.
[50] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, K.K. Leung, Dynamic service migration in mobile edge-clouds, in: 2015 IFIP Networking Conference, IFIP Networking, 2015, http://dx.doi.org/10.1109/IFIPNetworking.2015.7145316.
[51] L. Yala, P.A. Frangoudis, A. Ksentini, Latency and availability driven VNF placement in a MEC-NFV environment, in: 2018 IEEE Global Communications Conference, GLOBECOM, IEEE, 2018, pp. 1–7.
[52] R. Cziva, C. Anagnostopoulos, D.P. Pezaros, Dynamic, latency-optimal vNF placement at the network edge, in: IEEE Infocom 2018-Ieee Conference on Computer Communications, IEEE, 2018, pp. 693–701.
[53] G. Zhao, H. Xu, Y. Zhao, C. Qiao, L. Huang, Offloading tasks with dependency and service caching in mobile edge computing, IEEE Transactions on Parallel and Distributed Systems 32 (11) (2021) 2777–2792, http://dx.doi.org/10.1109/TPDS.2021.3076687.
[54] Z. Xu, Z. Zhang, J.C. Lui, W. Liang, Q. Xia, P. Zhou, W. Xu, G. Wu, Affinity-aware VNF placement in mobile edge clouds via leveraging GPUs, IEEE Trans. Comput. 70 (12) (2020) 2234–2248.
[55] Y. Yue, B. Cheng, M. Wang, B. Li, X. Liu, J. Chen, Throughput optimization and delay guarantee VNF placement for mapping SFC requests in NFV-enabled networks, IEEE Trans. Netw. Serv. Manag. 18 (4) (2021) 4247–4262, http://dx.doi.org/10.1109/TNSM.2021.3087838.
[56] S. Yang, F. Li, S. Trajanovski, X. Chen, Y. Wang, X. Fu, Delay-aware virtual network function placement and routing in edge clouds, IEEE Trans. Mob. Comput. 20 (2) (2019) 445–459.
[57] R. Zhou, N. Wang, Y. Huang, J. Pang, H. Chen, DPS: Dynamic pricing and scheduling for distributed machine learning jobs in edge-cloud networks, IEEE Trans. Mob. Comput. (2022).
[58] L. Qu, C. Assi, K. Shaban, M.J. Khabbaz, A reliability-aware network service chain provisioning with delay guarantees in NFV-enabled enterprise datacenter networks, IEEE Trans. Netw. Serv. Manag. 14 (3) (2017) 554–568.
[59] L. Qu, C. Assi, M.J. Khabbaz, Y. Ye, Reliability-aware service function chaining with function decomposition and multipath routing, IEEE Trans. Netw. Serv. Manag. 17 (2) (2019) 835–848.
[60] Y. Wang, L. Zhang, P. Yu, K. Chen, X. Qiu, L. Meng, M. Kadoch, M. Cheriet, Reliability-oriented and resource-efficient service function chain construction and backup, IEEE Trans. Netw. Serv. Manag. 18 (1) (2020) 240–257.
[61] M. Wang, B. Cheng, J. Chen, Joint availability guarantee and resource optimization of virtual network function placement in data center networks, IEEE Trans. Netw. Serv. Manag. 17 (2) (2020) 821–834.
[62] H. Tu, G. Zhao, H. Xu, Y. Zhao, Y. Qiu, L. Huang, RoNS: Robust network function services in clouds, Comput. Netw. (ISSN: 1389-1286) 215 (2022) 109212, http://dx.doi.org/10.1016/j.comnet.2022.109212, https://www.sciencedirect.com/science/article/pii/S1389128622002961.
[63] R. Sairam, S.S. Bhunia, V. Thangavelu, M. Gurusamy, NETRA: Enhancing IoT security using NFV-based edge traffic analysis, IEEE Sens. J. (2019) http://dx.doi.org/10.1109/JSEN.2019.2900097.
[64] W. Du, A. Li, P. Zhou, B. Niu, D. Wu, Privacyeye: A privacy-preserving and computationally efficient deep learning-based mobile video analytics system, IEEE Trans. Mob. Comput. 21 (9) (2021) 3263–3279.
[65] H. Wang, H. Xu, H. Huang, M. Chen, S. Chen, Robust task offloading in dynamic edge computing, IEEE Trans. Mob. Comput. (2021).
[66] W. Mingshi, L. Junqin, H. Tianxiang, W. Pingping, Y. Kang, H. Jiakai, Z. Diwen, Y. Yang, T. Riming, Failure prediction based VNF migration mechanism for multimedia services in power grid substation monitoring, in: 2022 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting, BMSB, IEEE, 2022, pp. 1–6.
[67] T. Alharbi, A. Aljuhani, H. Liu, Holistic DDoS mitigation using NFV, in: 2017 IEEE 7th Annual Computing and Communication Workshop and Conference, CCWC, 2017, http://dx.doi.org/10.1109/CCWC.2017.7868480.
[68] F. Nawab, Wedgechain: A trusted edge-cloud store with asynchronous (lazy) trust, in: 2021 IEEE 37th International Conference on Data Engineering, ICDE, IEEE, 2021, pp. 408–419.
[69] T. He, E.N. Ciftcioglu, S. Wang, K.S. Chan, Location privacy in mobile edge clouds: A chaff-based approach, IEEE J. Sel. Areas Commun. 35 (11) (2017) 2625–2636.
[70] T. Li, S. Hu, A. Beirami, V. Smith, Ditto: Fair and robust federated learning through personalization, in: International Conference on Machine Learning, PMLR, 2021, pp. 6357–6368.
[71] J. Li, W. Liang, M. Huang, X. Jia, Reliability-aware network service provisioning in mobile edge-cloud networks, IEEE Trans. Parallel Distrib. Syst. 31 (7) (2020) 1545–1558.

Jin Fang received the B.S. degree in the College of Computer Science and Electronic Engineering, Hunan University in 2020. He is pursuing the Ph.D. degree in computer software and theory at the University of Science and Technology of China. His current research interests include software-defined networks, distributed training systems and programmable networks.

Gongming Zhao received the Ph.D. degree in computer software and theory from the University of Science and Technology of China in 2020. He is currently an Associate Professor with the University of Science and Technology of China. His current research interests include software-defined networks and cloud computing.



Hongli Xu received the B.S. degree in computer science and the Ph.D. degree in computer software and theory from the University of Science and Technology of China in 2002 and 2007, respectively He is currently an Associate Professor with the School of Computer Science and Technology, University of Science and Technology of China. He has authored or coauthored over 70 papers, and held about 30 patents. His main research interest is software-defined networks, cooperative communication, and vehicular ad hoc network.

Huaqing Tu is currently pursuing the Ph.D. degree in computer science at the University of Science and Technology of China. Her main research interests are software-defined networks and cloud computing.

Haibo Wang is currently pursuing the Ph.D. degree in computer science from the University of Florida. His current research interests lies in the intersection between network measurement and data streaming algorithms, with a focus on the high-speed streamed big data analytics.