Server Placement for Edge Computing: A Robust Submodular Maximization Approach

Yuben Qu, Lihao Wang, Haipeng Dai, *Member, IEEE,* Weijun Wang, *Student Member, IEEE*
Chao Dong, *Member, IEEE,* Fan Wu, *Member, IEEE,* and Song Guo, *Fellow, IEEE*

Abstract—In this work, we study the problem of Robust Server Placement (RSP) for edge computing, *i.e.*, in the presence of uncertain edge server failures, how to determine a server placement strategy to maximize the expected overall workload that can be served by edge servers. We mathematically formulate the RSP problem in the form of robust max-min optimization, derived from two consequentially equivalent transformations of the problem that does not consider robustness and followed by a robust conversion. RSP is challenging to solve, because the explicit expression of the objective function in RSP is hard to obtain, and it is a robust max-min problem with knapsack constraints, which is still an unexplored problem in the literature. We reveal that the objective function is monotone submodular, and propose two solutions to RSP. Firstly, after proving that the involved constraints form a p-independence system constraint, where p is a parameter determined by the coefficients in the knapsack constraints, we propose an algorithm that achieves a provable approximation ratio in polynomial time. Secondly, we prove that one of the knapsack constraints is a matroid contraint, and propose another polynomial time algorithm with a better approximation ratio. Furthermore, we discuss the applicability of the aforementioned algorithms to the case with an additional server number constraint. Both synthetic and trace-driven simulation results show that, given any maximum number of server failures, our proposed algorithms outperform four state-of-the-art algorithms and approaches the optimal solution, which applies exhaustive exponential searches, while the proposed latter algorithm brings extra performance gains compared with the former one.

Index Terms—Edge computing networks, server placement, robust submodular optimization.

1 Introduction

Nowadays, mobile devices have been indispensable in everyday life and various mobile applications in fast growth have posed severe challenges to wireless access networks and cloud infrastructure. While those applications are becoming increasingly computation-intensive, the gap between required computing resources and computing capacity of mobile devices is becoming more remarkable. Although offloading them to remote clouds may bridge the gap, the incurred high latency could greatly affect the user experience [1], [2], [3], [4]. Mobile edge computing (MEC) [5] emerges as a promising paradigm to directly deliver computation services to mobile devices at the network edge, which can alleviate the pressure over cloud and break though the physical limitations of mobile devices simultaneously. A fundamental and critical issue in MEC is *edge server placement*, *i.e.*, determining where to deploy the edge servers to maximize the network performance.

Compared to mobile cloud computing (MCC) [6], MEC is faced with some unique uncertainties in the edge server placement as follows. Firstly, unlike the highly reliable large data center in MCC, MEC networks are usually heterogeneous and edge servers are less reliable [7], [8], [9], [10], [11], [21], *e.g.*, some edge servers can crash at any time [12]. Secondly, different from MCC tasks mostly being offloaded through reliable wired links, MEC tasks are usually offloaded to edge servers through failure-prone wireless links, as edge servers are generally co-located with the cellular base stations (BSs) or local wireless access points (APs) [2]. A recent measurement work shows a significant fraction of connection failures as large as 45% over 5 million users using WiFi networks in the urban area [13]. Once an offloading link or some BS/AP fails, the corresponding edge server is unreachable, which can be seen as kind of server failure even if it works normally. The above uncertain factors can fail any predefined optimized server placement strategy. Thus, instead of being offloaded to edge servers, some prescheduled tasks have to be executed locally or uploaded to the cloud, which may result in great performance loss. However, most existing studies about server placement for MEC do not consider such uncertainties.

In this paper, we study the problem of Robust Server Placement (RSP) for MEC, where robustness here refers to the ability of *servers* to provide an acceptable level of service in the face of possible hardware and software failures. Specifically, we investigate how to determine a server placement strategy to maximize the expected overall work-

Y. Qu and F. Wu are with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China (E-mail: quyuben@sjtu.edu.cn, fwu@cs.sjtu.edu.cn). Y. Qu is also with the Key Laboratory of Dynamic Cognitive System of Electromagnetic Spectrum Space, Ministry of Industry and Information Technology, Nanjing University of Aeronautics and Astronautics, China.
L. Wang, H. Dai (corresponding author), and W. Wang are with the Department of Computer Science and Technology, Nanjing University, China (E-mail: Lihao_Wang@hotmail.com, haipengdai@nju.edu.cn, weijunwang@smail.nju.edu.cn).
C. Dong is with the College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, China (E-mail: dch999@gmail.com).
S. Guo is with The Hong Kong Polytechnic University Shenzhen Research Institute and Department of Computing, The Hong Kong Polytechnic University, Hong Kong (E-mail: cssongguo@comp.polyu.edu.hk).

load processed by S edge servers deployed at N candidate locations, when at most $k \leq S-1$ servers fail actually. RSP is hard to formulate as it involves not only integral variables for edge server placement, but also continuous workload allocation variables, which are combinatorially coupled. Our formulation is based on the problem without considering robustness (Non-RSP). Firstly, we find out that in Non-RSP, the optimal workload allocation schedule can be solved efficiently given any fixed server placement strategy. Thus, Non-RSP is equivalent to a binary integer programming problem with respect to the server placement variable only. We further reformulate Non-RSP as a set function maximization problem with two knapsack constraints. Secondly, we mathematically formulate RSP in the popular form of robust max-min optimization problem after a robust conversion over the final expression of Non-RSP.

There exist two main technical challenges in solving RSP. First, the explicit objective function in RSP is hard to obtain in fact, since we cannot acquire the closed form of the optimal solution of workload allocation in Non-RSP given a fixed server placement strategy. Thus, some fundamental properties for set function optimizations such as monotonicity or submodularity in RSP are difficult to obtain. Secondly, existing works can handle robust max-min monotone submodular optimization problems either with a cardinality constraint [14], [15], [16], [17], [18] or with a single matroid constraint [19], [20], while the robust problem like RSP with two knapsack constraints is unsolved so far. This implies that even if there exist monotonicity and submodularity in RSP, the problem is still nontrivial.

In this work, instead of seeking the closed form of the optimal workload allocation, we regard the optimal workload allocation schedule as an implicit function of the server placement variable, and theoretically prove that the objective function in RSP is indeed monotone submodular using that implicit information. We then propose two approximation algorithms to RSP. Firstly, in light of one knapsack constraint being a matroid constraint, we prove that the two constraints form a p-independence system constraint, where p is a parameter determined by the ratio of the coefficients in another knapsack constraint. Therefore, RSP is proven to be a robust monotone submodular max-min problem with a p-independence system constraint. We propose a novel algorithm that achieves a provable approximation ratio in polynomial time for RSP. Secondly, we prove that the constraint about the candidate locations of any server is actually a matroid constraint, and then propose another polynomial time algorithm with a higher complexity and strictly better approximation ratio than the previous algorithm. Moreover, since the service provider might have the constraint of limiting at most one edge server at each candidate location, we also discuss how to extend the proposed algorithms to the new RSP problem with that additional constraint, and present the corresponding performance guarantees.

Our main contributions are summarized as follows:

- To the best of our knowledge, this is the first work that studies the problem of robust server placement for MEC (*i.e.*, RSP) considering uncertain edge server failures. The mathematical formulation is based on the twice equivalent transformations of the problem without considering robustness and a robust conversion. The RSP problem is reformulated as a robust max-min set function optimization problem with a matroid constraint and a knapsack constraint.
- We prove that in the RSP problem, its objective function is monotone and submodular, despite its implicit exact expression, and its constraints form a p-independence system constraint, where p is a parameter related to the ratio of the coefficients in the knapsack constraint about the budget. We then propose a novel robust algorithm that achieves a provable approximation ratio and polynomial time complexity. It is the first algorithm achieving an approximation ratio for the robust max-min submodular problem with a p-independence system constraint.
- We also propose another polynomial time algorithm for RSP that achieves a strictly better approximation ratio than the previous proposed algorithm, leveraging one of the knapsack constraints being a matroid constraint. We further discuss how to utilize the proposed algorithms to solve the new RSP problem with an additional server number constraint, with a rigorous theoretical analysis of their corresponding performance guarantees.
- We conduct both synthetic simulation and trace-driven simulation to validate the effectiveness of the proposed algorithms in terms of robustness guarantee. The results show that in the presence of any given number of server failures, our algorithms outperform four other benchmark algorithms in term of overall workload served by edge servers, and achieve about 90% workload of the exhaustive brute-force algorithm in the small network scale, while the proposed latter algorithm serves 6.7% extra workload compared to the proposed former algorithm.

The rest of this paper is organized as follows. Related works are reviewed in Section 2. In Section 3 and Section 4, we introduce the system model and problem statement, and mathematical formulation of RSP, respectively. In Section 5 and Section 6, we propose two robust server placement algorithms with different approximation ratios and time complexities. Section 7 discusses how to leverage the proposed algorithms to solve the new RSP problem with an additional constraint. The effectiveness of the proposed algorithms is validated by extensive simulations in Section 8. Section 9 concludes the paper.

2 Related Works

As far as we know, few existing works regarding edge server placement have considered the uncertain server failures in MEC environment. Comprehensive surveys about MEC can be found in [1], [2], [3], [4]. We classify the existing works related to server placement in MEC into three categories according to their different optimization objectives as follows.

Installation Expenditure Minimization for Server Placement: Many existing works aim to optimize installation expenditure of server placement in MEC [22], [23], [24], [25], [26], [27], [28], [29], [30]. Ceselli *et al.* [22], [23] studied the problem of planing edge servers for mobile networks in both static and dynamic scenarios, with the objective of

minimizing installation cost by optimizing cloudlet (a type of MEC network) placement as well as routing schedule. Fan *et al.* [27] aimed to optimize the tradeoff between the deployment cost and end-to-end delay. Meng *et al.* [29] studied the joint edge server placement and application configuration to minimize the weighted sum of edge server opening cost and service cost. In [24], [26], [25], [28], [30], several static edge server placement schemes are explored to minimize the installment cost and satisfy the capacity and latency constraints at the same time.

Access Delay Minimization: Besides, some other related works aim to optimize access delay in the edge server placement problem [31], [32], [33], [34], [35], [36]. Jia *et al.* [32] studied the optimal K cloudlet placement problem and user-to-cloudlet assignment in a WMAN, in order to minimize response latency and balance workload. Xu *et al.* [31], [33] proved that the capacitated cloudlet placement problem is NP-hard and proposed a heuristic placement algorithm to minimize the average cloudlet access delay. Zhao *et al.* [34] modeled each cloudlet as a $M/M/1$ queue and investigated the problem of minimizing the average cloudlet access delay. Considering the heterogeneity of edge servers and response time fairness, Cao *et al.* [35] proposed an approach with offline and online stages by integer linear programming (ILP) and mobility-aware game theory, respectively, to reduce the expected response latency as well as the response time fairness. Liu *et al.* [36] proposed a heuristic algorithm leveraging Kuhn-Munkres (KM) algorithm and greedy strategy to minimize the average bandwidth resource usage and average delay in edge computing networks.

Energy Consumption Minimization: Energy consumption is also considered as a critical factor for the edge server placement problem [37], [38], [39]. Li *et al.* [37] designed an edge server placement scheme with low energy consumption while keeping the access delay acceptable based on particle swarm optimization (PSO) techniques. Yang *et al.* [38] proposed a server placement strategy to jointly optimize cloudlet placement and workload allocation to minimize energy consumption under the latency constraint. Shen *et al.* [39] proposed an energy-efficient cloudlet placement strategy named ECPM to achieve the energy saving by effectively reducing the number of cloudlets.

To summarize, although the server placement problem for MEC has been extensively studied in existing works for various optimization objectives, the effect of uncertain server failures is seldom considered. Once several server failures occur due to uncertain factors, some pre-scheduled tasks have to be executed locally or uploaded to the cloud instead of being offloaded to edge servers. Since no redundancy design is considered in existing works, server failures can result in great performance loss for both service providers and end users. In this work, we study the robust server placement problem for MEC with the consideration of at most $k \leq S-1$ server failures when deploying S servers. To the best of our knowledge, this is the first work to study the robust edge server placement problem for MEC and propose two novel solutions with performance guarantees. Note that the first solution has been presented in the conference version of this paper [54].

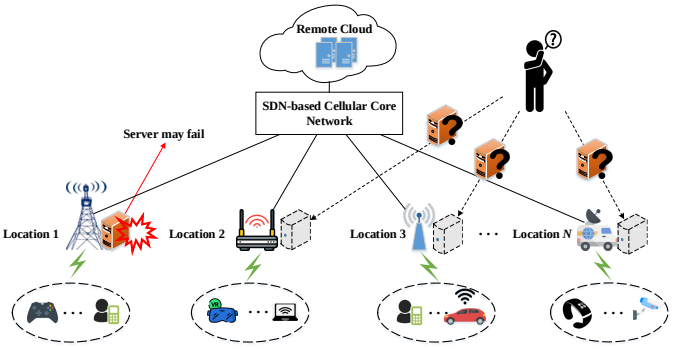


Fig. 1: An illustration of edge server placement.

3 Model and Problem Statement

In this section, we first introduce the basic model for server placement in MEC, and then present the formal definition of the robust server placement problem for MEC.

3.1 System Model

As shown in Fig. 1, we consider a typical edge computing network where N APs (*e.g.*, WiFi, BS, mobile communication vehicle and etc) are candidate placement locations for S edge servers. APs are interconnected by the software defined network (SDN) based cellular network [48], [27], [34], which provides efficient and flexible communications including data and control information among APs. Denote the sets of APs and edge servers as $\mathcal{N} := \{1, 2, ..., N\}$ and $\mathcal{S} := \{1, 2, ..., S\}$, respectively. We suppose the servers are heterogeneous in terms of capabilities and deployment costs at different locations. Let c_{js} represent the deployment cost to deploy server $s \in \mathcal{S}$ at AP $j \in \mathcal{N}$. We introduce a binary variable $x_{js} \in \{0, 1\}$ to indicate whether to deploy server $s \in \mathcal{S}$ at AP $j \in \mathcal{N}$ (*i.e.*, $x_{js} = 1$) or not (*i.e.*, $x_{js} = 0$). Thus, the overall deployment cost is $\sum_{j=1}^{N}\sum_{s=1}^{S} x_{js}c_{js}$, which should be no more than a predefined deployment budget C_0. Without considering the deployment budget, the service provider can deploy one server at each AP. However, in practice, the number of APs can be huge, which leads to high deployment cost. For instance, according to the trace data from Shanghai Telecom, there are 389 base stations in only 43 square kilometers of central urban area of Shanghai. In the context of ultra-dense networks in 5G, the number of APs can also be huge. We note that our model and solutions can apply to any ratio between servers and APs.

End users can obtain powerful computing services by sending their task-computing requests to edge servers via appropriate APs. APs usually are deployed at strategic locations such as markets, offices, train stations, schools and the workload at each AP is highly correlated with the population density and the user habit in that area. In the long term, those generally will not change dramatically, which enables us to accurately predict the workload with historical information by methods like linear regression [31], [33], [47], [49]. Let w_i be the expected workload at AP $i \in \mathcal{N}$. We denote the computing capacity of edge server $s \in \mathcal{S}$ as d_s. Due to the finite computing capacity and limited number of edge servers, some computing requests may not be served locally. Besides the locally deployed edge

TABLE 1: A list of frequently used notations.

Notation	Definition
$\mathcal{N}$	Set of APs
N	Number of APs
$\mathcal{S}$	Set of edge servers
S	Number of edge servers
c_{js}	Cost to deploy server s at AP j
C_0	Deployment budget
w_i	Expected workload at AP i
d_s	Computation capacity of edge server s
b_i	Task-required bandwidth at AP i
B_j^{up}	Available uplink bandwidth of AP j
B_j^{down}	Available downlink bandwidth of AP j
q_{ij}	Indicator of whether the request at AP i is permitted to be offloaded to AP j or not
x_{js}	Deployment decision variable for server s and AP j
$\mathbf{X}$	Deployment decision vector $\{x_{js}\}$
y_{ij}	Proportion of AP i's workload allocated to AP j
$\mathbf{Y}$	Workload allocation vector $\{y_{ij}\}$
P0	Problem Non-RSP
P1	Problem RSP
Ω	Set of all server-AP pairs

server, the user request can be offloaded to other nearby edge servers or the remote cloud. To serve the request with low latency, the request should not be offloaded to distant servers [44]. Considering the latency requirement, following [44], we introduce a binary constant q_{ij} to indicate whether the request at AP $i \in \mathcal{N}$ is permitted to be offloaded to servers at AP $j \in \mathcal{N}$ (*i.e.*, $q_{ij} = 1$) or not (*i.e.*, $q_{ij} = 0$).

Let a continuous variable $y_{ij} \in [0, 1]$ ($\forall i, j \in \mathcal{N}$) be the proportion of the AP i's workload that is allocated to the edge servers at AP j. Therefore, the actual workload to be performed at AP $j \in \mathcal{N}$ is $\sum_{i=1}^{N} w_i y_{ij}$, which should be no more than the overall computation capacities of edge servers placed at AP $j \in \mathcal{N}$. If there is no available edge server for some request, it will be uploaded to the remote cloud, which incurs higher latency. Serving task-computing requests not only consumes the computation resource of edge server, but also the communication resources. Denote the available bandwidth of AP $j \in \mathcal{N}$ as B_j and the total task-required bandwidth at AP $i \in \mathcal{N}$ as b_i. The total bandwidth occupied at AP $j \in \mathcal{N}$ is thus equal to $\sum_{i=1}^{N} b_i y_{ij}$. A list of frequently used notations in this paper is presented in Table 1.

3.2 Problem Statement

In this work, we aim to maximize the overall served workload of user requests in the long term, in the presence of possible edge server failures. Compared to the stable and reliable cloud servers, edge servers are heterogeneous and may fail [12], [7]. In other words, we aim to determine a server placement scheme in a robustness manner faced with server failures. Without considering robustness, the objective of maximizing the overall served workload can be formulated as: $\max \sum_{i=1}^{N} \sum_{j=1}^{N} w_i y_{ij}$. However, the RSP problem for edge computing is difficult to mathematically formulate because besides the integral deployment variable x_{js}, there is another continuous workload allocation variable y_{ij}, which are combinatorially coupled with each other. Based on the formulation as well as transformations of the problem without considering the robustness (*i.e.*, Non-RSP), we mathematically formulate the RSP problem in Section 4.

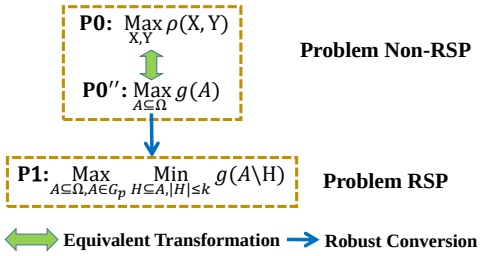


Fig. 2: A diagram of problem transformation.

4 Problem Formulation

In this section, as shown in Fig. 2, we first transform the mixed-integral Non-RSP problem into an equivalent set function optimization problem with respect to the integral deployment variable only. Then, we mathematically formulate the RSP problem based on the aforementioned set function optimization problem after a robust conversion.

4.1 Problem Transformations of Non-RSP

In problem Non-RSP, two kinds of decision variables are involved: server deployment variable $\mathbf{X} := \{x_{js}\}_{j\in\mathcal{N},s\in\mathcal{S}}$ indicating whether edge server s should be placed at AP j or not and workload allocation variable $\mathbf{Y} := \{y_{ij}\}_{i\in\mathcal{N},j\in\mathcal{N}}$ representing the proportion of AP i's workload permitted to be allocated to AP j, where the former is binary (*i.e.*, $x_{js} \in \{0, 1\}$) and the latter is continuous (*i.e.*, $y_{ij} \in [0, 1]$). The Non-RSP problem can be mathematically formulated as follows:

$$(\mathbf{P0}) \quad \max_{\mathbf{X},\mathbf{Y}} \quad \sum_{i=1}^{N}\sum_{j=1}^{N} w_i y_{ij} \tag{1}$$

$$\text{s.t.} \quad \sum_{j=1}^{N} x_{js} \le 1, \forall s \in \mathcal{S}, \tag{2}$$

$$\sum_{j=1}^{N} y_{ij} \le 1, \forall i \in \mathcal{N}, \tag{3}$$

$$\sum_{i=1}^{N} w_i y_{ij} \le \sum_{s=1}^{S} x_{js} d_s, \forall j \in \mathcal{N}, \tag{4}$$

$$\sum_{j=1}^{N}\sum_{s=1}^{S} x_{js} c_{js} \le C_0, \tag{5}$$

$$\sum_{i=1}^{N} b_i y_{ij} \le B_j^{down}, \forall j \in \mathcal{N}, \tag{6}$$

$$b_i \sum_{j=1}^{N} y_{ij} \le B_i^{up}, \forall i \in \mathcal{N}, \tag{7}$$

$$y_{ij} \le q_{ij}, \forall i, j \in \mathcal{N}, \tag{8}$$

$$x_{js} \in \{0, 1\}, y_{ij} \in [0, 1], \forall i, j \in \mathcal{N}, s \in \mathcal{S}, \tag{9}$$

where the objective is to maximize the expected overall workload served by edge servers (also maximize the expected overall profit for service providers). Constraint (2) ensures that any edge server can only be deployed on at most one of the APs. Constraint (3) constrains that the overall proportion of redirected workload from any AP is no more than 1. Constraint (4) means that the overall allocated workload at any AP is within the overall computation capability of the edge servers deployed at this AP. Constraint

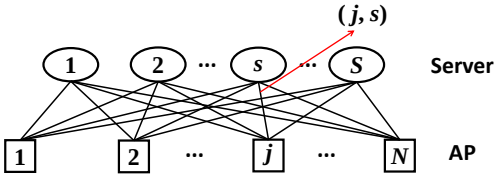


Fig. 3: Set Ω establishes a match between servers and APs.

(5) implies that the total deployment cost is no more than the deployment budget. Constraint (6) means that the total bandwidth required to receive the tasks at each AP cannot exceed the available downlink bandwidth of the AP. Constraint (7) implies that the total bandwidth required to offload the tasks at each AP to edge servers cannot exceed the available uplink bandwidth of the AP. Constraint (8) means that the requests can only be allocated to servers at nearby permitted APs. Constraint (9) specifies the basic range of the variables. Note that the latency has been considered in the above problem formulation. Specifically, constraint (4) guarantees the task execution latency in the target edge server, constraint (6) and constraint (7) guarantee the task's data transmission latency from an AP to its target edge server, and constraint (8) guarantees the latency between two APs for cooperative workload execution to be within a reasonable range. In sum, in light of those constraints, the proportion of offloading workload that is permitted to be successfully executed has been limited in terms of the latency.

Lemma 1. *For problem Non-RSP, given any fixed server deployment strategy, the optimal workload allocation can be obtained in polynomial time.*

Proof. Given any server deployment $\mathbf{X^0} = \{x^0_{ij}\}$, the problem of Non-RSP naturally turns into a workload allocation problem with respect to $\mathbf{Y}$ only in the following:

$$\begin{aligned}(\mathbf{P0}') \quad \max_{\mathbf{Y}} \quad & \sum_{i=1}^{N}\sum_{j=1}^{N} w_i y_{ij} \\ \text{s.t.} \quad & (3), (6), (7), (8), (9), \\ & \sum_{i=1}^{N} w_i y_{ij} \le \sum_{s=1}^{S} x^0_{js} d_s, \forall j \in \mathcal{N}. \end{aligned} \tag{10}$$

The above problem is a simple linear programming (LP) problem, which can be solved in polynomial time by some classical methods such as the Simplex algorithm [41] even in the large scale. □

Based on Lemma 1, we reformulate the Non-RSP problem as a set function optimization problem. Let $\rho(\mathbf{X}, \mathbf{Y})$ be the objective function of Non-RSP. Firstly, from Lemma 1, we know that for a given $\mathbf{X}$, we can always obtain the optimal value of $\mathbf{Y}$ in an efficient way, denoted as $\mathbf{Y}^*(\mathbf{X})$. Although the explicit expression of $\mathbf{Y}^*(\mathbf{X})$ cannot be obtained, it implies that solving the original Non-RSP problem is equivalent to solving the problem with respect to the deployment variable $\mathbf{X}$ only via substituting $\mathbf{Y}$ by $\mathbf{Y}^*(\mathbf{X})$. Thus, the objective function can be transformed to $\rho(\mathbf{X}, \mathbf{Y}^*(\mathbf{X}))$.

Secondly, let $\Omega := \{(j,s)|\forall j \in \mathcal{N}, s \in \mathcal{S}\}$, which establishes an one-one mapping between a server placement variable x_{js} and the element $e = (j,s) \in \Omega$ as shown in Fig. 3. To be specific, $x_{js} = 1$ means choosing element (j,s) from Ω, while $x_{js} = 0$ implies not choosing element (j,s) from Ω. Let $\mathcal{A} \subseteq \Omega$ represent the set of selected pairs of edge server and AP (deployment location), that is, $\mathcal{A} = \{(j,s)|x_{js} = 1, j \in \mathcal{N}, s \in \mathcal{S}\}$. For a feasible set $\mathcal{A} \subseteq \Omega$, we define $g(\mathcal{A}) := \rho(\mathbf{X}, \mathbf{Y}^*(\mathbf{X}))$, where for each x_{js} in $\mathbf{X}$, $x_{js} = 1$ iff $(j,s) \in \mathcal{A}$. Then, by introducing $\mathbb{1}$ as the indicator function, we reformulate Non-RSP as follows:

$$\begin{aligned}(\mathbf{P0}'') \quad \max_{\mathcal{A}\subseteq\Omega} \quad & g(\mathcal{A}) \\ \text{s.t.} \quad & \sum_{j:(j,s)\in\mathcal{A}} \mathbb{1}_{(j,s)\in\mathcal{A}} \le 1, \forall s \in \mathcal{S}, \end{aligned} \tag{11}$$

$$\sum_{(j,s)\in\mathcal{A}} c_{js} \le C_0. \tag{12}$$

In the next, we reveal some desirable properties of function $g(\mathcal{A})$. We first provide the basic definitions of non-negativity, monotonicity, and submodularity as follows.

Definition 1. *(Non-negativity, Monotonicity, Submodularity [42]) A set function $f : 2^\Omega \to \mathbb{R}$ (Ω is a finite ground set) is non-negative if $f(\emptyset) = 0$ and $f(\mathcal{A}) \ge 0$ for $\forall \mathcal{A} \subseteq \Omega$. $f(\cdot)$ is monotone if for $\forall \mathcal{A}_1 \subseteq \mathcal{A}_2 \subseteq \Omega, f(\mathcal{A}_1) \le f(\mathcal{A}_2)$. And $f(\cdot)$ is submodular if for $\forall \mathcal{A}_1 \subseteq \mathcal{A}_2 \subseteq \Omega$ and $\forall e \in \Omega \setminus \mathcal{A}_2$, $f(\mathcal{A}_1 \cup \{e\}) - f(\mathcal{A}_1) \ge f(\mathcal{A}_2 \cup \{e\}) - f(\mathcal{A}_2)$.*

Lemma 2. *The objective function $g(\mathcal{A})$ ($\mathcal{A} \subseteq \Omega$) in problem $\mathbf{P0}''$ is non-negative, monotone, and submodular.*

Proof. Firstly, the objective function $g(\mathcal{A})$ is non-negative, since if $\mathcal{A} = \emptyset$, the corresponding optimal workload allocation $y^*_{ij} = 0$ for $\forall i, j \in \mathcal{N}$ and accordingly $g(\emptyset) = 0$. And $g(\mathcal{A}) \ge 0$ for all $\mathcal{A} \subseteq \Omega$ due to the non-negative expression of the objective function in problem $\mathbf{P0}$. Secondly, $g(\mathcal{A})$ ($\mathcal{A} \subseteq \Omega$) is monotone, since if we add more elements into $\mathcal{A}$, the only affected constraint (10) will be relaxed and the feasible solution region of the LP problem for $\mathbf{Y}$ will be expanded, which results in the increased optimal value.

Lastly, according to Definition 1, to prove $g(\mathcal{A})$ is a submodular function, we need to show that for any feasible $\mathcal{A}_1, \mathcal{A}_2 \subseteq \Omega$ and any $(j_1, s_1) \in \Omega \setminus \mathcal{A}_2$ satisfying that $\mathcal{A}_1 \subseteq \mathcal{A}_2$ and $\mathcal{A}_2 \cup \{(j_1, s_1)\})$ is feasible, it holds:

$$g(\mathcal{A}_1 \cup \{(j_1, s_1)\}) - g(\mathcal{A}_1) \ge g(\mathcal{A}_2 \cup \{(j_1, s_1)\}) - g(\mathcal{A}_2). \tag{13}$$

To specify the effect of (j_1, s_1) on the objective function $g(\cdot)$, we unfold the workload allocation variable y_{ij} from the AP level to the server level. Specifically, we introduce a continuous variable $z_{ijs} \in [0,1]$ to indicate the proportion of AP $i \in \mathcal{N}$'s workload that is assigned to the edge server $s \in \mathcal{S}$ at AP $j \in \mathcal{N}$. Donate $\mathbf{Z} := \{z_{ijs}\}, i \in \mathcal{N}, j \in \mathcal{N}, s \in \mathcal{S}$. Whenever we get a feasible allocation solution $\mathbf{Y}$, we can always split each y_{ij} to z_{ijs} by computing the capacity of each server placed at AP j. Obviously we have $y_{ij} = \sum_{s=1}^{S} z_{ijs}, \forall i \in \mathcal{N}, j \in \mathcal{N}$. For any server $s \in \mathcal{S}$ placed at AP $j \in \mathcal{N}$, its workload is $\sum_{i=1}^{N} z_{ijs}$.

Suppose that $\mathbf{Y}^{(1)} = \{y^{(1)}_{ij}\}_{i\in\mathcal{N},j\in\mathcal{N}}$ and $\widehat{\mathbf{Y}}^{(1)} = \{\hat{y}^{(1)}_{ij}\}_{i\in\mathcal{N},j\in\mathcal{N}}$ is the optimal workload allocation solution obtained by solving $\mathbf{P0}'$ under the server placement variable $\mathcal{A}_1$ and $\mathcal{A}_1 \cup \{(j_1, s_1)\}$, respectively. Then we can get the optimal $\mathbf{Z}^{(1)} = \{z^{(1)}_{ijs}\}, i \in \mathcal{N}, j \in \mathcal{N}, s \in \mathcal{S}$ and $\widehat{\mathbf{Z}}^{(1)} = \{\hat{z}^{(1)}_{ijs}\}, i \in \mathcal{N}, j \in \mathcal{N}, s \in \mathcal{S}$ by unfolding $\mathbf{Y}^{(1)}$ and

$\widehat{\mathbf{Y}}^{(1)}$, respectively. Similarly, we can get the optimal workload allocation solution $\mathbf{Z}^{(2)} = \{z_{ijs}^{(2)}\}, i \in \mathcal{N}, j \in \mathcal{N}, s \in \mathcal{S}$ and $\widehat{\mathbf{Z}}^{(2)} = \{\hat{z}_{ijs}^{(2)}\}, i \in \mathcal{N}, j \in \mathcal{N}, s \in \mathcal{S}$ under the server placement variable $\mathcal{A}_2$ and $\mathcal{A}_2 \cup \{(j_1, s_1)\}$, respectively. Noting that the optimal workload allocation solution may be not unique, we specify $\widehat{\mathbf{Z}}^{(1)}$ and $\widehat{\mathbf{Z}}^{(2)}$ as the solutions that minimize the workload allocated to edge server s_1 at AP j_1, *i.e.*, $\sum_{i=1}^{N} w_i z_{ij_1s_1}$. Thus, we can rewrite the objective values of Non-RSP under $\mathcal{A}_1, \mathcal{A}_1 \cup \{(j_1, s_1)\}, \mathcal{A}_2, \mathcal{A}_2 \cup \{(j_1, s_1)\}$, respectively, as follows:

$$g(\mathcal{A}_1) = \sum_{(j,s)\in\mathcal{A}_1} \sum_{i=1}^{N} w_i z_{ijs}^{(1)}, \tag{14}$$

$$g(\mathcal{A}_1 \cup \{(j_1, s_1)\}) = \sum_{(j,s)\in\mathcal{A}_1} \sum_{i=1}^{N} w_i \hat{z}_{ijs}^{(1)} + \sum_{i=1}^{N} w_i \hat{z}_{ij_1s_1}^{(1)}, \tag{15}$$

$$g(\mathcal{A}_2) = \sum_{(j,s)\in\mathcal{A}_2} \sum_{i=1}^{N} w_i z_{ijs}^{(2)}, \tag{16}$$

$$g(\mathcal{A}_2 \cup \{(j_1, s_1)\}) = \sum_{(j,s)\in\mathcal{A}_2} \sum_{i=1}^{N} w_i \hat{z}_{ijs}^{(2)} + \sum_{i=1}^{N} w_i \hat{z}_{ij_1s_1}^{(2)}. \tag{17}$$

Then, we obtain:

$$\text{LHS of (13)} = \sum_{(j,s)\in\mathcal{A}_1} \sum_{i=1}^{N} w_i (\hat{z}_{ijs}^{(1)} - z_{ijs}^{(1)}) + \sum_{i=1}^{N} w_i \hat{z}_{ij_1s_1}^{(1)}, \tag{18}$$

$$\text{RHS of (13)} = \sum_{(j,s)\in\mathcal{A}_2} \sum_{i=1}^{N} w_i (\hat{z}_{ijs}^{(2)} - w_i z_{ijs}^{(2)}) + \sum_{i=1}^{N} w_i \hat{z}_{ij_1s_1}^{(2)}. \tag{19}$$

The first item in (18) is the difference in the workload served by the edge servers in $\mathcal{A}_1$ before/after placing the edge server s_1 at AP j_1. Since $\mathbf{Z}^{(1)}$ is the optimal workload allocation solution under $\mathcal{A}_1$, $\sum_{(j,s)\in\mathcal{A}_1} \sum_{i=1}^{N} w_i (\hat{z}_{ijs}^{(1)} - z_{ijs}^{(1)}) \leq 0$. Then we can proof $\sum_{(j,s)\in\mathcal{A}_1} \sum_{i=1}^{N} w_i (\hat{z}_{ijs}^{(1)} - z_{ijs}^{(1)}) = 0$ by contradiction. Assume that $\sum_{(j,s)\in\mathcal{A}_1} \sum_{i=1}^{N} w_i (\hat{z}_{ijs}^{(1)} - z_{ijs}^{(1)}) < 0$. Then we can infer that the workload served by the servers in $\mathcal{A}_1$ decreases after placing the edge server s_1 at AP j_1 because some workload allocated to $\mathcal{A}_1$ in $\mathbf{Z}^{(1)}$ is re-directed to s_1 in $\widehat{\mathbf{Z}}^{(1)}$. Then we always can return some workload from s_1 to servers in $\mathcal{A}_1$ while keeping the target value unchanged. Thus we can construct a workload allocation solution under $\mathcal{A}_1 \cup \{(j_1, s_1)\}$ with less workload allocated to server s_1 than $\widehat{\mathbf{Z}}^{(1)}$, which contradicts with the definition of $\widehat{\mathbf{Z}}^{(1)}$. Therefore, the first item of (18) is zero. Similarly, the first item of (19) is also zero.

The second items in (18) and (19) are the workloads served by the server (j_1, s_1) in the optimal workload allocation schemes $\widehat{\mathbf{Z}}^{(1)}$ and $\widehat{\mathbf{Z}}^{(2)}$, respectively. Since the workload served by the server (j_1, s_1) is minimized, adding $\mathcal{A}_2 \setminus \mathcal{A}_1$ to $\mathcal{A}_1 \cup (j_1, s_1)$ may offload workload of (j_1, s_1) to the servers in $\mathcal{A}_2 \setminus \mathcal{A}_1$. Thus, we have $\sum_{i=1}^{N} w_i \hat{z}_{ij_1s_1}^{(1)} \geq \sum_{i=1}^{N} w_i \hat{z}_{ij_1s_1}^{(2)}$, which implies that (13) holds and the lemma is proved. □

Definition 2. *(Independence system [43]) Denote Ω as a set of elements and $\mathcal{G} \subseteq 2^{\Omega}$ as a collection of subsets of Ω. The pair $(\Omega, \mathcal{G})$ is called an independence system, if the following conditions hold: 1) $\emptyset \in \mathcal{G}$; 2) if $\mathcal{A}_1 \in \mathcal{G}$ and $\mathcal{A}_2 \subseteq \mathcal{A}_1$, then $\mathcal{A}_2 \in \mathcal{G}$. The subsets of Ω in $\mathcal{G}$ are called independent. For any set A of elements, an inclusive-wise maximal subset $\mathcal{T}$ is called a basis of $\mathcal{A}$.*

Definition 3. *(p-independence system [43]) Given an independence system $(\Omega, \mathcal{G})$ and a subset $\mathcal{A} \subseteq \Omega$, the rank $r(\mathcal{A})$ is defined as the cardinality of the largest basis of $\mathcal{A}$, and the lower rank $l(\mathcal{A})$ is the cardinality of the smallest basis of $\mathcal{A}$. The independence system is called a p-independent system (or a p-system) if $\max_{\mathcal{A}\subseteq\Omega} \frac{r(\mathcal{A})}{l(\mathcal{A})} \leq p$.*

Lemma 3. *The constraints of (11) and (12) form a p-independence system for $p = 1 + \lceil \frac{\max\ c_{js}}{\min\ c_{js}} \rceil$.*

Proof. Consider a pair $(\Omega, \mathcal{G}_p)$, where $\Omega := \{(j, s) | \forall j \in \mathcal{N}, s \in \mathcal{S}\}$ and $\mathcal{G}_p$ is the set of all feasible sets of $\mathbf{P0}''$. Then, we have $\emptyset \in \mathcal{G}_p$ and if $Y \in \mathcal{G}_p$ and $X \subseteq Y$, $X \in \mathcal{G}_p$ as the subset of any feasible server placement is also feasible. By Definition 2, $(\Omega, \mathcal{G}_p)$ is an independence system. Consider any $\mathcal{A} \subseteq \Omega$ and any two maximal feasible edge server placement sets $\mathcal{A}_1, \mathcal{A}_2 \subseteq \mathcal{A}$. To add a pair $(j, s) \in \mathcal{A}_1 \setminus \mathcal{A}_2$ to $\mathcal{A}_2$, we need to take out several elements from $\mathcal{A}_2$ to meet the constraints. To satisfy constraint (11), we need to take out at most one element. To satisfy constraint (12), we need to take out at most $\lceil \frac{\max\ c_{js}}{\min\ c_{js}} \rceil$. Recall that c_{js} is the deployment cost to place server s at AP j, $c_{js} > 0, \forall j \in \mathcal{N}, s \in \mathcal{S}$. So, to add a pair $(j, s) \in \mathcal{A}_1 \setminus \mathcal{A}_2$ to $\mathcal{A}_2$, we need to take out at most $p = 1 + \lceil \frac{\max\ c_{js}}{\min\ c_{js}} \rceil$ elements from $\mathcal{A}_2$. Do that swapping operation to each element in $\mathcal{A}_1 \setminus \mathcal{A}_2$, and the number of the elements are reduced by at most p-fold in modifying $\mathcal{A}_2$ into $\mathcal{A}_1$. So the cardinality of $\mathcal{A}_1$ is at most p times the cardinality of $\mathcal{A}_2$. Since the above holds for any $\mathcal{A} \subseteq \Omega$ and any maximal independent subsets of $\mathcal{A}$, the constraints of (11) and (12) form a p-independence system $(\Omega, \mathcal{G}_p)$. □

4.2 Formulation of RSP and Problem Hardness

Based on $\mathbf{P0}''$ and Lemma 3, we mathematically formulate the RSP problem as follows:

$$(\mathbf{P1}) \quad \max_{\mathcal{A}\subseteq\Omega, \mathcal{A}\in\mathcal{G}_p} \ \min_{\mathcal{H}\subseteq\mathcal{A}, |\mathcal{H}|\leq k} \ g(\mathcal{A} \setminus \mathcal{H}), \tag{20}$$

where $(\Omega, \mathcal{G}_p)$ is the p-independence system formed by constraints of (11) and (12) and $k \leq S - 1$ is the maximum number of server failures that controls the degree of robustness. In practice, correlation among neighbor server failures may exist because of shared key configuration or similar environment between neighboring servers. However, we would like to point out that our proposed formulation incorporates no more than k server failures, no matter what correlation among these failures is. $\mathbf{P1}$ aims to maximize the expected total served workload, in spite of *worst-case failures* of the predetermined server placement strategy, which compromises the minimization of problem $\mathbf{P0}''$. Set $\mathcal{A}$ corresponds to the predetermined server placement strategy, where $(j, s) \in \mathcal{A}$ implies that server s should be deployed at AP j. Set $\mathcal{H}$ represents the removed set from $\mathcal{A}$, where $(j, s) \in \mathcal{H}$ means that the server s placed at AP j fails and the workload allocated to this server will be uploaded to the remote cloud. In practice, the failure of servers is unknown in advance. The formulation of $\mathbf{P1}$ is suitable for volatile scenarios where there is no prior information of the failure of edge servers due to various unpredictable factors.

Theorem 1. *Problem RSP (i.e., $\mathbf{P1}$) is NP-hard.*

Algorithm 1: Robust Server Placement Algorithm for Problem $\mathbf{P1}$

Input: Ground set Ω, p-independence system $(\Omega, \mathcal{G}_p)$, parameter k, value access to function $g(\mathcal{A})$.
Output: Set $\mathcal{A}$.

1 Initialize $\mathcal{A}_1 \leftarrow \emptyset$, $\mathcal{A}_2 \leftarrow \emptyset$, $\mathcal{T}_1 \leftarrow \emptyset$, $\mathcal{T}_2 \leftarrow \emptyset$.
2 **while** $\mathcal{T}_1 \neq \Omega$ and $|\mathcal{A}_1| < k$ **do**
3 $\{e\} \leftarrow \arg\max_{\{v\} \subseteq \Omega \setminus \mathcal{T}_1, \mathcal{A}_1 \cup \{v\} \in \mathcal{G}_p} g(\{v\})$.
4 $\mathcal{A}_1 \leftarrow \mathcal{A}_1 \cup \{e\}$.
5 $\mathcal{T}_1 \leftarrow \mathcal{T}_1 \cup \{e\}$.
6 $\mathcal{T}_2 \leftarrow \mathcal{T}_2 \cup \{e\}$.
7 **for** each $v \in \Omega \setminus \mathcal{T}_1$ **do**
8 **if** $\mathcal{A}_1 \cup \{v\} \notin \mathcal{G}_p$ **then**
9 $\mathcal{T}_1 \leftarrow \mathcal{T}_1 \cup \{v\}$.
10 $\mathcal{T}_2 \leftarrow \mathcal{T}_2 \cup \{v\}$.
11 **while** $\mathcal{T}_2 \neq \Omega$ **do**
12 $\{e\} \leftarrow \arg\max_{\{v\} \subseteq \Omega \setminus \mathcal{T}_1, \mathcal{A}_1 \cup \mathcal{A}_2 \cup \{v\} \in \mathcal{G}_p} g(\mathcal{A}_2 \cup \{v\})$.
13 $\mathcal{A}_2 \leftarrow \mathcal{A}_2 \cup \{e\}$.
14 $\mathcal{T}_2 \leftarrow \mathcal{T}_2 \cup \{e\}$.
15 **for** each $v \in \Omega \setminus \mathcal{T}_1$ **do**
16 **if** $\mathcal{A}_1 \cup \mathcal{A}_2 \cup \{v\} \notin \mathcal{G}_p$ **then**
17 $\mathcal{T}_2 \leftarrow \mathcal{T}_2 \cup \{v\}$.
18 $\mathcal{A} \leftarrow \mathcal{A}_1 \cup \mathcal{A}_2$.
19 **Return** $\mathcal{A}$.

Proof. The RSP problem is NP-hard, since when $k = 0$, it reduces to a submodular monotone maximization problem with a p-independence system constraint, which is NP-hard [42]. The theorem is thus proved. □

To our best, there exists no approximation algorithm for robust submodular max-min problems (with the similar *deletion-robust* objective expression as in $\mathbf{P1}$) with p-independence system constraints. Very recently, there emerge approximation solutions for robust submodular max-min problems with a cardinality constraint [14], [15], [16], [17], [18] or a single matroid constraint [19], [20], which cannot be applied to the RSP problem due to the p-independence system constraint.

5 Solution

In this section, we first give a detailed introduction of our algorithm to RSP, then theoretically analyze the performance of the algorithm, and lastly discuss how to distributively implement the proposed algorithm in practice.

5.1 Algorithm

The robust server placement algorithm is presented in detail in Algorithm 1. In the following, we first introduce its key idea, and then describe the steps in the algorithm.

5.1.1 Key Idea of Algorithm 1

Note that the objective of $\mathbf{P1}$ is to select set $\mathcal{A}$ towards maximizing value of function $g(\cdot)$, despite of a worst removal $\mathcal{H}$ from set $\mathcal{A}$. The value of function $g(\cdot)$ is actually evaluated by the set $\mathcal{A} \setminus \mathcal{H}$ instead of set $\mathcal{A}$. To deal with this situation, Algorithm 1 constructs set $\mathcal{A}$ as the union of two sets $\mathcal{A}_1$ and $\mathcal{A}_2$, whose meanings are provided in detail as follows.

Set $\mathcal{A}_1$ approximates the worst-case removed set $\mathcal{H}$ from $\mathcal{A}$. Set $\mathcal{A}_1$ is used to capture worst-case removal from all the elements that Algorithm 1 are going to select in the set $\mathcal{A}$. In other words, the set $\mathcal{A}_1$ is aimed to act as a "bait" to a rival who selects to remove the subset whose contribution to the objective value is maximized. Nevertheless, the problem of selecting a maximum-contribution set subject to a p-independence system constraint is an intractable problem [42]. Thus, Algorithm 1 uses the greedy method to approximate the maximum-contribution set by selecting elements with largest marginal contribution to the objective function value (lines 2-10).

Set $\mathcal{A}_2$ approximates the best set $\mathcal{A} \setminus \mathcal{A}_1$ with $\mathcal{A}_1$ removed from set $\mathcal{A}$ ("best" with respect to maximize the value of the objective function). Assuming that set $\mathcal{A}_1$ is the set selected to be removed from set $\mathcal{A}$. Algorithm 1 needs to select a set of elements from $\mathcal{A} \setminus \mathcal{A}_1$ which maximize the objective value. However, the problem of selecting such a best set over a p-independence system constraint is also an intractable problem. Accordingly, Algorithm 1 aims to approximate the best set with greedy procedure (lines 11-17).

5.1.2 Description of Steps in Algorithm 1

Next we describe the steps of Algorithm 1 in detail.

Initialization (line 1): Four auxiliary sets are defined, *i.e.*, $\mathcal{A}_1, \mathcal{A}_2, \mathcal{T}_1, \mathcal{T}_2$, which are initialized as empty sets. $\mathcal{A}_1$ and $\mathcal{A}_2$ are used to construct the set $\mathcal{A}$, which is the solution to $\mathbf{P1}$ selected by Algorithm 1. $\mathcal{T}_1$ and $\mathcal{T}_2$ are used to support the construction of set $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively. Specifically, $\mathcal{T}_1$ stores the elements which have already been added to $\mathcal{A}_1$ or cannot be included in $\mathcal{A}_1$. And $\mathcal{T}_2$ stores the elements which have already been added to $\mathcal{A}_2$ or cannot be included in $\mathcal{A}_2$.

Construction of set $\mathcal{A}_1$ (lines 2-10): Algorithm 1 sequentially constructs $\mathcal{A}_1$ by selecting elements greedily from Ω. Among all the feasible elements that satisfy the p-independence system constraint, Algorithm 1 picks the element with the highest marginal value of $g(\cdot)$ and adds it to $\mathcal{A}_1$ in each iteration. Meanwhile, the cardinality of $\mathcal{A}_1$ is no more than k.

Construction of set $\mathcal{A}_2$ (lines 11-17): $\mathcal{A}_2$ is also sequentially constructed by greedily selecting one element from $\Omega \setminus \mathcal{A}_1$ at each time. Each added element e should maximize the value of $g(\mathcal{A}_2 \cup \{e\})$ with $\mathcal{A}_1 \cup \mathcal{A}_2$ belonging to $\mathcal{G}_p$.

Construction of set $\mathcal{A}$ (line 18) : Finally, set $\mathcal{A}$ is constructed as the union of sets $\mathcal{A}_1$ and $\mathcal{A}_2$.

In sum, Algorithm 1 sequentially constructs the sets $\mathcal{A}_1$ and $\mathcal{A}_2$ with their union as the solution to $\mathbf{P1}$.

5.2 Performance Analysis

In this subsection, we analyze the performance of Algorithm 1 in terms of approximation ratio and time complexity. We first provide the definition of curvature for monotone submodular functions in the following.

Definition 4. *(Curvature of monotone submodular functions [20]) Consider a non-decreasing submodular set function $f : 2^{\Omega} \mapsto \mathbb{R}$ over a finite set Ω. Assume for any element $e \in \Omega$, $f(\{e\}) \neq 0$. The curvature of function $f(\cdot)$ is defined by $\kappa_f := 1 - \min_{e \in \Omega} \frac{f(\Omega) - f(\Omega \setminus \{e\})}{f(\{e\})}$.*

According to the above definition, for any non-decreasing submodular function $f : 2^{\Omega} \mapsto \mathbb{R}$, $0 \leq \kappa_f \leq 1$

holds. The value of κ_f actually measures how far $f(\cdot)$ is from modularity. If $\kappa_f = 0$, f is modular, since for any element $e \in \mathcal{A} \in \mathcal{I}$, we have $f(\mathcal{A}) - f(\mathcal{A}\backslash\{e\}) = f(\{e\})$. In another extreme case, if $\kappa_f = 1$, there exists an element $e \in \Omega$ such that $f(\Omega\backslash\{e\}) = f(\Omega)$, which implies that e loses its contribution to $f(\Omega)$ in the presence of $\Omega \setminus \{e\}$. Based on the definition of curvature, the approximation performance of Algorithm 1 is analyzed as follows.

Theorem 2. *Given a set $\mathcal{A}$ as a feasible solution to the RSP problem (i.e., $\mathbf{P1}$), let $\mathcal{H}^*(\mathcal{A})$ be an optimal set removal from $\mathcal{A}$ in $\mathbf{P1}$, that is, $\mathcal{H}^*(\mathcal{A}) \in \arg\min_{\mathcal{H}\subseteq\mathcal{A},|\mathcal{H}|\leq k} g(\mathcal{A}\setminus\mathcal{H})$. Algorithm 1 achieves an approximation ratio of $\frac{1}{1+p}\max\{1-\kappa_g, \frac{1}{S-k}, \frac{1}{1+k}\}$, i.e.,*

$$\frac{g(\mathcal{A}\setminus\mathcal{H}^*(\mathcal{A}))}{g(\mathcal{A}^*\setminus\mathcal{H}^*(\mathcal{A}^*))} \geq \frac{1}{1+p}\max\{1-\kappa_g, \frac{1}{S-k}, \frac{1}{1+k}\}, \quad (21)$$

where $\mathcal{A}$ is the output of Algorithm 1, $g(\mathcal{A}^*\setminus\mathcal{H}^*(\mathcal{A}^*))$ is the optimal value of $\mathbf{P1}$, p is the parameter defined in Lemma 3, and κ_g is the curvature of function $g(\cdot)$.

Proof. (Sketch) Recall that the constraints of problem $\mathbf{P1}$ consist of a p-independence system $(\Omega, \mathcal{G}_p)$, and $\mathcal{A}_1$ and $\mathcal{A}_2$ are the sets constructed by Algorithm 1 in lines 2-10 and lines 11-17, respectively. First, we prove that after removing a set from $\mathcal{G}_p$, it is still a p-independence system. That is, $(\Omega\setminus\mathcal{V}, \mathcal{G}'_p)$ is a p-independence system, where $\mathcal{V} \in \mathcal{G}_p$ and $\mathcal{G}'_p := \{\mathcal{U} : \mathcal{U} \subseteq \Omega\setminus\mathcal{V}, \mathcal{U}\cup\mathcal{V} \in \mathcal{G}_p\}$. Based on that, we then prove that $g(\mathcal{A}_2) \geq \frac{1}{1+p}\max_{\mathcal{U}\subseteq\Omega\setminus\mathcal{A}_1, \mathcal{U}\cup\mathcal{A}_1\in\mathcal{G}_p} g(\mathcal{U})$ holds. We further prove that $\max_{\mathcal{U}\subseteq\Omega\setminus\mathcal{A}_1, \mathcal{U}\cup\mathcal{A}_1\in\mathcal{G}_p} g(\mathcal{U}) \geq g(\mathcal{A}^*\setminus\mathcal{H}^*(\mathcal{A}^*))$, which is used to prove the first part in the approximation ratio. Second, we prove that for any element $a \in \mathcal{A}_1$ and $a' \in \mathcal{A}_2$, $g(a) \geq g(a')$ always holds. Based on that and some fundamental submodular properties, we prove the second part and last part in the approximation ratio, respectively. The complete proof is provided in the Appendix. □

Remark 1: Note that in Theorem 2, the achieved approximation ratio is not a constant since the inside parameters such as p and κ_g are problem-specific and vary with different instance of the RSP problem. However, it is fixed under any given problem instance because the value of any inside parameter in the approximation ratio is fixed for any given RSP problem.

Remark 2: When $k = 0$, RSP is reduced to a monotone submodular maximization problem with a p-independence system constraint and the approximate ratio equals to $\frac{1}{1+p}$, which is in accordance with the conclusion of [42]. The approximation ratio of Algorithm 1 can be rewritten as $\frac{1}{1+p}\max\{1-\kappa_g, \max\{\frac{1}{S-k}, \frac{1}{1+k}\}\}$. With the increase of k, the k-dependent approximation bound $\max\{\frac{1}{S-k}, \frac{1}{1+k}\}$ decreases when $k \leq \frac{S-1}{2}$, and increases when $k \geq \frac{S-1}{2}$. Besides, when the curvature of function $g(\cdot)$ is low, RSP can be well approximated.

Theorem 3. *The time complexity of Algorithm 1 is $O(NS^2\tau_g)$, where N is the number of candidate APs, S is the number of edge servers, and τ_g is the time of solving the LP problem $\mathbf{P0}'$ to obtain the evaluation of the objective function $g(\cdot)$ in problem $\mathbf{P0}''$.*

Proof. Since the value of the objective function $g(\mathcal{A})$ is not in its explicit form and can be obtained after solving the LP problem $\mathbf{P0}'$ given $\mathbf{X}$ from set $\mathcal{A} \subseteq \Omega$, we let the time of solving $\mathbf{P0}'$ to obtain the value of $g(\cdot)$ as τ_g, which can be done in polynomial time of N. We first evaluate the running time of lines 2-10: it needs at most $k[|\Omega|\tau_g + |\Omega|\log(|\Omega|) + |\Omega| + O(\log(|\Omega|)) + |\Omega|]$ time, as they are repeated k times at most, and line 3 needs at most $|\Omega|$ evaluations of function $g(\cdot)$ (*i.e.*, $|\Omega|\tau_g$), and the sorting requires another $|\Omega|\log(|\Omega|) + |\Omega| + O(\log(|\Omega|))$ time using some classical algorithm such as the merge sort algorithm, and lines 7-10 needs at most $|\Omega|$ time. The running time of lines 11-17 is very similar to that of lines 2-10 except that lines 11-17 are repeated at most $S-k$ times. Therefore, the total time complexity of Algorithm 1 is $(k+S-k)[|\Omega|\tau_g + |\Omega|\log(|\Omega|) + |\Omega| + O(\log(|\Omega|)) + |\Omega|] = O(S|\Omega|\tau_g) = O(NS^2\tau_g)$, which is polynomial of N and S. The theorem is thus proved. □

5.3 Distributed Implementation Issues of Algorithm 1

Although Algorithm 1 is proven to be time polynomial over the number of candidate APs N and number of edge servers S, running it may be prohibitively expensive to cover a wide geographic area in real world. For example, there may be tens of thousands of candidate APs in a metropolitan area. Our proposed algorithm can be implemented using the double partition methods [50], [51], in a distributed way as follows. First, we can divide the whole area into multiple independent small subareas such that we can consider the robust server placement problem in each of them separately. The performance loss arisen from some partition that makes some user request previously covered by multiple APs served by only some of the APs can be analyzed and bounded similar to [51]. Second, under the given budget of servers, we should intelligently determine the number of placed servers in each subarea and check the deployment budget constraint to maximize the overall served workload. To be specific, we can greedily determine the number of placed servers in each subarea such that the performance is bounded by a constant factor, which results in the algorithm distributively implemented with a constant-ratio performance loss.

6 Enhanced Solution with Improved Approximation Ratio

In this section, instead of treating constraints (11) and (12) as a whole p-independence system, we explore the individual property of each constraint, and find that the former one is a matroid constraint while the second one is obviously a knapsack constraint. Thus, motivated by this, we propose a novel "greedy-and-local search" algorithm for problem $\mathbf{P1}$ in Alg. 2, which is proved to achieve strictly better approximation ratio than Alg. 1.

6.1 Analysis of Constraints

To begin with, constraint (12) is obviously a knapsack constraint. Next, we prove that constraint (11) is a matroid constraint. We first give some fundamental definitions in the following.

Definition 5. *(Matroid [52]) Let Ω and $\mathcal{G} \subseteq 2^\Omega$ denote a set of elements and a collection of all subsets of Ω, respectively. The pair*

$(\Omega, \mathcal{G})$ is called a matroid if the following three conditions hold: 1) $\mathcal{G} \neq \emptyset$ (non-empty); 2) if $\mathcal{A}_1 \in \mathcal{G}$ and $\mathcal{A}_2 \subseteq \mathcal{A}_1$, then $\mathcal{A}_2 \in \mathcal{G}$ (monotone); 3) if $\mathcal{A}_1, \mathcal{A}_2 \in \mathcal{G}$ and $|\mathcal{A}_1| < |\mathcal{A}_2|$, then there exists an element $a \in \mathcal{A}_2$, such that $\mathcal{A}_1 \cup \{a\} \in \mathcal{G}$ (exchange).

Lemma 4. *Recall set $\Omega = \{(j,s)|\forall j \in \mathcal{N}, s \in \mathcal{S}\}$ and define $\mathcal{G} := \{\mathcal{A}|\mathcal{A} \subseteq \Omega, \forall a_1 := (j_1, s_1), a_2 := (j_2, s_2) \in \mathcal{A}, s_1 \neq s_2\}$. Then, constraint (11) is a matroid constraint.*

Proof. Assume at least two APs and two edge servers exist in the network, *i.e.*, $N \geq 2$ and $S \geq 2$, respectively; otherwise, the problem is trivial. We only need to prove that the defined pair $(\Omega, \mathcal{G})$ is a matroid. Specifically, we prove that $(\Omega, \mathcal{G})$ satisfies the properties of a matroid by Definition 5 one by one. Firstly, the non-emptiness of $\mathcal{G}$ holds due to the aforementioned assumption. Secondly, for the monotonicity, if set $\mathcal{A}_1 \subseteq \mathcal{A}_2 \in \mathcal{G}$, we have $\mathcal{A}_1 \in \mathcal{G}$. Otherwise, there exist $a_1, a_2 \in \mathcal{A}_1$ ($a_1 \neq a_2$) with the same second components therein. Owing to $\mathcal{A}_1 \subseteq \mathcal{A}_2$, we have $a_1, a_2 \in \mathcal{A}_2$, which obviously contradicts with $A_2 \in \mathcal{G}$. Thirdly, suppose $\mathcal{A}_1, \mathcal{A}_2 \in \mathcal{G}$ and $|\mathcal{A}_1| < |\mathcal{A}_2|$. If there exists no element $a \in \mathcal{A}_2$ such that $\mathcal{A}_1 \cup \{a\} \in \mathcal{G}$, that is to say, for $\forall a \in \mathcal{A}_2$, $\mathcal{A}_1 \cup \{a\} \notin \mathcal{G}$ holds. Because of $\mathcal{A}_1 \in \mathcal{G}$, any element belonging to $\mathcal{A}_2$ shares the exactly same first component with some element in $\mathcal{A}_1$. Due to $|\mathcal{A}_1| < |\mathcal{A}_2|$, there must exist two different elements $a_1, a_2 \in \mathcal{A}_2$ and an element $a \in \mathcal{A}_1$, all of which share the identical second component. This results in that the second components of a_1 and a_2 are also the same, which contradicts with $a_1, a_2 \in \mathcal{A}_2 \in \mathcal{G}$. Therefore, the exchange property also holds, which concludes this lemma. □

6.2 Algorithm

The "greedy&local search" algorithm for problem $\mathbf{P1}$ is described in detail in Alg. 2. In the following, we first introduce the key idea, and then explain the meaning of each step in the proposed algorithm.

6.2.1 Key Idea of Algorithm 2

Different from the "twice greedy" approach in Algorithm 1, we here propose a novel "greedy&local search" approach to achieve better approximation performance. To be specific, in Alg. 2, we similarly construct two sets $\mathcal{A}_1$ and $\mathcal{A}_2$ whose union forms the output set $\mathcal{A}$, while they are respectively constructed by "greedy" and "local search". The key insight is that, Alg. 2 utilizes the local search technology in the second set construction in view of the matroid constraint, to achieve better approximation ratio than Alg. 1.

6.2.2 Description of Steps in Algorithm 2

Initialization (line 1): similar to Alg. 1, three auxiliary sets $\mathcal{A}_1$, $\mathcal{A}_2$, and $\mathcal{T}$ are initialized first, where the union of the former two is used to construct the output set $\mathcal{A}$, while the last one records the elements of Ω that have been included or cannot be included in $\mathcal{A}_1$.

Construction of set $\mathcal{A}_1$ (lines 2-8): set $\mathcal{A}_1$ is constructed similar to that in lines 2-10 in Alg. 1, while the sole difference is that satisfying the p-independence system constraint becomes into satisfying the matroid constraint as well as the knapsack constraint.

Algorithm 2: Greedy-and-Local Search Algorithm for Problem $\mathbf{P1}$

Input: Matroid $(\Omega, \mathcal{G})$, budget C_0, parameters k and $\theta \in (0, \frac{4}{e^2-1}]$, value access to function $g(\mathcal{A})$.
Output: Set $\mathcal{A}$.

1 Initialize $\mathcal{A}_1 \leftarrow \emptyset$, $\mathcal{A}_2 \leftarrow \emptyset$, $\mathcal{T} \leftarrow \emptyset$.
2 **while** $\mathcal{T} \neq \Omega$ and $|\mathcal{A}_1| < k$ **do**
3 $\{a\} \leftarrow \arg\max_{\{v\} \subseteq \Omega \setminus \mathcal{T}, \mathcal{A}_1 \cup \{v\} \in \mathcal{G}, C_{\mathcal{A}_1 \cup \{v\}} \leq C_0, |\mathcal{A}_1 \cup \{v\}| \leq k} g(\{v\})$.
4 $\mathcal{A}_1 \leftarrow \mathcal{A}_1 \cup \{a\}$.
5 $\mathcal{T} \leftarrow \mathcal{T} \cup \{a\}$.
6 **for** each $v \in \Omega \setminus \mathcal{T}$ **do**
7 **if** $\mathcal{A}_1 \cup \{v\} \notin \mathcal{G} || C_{\mathcal{A}_1 \cup \{v\}} > C_0$ **then**
8 $\mathcal{T} \leftarrow \mathcal{T} \cup \{v\}$.
9 Let $\mathcal{A}_2 = \{a_1, a_2\}$, where $a_1 := \arg\max_{v \in \Omega \setminus \mathcal{A}_1, \mathcal{A}_1 \cup \{v\} \in \mathcal{G}, C_{\mathcal{A}_1 \cup \{v\}} \leq C_0} g(\{v\})$, $a_2 := \arg\max_{v \in \Omega \setminus \mathcal{A}_1', \mathcal{A}_1' \cup \{v\} \in \mathcal{G}, C_{\mathcal{A}_1' \cup \{v\}} \leq C_0} g(\{a_1, v\}) - g(\{a_1\})$, $\mathcal{A}_1' = \mathcal{A}_1 \cup \{a_1\}$, and set $exchange \leftarrow true$.
10 **while** *exchange* **do**
11 $exchange \leftarrow false$.
12 **while** *not(exchange)* **do**
13 **if** there exist $a \in (\Omega \setminus \mathcal{A}_1) \setminus \mathcal{A}_2$ and $a' \in \mathcal{A}_2 \cup \{\emptyset\}$, such that $\mathcal{A}_1 \cup (\mathcal{A}_2 \setminus \{a'\}) \cup \{a\} \in \mathcal{G}, C_{\mathcal{A}_1 \cup (\mathcal{A}_2 \setminus \{a'\}) \cup \{a\}} \leq C_0$, and $g((\mathcal{A}_2 \setminus \{a'\}) \cup \{a\}) > (1 + \frac{\theta}{(NS - |\mathcal{A}_1|)^2}) g(\mathcal{A}_2)$ hold **then**
14 $\mathcal{A}_2 \leftarrow (\mathcal{A}_2 \setminus \{a'\}) \cup \{a\}$.
15 $exchange \leftarrow true$.
16 $\mathcal{A} \leftarrow \mathcal{A}_1 \cup \mathcal{A}_2$.
17 **Return** $\mathcal{A}$.

Construction of set $\mathcal{A}_2$ (lines 9-14): set $\mathcal{A}_2$ is constructed by applying a local search over $\Omega \setminus \mathcal{A}_1$ under the matroid and knapsack constraints. To be specific, Alg. 2 firstly initializes $\mathcal{A}_2$ by $\{a_1, a_2\}$, where the two elements obtain the largest and second largest marginal value of $g(\cdot)$ in $\Omega \setminus \mathcal{A}_1$, respectively (line 9). Next, a local greedy swap between $(\Omega \setminus \mathcal{A}_1) \setminus \mathcal{A}_2$ and $\mathcal{A}_2 \cup \{\emptyset\}$ is applied to expand $\mathcal{A}_2$ within $\Omega \setminus \mathcal{A}_1$ in an iterative manner, only if these conditions hold (lines 10-14): i) after swap the new set $\mathcal{A}_2$ satisfies both the matroid and knapsack constraints; ii) the increased value of $g(\mathcal{A}_2)$ is at least by a factor of $1 + \frac{\theta}{|NS - \mathcal{A}_1|^2}$.

6.3 Performance Analysis

The performance of Alg. 2 including approximation ratio and time computation complexity can be analyzed in the following theorem.

Theorem 4. *Given a set $\mathcal{A}$ as a feasible solution to the RSP problem (i.e., $\mathbf{P1}$), let $\mathcal{H}^*(\mathcal{A})$ be an optimal set removal form $\mathcal{A}$ in $\mathbf{P1}$, that is, $\mathcal{H}^*(\mathcal{A}) \in \arg\min_{\mathcal{H} \subseteq \mathcal{A}, |\mathcal{H}| \leq k} g(\mathcal{A} \setminus \mathcal{H})$. Alg. 1 achieves an approximation ratio of $\frac{1-e^{-2}}{2} \max\{1 - \kappa_g, \frac{1}{S-k}, \frac{1}{1+k}\}$, i.e.,*

$$\frac{g(\mathcal{A} \setminus \mathcal{H}^*(\mathcal{A}))}{g(\mathcal{A}^* \setminus \mathcal{H}^*(\mathcal{A}^*))} \geq \frac{1-e^{-2}}{2} \max\{1 - \kappa_g, \frac{1}{S-k}, \frac{1}{1+k}\}, \quad (22)$$

where $\mathcal{A}$ is the output of Alg. 2, $g(\mathcal{A}^* \setminus \mathcal{H}^*(\mathcal{A}^*))$ is the optimal value of $\mathbf{P1}$ and κ_g is the curvature of function $g(\cdot)$. The time complexity is $O(N^6 S^6 \tau_g)$, where N is the number of candidate APs, S is the number of edge servers, and τ_g is the time of solving the LP problem $\mathbf{P0}'$ to obtain the evaluation of the objective function $g(\cdot)$ in problem $\mathbf{P0}''$.

Proof. The theorem can be proved similar to Theorem 2, with one major difference as follows. We need to prove that $g(\mathcal{A}_2) \geq \frac{1-e^{-2}}{2} \max_{\mathcal{U}\subseteq\Omega\setminus\mathcal{A}_1,\mathcal{U}\cup\mathcal{A}_1\in\mathcal{G},C_{\mathcal{U}\cup\mathcal{A}_1}\leq C_0} g(\mathcal{U})$ holds, which mainly exploits that, both the matroid system and knapsack constraint still maintain. Specifically, for the set $\mathcal{A}_1$ constructed by Alg. 2 in lines 2-8, $\mathcal{A}_1 \subseteq \Omega, \mathcal{A}_1 \in \mathcal{G}$, and $C_{\mathcal{A}_1} \leq C_0$ naturally hold. If defining $\mathcal{G}' := \{\mathcal{U} : \mathcal{U} \subseteq \Omega \setminus \mathcal{A}_1, \mathcal{U} \cup \mathcal{A}_1 \in \mathcal{G}\}$, then $(\Omega \setminus \mathcal{A}_1, \mathcal{G}')$ is still a matroid (Lemma 7, [20]). And for $\forall \mathcal{U} \in \mathcal{G}', C_{\mathcal{U}} \leq C_{\mathcal{U}\cup\mathcal{A}_1} \leq C_0$ holds. Based on the above conclusions, after set $\mathcal{A}_1$ is constructed, we conclude that $\max_{\mathcal{U}\subseteq\Omega\setminus\mathcal{A}_1,\mathcal{U}\cup\mathcal{A}_1\in\mathcal{G},C_{\mathcal{U}\cup\mathcal{A}_1}\leq C_0} g(\mathcal{U})$ is a classical monotone submodular maximization problem under a matroid constraint and knapsack constraint. Note that in the construction of set $\mathcal{A}_2$ in lines 9-15, we follow the greedy local search steps over the new ground set $\Omega\setminus\mathcal{A}_1$ for the aforementioned optimization problem, which achieves an approximation ratio of $\frac{1-e^{-2}}{2}$ with respect to $\mathcal{A}_2$ in light of Theorem 1 in [53]. The rest proof is similar to that of Theorem 2. For the time computation complexity, the construction of set $\mathcal{A}_2$ dominates the whole complexity, which is $O(N^6S^6\tau_g)$ according to Theorem 7 in [53]. This theorem thus concludes. □

Based on Theorem 2 and Theorem 4, we have the following corollary.

Corollary 1. *For problem* $\mathbf{P1}$*, the approximation ratio achieved by Alg 2 is strictly better than that by Alg. 1.*

Proof. The corollary holds owing to the following inequality: $\frac{1}{1+p} <= \frac{1}{1+2} \approx 0.333 < 0.432 \approx \frac{1-e^{-2}}{2}$ $(p >= 2)$. □

7 Extension to the Case with Additional Server Number Constraint

Note that in the considered problem in previous sections, we assume that more than one edge server can be deployed at the same AP, while in some case the service provider might deploy at most one edge server at one single AP. This leads to an additional server number constraint in the original problem $\mathbf{P0}$, which is mathematically formulated in the following:

$$\begin{aligned}(\mathbf{P2}) \quad \max_{\mathbf{X},\mathbf{Y}} \quad & \sum_{i=1}^{N}\sum_{j=1}^{N} w_i y_{ij} \\ \text{s.t.} \quad & (2)-(9) \\ & \sum_{s=1}^{S} x_{js} \leq 1, \forall j \in \mathcal{N}. \end{aligned} \tag{23}$$

We next discuss how to extend the aforementioned solution framework to address problem $\mathbf{P2}$. To begin with, we notice that constraint (23) has the same structure as constraint (2). Based on that insight, we have the two following conclusions. Firstly, similar to Lemma 3, we prove that the transformed constraint of (23) by a similar reformulation from problem $\mathbf{P0}'$ to problem $\mathbf{P0}''$, combined with (11) and (12), forms another p-independence system. This makes Alg. 1 still applicable to problem $\mathbf{P2}$ with a different approximation ratio. Secondly, we prove that constraint (23) is also a matroid constraint similar to (11). Alg. 2 could be applicable to problem $\mathbf{P2}$ after minor modification.

7.1 Performance Analysis of Alg. 1 for Problem $\mathbf{P2}$

Similar to (11), constraint (23) can be equivalently reformulated as

$$\sum_{s:(j,s)\in\mathcal{A}} \mathbb{1}_{(j,s)\in\mathcal{A}} \leq 1, \forall j \in \mathcal{N}. \tag{24}$$

The new Non-RSP problem is reformulated as follows:

$$\begin{aligned}(\mathbf{P2}'') \quad \max_{\mathcal{A}\subseteq\Omega} \quad & g(\mathcal{A}) \\ \text{s.t. } & (11), (12), (24).\end{aligned}$$

Lemma 5. *The constraints of (11), (12), and (24) form a p'-independence system for $p' = 2 + \lceil \frac{\max c_{js}}{\min c_{js}} \rceil$.*

Proof. The proof is similar to that of Lemma 3. The key point is as follows. To satisfy constraints (11), (12), and (24), we need to take out at most one element, $\lceil \frac{\max c_{js}}{\min c_{js}} \rceil$ elements, and one element, respectively, which results in at most $p' = 2+\lceil \frac{\max c_{js}}{\min c_{js}} \rceil$ elements to be taken out. The lemma thus concludes. □

Based on $\mathbf{P2}''$ and Lemma 5, the new RSP problem $\mathbf{P2}$ can be equivalently formulated as follows:

$$(\mathbf{P3}) \quad \max_{\mathcal{A}\subseteq\Omega,\mathcal{A}\in\mathcal{G}_{p'}} \min_{\mathcal{H}\subseteq\mathcal{A},|\mathcal{H}|\leq k} g(\mathcal{A}\setminus\mathcal{H}),$$

where $(\Omega, \mathcal{G}_{p'})$ is the p'-independence system formed by constraints of (11), (12), and (24), and $k \leq S-1$ is the maximum number of server failures that controls the degree of robustness. Then, according to Lemma 3, Lemma 5, Theorem 2, and Theorem 3, we have the following performance guarantee of Alg. 1 for problem $\mathbf{P3}$.

Theorem 5. *Given a set $\mathcal{A}$ as a feasible solution to the new RSP problem (i.e., $\mathbf{P3}$), let $\mathcal{H}^*(\mathcal{A})$ be an optimal set removal from $\mathcal{A}$ in $\mathbf{P3}$, that is $\mathcal{H}^*(\mathcal{A}) \in \arg\min_{\mathcal{H}\subseteq\mathcal{A},|\mathcal{H}|\leq k} g(\mathcal{A}\setminus\mathcal{H})$. Algorithm 1 achieves an approximation ratio of $\frac{1}{1+p'}\max\{1-\kappa_g, \frac{1}{S-k}, \frac{1}{1+k}\}$, i.e.,*

$$\frac{g(\mathcal{A}\setminus\mathcal{H}^*(\mathcal{A}))}{g(\mathcal{A}^*\setminus\mathcal{H}^*(\mathcal{A}^*))} \geq \frac{1}{1+p'}\max\{1-\kappa_g, \frac{1}{S-k}, \frac{1}{1+k}\},$$

where $\mathcal{A}$ is the output of Algorithm 1, $g(\mathcal{A}^*\setminus\mathcal{H}^*(\mathcal{A}^*))$ is the optimal value of $\mathbf{P3}$, p' is the parameter defined in Lemma 5, and κ_g is the curvature of function $g(\cdot)$.

Proof. The proof is extremely similar to that of Theorem 2, while we only need to change the p-independence system by the p'-independence system. The proof is omitted here to avoid repetition. □

7.2 Performance Analysis of Alg. 2 for Problem $\mathbf{P2}$

Similar to Lemma 4, we prove that the constraint (24) is also a matroid constraint in the following lemma.

Lemma 6. *Recall set $\Omega = \{(j,s)|\forall j \in \mathcal{N}, s \in \mathcal{S}\}$ and define $\mathcal{G}' := \{\mathcal{A}|\mathcal{A} \subseteq \Omega, \forall a_1 := (j_1,s_1), a_2 := (j_2,s_2) \in \mathcal{A}, j_1 \neq j_2\}$. Then, constraint (24) is a matroid constraint.*

Proof. The proof is similar to that of Lemma 4, which is omitted here to avoid repetition. □

Based on Lemma 6 as well as the aforementioned conclusions in Section 6, we conclude that problem $\mathbf{P3}$ is actually a robust submodular max-min problem with two matroid

constraints and a knapsack constraint. After applying Alg. 2, it achieves the following performance guarantee.

Theorem 6. *Given a set $\mathcal{A}$ as a feasible solution to the new RSP problem (i.e., $\mathbf{P3}$), let $\mathcal{H}^*(\mathcal{A})$ be an optimal set removal form $\mathcal{A}$ in $\mathbf{P3}$, that is, $\mathcal{H}^*(\mathcal{A}) \in \arg\min_{\mathcal{H}\subseteq\mathcal{A},|\mathcal{H}|\leq k} g(\mathcal{A}\setminus\mathcal{H})$. Alg. 2 achieves an approximation ratio of $\frac{1-e^{-3}}{3}\max\{1-\kappa_g, \frac{1}{S-k}, \frac{1}{1+k}\}$, i.e.,*

$$\frac{g(\mathcal{A}\setminus\mathcal{H}^*(\mathcal{A}))}{g(\mathcal{A}^*\setminus\mathcal{H}^*(\mathcal{A}^*))} \geq \frac{1-e^{-3}}{3}\max\{1-\kappa_g, \frac{1}{S-k}, \frac{1}{1+k}\},$$

where $\mathcal{A}$ is the output of Alg. 2, $g(\mathcal{A}^*\setminus\mathcal{H}^*(\mathcal{A}^*))$ is the optimal value of $\mathbf{P3}$ and κ_g is the curvature of function $g(\cdot)$.

Proof. The proof is similar to that of Theorem 4, while we only need to prove that $g(\mathcal{A}_2) \geq \frac{1-e^{-3}}{3}\max_{\mathcal{U}\subseteq\Omega\setminus\mathcal{A}_1,\mathcal{U}\cup\mathcal{A}_1\in\mathcal{G}\cup\mathcal{G}',C_{\mathcal{U}\cup\mathcal{A}_1}\leq C_0} g(\mathcal{U})$ holds. It mainly exploits that, after determining $\mathcal{A}_1$, $\max_{\mathcal{U}\subseteq\Omega\setminus\mathcal{A}_1,\mathcal{U}\cup\mathcal{A}_1\in\mathcal{G}\cup\mathcal{G}',C_{\mathcal{U}\cup\mathcal{A}_1}\leq C_0} g(\mathcal{U})$ is a classical monotone submodular maximization problem under two matroid constraints and a knapsack constraint, and Theorem 8 in [53]. The rest is similar to that of Theorem 4, which concludes this theorem. □

Corollary 2. *For problem $\mathbf{P3}$, the approximation ratio achieved by Alg 2 is strictly better than that by Alg. 1.*

Proof. The corollary holds owing to the following inequality: $\frac{1}{1+p'} <= \frac{1}{1+3} \approx 0.25 < 0.317 \approx \frac{1-e^{-3}}{3}$ ($p' >= 3$). □

8 Performance Evaluation

In this section, we validate the effectiveness of Alg. 1 (labeled as **Robust**) and Alg. 2 (labeled as **Robust+**) by synthetic and trace-driven simulations faced with no more than k server failures.

8.1 Benchmarks

In the performance comparison, we involve five benchmark algorithms as follows.

- **Brute-Force**: It finds the *optimal* placement scheme by exhaustive search over $2^{N\times S}$ possible decisions in the presence of 2^k possible server failures. The computational complexity of this algorithm is so high that it works only in a small network scale.
- **Random**: Each server is randomly placed at one of the feasible APs with equal possibility subject to the server deployment budget constraint.
- **LP-Relaxation**: Following [23], we design a heuristic algorithm for the RSP problem. The procedure is as follows. (i) Solve the LP-relaxation of $\mathbf{P0}$ and get a fractional solution $(\ddot{\mathbf{X}}, \ddot{\mathbf{Y}})$. (ii) Select the deployment variable $\ddot{x}_{js}$ with highest fractional value in $\ddot{\mathbf{X}}$ that has not been fixed yet. Fix the variable to one if the budget constraint is satisfied, otherwise fix the variable to zero. (iii) Continue rounding by fixing to zero all variables that would lead to infeasible solutions when set to one. (iv) If all the deployment variables are fixed, solve $\mathbf{P0'}$ and output the solution. Otherwise, given the fixed variables, re-solve the LP-relaxation of $\mathbf{P0}$ and repeat rounding from step (ii).
- **Greedy**: It greedily selects the deployment decision with largest marginal contribution to the objective value until all the servers are deployed or the budget exhausts. Greedy is a variant of the proposed Robust algorithm that applies the one-step greedy strategy by setting $k = 0$.
- **PRo** [16]: It firstly defines a subroutine A with input m and a set T, which outputs an ordered set with size m chosen from T based on the classic greedy algorithm. Then it refers to k as the number of servers that may fail, V as the set whose elements satisfy the knapsack constraints, and the set S_0 as the robust set. S_0 consists of $\lceil\log k\rceil + 1$ partitions and each of the partition have $\lceil k/2^i\rceil$ buckets. Every bucket in partition i is obtained through the subroutine A with input 2^i and $V\setminus S_0$. After the generation of S_0, it chooses the set S_1 which contains all the remaining feasible placements through the subroutine A with input set $V\setminus S_0$. The output of this algorithm is the set $S_0 \cup S_1$.

Note that in all algorithms, the schedule for workload allocation is obtained by solving $\mathbf{P0'}$ with the corresponding obtained server deployment strategy.

8.2 Simulation Setup

For synthetic simulation, we consider a network consisting of N APs and S edge servers, where N varies from 10 to 200, and S varies from 5 to 50, respectively. Since the actual deployment cost is not necessary in this model, the relative deployment cost is drawn uniformly from $[0.5, 1]$ and the budget is set to $0.6\times S$. q_{ij} is set to only permit task offloading between APs within two hops. We introduce $\lambda_i, i \in \mathcal{N}$ as the request rate, $\bar{w}$ as the workload of single task and $\bar{b}$ as the requested bandwidth of each task, so that the workload and requested bandwidth of each AP $i \in \mathcal{N}$ can be estimated as $\lambda_i\bar{w}$ and $\lambda_i\bar{b}$, respectively. Following [44], the values of λ_i, $\bar{w}$ and $\bar{b}$ are drawn uniformly from the intervals $[3, 5]$ times/s, $[0.5, 1]$ MFLOPs (mega floating-point operations), $[0.5, 1]$ KB/s, respectively and the available uplink bandwidth B_i^{up} and the available downlink bandwidth B_i^{down} of AP i follow uniform distribution $U[16, 24]$ KB/s. For the edge server settings, the computing capacity of each edge server s follows the uniform distribution with $d_s \sim U[32, 48]$ MFLOPS (MFLOPs per second).

For trace-driven simulation, we extract workload of APs from real Internet-access logs. We utilize the dataset from Shanghai Telecom, which contains 1.2 million records for 8411 mobile users who access about 3200 base stations in June 2014 [45], [46]. Every Internet access record consists of start time, end time, user id and location of the BS. After filtering out some invalid BS records, we randomly select 10 and 200 BSs from the dataset in the simulation. Following [47], [37], [45], the workload of BS is proportional to total request time of the BS. Other involved settings are the same as that in the synthetic case.

8.3 Results

8.3.1 Near-Optimality of Robust and Robust+

We first validate the near-optimality of Robust and Robust+ by extensive simulations under the small network scale,

where the optimal Brute-Force algorithm can output effective results. Fig. 4 and Fig. 5 present the performances of the seven algorithms over the synthetic trace and Shanghai Telecom trace, respectively. Note that in the latter trace, there is no result with the variation of the BS's workload, since under the real-world trace, the workload of each BS is fixed and obtained from the trace. According to Fig. 4 and Fig. 5, both Robust and Robust+ significantly outperform Random, LP-Relaxation, and Greedy, when the maximum number of server failures, workload of each BS, or total budget varies, while PRo achieves almost identical performances as Robust and Robust+ at $k = 2$ and $k = 3$ mainly due to the similar design of a guess of worst removal subset in PRo. More importantly, Robust and Robust+ achieve on average about 87.24% and 90.11% of the optimal performance by Brute-Force in Fig. 4, respectively, and they achieve on average about 90.25% and 91.67% in Fig. 5, respectively. And Robust achieves at most 90.41% of the optimal performance in all cases, while Robust+ obtains at most 91.94%. The above results demonstrate the near-optimality of both Robust and Robust+, which also validates the proposed approximation ratio analysis.

8.3.2 Effectiveness of Robust and Robust+

We next present the performances of Robust and Robust+ under the large network scale to validate their effectiveness. Firstly, we evaluate the performance of the proposed Robust by varying the maximum number of server failures, the workload of each AP and the overall budget under $N = 200$ APs and $S = 50$ edge servers. When the number of maximum server failures varies in Fig. 6 (a), the result shows that Robust on average outperforms Random, LP-Relaxation, Greedy, and PRo by 92%, 5%, 11%, and 35% over the synthetic trace, respectively. In Fig. 6 (b) and Fig. 6 (c), we respectively vary the workload of each AP and overall budget when there are at most $k = 2$ server failures. The result shows that Robust on average outperforms Random, LP-Relaxation, Greedy, and PRo by over 77%, 7%, 9%, and 32% according to Fig. 6 (b). The result of Fig. 6 (c) is similar with Fig. 6 (b).

Secondly, similarly with results in synthetic simulation, Robust outperforms Random, Greedy, LP-Relaxation, and PRo over the real-world trace, as illustrated in Fig. 7. Specifically, Robust outperforms Random, LP-Relaxation, Greedy, and PRo by 97%, 34%, 7%, and 33% on average, respectively when we vary the maximum number of server failures in Fig. 7 (a). Besides, when there are at most $k = 2$ server failures in Fig. 7 (b), Robust outperforms Random, LP-Relaxation, Greedy, and PRo by 204%, 23%, 12%, and 40% on average, respectively, which validates the effectiveness of the proposed algorithm in practice. Lastly, Fig. 8 illustrates the superiority of Robust+ over Robust. Specifically, according to Fig. 8 (a) and Fig. 8 (b), Robust+ outperforms Robust by 6.7% and 3.6% on average, respectively.

8.3.3 Effectiveness under the Extended Case

We also evaluate the performance of the proposed solution framework under the extended case discussed in Section 7, by presenting the performance of Robust under the large network scale over the Shanghai Telecom trace in Fig. 9 as a representative example. Based on Fig. 9 (a), Robust outperforms Random, LP-Relaxation, Greedy, and PRo by 101%, 35%, 7.6%, and 33% on average, respectively, when the maximum number of server failures varies from 1 to 4. And as shown in Fig. 9 (b), Robust outperforms Random, LP-Relaxation, Greedy, and PRo by 138%, 24%, 13%, and 30% on average, respectively, when the budget changes from 21 to 27.

9 Conclusion

In this paper, we have studied the robust server placement (RSP) for mobile edge computing in the presence of uncertain server failures. We have formulated the RSP problem in the form of robust max-min optimization problem. First, we have revealed that the objective function in RSP is monotone submodular. Then, we have proved that the involved constraints are equivalent to a p-independence system constraint. Next, we have proposed two novel algorithms (*i.e.*, Robust and Robust+) that achieve provable approximation ratios in polynomial time. Finally, we have conducted extensive simulations to validate the effectiveness of the proposed algorithms. Some promising future works include how to obtain robust server placement scheme in a dynamic scenario considering server mobility and the edge server placement problem considering the reliable federated edge learning [55], [56].

Acknowledgement

This work is supported in part by the National Key R&D Program of China No. 2018YFB1800800, in part by the National Natural Science Foundation of China under Grant No. 61931011, Grant No. 62072303, Grant No. 62002377, in part by National Postdoctoral Program for Innovative Talents of China No. BX20190202, in part by the Open Project Program of the Key Laboratory of Dynamic Cognitive System of Electromagnetic Spectrum Space No. KF20202105.

References

[1] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys and Tutorials*, vol. 19, no. 3, pp. 1628-1656, 2017.

[2] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys and Tutorials*, vol. 19, no. 4, pp. 2322-2358, 2017.

[3] D. Xu, Y. Li, X. Chen, J. Li, P. Hui, S. Chen, and J. Crowcroft, "A survey of opportunistic offloading," *IEEE Communications Surveys and Tutorials*, vol. 20, no. 3, pp. 2198-2236, 2018.

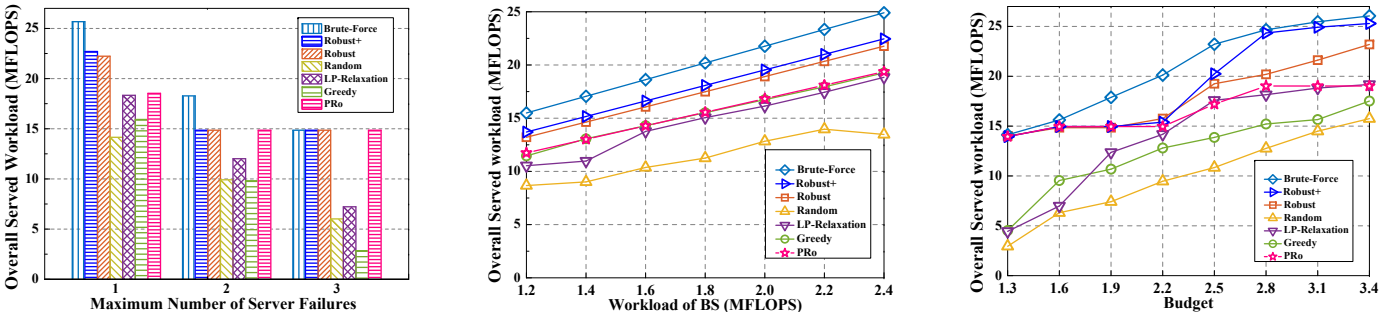
[4] Z. Tao, Q. Xia, Z. Hao, C. Li, L. Ma, S. Yi, and Q. Li, "A survey of virtual machine management in edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1482-1499, 2019.

[5] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing-A key technology towards 5g," *ETSI White Paper*, vol. 11, no. 11, pp. 1-16, 2015.

[6] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: Architecture, applications, and approaches," *Wiley Wireless Communications and Mobile Computing*, vol. 13, no. 18, pp. 1587-1611, 2013.

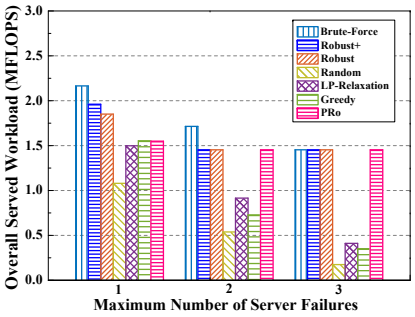
[7] Z. Hao, S. Yi, and Q. Li, "Edgecons: Achieving efficient consensus in edge computing networks," *Proc. USENIX Workshop on Hot Topics in Edge Computing (HotEdge)*, 2018.

[8] L. Wang and K. S. Trivedi, "Architecture-based reliability-sensitive criticality measure for fault-tolenrance cloud applications," *IEEE Transactions on Parallel and Distribtued Systems*, vol. 30, no. 11, pp. 2408-2421, 2019.

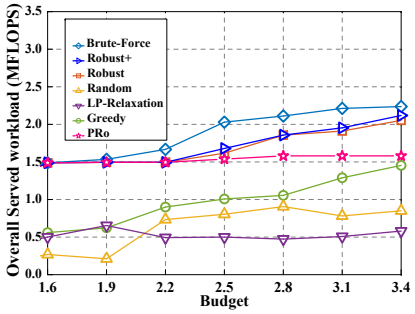


(a) Overall served workload vs. k (b) Overall served workload vs. w_i (c) Overall served workload vs. C_0

Fig. 4: Performance of Robust and Robust+ under the small network scale over the synthetic trace.



(a) Overall served workload vs. k



(b) Overall served workload vs. C_0

Fig. 5: Performance of Robust and Robust+ under the small network scale over the Shanghai Telecom trace.

[9] G. Yao, Y. Ding, and K. Hao, "Using imbalance characteristic for fault-tolerant workflow scheduling in cloud systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 12, pp. 3671-3683, 2017.

[10] W. Chen, R. F. da Silva, E. Deelman, and T. Fahringer, " Dynamic and fault-tolerant clustering for scientific workflows," *IEEE Transactions on Cloud Computing*, vol. 4, no. 1, pp. 49-62, 2015.

[11] A. Zhou, S. Wang, B. Cheng, Z. Zheng, F. Yang, R. N. Chang, M. R. Lyu, and R. Buyya, "Cloud service reliability enhancement via virtual machine placemnet optimization," *IEEE Transactions on Services Computing*, vol. 10, no. 6, pp. 902-913, 2016.

[12] M. Le, Z. Song, Y.-W. Kwon, and E. Tilevich, "Reliable and efficient mobile edge computing in highly dynamic and volatile environments," *Proc. IEEE International Conference on Fog and Mobile Edge Computing (FMEC)*, pp. 113-120, 2017.

[13] C. Pei, Z. Wang, Y. Zhao, Z. Wang, Y. Meng, D. Pei, Y. Peng, W. Tang, and X. Qu, "Why it takes so long to connect to a wifi access point," *Proc. IEEE International Conference on Computer Communications (INFOCOM)*, pp. 1-9, 2017.

[14] E. Kazemi, M. Zadimoghaddam, and A. Karbasi, "Scalable deletion-robust submodular maximization: Data summarization with privacy and fairness constraints," *Proc. International Conference on Machine Learning (ICML)*, pp. 2549-2558, 2018.

[15] J. B. Orlin, A. S. Schulz, and R. Udwani, "Robust monotone submodular function maximization," *Springer Mathematical Programming*, vol. 172, no. 1-2, pp. 505-537, 2018.

[16] I. Bogunovic, S. Mitrovi'c, J. Scarlett, and V. Cevher, "Robust submodular maximization: A non-uniform partitioning approach," *Proc. International Conference on Machine Learning (ICML)*, vol. 70, pp. 508-516, 2017.

[17] B. Mirzasoleiman, A. Karbasi, and A. Krause, "Deletion-robust submodular maximization: Data summarization with the right to be forgotten," *Proc. International Conference on Machine Learning (ICML)*, vol. 70, pp. 2449-2458, 2017.

[18] S. Mitrovic, I. Bogunovic, A. Norouzi-Fard, J. M. Tarnawski, and V. Cevher, "Streaming robust submodular maximization: A partitioned thresholding approach," *Proc. Advances in Neural Information Processing Systems (NIPS)*, pp. 4557-4566, 2017.

[19] L. Zhou, V. Tzoumas, G.J. Pappas, and P. Tokekar, "Resilient active target tracking with multiple robots," *IEEE Robotics and Automation Letters (RA-L)*, vol. 4, no. 1, pp. 129-136, 2018.

[20] V. Tzoumas, A. Jadbabaie, and G. J. Pappas, "Resilient nonsubmodular maximization over matroid constraints," *arXiv preprint* arXiv:1804.01013, 2018.

[21] L. Yuan, Q. He, S. Tan, B. Li, J. Yu, F. Chen, H. Jin, and Y. Yang, "CoopEdge: A decentralized blockchain-based platform for cooperative edge computing," *Proc. 30th The Web Conference (WWW 2021)*, pp. 2245-2257, 2021.

[22] A. Ceselli, M. Premoli, and S. Secci, "Cloudlet network design optimization," *Proc. IEEE IFIP Networking Conference*, pp. 1-9, 2015.

[23] A. Ceselli, M. Premoli, and S. Secci, "Mobile edge cloud network design optimization," *IEEE/ACM Transactions on Networking*, vol. 25, no. 3, pp. 1818-1831, 2017.

[24] S. Mondal, G. Das, and E. Wong, "CCOMPASSION: A hybrid cloudlet placement framework over passive optical access networks," *Proc. IEEE International Conference on Computer Communications (INFOCOM)*, pp. 216-224, 2018.

[25] F. Zeng, Y. Ren, X. Deng, and W. Li, "Cost-effective edge server placement in wireless metropolitan area networks," *MDPI Sensors*, vol. 19, no. 1, pp. 32, 2019.

[26] S. Mondal, G. Das, and E. Wong, "Cost-optimal cloudlet placement frameworks over fiber-wireless access networks for low-latency applications," *Elsevier Journal of Network and Computer Applications*, vol. 138, pp. 27-38, 2019.

[27] Q. Fan and N. Ansari, "Cost aware cloudlet placement for big data processing at the edge," *Proc. IEEE International Conference on Communications (ICC)*, pp. 1-6, 2017.

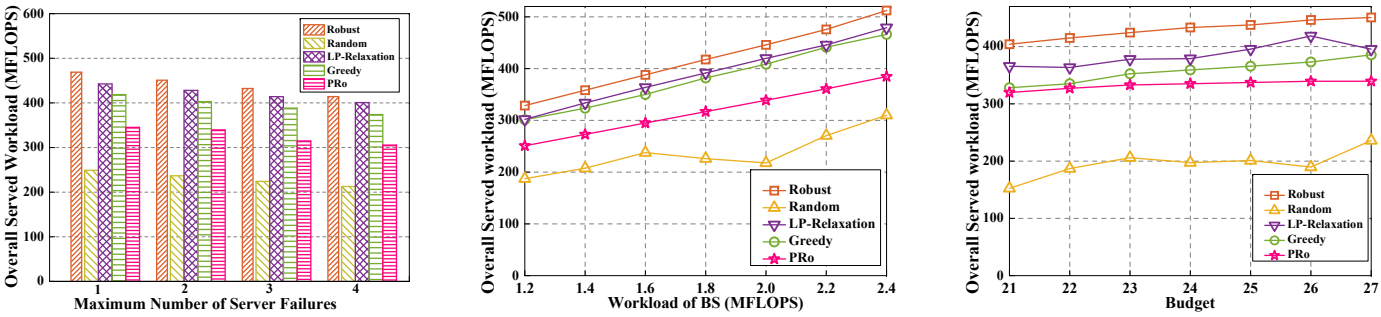
[28] D. Bhatta and L. Mashayekhy, "Cost-aware cloudlet placement in edge computing systems," *Proc. 4th ACMIEEE Symposium on Edge Computing (SEC)*, pp. 292-294, 2019.

[29] J. Meng, C. Zeng, H. Tan, Z. Li, B. Li, and X.-Y. Li, "Joint heterogeneous server placement and application configuration in edge computing," *Proc. 25th IEEE International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 488-497, 2019.

[30] Y. Ren, F. Zeng, W. Li, and L. Meng, "A low-cost edge server placement strategy in wireless metropolitan area networks," *Proc. 27th International Conference on Computer Communication and Networks (ICCCN)*, pp. 1-6, 2018.

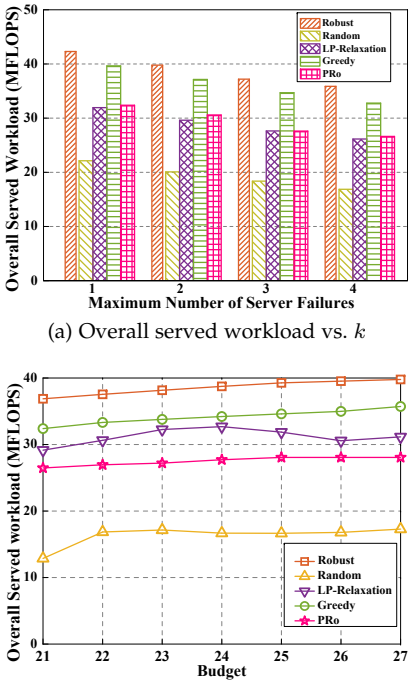
[31] Z. Xu, W. Liang, W. Xu, M. Jia, and S. Guo, "Efficient algorithms for capacitated cloudlet placements," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 10, pp. 2866-2880, 2015.

[32] M. Jia, J. Cao, and W. Liang, "Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks," *IEEE Transactions on Cloud Computing*, vol. 5, no. 4, pp. 725-737, 2015.



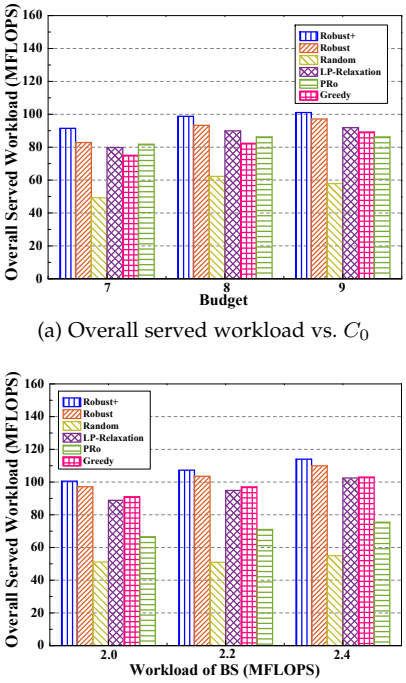
(a) Overall served workload vs. k (b) Overall served workload vs. w_i (c) Overall served workload vs. C_0

Fig. 6: Performance of Robust under the large network scale over the synthetic trace.



(a) Overall served workload vs. k

(b) Overall served workload vs. C_0



(a) Overall served workload vs. C_0

(b) Overall served workload vs. w_i

Fig. 7: Performance of Robust under the large network scale over the Shanghai Telecom trace.

Fig. 8: Performance comparison of Robust and Robust+.

[33] Z. Xu, W. Liang, W. Xu, M. Jia, and S. Guo, "Capacitated cloudlet placements in wireless metropolitan area networks," *Proc. IEEE Conference on Local Computer Networks (LCN)*, pp. 570-578, 2015.

[34] L. Zhao, W. Sun, Y. Shi, and J. Liu, "Optimal placement of cloudlets for access delay minimization in sdn-based internet of things networks," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1334-1344, 2018.

[35] K. Cao, L. Li, Y. Cui, T. Wei, and S. Hu, "Exploring placement of heterogeneous edge servers for response time minimization in mobile edge-cloud computing," *IEEE Transactions on Industry Informatics*, vol. 17, no. 1, pp. 494-503, 2021.

[36] Z. Liu, J. Zhang, and J. Wu, "Joint optimization of server placement and conten caching in mobile edge computing networks," *Proc. 8th International Conference on Networks, Communication and Computing (ICNCC)*, pp. 149-153, 2019.

[37] Y. Li and S. Wang, "An energy-aware edge server placement algorithm in mobile edge computing," *Proc. IEEE International Conference on Edge Computing (EDGE)*, pp. 66-73, 2018.

[38] S. Yang, F. Li, M. Shen, X. Chen, X. Fu, and Y. Wang, "Cloudlet placement and task allocation in mobile edge computing," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5853-5863, 2019.

[39] C. Shen, S. Xue, and S. Fu, "ECPM: An energy-efficient cloudlet placement method in mobile cloud environment," *EURASIP Journal on Wireless Communications and Networking*, vol. 141, 2019.

[40] H. Yin, X. Zhang, H. H. Liu, Y. Luo, C. Tian, S. Zhao, and F. Li, "Edge provisioning with flexible server placement," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 4, pp. 1031-1045, 2016.

[41] J. Dongarra and F. Sullivan, "Guest editors' introduction: The top 10 algorithms," *AIP Computing in Science & Engineering*, vol. 2, no. 1, pp. 22, 2000.

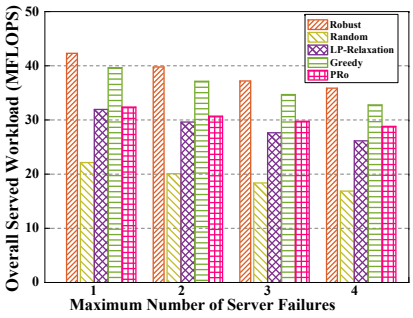
[42] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey, "An analysis of approximations for maximizing submodular set functions-II," *Springer Mathematical Programming Studies*, vol. 8, pp. 73-87, 1978.

[43] A. Gupta, A. Roth, G. Schoenebeck, and K. Talwar, "Constrained non-monotone submodular maximization: Offline and secretary algorithms," *Proc. Springer International Workshop on Internet and Network Economics*, pp. 246-257, 2010.

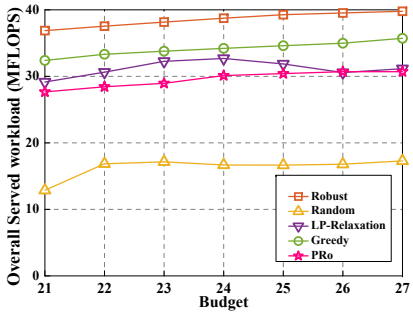
[44] V. Farhadi, F. Mehmeti, T. He, T. La Porta, H. Khamfroush, S. Wang, and K. S. Chan, "Service placement and request scheduling for dataintensive applications in edge clouds," *Proc. IEEE International Conference on Computer Communications (INFOCOM)*, pp. 1279-1287, 2019.

[45] Y. Guo, S. Wang, A. Zhou, J. Xu, J. Yuan, and C.-H. Hsu, "User allocation-aware edge cloud placement in mobile edge computing," *Wiley Software: Practice and Experience*, vol. 50, no. 5, pp. 489-502, 2020.

[46] S. Wang, Y. Zhao, L. Huang, J. Xu, and C.-H. Hsu, "Qos prediction for service recommendations in mobile edge computing," *Elsevier*



(a) Overall served workload vs. k



(b) Overall served workload vs. C_0

Fig. 9: Performance of Robust under the extended case.

Journal of Parallel and Distributed Computing, vol. 127, pp. 134-144, 2019.

[47] S. Wang, Y. Zhao, J. Xu, J. Yuan, and C.-H. Hsu, "Edge server placement in mobile edge computing," *Elsevier Journal of Parallel and Distributed Computing*, vol. 127, pp. 160-168, 2019.

[48] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 587-597, 2018.

[49] J. Meng, W. Shi, H. Tan, and X. Li, "Cloudlet Placement and Minimum-Delay Routing in Cloudlet Computing," *Proc. IEEE International Conference on Big Data Computing and Communications (BIGCOM)*, pp. 297-304, 2017.

[50] D.-Z. Du, K.-I. Ko, and X. Hu, "Design and analysis of approximation algorithms," Springer, vol. 62, 2011.

[51] H. Dai, Y. Liu, A.X. Liu, L. Kong, G. Chen, and T. He, "Radiation constrained wireless charger placement," *Proc. IEEE International Conference on Computer Communications (INFOCOM)*, pp. 1-9, 2016.

[52] A. Schrijver, "Combinatorial optimization: Polyhedra and efficiency," *Springer Science & Business Media*, vol. 24, 2003.

[53] K. Sarpatwar, B. Schieber, and H. Shachnai, "Constrained submodular maximization via greedy local search," *Operations Research Letters*, vol. 47, no. 1, pp. 1-6, 2019.

[54] D. Lu, Y. Qu, F. Wu, H. Dai, C. Dong, and G. Chen, "Robust server placement for edge computing," *Proc. IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 285-294, 2020.

[55] J. Kang, Z. Xiong, D. Niyato, Y. Zou, Y. Zhang, and M. Guizani, "Reliable Federated Learning for Mobile Networks," *IEEE Wireless Communications*, vol. 27, no. 2, pp. 72-80, 2020.

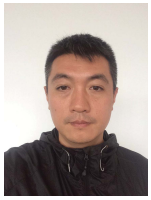
[56] J. Kang, Z. Xiong, X. Li, Y. Zhang, D. Niyato, C. Leung, and C. Miao, "Optimizing Task Assignment for Reliable Blockchain-Empowered Federated Edge Learning," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 2, pp. 1910-1923, 2021.

Yuben Qu received the B.S. degree in Mathematics and Applied Mathematics from Nanjing University, and both the M.S. degree in Communication and Information Systems and the Ph.D degree in Computer Science and Technology from Nanjing Institute of Communications, in 2009, 2012 and 2016, respectively. He is currently a post-doc in the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. From October 2015 to January 2016, he was a visiting research associate in the School of Computer Science and Engineering, The University of Aizu, Japan. His research interests include mobile edge computing, air-ground integrated networks, D2D communications, and crowdsensing.

Lihao Wang received the B.S. degree through the Tang Aoqing Honors Program (computer science) with the College of Computer Science and Technology, Jilin University, Changchun, China, in 2020. He is currently working towards his M.S. degree in the Department of Computer Science and Technology in Nanjing University, China. His research interests include mobile edge computing, wireless sensor networks, and Internet of Things.

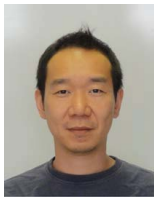
Haipeng Dai received the B.S. degree in the Department of Electronic Engineering from Shanghai Jiao Tong University, Shanghai, China, in 2010, and the Ph.D. degree in the Department of Computer Science and Technology in Nanjing University, Nanjing, China, in 2014. His research interests are mainly in the areas of wireless charging, mobile computing, and data mining. He is an associate professor in the Department of Computer Science and Technology in Nanjing University. His research papers have been published in many prestigious conferences and journals such as ACM MobiSys, ACM MobiHoc, ACM VLDB, ACM SIGMETRICS, ACM Ubicomp, IEEE INFOCOM, IEEE ICDCS, IEEE ICNP, IEEE SECON, IEEE IPSN, IEEE JSAC, IEEE/ACM TON, IEEE TMC, IEEE TPDS, and IEEE TOSN. He is an IEEE and ACM member. He serves/ed as Poster Chair of the IEEE ICNP'14, Track Chair of the ICCCN'19, TPC member of the IEEE ICNP'14, IEEE ICC'14-18, IEEE ICCCN'15-18 and the IEEE Globecom'14-18. He received Best Paper Award from IEEE ICNP'15, Best Paper Award Runner-up from IEEE SECON'18, and Best Paper Award Candidate from IEEE INFOCOM'17.

Weijun Wang received the B.S. degree in the Department of Computer and Software from Nanjing University of Post and Telecommunication, Nanjing, China, in 2014 and M.E. degree in the Department of Communication Engineering from PLA University of Science and Technology, Nanjing, China, in 2017. He is studying towards the Ph.D degree in the Department of Computer Science and Technology in Nanjing University. His research interests focus on UAV monitoring problem, cognitive radio ,and MAC protocols in UAV hoc networks. He is a student member of IEEE.



Chao Dong received his Ph.D degree in Communication Engineering from PLA University of Science and Technology, China, in 2007. From 2008 to 2011, he worked as a post Doc at the Department of Computer Science and Technology, Nanjing University, China. From 2011 to 2017, he was an Associate Professor with the Institute of Communications Engineering, PLA University of Science and Technology, Nanjing, China. He is now a full professor with the College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, China. His current research interests include D2D communications, UAVs swarm networking and anti-jamming network protocol. He is a member of IEEE, ACM and IEICE.

Fan Wu is a professor in the Department of Computer Science and Engineering, Shanghai Jiao Tong University. He received his B.S. in Computer Science from Nanjing University in 2004, and Ph.D. in Computer Science and Engineering from the State University of New York at Buffalo in 2009. He has visited the University of Illinois at Urbana-Champaign (UIUC) as a Post Doc Research Associate. His research interests include wireless networking and mobile computing, algorithmic game theory and its applications, and privacy preservation. He has published more than 180 peer-reviewed papers in technical journals and conference proceedings. He is a recipient of the first class prize for Natural Science Award of China Ministry of Education, NSFC Excellent Young Scholars Program, ACM China Rising Star Award, CCF-Tencent "Rhinoceros bird" Outstanding Award, CCF-Intel Young Faculty Researcher Program Award, Pujiang Scholar, and Tang Scholar. He has served as the chair of CCF YOCSEF Shanghai, on the editorial board of IEEE Transactions on Mobile Computing and Elsevier Computer Communications, and as the member of technical program committees of more than 60 academic conferences. For more information, please visit www.cs.sjtu.edu.cn/∼fwu/.



Song Guo is a Full Professor at Department of Computing, The Hong Kong Polytechnic University. His research interests are mainly in the areas of big data, cloud computing, mobile computing, and distributed systems with over 450 papers published in major conferences and journals. His work was recognized by the 2016 Annual Best of Computing: Notable Books and Articles in Computing in ACM Computing Reviews. He is the recipient of the 2018 IEEE TCGCC Best Magazine Paper Award, 2017 IEEE Systems Journal Annual Best Paper Award, and other six Best Paper Awards from IEEE/ACM conferences. Prof. Guo was an Associate Editor of IEEE Transactions on Parallel and Distributed Systems and an IEEE ComSoc Distinguished Lecturer. He is now an Associate Editor of IEEE Transactions on Cloud Computing, IEEE Transactions on Emerging Topics in Computing, IEEE Transactions on Sustainable Computing, IEEE Transactions on Green Communications and Networking, and IEEE Network. Prof. Guo also served as General and Program Chair for numerous IEEE conferences. He currently serves as a Director and Member of the Board of Governors of IEEE ComSoc. He is a fellow of the IEEE.