


Mechanisms for Resource Allocation and Pricing in Mobile Edge Computing Systems

Tayebeh Bahreini, *Student Member, IEEE*,
Hossein Badri, *Student Member, IEEE*, and Daniel Grosu , *Senior Member, IEEE*

Abstract—In this article, we address the resource allocation and monetization challenges in Mobile Edge Computing (MEC) systems, where users have heterogeneous demands and compete for high quality services. We formulate the Edge Resource Allocation Problem (ERAP) as a Mixed-Integer Linear Program (MILP) and prove that ERAP is NP-hard. To solve the problem efficiently, we propose two resource allocation mechanisms. First, we develop an auction-based mechanism and prove that the proposed mechanism is *individually-rational* and produces *envy-free allocations*. We also propose an LP-based approximation mechanism that does not guarantee envy-freeness, but it provides solutions that are guaranteed to be within a given distance from the optimal solution. We evaluate the performance of the proposed mechanisms by conducting an extensive experimental analysis on ERAP instances of various sizes. We use the optimal solutions obtained by solving the MILP model using a commercial solver as benchmarks to evaluate the quality of solutions. Our analysis shows that the proposed mechanisms obtain near optimal solutions for fairly large size instances of the problem in a reasonable amount of time.

Index Terms—Edge computing, resource allocation, pricing, envy-free mechanism, approximation algorithm

1 INTRODUCTION

IN Mobile Cloud Computing (MCC), centralized cloud servers are employed as powerful resources for executing computational tasks of mobile applications [1]. These systems have been able to mitigate some of the existing limitations and challenges in mobile devices such as storage capacity, computation power, and battery life. The long distance between the centralized servers and end users results in high response latency which makes MCC systems unsuitable for many applications with low latency requirements. In recent years, Mobile Edge Computing (MEC) has been introduced to mitigate the existing challenges in MCC. In a MEC system, data, computation, and storage are migrated from mobile devices to the servers located at the edge of the network [2]. Despite the fact that MEC systems are in their early stages of development, a wide range of applications is expected to benefit from this technology. Autonomous driving [3], [4], healthcare [5], [6], and entertainment are just few examples of such applications. Compared to cloud data centers, edge systems have much more limited resources leading to increased competition among users who desire to acquire high quality services. Due to this challenge, the efficiency of MEC systems depends heavily on the utilized resource allocation mechanisms. Any inefficiencies in resource allocation might lead to low Quality

of Service (QoS), high energy consumption, and increased operating costs.

Despite all the efforts focused on designing efficient resource allocation algorithms in MEC systems, monetization of services, that is, developing incentive schemes for mobile users and edge providers, is still a significant challenge in the development of MEC systems. A report by National Science Foundation (NSF) on grand challenges in edge computing [7], identified incentives and monetization as one of the five grand challenges in the development of edge computing systems. Decentralized distribution of MEC servers, heterogeneity of resource requirements, and the competition between users to acquire high quality services make the resource allocation and pricing in MEC systems a challenging problem. Many auction mechanisms developed for MCC systems are not directly applicable to MEC settings. In MEC systems, resources are distributed over multiple levels, users typically request bundles of multiple types of resources, and they have different valuations for the services provided from different network levels (i.e., cloud and edge).

In this paper, we address the monetization challenge in MEC systems by developing auction-based mechanisms for resource allocation and pricing. We consider a telecom-centric MEC system, where the edge resources are located at the first level of aggregation in the network. Such type of telecom-centric MEC architecture received considerable attention in the literature and significant support from industry and ETSI [8]. Deploying servers at the edge of the network in a telecom-centric MEC system is expensive, and therefore, a limited number of such servers are made available to mobile users. The scarcity of resources at the edge of the network creates a competitive environment for the mobile users, and therefore, there is an urgent need to develop incentive-based resource allocation and pricing mechanisms for MEC.

- The authors are with the Department of Computer Science, Wayne State University, Detroit, MI 48202 USA. E-mail: {tayebeh.bahreini, hossein.badri, dgrosu}@wayne.edu.

Manuscript received 17 May 2020; revised 13 July 2021; accepted 20 July 2021.
Date of publication 26 July 2021; date of current version 11 Aug. 2021.

(Corresponding author: Daniel Grosu.)

Recommended for acceptance by Q. Zheng.

Digital Object Identifier no. 10.1109/TPDS.2021.3099731

1.1 Our Contributions

This paper is an extended version of our previous work on envy-free auction mechanisms for resource allocation in edge computing systems [9]. In our previous work, we developed a greedy resource allocation and pricing mechanism for edge computing systems which considers heterogeneous servers and resources of different types. We assumed that the computing resources have been statically provisioned in the form of virtual machines (VMs). The mechanism combines features from both position [10] and combinatorial auctions [11] and handles heterogeneous resource requests from mobile users and heterogeneous types of resources. It determines *envy-free allocations* (i.e., allocations in which no user can improve her utility by exchanging bids with any user with the same request for resources) and prices that lead to close to optimal social welfare for the users.

Different from our previous work, in this paper, we consider dynamic provisioning of computing resources. Depending on the user demand, the computing resources are provisioned as VM instances. For this new setup, we propose two allocation and pricing mechanisms. The first mechanism is an extension of our previous mechanism [9]. We prove that the proposed mechanism is individually-rational and produces envy-free allocations. In addition, the mechanism requires very small execution time even for very large problem instances with hundreds of users. The second, is a linear programming (LP) based approximation mechanism that does not guarantee envy-freeness, but it provides a solution that is guaranteed to be within a given distance from the optimal solution. We evaluate the performance of the proposed mechanisms through an extensive experimental analysis. For small-size instances, we compare the solutions obtained by our proposed mechanisms with the optimal solutions obtained by solving the Mixed-Integer Linear Programming (MILP) model of the problem using the CPLEX software. Since for large-size instances obtaining the optimal solution in a reasonable amount of time is not feasible, we compare the performance of the proposed mechanisms against each other. We also investigate the impact of dynamic provisioning by comparing the performance of the system under both static and dynamic provisioning.

1.2 Organization

The rest of the paper is organized as follows. In Section 2, we review the recent work on pricing and allocation in MEC systems. In Section 3, we formulate the problem of allocation and pricing. In Section 4, we introduce the proposed envy-free allocation and pricing mechanism. In Section 5, we introduce the proposed approximation mechanism. In Section 6, we describe the experimental setup and discuss the experimental results. In Section 7, we conclude the paper and suggest possible directions for future work.

2 RELATED WORK

A wide range of metrics could be considered when optimizing resource allocation in MEC systems, and depending on the network system, business rules, and assumptions, different constraints should be taken into account. Due to this fact, researchers have addressed a broad spectrum of resource allocation problems in these systems. Some researchers have

devoted their efforts on developing novel algorithms for computation offloading, where computation requirements of applications, network conditions, and users' preferences are taken into account to decide which tasks must be migrated to remote servers [12], [13], [14], [15], [16], [17]. Service placement has been another area of interest for researchers, that is, determining the most efficient server to run a service, where servers could be located in clouds or at the edge of the network [18], [19], [20], [21], [22].

Monetization of services has been identified as a grand challenge in edge computing systems [7]. This fact has led several researchers to concentrate their efforts on designing incentive-based resource allocation mechanisms. In contrast to the significant efforts focused on developing incentive-based resource allocation mechanisms for MCC systems [23], [24], [25], there are only few papers that have recently addressed this challenge in MEC systems.

Xiong *et al.* [26] proposed a pricing mechanism for running mining processes of mobile blockchain applications on the edge servers. They formulated the problem as a two-stage Stackelberg game with the aim of maximizing the profit of the service provider and the individual utilities of the miners. In another work on employing MEC for mobile blockchain, Jiao *et al.* [27] developed a truthful mechanism that maximizes the social welfare. The authors have considered a single service provider and multiple users who are competing for a single type of computational resource on the edge servers. Luong *et al.* [28] also studied the problem of resource allocation in edge computing systems for mobile blockchain applications. They adopted a deep learning approach based on a multi-layer neural network architecture to optimize the loss function which has been defined as the expected, negated revenue of the service provider. Li *et al.* [29] developed a learning-based pricing mechanism in which no profit information of users is required. At equilibrium the edge server induces self-interested users to choose the correct priority class (based on their delay sensitivity) and make socially optimal offloading decisions. However, none of the learning-based approaches consider relative valuations for different computing resources. Baek *et al.* [30] developed an auction-based mechanism for resource allocation in edge computing in which users bid to get more CPU cycles from the edge server. The authors showed that there exists a unique Nash Equilibrium (NE) in the system. Therefore, the payments of users will tend to be fixed after competitions, regardless of the initial payments, and the edge seller can predict the strategies of users and determine the amount of CPU cycles they need. Chen *et al.* [31] studied the problem of multiple resource allocation and pricing in edge computing systems. They decomposed the problem into a set of sub-problems in which each sub-problem only considers a single type of resources. They constructed a Stackelberg game framework for each subproblem and developed algorithms to compute the Stackelberg equilibrium for each type of resource.

To the best of our knowledge, the closest work to ours is by Kiani *et al.* [32] who proposed a three-level hierarchical architecture for mobile edge computing and an auction-based mechanism for VM pricing and resource allocation. Their pricing mechanism is based on the Amazon's Elastic Compute Cloud (EC2) spot pricing. The system determines the price of each type of VMs in each access point. However,

their problem assumes a non-combinatorial auction (NC-Auction) setting, where requests are placed for only one VM instance of a single type. In that setting, bundles of VM requests are not allowed, and if a user is willing to request multiple VM instances of the same type, he/she needs to submit multiple bids. Submitting multiple bids for individual VM instances involves the risk of ending up obtaining only a subset of the requested set of VM instances. However, our mechanisms allow requests for bundles of VM instances and for multiple VM instances of the same type.

Several researchers have applied some of the traditional auction models for solving resource allocation problems in cloud computing systems [33], [34], [35]. The Vickrey-Clarke-Groves (VCG) auction has been one of the most popular truthful auctions [36]. The VCG auction is known to be incentive-compatible and socially optimal. Since achieving truthfulness in VCG requires the optimal solution of the social welfare maximization problem, it is practically infeasible to be applied for problems where the exact solution to the social welfare maximization cannot be obtained in a reasonable amount of time. Also, other traditional auction models are not directly applicable to the edge computing settings where resources are distributed over multiple levels, users typically request bundles of multiple types of resources, and they have different valuations for the services provided from different levels. Since, users request bundles of multiple types of resources, the MEC auction mechanisms fall into the class of combinatorial auctions. In addition, since users have different valuations for resources at different levels, the mechanisms can be classified as position auctions. An example of a non-truthful position auction mechanism is the generalized second price auction employed by Google to sell online advertising [37].

In this paper, we extend our previous work [9] and develop efficient pricing and resource allocation mechanisms for a two-level edge computing system with multiple resources and heterogeneous demands. We consider heterogeneous requests from users, where each user requests for a bundle of different type of resources such as CPU, memory, and disk. In this system, users are single minded. This means that all the bundle requests must be allocated and partial allocation does not have any value for the user. Also, we assume that the competition is on the two levels of resources (edge and cloud), where one level (edge) has more priority for users. This setup is different from the papers discussed above, where only the edge level servers have been considered. Thus, our proposed mechanisms are unique in the sense that they handle the allocation of resources available at the two-levels of the system by combining features from both position and combinatorial auctions. Compared to our previous paper [9], our proposed mechanisms allow dynamic provisioning of VMs, and do not require pre-provisioning the VMs.

3 EDGE RESOURCE ALLOCATION AND PRICING PROBLEM

We address the *Edge Resource Allocation Problem* (ERAP) for an edge computing system with one provider that offers computing resources in the form of Virtual Machines (VMs). In this network, servers are located either at the edge or on the clouds. Since the edge servers are closer to the users, they prefer to run their applications on the edge servers to obtain

TABLE 1
Types of VM Instances Used in the Experiments

Type	vCPU	Memory (GB)	SSD Storage (GB)
medium	1	3.75 (1 unit)	1×4 (1 unit)
Large	2	7.5 (2 units)	1×32 (8 units)
Xlarge	4	15 (4 units)	2×40 (20 units)
2xlarge	8	30 (8 units)	2×80 (40 units)

better performance. In fact, due to low latency requirements of users' applications, it is always more desirable for them to run their applications on edge servers that guarantee lower latency. However, the capacity of edge resources is more restricted than that of the cloud servers. Therefore, users have to compete to obtain computing resources on the edge servers. The provider uses an auction model to provide VM instances, where the price of resources is not pre-defined, but determined by employing a mechanism. The system allows dynamic provisioning of VMs, and does not require pre-provisioning the VMs. Therefore, the provider can fulfill dynamic market demands efficiently.

In this network, there are n users who are competing for resources situated at two levels, edge ($l = 1$), and cloud ($l = 2$), where l denotes the level. In both edge and cloud servers, m types of VM instances are available to serve users, where each type of VM instance is characterized by d types of resources. A VM instance of type j provides q_{jk} units of resource of type k . Here, we consider the aggregated resource capacity at each level, that is, the total capacity at the edge or cloud level. The total amount of resource of type k available at level l is denoted by C_{kl} . As an example, let us consider Amazon's Elastic Compute Cloud (EC2) M3 family of VM instances (see Table 1). In these instances, there are three types of resources ($d = 3$): vCPU ($k = 1$), memory ($k = 2$), and storage ($k = 3$). A 'large' VM instance of type $j = 2$ consists of 2 units of vCPU, 2 units of memory, and 8 units of storage. That is, the instance of type $j = 2$ is characterized by $q_{21} = 2$, $q_{22} = 2$, and $q_{23} = 8$.

User i submits a request for a bundle of VM instances which is denoted by $\theta_i = (b_i; \{r_{ij}\}) = (b_i; r_{i1}, \dots, r_{im})$, where b_i is the bid and r_{ij} is the number of VM instances of type j requested by user i . The main reason for considering the requests as bundles of heterogeneous VMs is that users often require to execute tasks with different functionalities and roles on a set of VMs of different types [25]. It is safe to assume that the total request of a user for each type of resources is less than the available capacity for that type. The requests are submitted to a mechanism that determines the provisioning and allocation of VM instances to users and the price they have to pay for their allocations.

A resource allocation and pricing mechanism for the above setting can be viewed as a hybrid of a multi-unit combinatorial auction [11] and a position auction [10]. It is a position auction because, (i) the auctioneer never allocates the bundle request of a user to more than one level, and (ii) users have different preferences for each level (position) of resources, with resources at the edge being more preferred. To characterize the user's preference for the edge and cloud level, we define α_l as the *preference factor* for level l which is determined based on the average distance between users

and resources at level l (obviously, $\alpha_1 \geq \alpha_2$). Since the latency is determined by the distance between users and servers, the preference factor captures the service latency as well. Furthermore, since there are several VM instances of the same type in each level and users bid for a bundle of resources, the problem can also be viewed as a multi-unit combinatorial auction.

We assume that users are not interested in partially executing their tasks. Also, they do not obtain any value by partially executing on edge and partially on the cloud level (due to the undesirable variable latencies that may occur by executing their tasks on different levels of the network). In other words, we assume that users are single minded [38], that is, a user is only interested in a single bundle. The user values this bundle and any superset of it at the same amount, and all other bundles at zero. If the allocation function allocates the requested bundle of a user and any superset of it in the first level, then the user values the bundle and any superset of that bundle at the same amount. If the allocation function allocates the requested bundle of a user and any superset of it in the second level, then the user values it at the same amount (but this value is less than the value in the first case); otherwise, the user values the allocation at zero.

Let $a_i = \{a_i^1, a_i^2\}$ be the allocation for user i determined by the allocation mechanism, where $a_i^l = (a_{i1}^l, \dots, a_{im}^l)$ is the allocation of resources at level l for user i , and a_{ij}^l is the number of VM instances of type j allocated to this user on level l . Thus, the valuation function for user i is as follows:

$$v_i(a_i) = \begin{cases} \alpha_1 b_i & \text{if } r_i \leq a_i^1 \\ \alpha_2 b_i & \text{if } r_i \leq a_i^2 \text{ and } r_i \not\leq a_i^1, \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where $r_i \leq a_i^l$, if $r_{ij} \leq a_{ij}^l, \forall j \in \{1, \dots, m\}$. The valuation represents the maximum price that a user is willing to pay for the requested bundle at each level. Since $\alpha_1 > \alpha_2$, users prefer the allocation at the edge level instead of that at the cloud level.

In our system, the goal is to maximize the social welfare. The social welfare, V , is the sum of users' valuations, $V = \sum_{l=1}^2 \sum_{i=1}^n \alpha_l \cdot b_i \cdot x_{il}$, where x_{il} is a binary decision variable that is 1, if the bundle requested by user i is allocated at level l ; and 0, otherwise. Maximizing social welfare helps resource providers increase their revenue, due to the fact that the system allocates the available VMs to the users who value them the most [38]. Table 2 shows the notation we use in the formulation of the problem. The ERAP can be formulated as a mixed-integer linear program (MILP) as follows,

ERAP-MILP:

$$\text{maximize } V = \sum_{i=1}^n \sum_{l=1}^2 \alpha_l \cdot b_i \cdot x_{il} \quad (2)$$

$$\text{subject to: } \sum_{i=1}^n \sum_{j=1}^m r_{ij} \cdot q_{jk} \cdot x_{il} \leq C_{kl} \quad \forall k, \forall l \quad (3)$$

$$\sum_{l=1}^2 x_{il} \leq 1 \quad \forall i \quad (4)$$

TABLE 2
Notation

Notation	Description
n	Number of users.
m	Types of VM instances.
d	Number of resource types.
α_l	Preference factor for level l .
b_i	Bid of user i .
r_{ij}	Number of VM instances of type j requested by user i .
C_{kl}	Capacity of resource of type k available at level l .
q_{jk}	Amount of resource of type k for a VM of type j .
w_k	Weight of resource of type k .
π_i	Base price for user i .
p_i	Payment of user i .

$$x_{il} \in \{0, 1\} \quad \forall i, \forall l. \quad (5)$$

The objective function (2) is to maximize the *welfare*, V , where the welfare is the sum of users' valuations. Constraints (3) ensure that the total allocated requests of each type of resources to each level does not exceed the capacity of that level. Constraints (4) guarantee that the request of user i is not allocated to more than one level. Finally, constraints (5) guarantee the integrality of the decision variables.

A pricing mechanism $M = (A, P)$ consists of an allocation function $A = (a_1, \dots, a_n)$ and a payment rule $P = (p_1, \dots, p_n)$. The mechanism determines the allocation of requested bundles and the payments based on the users' bids and the available resources in the system. It determines a base price for each user i , denoted by π_i . Based on the base price, the price of a unit of resource of type k for user i is defined as $\pi_i \cdot w_k$, where w_k is the weight of the resource of type k . Here, w_k is used to differentiate the values of a unit of different types of resources. Therefore, the price of a VM instance of type j for user i is $\sum_{k=1}^d \pi_i \cdot w_k \cdot q_{jk}$.

We define the payment that user i has to pay to the provider as

$$p_i = \sum_{j=1}^m \sum_{k=1}^d r_{ij} \cdot q_{jk} \cdot \pi_i \cdot w_k. \quad (6)$$

We assume that users have quasi-linear utilities (i.e., $u_i = v_i - p_i$) and are rational, in the sense that their goal is to maximize their utility.

3.1 Complexity of ERAP

Here, we prove that the decision version (ERAP-D) of ERAP is NP-complete. This implies that ERAP is NP-hard. An instance of ERAP-D consists of: a set of users, users' requests $\theta_i = \{b_i, \{r_{ij}\}\}$, $i = 1, \dots, n$, k computational resources of given capacities $\{C_{kl}\}$, at both levels (i.e., edge and cloud), preference factor α_l at level l , and a bound $B \in \mathbb{R}^+$. The decision problem is to determine whether there exists an allocation of users such that the social welfare of the assignment (Equation (2)) is at least B , and no capacity constraint is violated.

Theorem 1. ERAP-D is NP-complete.

Proof. We prove that ERAP-D is NP-complete by showing that: (i) ERAP-D belongs to NP, and, (ii) a well known NP-complete problem is reduced to ERAP-D in polynomial time. For any arbitrary allocation of requests, the feasibility check of the capacity constraints and computing the social welfare (Equation (2)) could be performed in polynomial time, which implies that ERAP-D is in NP.

For the second condition, we consider a special case of the Multiple Multi-dimensional Knapsack Problem, MMKP, with a fixed number of knapsacks (two knapsacks, one for each of the two network levels) and a fixed number of dimensions. For simplicity, we denote this special case by 2D-MMKP. The multi-dimensional knapsack problem for any fixed number of dimensions is NP-complete [39]. Also, it is well known that the multiple knapsack problem is NP-complete even when the number of knapsacks is two [40]. Therefore, 2D-MMKP is NP-complete.

Now, we show that 2D-MMKP can be reduced to ERAP-D in polynomial time. Let us define an arbitrary instance of 2D-MMKP with n' items. Each item i has a value v_i and a weight ω_{ij} in dimension j . The capacity of knapsack l in dimension k is C'_{kl} . The decision problem is to determine whether there is an assignment of items to each of the two knapsacks, such that the total value of the items is at least L , while the total weight on each dimension of each knapsack does not exceed the capacity.

We construct an instance of ERAP-D, called Q , based on an arbitrary instance of 2D-MMKP, called Q' , such that the total social welfare of Q is at least B , if and only if, the total value of Q' is at least L . Instance Q consists of n items such that $n = n'$. The capacity of resources at level l for resources of type k is defined as $C_{kl} = C'_{kl}$. Let us assume $\alpha_1 = \alpha_2 = 1$. The bid of user i is defined as $b_i = v_i$. Also, we let $r_{ij} = \omega_{ij}$, and $q_{jk} = 1$. We claim that Q has a feasible assignment with social welfare greater than or equal to B , if and only if, Q' has a feasible assignment with the total value greater than or equal to L . Since we assume $q_{jk} = 1$, any feasible solution for Q is also a feasible solution for Q' . Also, since we assume that $\alpha_1 = \alpha_2 = 1$, the social welfare of the solution for Q is equivalent to the total value obtained for Q' . Conversely, suppose that there is an assignment in Q' with total value greater than or equal to L . We assign the corresponding users to the edge/cloud levels. Clearly, any feasible solution for Q' is also a feasible solution for Q and the total value of the solution for Q' is equivalent to the social welfare of the solution for Q . \square

We proved that ERAP is NP-hard, that is, it is not possible to find an optimal solution in polynomial time, unless $P = NP$. On the other hand, incorporating pricing into a resource allocation model exacerbates the complexity of the problem. Given the fact that a resource allocation mechanism has to be used efficiently for instances with a large number of requests and in order to solve the problem in reasonable time, we have to leverage heuristic methods instead of using commercial solvers. In this paper, we design two mechanisms for edge resource allocation and pricing problem, G-ERAP and APX-ERAP. G-ERAP is a greedy mechanism for resource allocation and pricing in edge systems, that is individually-rational and envy-free. APX-ERAP is an LP-based

$\left(\frac{1}{2d+1}\right)$ -approximation mechanism for resource allocation and pricing in edge systems. In the following sections, we describe the proposed mechanisms and discuss their properties.

4 ENVY-FREE RESOURCE ALLOCATION AND PRICING MECHANISM

In this section, we design a greedy mechanism for resource allocation and pricing, called G-ERAP. G-ERAP, which is given in Algorithm 1, considers uniform base prices for each type of VM in the same level. In other words, the price per unit of each VM type is the same for winners at the same level of the system, but winners at different levels should pay differently for the same type of VM instance. The mechanism is invoked periodically at time intervals of a specified duration. The allocation and price determined by the mechanism is valid for the current time interval.

Algorithm 1. G-ERAP Mechanism

Input: Vector of requests: $\theta_i = (b_i; \{r_{ij}\})$
 Vector of resources' capacities: $C = \{C_{lk}\}$
output: Allocation matrix: $X = \{x_{il}\}$
 Total welfare: V
 Payment vector: $P = \{p_i\}$

- 1: $V \leftarrow 0$
- 2: $X \leftarrow 0$
- 3: $B_i \leftarrow \frac{b_i}{\sum_{j=1}^m \sum_{k=1}^d r_{ij} \cdot q_{jk} \cdot w_k} \quad \forall i \in \{1, \dots, n\}$
- 4: Sort users in non-increasing order of their B_i
- 5: $u \leftarrow 0, u' \leftarrow 0, l \leftarrow 1, i \leftarrow 1$
- 6: **while** $i \leq n$ **do**
- 7: **if** $C_{lk} \geq \sum_{j=1}^m r_{ij} \cdot q_{jk} \quad \forall k \in \{1, \dots, d\}$ **then**
- 8: $C_{lk} \leftarrow C_{lk} - \sum_{j=1}^m r_{ij} \cdot q_{jk} \quad \forall k \in \{1, \dots, d\}$
- 9: $V \leftarrow V + \alpha_l \cdot b_i$
- 10: $x_{il} \leftarrow 1$
- 11: $i \leftarrow i + 1$
- 12: **else**
- 13: **if** $l = 1$ **then**
- 14: $u \leftarrow i - 1$
- 15: $l \leftarrow l + 1$
- 16: **else**
- 17: $u' \leftarrow i$
- 18: **break**
- 19: **if** $u' < n$ **then**
- 20: $B^* \leftarrow B_{u'+1}$
- 21: **else**
- 22: $B^* \leftarrow B_u - \epsilon$
- 23: **for each** $i \in \{1, \dots, n\}$ **do**
- 24: **if** $x_{i2} = 1$ **then**
- 25: $\pi_i \leftarrow \alpha_2 \cdot B^* + \frac{(\alpha_1 - \alpha_2)}{2} (B_u + B_{u+1})$
- 26: **else**
- 27: **if** $x_{i1} = 1$ **then**
- 28: $\pi_i \leftarrow \alpha_2 \cdot B^*$
- 29: **else**
- 30: $\pi_i \leftarrow 0$
- 31: $p_i \leftarrow \sum_{j=1}^m \sum_{k=1}^d r_{ij} \cdot q_{jk} \cdot \pi_i \cdot w_k$

The mechanism collects the requests of users and prioritize them based on their *bid densities*, that is, the average bid per unit of resource. The bid density of user i is defined as

$$B_i = \frac{b_i}{\sum_{j=1}^m \sum_{k=1}^d r_{ij} \cdot q_{jk} \cdot w_k}. \quad (7)$$

The mechanism starts the allocation from the edge level for the requests with relatively high density bid. Once it reaches to a request which is not fitted to the edge level (according to the size of request), the mechanism starts the allocation on the cloud level.

The input to G-ERAP consists of a vector of requests θ_i , $i = 1, \dots, n$ from users, and a vector of resource capacities, C . G-ERAP determines how these resources are allocated to users. The output of the mechanism consists of the allocation matrix X , where $X = [x_{il}]$, $i = 1, \dots, n$, and $l = 1, 2$, the social welfare V , and the payment vector $P = \{p_i\}$. First, the mechanism determines the bid density for each user (line 3). Then, users are sorted in non-increasing order of their bid densities (line 4), and VM instances are allocated to users starting from the first level (i.e., the edge level). For the current user, the mechanism checks if there are enough resources at the current level. If so, the requested VM instances are allocated to the user, and the social welfare and the capacity are updated (lines 7-11). If there are not enough resources to allocate the requested bundle at the first level, then the index of the level is increased by one (i.e., starting allocation of VM instances on the second level), and the index of the user is stored as the first allocated user in the second level. This user is denoted by u (lines 12-15). The allocation stops once it reaches a user (denoted by u') for which there are not enough resources to satisfy the requested bundle in the second level (lines 16-18).

Next, G-ERAP determines the payments for each user (lines 19-31). According to the mechanism, user u is the last user in the sorted order which is allocated to the first level. Therefore, user $u + 1$ is the first user in the list that is to be allocated on the second level. The base price for each user i allocated at edge level is determined as follows:

$$\pi_i = \alpha_2 B^* + \frac{(\alpha_1 - \alpha_2)}{2} (B_u + B_{u+1}). \quad (8)$$

The base price for each user i allocated at cloud level is determined as follows:

$$\pi_i = \alpha_2 B^*, \quad (9)$$

where

$$B^* = \begin{cases} B_{u'+1} & \text{if } u' < n \\ B_{u'} - \epsilon & \text{otherwise} \end{cases}, \quad (10)$$

and u' is the last allocated user at the cloud level ($l = 2$). If some users in the sorted list cannot be allocated at the cloud level, then B^* is defined based on the bid density of the first unallocated user in the sorted list. But, if all the remaining users from the first level are allocated at the second level, then B^* has a value a bit less than the weighted bid of the last allocated user (i.e., $B_{u'} - \epsilon$, where ϵ is a very small positive real number). Note that if a user is not allocated the requested bundle, then the base payment is 0 (line 30). After determining the base payments, G-ERAP determines the payment of users according to Equation (6) (line 31).

4.1 Properties of G-ERAP

G-ERAP combines features from both position [10] and combinatorial auctions [11] and is not truthful. Here, we do not target truthfulness for our mechanism but instead, we guarantee that it produces envy-free allocations. Truthful mechanisms are desirable from the user perspective because truth-telling is the dominant strategy. Thus, users know that by submitting their true valuation they maximize their utilities independent of the bids of the other users. However, truthful mechanisms are not necessarily the most desirable mechanisms from the auctioneer's perspective. For example, search engines, such as Google prefer to employ the Generalized Second Price (GSP) mechanism to sell online advertising [37]. GSP is an envy-free mechanism and is not truthful, but it generates more revenue than truthful mechanisms such as Vickrey-Clarke-Groves (VCG). In addition, GSP has a simple design and is very fast.

Envy freeness allows us to design a computationally efficient mechanism suitable for providing services in a two-level edge computing systems, where the edge resources are at the first level and the cloud resources at the second. In the following, we prove that G-ERAP mechanism is *individually-rational* and produces *envy-free allocations* which are two important and desirable properties of a mechanism [36]. The first property guarantees that users are willing to participate in the mechanism, while the second guarantees that when the auction is terminated, no user would be happier with the outcome of another user.

Definition 1 (Individual Rationality). *A mechanism is individually rational if for each user i bidding her true valuation for the bundle, $v_i - p_i \geq 0$ (i.e., a user reporting her true valuation for the bundle never incurs a loss).*

Lemma 2. *The base price of a user allocated at the edge level, satisfies $\pi_i \leq \alpha_1 B_u$, where u is the last user allocated at the edge level ($l = 1$).*

Proof. The base price of a user i allocated at the edge level is given by:

$$\begin{aligned} \pi_i &= \alpha_2 B^* + \frac{(\alpha_1 - \alpha_2)}{2} (B_u + B_{u+1}) \\ &\leq \alpha_2 B^* + \frac{(\alpha_1 - \alpha_2)}{2} 2B_u \leq \alpha_2 B^* + (\alpha_1 - \alpha_2) B_u \\ &\leq \alpha_1 B_u + \alpha_2 (B^* - B_u). \end{aligned}$$

Because user u is the last user allocated at the edge level, $B^* \leq B_u$, and thus, $\pi_i \leq \alpha_1 B_u$. \square

Theorem 3. *G-ERAP is individually-rational.*

Proof. To prove this theorem, we need to show that the utility of a user reporting (bidding) her true valuation for the requested bundle is non-negative. There are three possible outcomes for a participant user denoted by i :

Case I. User i is allocated to the edge level ($B_i \geq B_u$)

$$v_i - p_i = \alpha_1 B_i \sum_{j=1}^m \sum_{k=1}^d w_k \cdot r_{ij} \cdot q_{jk} - \pi_i \sum_{j=1}^m \sum_{k=1}^d w_k \cdot r_{ij} \cdot q_{jk}$$

$$\geq \alpha_1 B_u \sum_{j=1}^m \sum_{k=1}^d w_k \cdot r_{ij} \cdot q_{jk} - \pi_i \sum_{j=1}^m \sum_{k=1}^d w_k \cdot r_{ij} \cdot q_{jk}.$$

Using the result of Lemma 2, we have, $v_i - p_i \geq 0$.

Case II. User i is allocated to the cloud level ($B_u \geq B_i \geq B^*$)

$$v_i - p_i = \alpha_2 B_i \sum_{j=1}^m \sum_{k=1}^d w_k \cdot r_{ij} \cdot q_{jk} - \pi_i \sum_{j=1}^m \sum_{k=1}^d w_k \cdot r_{ij} \cdot q_{jk}$$

$$\geq \alpha_2 B^* \sum_{j=1}^m \sum_{k=1}^d w_k \cdot r_{ij} \cdot q_{jk} - \pi_i \sum_{j=1}^m \sum_{k=1}^d w_k \cdot r_{ij} \cdot q_{jk}.$$

According to Equation (9), $\pi_i = \alpha_2 B^*$. Thus, $v_i - p_i \geq 0$.

Case III. User i loses the auction ($B^* \geq B_i$). According to Equation (1), $v_i = 0$. Because user i is not allocated any bundle of VMs, $p_i = 0$. Thus, $v_i - p_i = 0$. \square

Definition 2 (Envy-Freeness). An allocation is envy-free if no user can improve her utility by exchanging bids with any user with the same request for VM instances.

Theorem 4. G-ERAP produces envy-free allocations.

Proof. Let us assume that user i is allocated to the first level (edge), user i' is allocated to the second level (cloud), and user i'' loses the auction, and all of them have the same request for VM instances. It is obvious that a user cannot improve her utility by exchanging bids with another user with an identical request for VM instances that is allocated to the same level. Therefore, we only need to show three cases.

Case I. Users i and i' cannot improve their utilities by exchanging their bids. By exchanging their bids, user i' is allocated to the first level (edge) and user i is allocated to the second level (cloud). In this case, we need to show that

$$v_i(a_i) - p_i \geq v_i(a_{i'}) - p_{i'}, \quad (11)$$

and

$$v_{i'}(a_{i'}) - p_{i'} \geq v_{i'}(a_i) - p_i. \quad (12)$$

These equations are equivalent to

$$\begin{aligned} \alpha_1 b_i - \pi_i \sum_{j=1}^m \sum_{k=1}^d w_k \cdot r_{ij} \cdot q_{jk} &\geq \\ \alpha_2 b_i - \pi_{i'} \sum_{j=1}^m \sum_{k=1}^d w_k \cdot r_{ij} \cdot q_{jk}, &\quad (13) \end{aligned}$$

and

$$\begin{aligned} \alpha_2 b_{i'} - \pi_{i'} \sum_{j=1}^m \sum_{k=1}^d w_k \cdot r_{i'j} \cdot q_{jk} &\geq \\ \alpha_1 b_{i'} - \pi_i \sum_{j=1}^m \sum_{k=1}^d w_k \cdot r_{i'j} \cdot q_{jk}. &\quad (14) \end{aligned}$$

According to the definition of B_i (Equation (7)), we rewrite Equation (13) as

$$\begin{aligned} \alpha_1 B_i \sum_{j=1}^m \sum_{k=1}^d w_k \cdot r_{ij} \cdot q_{jk} - \pi_i \sum_{j=1}^m \sum_{k=1}^d w_k \cdot r_{ij} \cdot q_{jk} &\geq \\ \alpha_2 B_i \sum_{j=1}^m \sum_{k=1}^d w_k \cdot r_{ij} \cdot q_{jk} - \pi_{i'} \sum_{j=1}^m \sum_{k=1}^d w_k \cdot r_{ij} \cdot q_{jk} &\implies \alpha_1 B_i - \pi_i \geq \alpha_2 B_i - \pi_{i'}. \end{aligned} \quad (15)$$

Similarly, Equation (14) is equivalent to

$$\alpha_2 B_{i'} - \pi_{i'} \geq \alpha_1 B_{i'} - \pi_i. \quad (16)$$

Based on Equations (15) and (16), we only need to show that

$$B_{i'}(\alpha_1 - \alpha_2) \leq \pi_i - \pi_{i'} \leq B_i(\alpha_1 - \alpha_2). \quad (17)$$

Using the definition of B_u and B^* from Equations (8) and (10) we obtain, $B^* \leq B_{i'} \leq B_{u+1} \leq B_u \leq B_i$. Also, based on Equations (8) and (9)

$$\pi_i - \pi_{i'} = \frac{(\alpha_1 - \alpha_2)}{2} (B_u + B_{u+1}). \quad (18)$$

Thus

$$\begin{aligned} \frac{(\alpha_1 - \alpha_2)}{2} (B_{i'} + B_{i'}) &\leq \pi_i - \pi_{i'} \leq \frac{(\alpha_1 - \alpha_2)}{2} (B_i + B_i) \\ (\alpha_1 - \alpha_2) B_{i'} &\leq \pi_i - \pi_{i'} \leq (\alpha_1 - \alpha_2) B_i. \end{aligned} \quad (19)$$

Therefore, for this case, G-ERAP produces envy-free allocations.

Case II. Users i and i'' cannot improve their utilities by exchanging their bids. In this case, user i loses the auction and user i'' is allocated to the first level (edge). It is obvious that the utility of user i does not improve, thus, we only need to show that the utility of user i'' does not improve. In this case, we need to show that, $v_{i''}(a_{i''}) - p_{i''} \geq v_{i''}(a_i) - p_i$. This is equivalent to show that

$$0 \geq \alpha_1 B_{i''} \sum_{j=1}^m \sum_{k=1}^d w_k \cdot r_{i''j} \cdot q_{jk} - \pi_i \sum_{j=1}^m \sum_{k=1}^d w_k \cdot r_{i''j} \cdot q_{jk}, \quad (20)$$

which is equivalent to show that

$$0 \geq \alpha_1 B_{i''} - \pi_i. \quad (21)$$

Also, based on Equation (8)

$$\pi_i \geq \alpha_2 B^* + \frac{(\alpha_1 - \alpha_2)}{2} (2B^*) \implies \pi_i \geq \alpha_1 B^* \geq \alpha_1 B_{i''}, \quad (22)$$

which satisfies Equation (21).

Case III. User i' and user i'' cannot improve their utility by exchanging their bids. By exchanging their bids, user i' loses the auction and user i'' is allocated to the second level

(i.e., cloud). In this case, the utility of user i' does not improve, thus, we only need to show that the utility of user i'' does not improve. For this purpose, we need to show that, $v_{i''}(a_{i''}) - p_{i''} \geq v_{i'}(a_{i'}) - p_{i'}$. This is equivalent to show that

$$0 \geq \alpha_2 B_{i''} \sum_{j=1}^m \sum_{k=1}^d w_k \cdot r_{i''j} \cdot q_{jk} - \pi_{i'} \sum_{j=1}^m \sum_{k=1}^d w_k \cdot r_{i''j} \cdot q_{jk}, \quad (23)$$

which is equivalent to show that

$$0 \geq \alpha_2 B_{i''} - \pi_{i'}. \quad (24)$$

According to Equation (10), $B^* \geq B_{i''}$. Therefore, based on Equation (9), $\pi_{i'} = \alpha_2 B^* \geq \alpha_2 B_{i''}$ which satisfies Equation (24). \square

Now, we investigate the time complexity of G-ERAP. The time complexity of computing the bid densities (line 3) is $O(nmd)$. Sorting the users (line 4) takes $O(n \log n)$. The main part of the algorithm consists of the loop in lines 7-19 which executes n times. In each iteration, the available capacity for each type of resources is compared with the total request of the user for that resource type (lines 8-9), which takes $O(md)$. Therefore, the time complexity of G-ERAP is $O(n \log n + nmd)$.

5 LP-BASED APPROXIMATION MECHANISM FOR RESOURCE ALLOCATION AND PRICING

In this section, we focus on designing a mechanism that provides guarantees with respect to how far from the optimal is the solution obtained by the mechanism. We develop an LP-based approximation mechanism, APX-ERAP, for solving the ERAP. APX-ERAP is an extension of a $(\frac{1}{d+1})$ -approximation algorithm [41] for the d -dimensional knapsack problem, MDKP. In fact, ERAP can be viewed as a weighted multi-dimensional multiple knapsack problem composed of two knapsacks (two levels of resources). Each knapsack has d dimensions (d types of resources) with a limited capacity, C_{lk} , for each dimension k . There are n items (n users) that are to be assigned to the knapsacks. The size of item i in dimension k is $\sum_{j=1}^m r_{ij} \cdot q_{jk}$, the profit of item i in knapsack l is $\alpha_l \cdot b_{il}$ and the objective is to maximize the total profit which is $\sum_{i=1}^n \alpha_l \cdot b_{il} \cdot x_{il}$.

The original algorithm [41] first solves the LP relaxation of MDKP which considers only one d -dimensional knapsack. The LP relaxation solution contains a set of completely assigned items, I , and a set of at most d fractionally assigned items, F . Then, the algorithm considers two feasible solutions for the knapsack: assigning all items in I to the knapsack or assigning the most profitable item in F to the knapsack. The algorithm selects the solution that maximizes the profit. In APX-ERAP, we extend this idea to solve the ERAP, which considers two d -dimensional knapsacks. The LP relaxation solution of ERAP-MILP contains a set of completely assigned users at the edge level, I_1 , a set of completely assigned users at the cloud level, I_2 , a set of at most d fractionally assigned items at the edge level, F_1 , and a set of at most d fractionally assigned items at the cloud level, F_2 . Then, based on these sets, the algorithm determines the

most valuable allocations for the edge level and the cloud level.

Algorithm 2. APX-ERAP

Input: Vector of requests: $\theta_i = (b_i; \{r_{ij}\})$
 Vector of resources' capacities: $C = \{C_{lk}\}$
output: Allocation matrix: $X = \{x_{il}\}$
 Total welfare: V
 Payment vector: $P = \{p_i\}$

- 1: $V \leftarrow 0$
- 2: $X \leftarrow \{0\}$
- 3: $B_i \leftarrow \frac{b_i}{\sum_{j=1}^m \sum_{k=1}^d w_k \cdot r_{ij} \cdot q_{jk}} \quad \forall i \in \{1, \dots, n\}$
- 4: Sort users in non-increasing order of B_i
- 5: $\bar{X} \leftarrow \text{LP-SOLVE(ERAP)}$
- 6: $F_l \leftarrow \{i : 0 < \bar{x}_{il} < 1\}, l = 1, 2$
- 7: $I_l \leftarrow \{i : \bar{x}_{il} = 1\}, l = 1, 2$
- 8: $v_l \leftarrow \sum_{i \in I_l} b_i, l = 1, 2$
- 9: $max_1 \leftarrow \arg\max_{i \in F_1 \cup F_2} b_i$
- 10: $max_2 \leftarrow \arg\max_{i \in F_1 \cup F_2 \setminus \{max_1\}} b_i$
- 11: **case1:** $v_1 < b_{max_1}$ and $v_2 < b_{max_2}$
- 12: $V \leftarrow \alpha_1 \cdot b_{max_1} + \alpha_2 \cdot b_{max_2}$
- 13: $x_{max_1 1} \leftarrow 1$
- 14: $x_{max_2 2} \leftarrow 1$
- 15: **case 2:** $v_1 \geq b_{max_1}$ and $v_2 \geq b_{max_2}$
- 16: $V \leftarrow \alpha_1 \cdot v_1 + \alpha_2 \cdot v_2$
- 17: $x_{i1} \leftarrow 1 \quad \forall i \in I_1$
- 18: $x_{i2} \leftarrow 1 \quad \forall i \in I_2$
- 19: **case3:** $v_1 < b_{max_1}$ and $v_2 \geq b_{max_2}$
- 20: $V \leftarrow \alpha_1 \cdot b_{max_1} + \alpha_2 \cdot v_2$
- 21: $x_{max_1 1} \leftarrow 1$
- 22: $x_{i2} \leftarrow 1 \quad \forall i \in I_2$
- 23: **case4:** $v_1 \geq b_{max_1}$ and $v_2 < b_{max_2}$
- 24: $V \leftarrow \alpha_1 v_1 + \alpha_2 \cdot b_{max_2}$
- 25: $x_{i1} \leftarrow 1 \quad \forall i \in I_1$
- 26: $x_{max_2 2} \leftarrow 1$
- 27: **for each** $i \in \{1, \dots, n\}$ **do**
- 28: **if** $x_{i1} = 1$ **then**
- 29: $\pi_i \leftarrow \alpha_1 B_{i+1}$
- 30: **else**
- 31: **if** $x_{i2} = 1$ **then**
- 32: $\pi_i \leftarrow \alpha_2 B_{i+1}$
- 33: **else**
- 34: $\pi_i \leftarrow 0$
- 35: $p_i \leftarrow \sum_{j=1}^m \sum_{k=1}^d r_{ij} \cdot q_{jk} \cdot \pi_i \cdot w_k$

APX-ERAP is given in Algorithm 2. The input to APX-ERAP consists of a vector of requests from users, θ_i , and a vector of resource capacities, C . The output of the mechanism consists of the allocation matrix, $X = \{x_{il}\}$, the social welfare V , and the vector of payments of users, $P = \{p_i\}$. The mechanism first sorts the users in non-increasing order of their bid densities (lines 3-4). This order is used when the algorithm determines the payment of users. Then, the algorithm solves the LP relaxation of ERAP-MILP by calling LP-SOLVE(ERAP) and saves the solution in matrix $\bar{X} = \{\bar{x}_{il}\}$ (line 5). Then, it defines $F_l = \{i : 0 < \bar{x}_{il} < 1\}$ as the set of users that have fractional allocations at level l , $I_l = \{i : \bar{x}_{il} = 1\}$, as the set of users that are completely allocated on level l , and v_l as the total bids of users who are completely allocated on level l (lines 6-7). One feasible solution for the problem is to allocate users in I_1 at the edge level and users in I_2 at the cloud

level. An alternative solution is to select two users from $F_1 \cup F_2$ and allocate one of them at the edge level and the other one at the cloud level. This solution is also feasible, because we assume that the size of each user's request is less than the capacity. Based on these facts, the mechanism determines the allocation at the edge level and the cloud level.

To obtain the maximum social welfare, the mechanism compares the two highest bids of users in $F_1 \cup F_2$ with the sum of the bids of users in I_1 , v_1 , and the sum of the bids of users in I_2 , v_2 . For this purpose, it determines the index of the two highest bidders in $F_1 \cup F_2$, denoted by max_1 and max_2 (lines 9-10). Then, it considers four possible cases for values of v_1 , v_2 , b_{max_1} , and b_{max_2} (lines 11-26). In each case, the mechanism allocates users in such a way that the social welfare is maximized. In the first case, since $v_1 < b_{max_1}$ and $v_2 < b_{max_2}$, the mechanism allocates the user with index max_1 at the edge level and user with index max_2 at the cloud level. In the second case, since $v_1 \geq b_{max_1}$ and $v_2 \geq b_{max_2}$, the mechanism allocates users in I_1 at the edge level and users in I_2 at the cloud level. In the third case, since $v_1 < b_{max_1}$ and $v_2 \geq b_{max_2}$, it allocates the user with index max_1 at the edge level and users in I_2 at the cloud level. Finally, in the last case, since $v_1 \geq b_{max_1}$ and $v_2 < b_{max_2}$, the mechanism assigns users in I_1 at the edge level and user with index max_1 at the cloud level.

Next, APX-ERAP determines the payments for users (lines 27-35). The algorithm considers variable base prices in which winners of the same level may have different base prices. For user i allocated at level l , the base price is calculated as

$$\pi_i = \alpha_l B_{i+1}, \quad (25)$$

where B_{i+1} is the highest bid density of the user after user i in the non-increasing order of bids.

5.1 Properties of APX-ERAP

Here, we prove that APX-ERAP is a $\left(\frac{1}{2d+1}\right)$ -approximation mechanism, where d is the number of types of physical resources. We also show that the mechanism is *individually-rational*.

Lemma 5. *In the LP relaxation of ERAP, where $0 \leq x_{il} \leq 1$, there are at most d users that are fractionally allocated at the edge level and at most $2d$ users that are fractionally allocated at the cloud level.*

Proof. Let $\bar{x} = \{\bar{x}_i^l\}$ be the solution obtained by LP-relaxation, and $F_l = \{i : 0 < \bar{x}_{il} < 1\}$ be the set of users that are fractionally allocated at level l . For user i that is fractionally allocated at the edge level ($i \in F_1$), Constraint (4) and Constraint ($0 \leq x_{il} \leq 1$) are redundant. The reason is that, since $\alpha_1 > \alpha_2$, the only constraint that does not allow a higher fraction of allocation for user i at the edge level is the limited capacity (Constraint (3)). Since Constraint (3) is the only set of constraints in the LP relaxation model, the total number of constraints related to the edge level is d . Therefore, any basic feasible solution of this relaxation has at most d fractional variables at the edge level (i.e., $|F_1| \leq d$).

For the set of users that are fractionally allocated at the cloud level ($i \in F_2$), there are two possible subsets: (i) the set of users that are fractionally allocated at both edge

level and cloud level (i.e., $F_1 \cap F_2$); (ii) the set of users that are fractionally allocated at the cloud level only (i.e., $F_2 \setminus F_1$). It is obvious that the number of users in the first set is not more than d . In the second set, the only constraint that does not allow a higher allocation is the capacity constraint (Constraint (3)). Thus, Constraint (4) and Constraint ($0 \leq x_{il} \leq 1$) are redundant for users in this set. Therefore, there are at most d fractional variables associated with the users in this set. Therefore, the total number of users fractionally allocated at the cloud level is at most $2d$. \square

Theorem 6. *APX-ERAP is a $\left(\frac{1}{2d+1}\right)$ -approximation mechanism for ERAP.*

Proof. Let X^* be the optimal allocation matrix and OPT be the total social welfare of the optimal solution. The solution to the LP relaxation is an upper bound on the optimal solution

$$OPT \leq LP \leq \alpha_1 \cdot \left(\sum_{i \in F_1} b_i + \sum_{i \in I_1} b_i \right) + \alpha_2 \cdot \left(\sum_{i \in F_2} b_i + \sum_{i \in I_2} b_i \right). \quad (26)$$

For the four possible cases mentioned in Algorithm 2, we can easily show that

$$V \geq \alpha_1 \cdot b_i + \alpha_2 \cdot b_j \quad \forall i \in F_1, j \in F_2. \quad (27)$$

According to Lemma 5, there are at most d items in F_1 and at most $2d$ items in F_2 . Therefore

$$2d \cdot V \geq \alpha_1 \cdot \sum_{i \in F_1} b_i + \alpha_2 \cdot \sum_{i \in F_2} b_i. \quad (28)$$

Also, $V \geq \alpha_1 \cdot v_1 + \alpha_2 \cdot v_2$. Thus

$$(2d+1) \cdot V \geq \alpha_1 \sum_{i \in F_1} b_i + \alpha_2 \sum_{i \in F_2} b_i + \alpha_1 v_1 + \alpha_2 v_2 \geq OPT. \quad (29)$$

Therefore, $V \geq \frac{OPT}{2d+1}$. \square

Theorem 7. *APX-ERAP is individually-rational.*

Proof. To prove this theorem, we need to show that the utility of a user reporting her true valuation for the requested bundle is non-negative. There are two possible outcomes of each user i :

Case I. User i is allocated to level l

$$v_i - p_i = \alpha_l B_i \sum_{j=1}^m \sum_{k=1}^d w_k \cdot r_{ij} \cdot q_{jk} - \alpha_l B_{i+1} \sum_{j=1}^m \sum_{k=1}^d w_k \cdot r_{ij} \cdot q_{jk}.$$

Therefore, since $B_i \geq B_{i+1}$, we have, $v_i - p_i \geq 0$.

Case II. User i loses the auction. According to Equation (1), $v_i = 0$. Because user i is not allocated any bundle of VMs, $p_i = 0$. Thus, $v_i - p_i = 0$. \square

Now, we investigate the time complexity of APX-ERAP. The most time consuming part of the algorithm is solving the LP relaxation of ERAP-MILP, which takes polynomial time [42]. The other parts of the algorithm also have polynomial time complexity. Therefore, the time complexity of the algorithm is polynomial.

6 EXPERIMENTAL ANALYSIS

We perform an extensive experimental analysis to evaluate the performance of the proposed mechanisms, G-ERAP and APX-ERAP with respect to several key metrics. First, we compare the performance of the mechanisms with that of the optimal solution obtained by solving small-size instances of the ERAP-MILP problem using the CPLEX solver [43]. Second, we compare the performance of the two proposed mechanisms for large-size problem instances. Since for the large-size instances, obtaining the optimal solution for ERAP-MILP within a reasonable amount of time is not possible, we compare the performance of the proposed mechanisms with that of the LP relaxation of ERAP-MILP. Furthermore, to investigate the impact of dynamic provisioning, we compare the performance of the mechanisms under both dynamic provisioning and static pre-provisioning. In the following, we describe the experimental setup and analyze the experimental results.

6.1 Experimental Setup

We generate several problem instances with different number of users, demands, and capacities for the edge and cloud. In our experiments, the provider offers four types of VM instances as shown in Table 1. These types of VM instances are based on Amazon's Elastic Compute Cloud (EC2) M3 family of instances. In this family, four types of VM instances are defined, medium, large, xlarge, and 2xlarge. Each type of VM instance provides a combination of three types of resources, vCPU, memory, and storage. We assume that the same types of VMs can be provided at both the edge and the cloud levels. However, depending on the capacity of resources, during the dynamic provisioning, some types of VMs may not be provisioned at the edge or cloud level. We define 3.75 GB of memory as one unit of memory and every 4GB of storage as one unit of storage. Therefore, in the EC2 VM instances, the size of CPU and memory varies from 1 unit to 8 units, while the size of storage varies from 1 unit to 40 units. The distribution of the other parameters that are used to generate problem instances in our simulation experiments are shown in Table 3. In this table, we denote by $U[a, b]$, the uniform distribution within interval $[a, b]$. The bid b_i of user i must be proportional to the total amount of request of users i . Therefore, to generate bid b_i , we first draw from $U[1, 10]$, the bid per unit of resource. Then, we multiply this value by the total size of the request of user i , $R_i = \sum_{j=1}^m \sum_{k=1}^d w_k \cdot q_{jk} \cdot r_{ij}$. We also use the uniform distribution to generate r_{ik} , the number of VM instances of each type that are requested by user i . We evaluate the performance of the mechanisms for both small-size instances and large-size instances. For small-size instances, the number of users varies from 100 to 500. Since the number of VM instances of each type of VM is drawn from $U[0, 10]$, the total request of users for each type of VM varies from $100 \cdot U[0, 10]$ to $500 \cdot U[0, 10]$. Thus, to cover the requests of a reasonable number of users, the available capacity of each type of resources must be proportional to these values. For CPU and memory, the available capacity is 25000, while for the storage the available capacity we use 200000. For large-size instances, the number of users varies from 5000 to 50000. Thus, to cover the requests of a reasonable number of

TABLE 3
Simulation Parameters for Small Instances

Parameter	Distribution
C_1, C_2	small: 25000 large: 250000
C_3	small: 200000 large: 2000000
r_{ik}	$U[0, 10]$
w	$[1, 1, 1]$

users, the available capacity of CPU and memory is 250000, while for the storage the available capacity we use 2000000. The vector of resource weights w , for the three types of resource is given in the last row of the table. For each size of instances, we execute G-ERAP and APX-ERAP, and CPLEX for 10 randomly generated instances and perform our analysis based on the average values.

One of the important factors on the performance of the mechanisms is the amount of available resources at the edge level and the cloud level. Therefore, we define a parameter ρ_{EC} , called *edge-cloud resource capacity ratio*, which is the sum of the ratios of the capacity for each type of resources at the edge and the capacity at the cloud level, $\rho_{EC} = \sum_{k=1}^d \left(\frac{C_{k1}}{C_{k2}} \right)$.

Social welfare and revenue are two important measures for the efficiency of resource allocation mechanisms. To characterize the welfare and revenue obtained by G-ERAP and APX-ERAP, we define two relative metrics: (i) the *social welfare ratio*, $\mathcal{V}^r = \frac{V}{V^*}$, where V is the social welfare obtained by G-ERAP or APX-ERAP, and V^* is the social welfare of the optimal solution obtained by CPLEX; and, (ii) the *revenue ratio*, $\mathcal{R}^r = \frac{R}{R^*}$, where R is the revenue obtained by G-ERAP or APX-ERAP, and R^* is the revenue obtained by the optimal solution obtained by CPLEX. R^* and R are calculated using Equation (6). Note that, to determine the revenue of the CPLEX solution, we use the pricing rule defined for APX-ERAP.

To investigate the impact of the dynamic provisioning on the performance of the mechanisms, we compare their performance considering two types of MEC systems: one that allows dynamic provisioning, and another one in which the VMs are pre-provisioned. Both systems have the same amount of physical resources. In the system with pre-provisioning, the resources are equally provisioned into four types of VMs. In fact, since we assume the same range of demand for each type of VMs (the number of requests for each type of VMs for each user is drawn from the same distribution), we equally distribute the resources among the four types of VMs.

The mechanisms are implemented in C++ and the experiments are conducted on an Intel 1.6GHz Core i5 system with 8 GB RAM. For solving ERAP-MILP, we use the CPLEX 12 solver provided by IBM ILOG CPLEX optimization studio for academics initiative [43].

6.2 Analysis of Results

In this section, we compare the performance and the scalability of the mechanisms for different types of problem

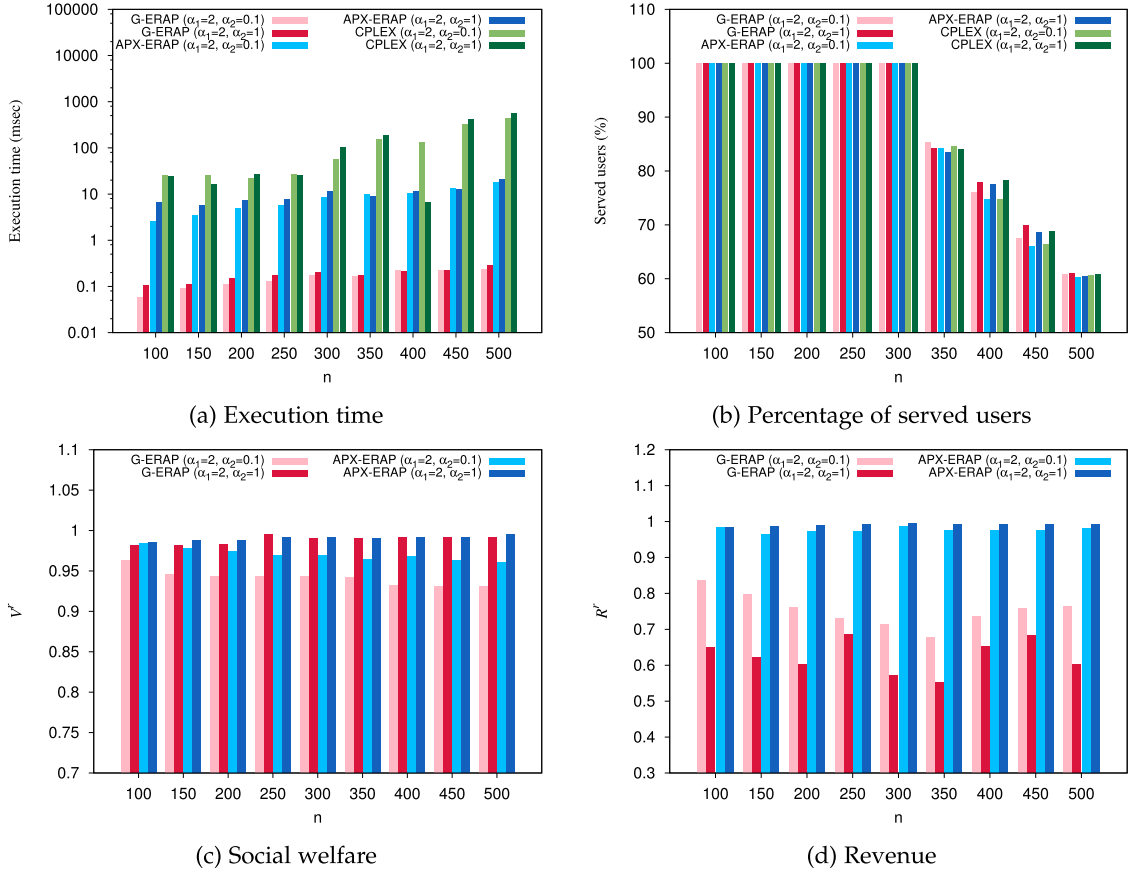


Fig. 1. The effect of the total number of users, n , on (a) the execution time; (b) the percentage of served users; (c) the social welfare; and (d) the revenue (small-size instances).

instances. First, we investigate the performance of the mechanisms for small-size instances by comparing with the CPLEX solution. Then, we investigate the scalability and performance of the mechanisms for large-size instances. Finally, we investigate the performance of the mechanisms on systems with different edge-cloud resource capacity ratio values.

Performance With Respect to the Number of Users (Small-Size Instances). First, we investigate the effects of the number of users on the performance of the mechanisms. In this experiment, we compare the performance of the mechanisms against the optimal solutions obtained by solving ERAP-MILP using CPLEX. For this purpose, we only consider small-size problem instances. We consider a fixed amount of total capacity for each type of physical resources at both edge and cloud levels. We set ρ_{EC} for each type of resource instance to $1/9$, that is, 90 percent of the total capacity of each type of resources is at the cloud level, and 10 percent at the edge level. We perform experiments with two sets of problem instances. In order to investigate the performance of the proposed mechanisms for applications with different latency requirements, we perform our experiments using two sets of values for (α_1, α_2) . For the first set, we consider the preference factors $\alpha_1 = 2$, and $\alpha_2 = 1$, while for the second set, we consider $\alpha_1 = 2, \alpha_2 = 0.1$. We consider different number of users, n , ranging from 100 to 500. Fig. 1a shows the execution time of G-ERAP, APX-ERAP, and CPLEX for the two sets of problem instances. In all cases, the execution time of G-ERAP is less than one millisecond, significantly smaller than the time required by CPLEX to obtain the solution. Also, the execution time of APX-ERAP is less than 11

milliseconds. We observe an increase in the execution time of G-ERAP with the increase in the number of users. The reason is that the running time of the mechanisms grows linearly with the number of users. Overall, both G-ERAP and APX-ERAP are not very sensitive to the number of users. In contrast, the execution time of CPLEX is very sensitive to the number of users and it increases significantly as the number of users in the system increases. Fig. 1b shows the percentage of users served by the provider when employing the solution determined by G-ERAP, APX-ERAP, and CPLEX. For instances with a small number of users, there is no difference between the solution of the mechanisms in terms of the number of served users. With an increase in the number of users we observe that G-ERAP serves more users. The reason is that G-ERAP greedily allocates users that have high bid density. Therefore, it may assign users that have relatively higher bid and smaller request size.

Fig. 1c shows the social welfare ratio for the two sets of problem instances. The results obtained by G-ERAP and APX-ERAP for both sets of problem instances are fairly close to those of CPLEX. Furthermore, the value of social welfare ratio is above 0.93. In most cases, the performance of APX-ERAP is better than that of G-ERAP, specially for the second set of problem instances ($\alpha_1 = 2, \alpha_2 = 0.1$) in which the edge level is more preferred. In fact, both G-ERAP and APX-ERAP are more sensitive to the number of users for the second set of instances. The reason is that when applications have significantly low latency requirements, $\alpha_1 \gg \alpha_2$, any inappropriate allocation has significant effects on the social welfare. Fig. 1d shows the revenue ratios. We observe no significant gap between the

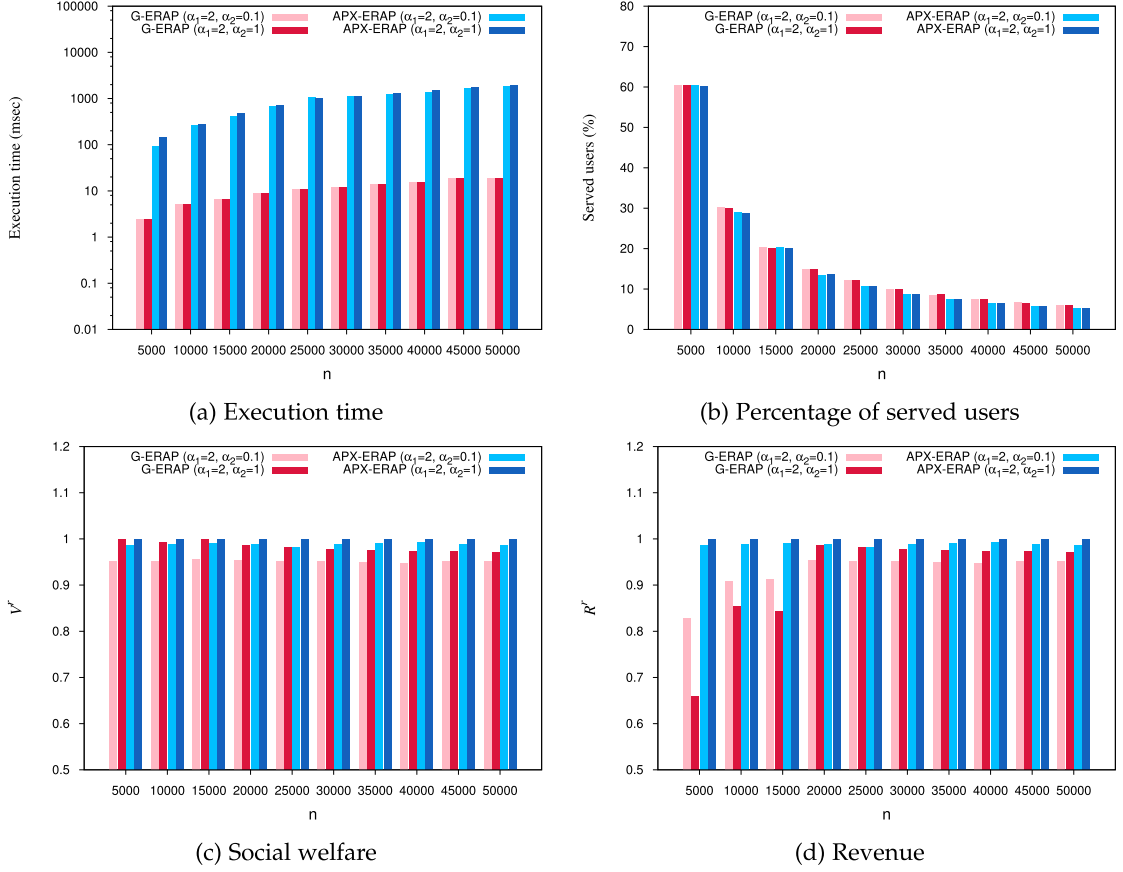


Fig. 2. The effect of the total number of users, n , on: (a) the execution time; (b) the percentage of served users; (c) the social welfare; and (d) the revenue (large-size instances).

performance of APX-ERAP and that of CPLEX. The reason is that APX-ERAP and CPLEX use the same pricing rule. However, G-ERAP obtains lower revenue than the other mechanism. The reason is that G-ERAP considers a lower payment for the allocated users. Furthermore, we observe that for $n \leq 350$, the revenue obtained by G-ERAP decreases by increasing the number of users, while for $n > 350$, the revenue increases by increasing the number of users. The reason is that for $n \leq 350$ the provider has enough capacity to allocate most of the users. Thus, by increasing the number of users, more users are allocated. This results in a higher revenue for CPLEX which is obtained by adding the bids of the winners. However, the revenue of G-ERAP which is obtained based on the bids of the last allocated bidder and the first losing bidder on the cloud and the edge level, does not grow at the same rate. For $n > 350$ when the number of losers increases (see Fig. 1b), the ratio between the highest bidder that loses and the average bids of winners decreases. Thus, the revenue of G-ERAP, which is affected by the bids of losers, gets closer to the revenue of CPLEX which is based on the bids of the winners. Another observation is that the revenue obtained by G-ERAP is closer to that of CPLEX for the case $\alpha_1 \gg \alpha_2$. The reason is that for these settings the low revenue obtained by G-ERAP from the cloud side does not affect the total revenue of the system (due to the small value of α_2).

Performance and Scalability With Respect to the Number of Users (Large-Size Instances). We now evaluate the performance of G-ERAP and APX-ERAP by varying the number of users. We assume a fixed capacity at both edge and cloud level and vary

the number of requests from 5000 to 50000. In this set of experiments, we assume that 90 percent of resources are at the cloud level while only 10 percent of resources are available at the edge level. The values of other parameters are given in Table 3. Since CPLEX is not feasible to be used for solving such large instances, we will not compare the performance of our mechanisms against the performance of the optimal solution obtained by solving ERAP-MILP. Instead, we will compare the performance of our mechanisms against the LP relaxation of ERAP-MILP (which gives the upper bound on the optimal solution). In order to do this, we redefine the *social welfare ratio* as the ratio between the social welfare obtained by G-ERAP or APX-ERAP, and the social welfare obtained by the LP relaxation solution. We also redefine the *revenue ratio*, as the ratio between the revenue obtained by G-ERAP or APX-ERAP, and the revenue obtained by the LP relaxation solution.

Fig. 2a shows the average execution time of both G-ERAP and APX-ERAP, for the two sets of problem instances. We observe that by increasing the number of users, the execution time of both G-ERAP and APX-ERAP increases linearly. For example, the average execution time of G-ERAP for $n = 5000$ is about 2.5 milliseconds and for $n = 50000$, the average execution time is about 18 milliseconds. We observe a similar behavior for APX-ERAP. The average execution time of APX-ERAP for $n = 5000$ is about 120 milliseconds and for $n = 50000$, the average execution time is about 1790 milliseconds. Fig. 2b shows the average percentage of served users for various numbers of users. We observe that by increasing the number of users, a lower percentage of users are served. This is

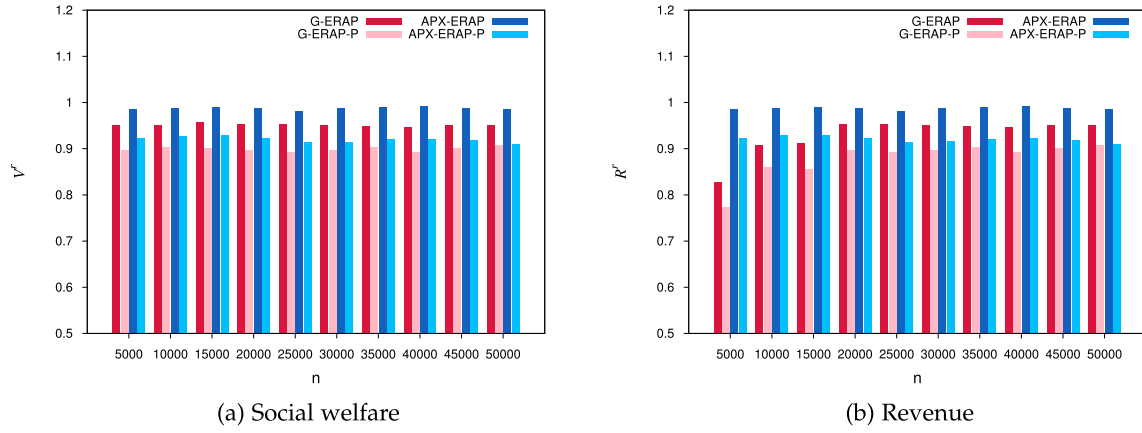


Fig. 3. The effect of the dynamic provisioning on the social welfare and the revenue ($\alpha_1 = 2, \alpha_2 = 1$).

because the total capacity is fixed and the system is not able to serve all the requests. Again, we observe that by increasing the number of users, G-ERAP serves more users. The reason is that G-ERAP may greedily assign users that have relatively higher bids and smaller request sizes. Fig. 2c shows the effect of the number of users on the social welfare for the two sets of problem instances. Both G-ERAP and APX-ERAP are able to maintain a welfare ratio above 0.95. A similar behavior is observed for APX-ERAP when we consider the effects of the number of users on the revenue (Fig. 2d), where the revenue ratio is kept above 0.95. However, G-ERAP obtains a lower revenue compared to APX-ERAP. The reason is that the LP relaxation and APX-ERAP use the same pricing rule to determine the revenue while G-ERAP considers a lower payment for the allocated users. We also observe that by increasing the number of users, the revenue of G-ERAP gets closer to that obtained by the LP relaxation. The reason is that by increasing the number of users, the number of losers increases. Thus, with a high probability, the ratio between the bid of the highest bidder that loses and the average bid of the winners decreases. Therefore, the revenue of G-ERAP, which is affected by the bids of losers, becomes closer to the revenue obtained by the LP relaxation, which is based on the bids of the winners. The experimental results on large-size instances show that G-ERAP can solve fairly large-size problem instances within 20 milliseconds while keeping the social welfare and revenue within an acceptable distance from those obtained by the LP relaxation solution. These results show that G-ERAP can be employed efficiently in real world systems with a large number of users.

Performance With Respect to Dynamic Provisioning. Here, we investigate the effects of the dynamic provisioning on the performance of the system. For this purpose, we consider a system in which the number of VM types at each level is pre-determined (as explained in Section 6.1). In this set of experiments, we consider the same setup as that for large size instances. We vary the number of requests from 5000 to 50000. For more readability, in Fig. 3 we show the performance of the mechanisms in the pre-provisioning (denoted by G-ERAP-P and APX-ERAP-P) and dynamic provisioning cases only for preference factors $\alpha_1 = 2, \alpha_2 = 1$. Fig. 3a shows that the social welfare of the system in the pre-provisioning case is lower than in the case of dynamic provisioning for both G-ERAP and APX-ERAP. Fig. 3b shows a similar behavior of the mechanisms in terms of the revenue. The reason behind

this observation is that in a system with pre-provisioning, some VMs that are statically provisioned may not be used as they are the surplus to the demand, while some other VMs that are more wanted are scarce. Therefore, the resource manager is not able to utilize its resources according to the demand, and the social welfare and the revenue of the system decreases.

Performance With Respect to the Edge-Cloud Resource Capacity Ratio (Large-Size Instances). In this set of experiments, we investigate the effects of the edge-cloud resource capacity ratio, ρ_{EC} on large-size problem instances with 10000 users. We draw the total capacity of each type of physical resources from the distributions given in Table 3 and keep it fixed. We allocate the capacity to each level according to the following edge-cloud resource capacity ratios, $\rho_{EC} = \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \frac{1}{64}, \frac{1}{128}, \frac{1}{256}, \frac{1}{512}$, and $\frac{1}{1024}$. These ratios will allow us to investigate the performance of G-ERAP and APX-ERAP on systems with plenty of available resources at the edge level ($\rho_{EC} = \frac{1}{2}$), and systems with very few resources at the edge level ($\rho_{EC} = \frac{1}{1024}$). We perform our experiments with two sets of problem instances. In the first set, we consider the preference factors $\alpha_1 = 2$, and $\alpha_2 = 1$, while in the second set, we consider $\alpha_1 = 2, \alpha_2 = 0.1$.

Fig. 4a shows the effects of ρ_{EC} on the average execution time of both G-ERAP and APX-ERAP, for the two sets of problem instances. For all values of ρ_{EC} , the execution time of G-ERAP is less than 10 milliseconds, while the execution time of APX-ERAP is less than 300 millisecond. Also, the execution time of both G-ERAP and APX-ERAP are not significantly sensitive to variations of ρ_{EC} . Fig. 4b shows the effects of ρ_{EC} on the percentage of served users when considering the solution obtained by G-ERAP and APX-ERAP. Similarly to the execution time, we observe that ρ_{EC} does not have significant effects on the number of served users for both G-ERAP and APX-ERAP solutions. Fig. 4c shows the effect of ρ_{EC} on the social welfare for the two sets of problem instances. Both G-ERAP and APX-ERAP are able to maintain a welfare ratio above 0.9 even in the worst cases, where resources are very scarce at the edge level ($\rho_{EC} = \frac{1}{1024}$). A similar behavior is observed for APX-ERAP when we consider the effects of ρ_{EC} on the revenue (Fig. 4d), where the revenue ratio is kept above 0.9 even when the resources at the edge level are very scarce. The reason is that the same pricing rule is used to determine the revenue of the LP solution and that of APX-ERAP. However, G-ERAP obtains a lower revenue than the other algorithms due to considering a lower payment for allocated users. We also observe

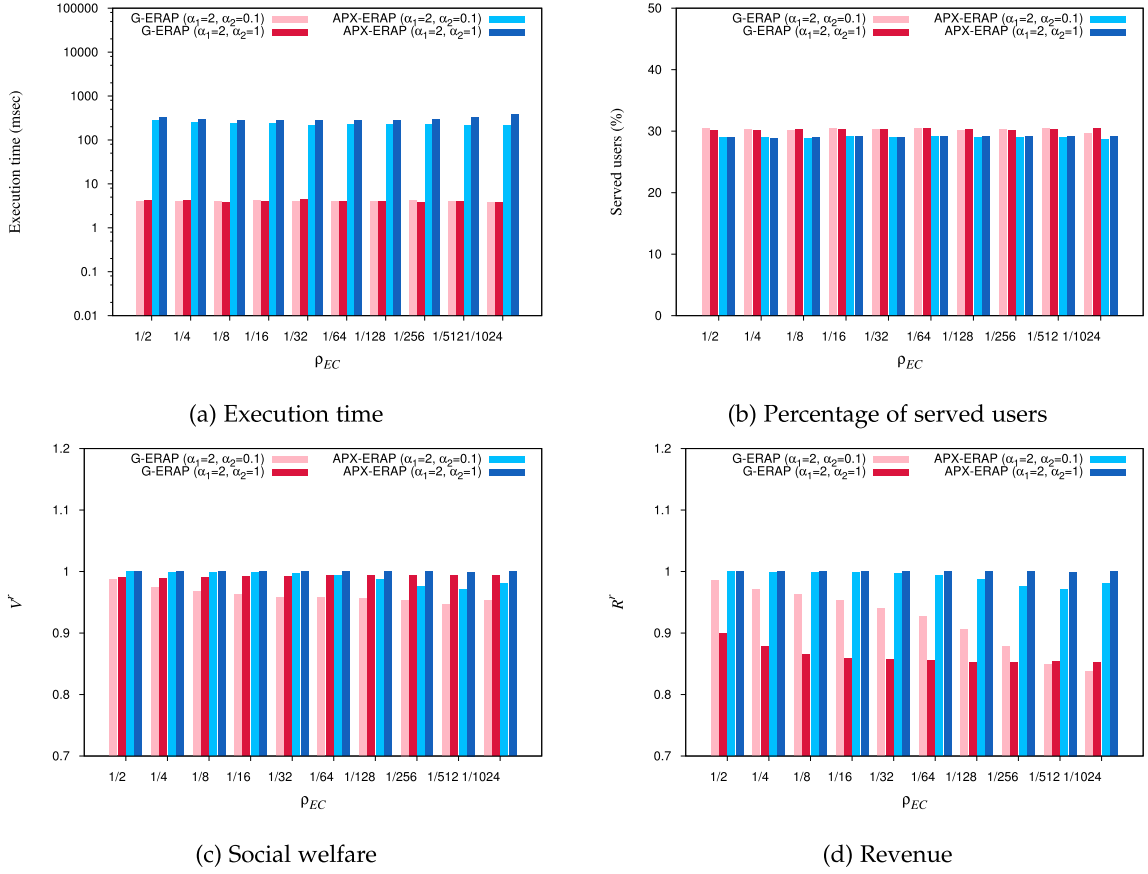


Fig. 4. The effect of the capacity ratio on: (a) the execution time; (b) the percentage of served users; (c) the social welfare; and (d) the revenue (large-size instances).

that when the edge level is more preferred ($\alpha_1 \gg \alpha_2$), the efficiency of the proposed mechanisms is affected by the distribution of the resources among the levels. This is because the valuation for the edge level resources is higher and those resources are scarcer.

Performance Comparison With Non-Combinatorial Auctions. In order to investigate the impact of non-combinatorial bidding on the performance metrics, such as that of the bidding considered by Kiani *et al.* [32], we compare the performance of G-ERAP and APX-ERAP algorithms against that of a class of non-combinatorial auctions (called NC-Auction). In a non-combinatorial auction the user who needs a bundle of VM instances composed of VM instances of various types, submits a separate bid for each individual type of VM instances in the bundle. This is in contrast with combinatorial auctions (G-ERAP and APX-ERAP) in which a user submits a single bid for the whole bundle of VM instances. Kiani *et al.* [32] employed such a type of non-combinatorial auction. In their approach, a user submits a bid for each type of VM and the users' requests are ordered in descending order of their bids and are allocated accordingly to the edge level and the cloud level. Furthermore, their approach requires static provisioning in which VMs are provisioned in advance. We generate the bids of users for the whole bundle of the request as described in Section 6.1. Thus, the valuation of user i for the whole bundle is obtained by multiplying the bid per unit of resource and the total size of the request of user i . We assume that the VMs are complementary goods (i.e., the valuation for two VMs A and B is greater than or equal to the sum of

individual valuations of A and B), and thus in the case of NC-Auction, the valuation per unit of resource of a partially allocated bundle is reduced between 5-15 percent (randomly drawn) compared to the case in which the whole bundle is allocated.

Fig. 5a shows the social welfare ratio obtained by G-ERAP, APX-ERAP, and NC-Auction for the two sets of problem instances with $(\alpha_1 = 2, \alpha_2 = 1)$ and $(\alpha_1 = 2, \alpha_2 = 0.1)$. In all instances, the social welfare ratio obtained by G-ERAP and APX-ERAP is higher than that obtained by NC-Auction. The reason is that NC-Auction allocates VMs based on the individual bids, while G-ERAP and APX-ERAP consider the bid for the whole bundle. Thus, NC-Auction may greedily allocate a VM instance to a user with a relatively high individual bid for the VM, while the bids for the other VMs needed by the user are low. If we consider this as a bundle the sum of the bids in the bundle is low. On the other hand the G-ERAP and APX-ERAP allocate higher value bundles and will obtain a higher social welfare. Furthermore, the valuation of a user for the whole bundle of the request is higher than the sum of the individual bids. Thus, there might be a case in which user i has a higher individual bid for each type of VM compared to user i' ; but the bid for the whole bundle of user i is lower than that of user i' . In this case, NC-Auction greedily allocates user i , while G-ERAP and APX-ERAP allocate resources to user i' and obtains a higher social welfare.

Fig. 5b shows the percentage of the served users for the two sets of problem instances. We observe that when the number

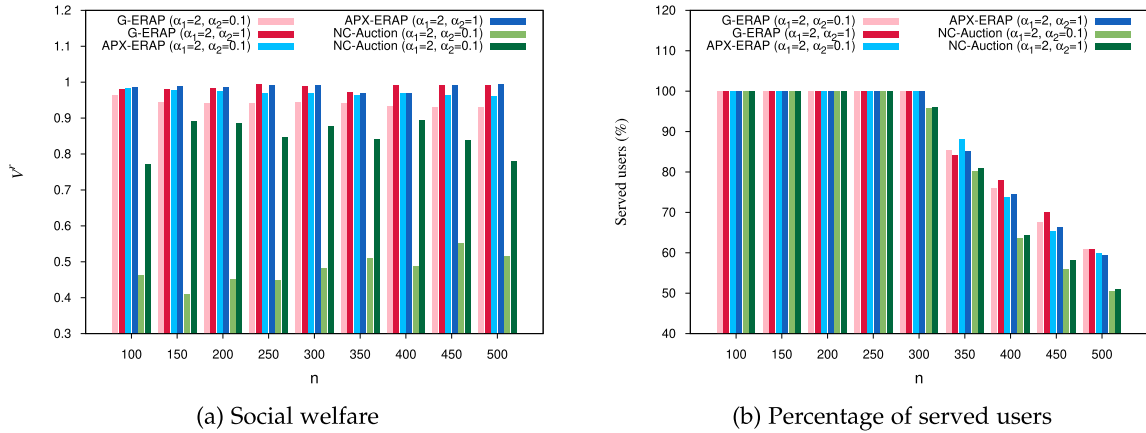


Fig. 5. Comparison with NC-Auction: (a) social welfare; and (b) the percentage of served users; (small-size instances).

of users is relatively low ($n \leq 250$), G-ERAP, APX-ERAP, and NC-Auction serve the same percentage of users. The reason is that there is enough capacity at both edge and cloud levels to serve the requests. As the number of users increases, NC-Auction allocates a lower percentage of users compared to G-ERAP and APX-ERAP. NC-Auction may not be able to utilize resources according to the demand, thus the number of served users decreases. NC-Auction allocates VMs one by one based on their individual bids. Thus, a relatively high percentage of users may receive a partial allocation, while G-ERAP and APX-ERAP only allocate whole bundles.

Our experimental results showed that for small problem instances, both G-ERAP and APX-ERAP yield solutions close to those obtained by the CPLEX optimal solution. Compared to APX-ERAP, G-ERAP has a very small execution time. However, in systems where users have high preference for the edge level ($\alpha_1 \gg \alpha_2$), APX-ERAP is more efficient than G-ERAP in terms of social welfare and revenue. Overall, the low execution time and the acceptable distance from the optimal solution make G-ERAP mechanism suitable for edge computing systems with a large number of users.

7 CONCLUSION

Monetization of services is one of the grand challenges in edge computing systems. In this paper, we proposed two resource allocation and pricing mechanisms in edge computing systems, where users have heterogeneous requests and compete for high quality services. We proved the properties of the proposed mechanisms and evaluated their efficiency by performing an extensive experimental analysis. For small-size instances, we compared the solutions obtained by the proposed mechanisms with the optimal solutions obtained by the CPLEX solver with respect to execution time, percentage of served users, social welfare and revenue. For large-size instances, we compared the performance of the two proposed mechanisms with respect to the same metrics used for the analysis on small-size instances. The experimental results showed that the resource allocation obtained by the proposed mechanisms yield near optimal solutions. In addition to the quality of solutions, the small execution time makes the proposed mechanisms promising for deployment in edge computing systems. As a future research, we plan to design and implement resource allocation and pricing mechanisms

for edge computing systems with different network structures. In this research, we assumed that a user is allocated either at the cloud or edge level, but not at both. One direction for future research is to allow allocation of a user request at both the edge and the cloud level. Another possible direction for future research could be considering a setting with multiple edge providers.

ACKNOWLEDGMENTS

This article is a revised and extended version of [9] presented at The Third ACM/IEEE Symposium on Edge Computing (SEC 2018). This work was supported in part by the US National Science Foundation under Grant IIS-1724227.

REFERENCES

- [1] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless Comm. Mobile Comp.*, vol. 13, no. 18, pp. 1587–1611, Dec. 2013.
- [2] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tut.*, vol. 19, no. 3, pp. 1628–1656, Jul.–Sep. 2017.
- [3] Q. Yuan, H. Zhou, J. Li, Z. Liu, F. Yang, and X. S. Shen, "Toward efficient content delivery for automated driving services: An edge computing solution," *IEEE Netw.*, vol. 32, no. 1, pp. 80–86, Jan./Feb. 2018.
- [4] T. Bahreini, M. Brocanelli, and D. Grosu, "Energy-aware speculative execution in vehicular edge computing systems," in *Proc. 2nd Workshop Edge Syst., Anal. Netw.*, 2019, pp. 18–23.
- [5] A. H. Sodhro, Z. Luo, A. K. Sangaiah, and S. W. Baik, "Mobile edge computing based QoS optimization in medical healthcare applications," *Int. J. of Inf. Manag.*, vol. 45, pp. 308–318, Apr. 2019.
- [6] H. Wang, J. Gong, Y. Zhuang, H. Shen, and J. Lach, "Healthedge: Task scheduling for edge computing with health emergency and human behavior consideration in smart homes," in *Proc. IEEE Int. Conf. Big Data*, 2017, pp. 1213–1222.
- [7] NSF, "NSF Workshop Report on Grand Challenges in Edge Computing," 2016. [Online]. Available: <http://iot.eng.wayne.edu/edge/NSF%20Edge%20Workshop%20Report.pdf>
- [8] M. Patel et al., "Mobile-edge computing - Introductory Technical White Paper," Sophia Antipolis, France, Eur. Telecommun. Standards Inst., White Paper, Sep. 2014. [Online]. Available: https://portal.etsi.org/Portals/0/TBpages/MEC/Docs/Mobile-edge_Computing_-_Introductory_Technical_White_Paper_V1%2018-09-14.pdf
- [9] T. Bahreini, H. Badri, and D. Grosu, "An envy-free auction mechanism for resource allocation in edge computing systems," in *Proc. ACM/IEEE Symp. Edge Comput.*, 2018, pp. 313–322.
- [10] H. R. Varian, "Position auctions," *Int. J. Ind. Org.*, vol. 25, no. 6, pp. 1163–1178, Dec. 2007.

- [11] P. Cramton, Y. Shoham, and R. Steinberg, *Combinatorial Auctions*. Cambridge, MA, USA: MIT Press, 2006.
- [12] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 587–597, Mar. 2018.
- [13] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 856–868, Jan. 2019.
- [14] E. Cuervo *et al.*, "Maui: Making smartphones last longer with code offload," in *Proc. 8th ACM Int. Conf. Mobile Syst., Appl. Serv.*, 2010, pp. 49–62.
- [15] Y. Mao, J. Zhang, and K. B. Letaief, "Joint task offloading scheduling and transmit power allocation for mobile-edge computing systems," in *Proc. IEEE Wireless Comm. Netw. Conf.*, 2017, pp. 1–6.
- [16] M.-R. Ra, A. Sheth, L. Mummert, P. Pillai, D. Wetherall, and R. Govindan, "Odessa: Enabling interactive perception applications on mobile devices," in *Proc. 9th ACM Int. Conf. Mobile Syst. Appl. Serv.*, 2011, pp. 43–56.
- [17] C.-F. Liu, M. Bennis, M. Debbah, and H. V. Poor, "Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing," *IEEE Trans. Commun.*, vol. 67, no. 6, pp. 4132–4150, Jun. 2019.
- [18] H. Badri, T. Bahreini, D. Grosu, and K. Yang, "Energy-aware application placement in mobile edge computing: A stochastic optimization approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 4, pp. 909–922, Apr. 2020.
- [19] H. Badri, T. Bahreini, D. Grosu, and K. Yang, "A sample average approximation-based parallel algorithm for application placement in edge computing systems," in *Proc. IEEE Int. Conf. Cloud Eng.*, 2018, pp. 198–203.
- [20] Y. Qian, L. Hu, J. Chen, X. Guan, M. M. Hassan, and A. Alelaiwi, "Privacy-aware service placement for mobile edge computing via federated learning," *Inf. Sci.*, vol. 505, pp. 562–570, Dec. 2019.
- [21] T. Bahreini and D. Grosu, "Efficient placement of multi-component applications in edge computing systems," in *Proc. 2nd ACM/IEEE Symp. Edge Comput.*, 2017, pp. 5:1–5:11.
- [22] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, and L. Tassiulas, "Joint service placement and request routing in multi-cell mobile edge computing networks," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 10–18.
- [23] L. Mashayekhy, M. M. Nejad, and D. Grosu, "Physical machine resource management in clouds: A mechanism design approach," *IEEE Trans. Cloud Comput.*, vol. 3, no. 3, pp. 247–260, Jul.–Sep. 2015.
- [24] S. Zaman and D. Grosu, "Combinatorial auction-based allocation of virtual machine instances in clouds," *J. Parallel Distrib. Comput.*, vol. 73, no. 4, pp. 495–508, Apr. 2013.
- [25] L. Zhang, Z. Li, and C. Wu, "Dynamic resource provisioning in cloud computing: A randomized auction approach," in *Proc. IEEE Conf. Comput. Commun.*, 2014, pp. 433–441.
- [26] Z. Xiong, S. Feng, D. Niyato, P. Wang, and Z. Han, "Edge computing resource management and pricing for mobile blockchain," 2017, *arXiv:1710.01567*.
- [27] Y. Jiao, P. Wang, D. Niyato, and Z. Xiong, "Social welfare maximization auction in edge computing resource allocation for mobile blockchain," in *Proc. IEEE Int. Conf. Comm.*, 2018, pp. 1–6.
- [28] N. C. Luong, Z. Xiong, P. Wang, and D. Niyato, "Optimal auction for edge computing resource management in mobile blockchain networks: A deep learning approach," in *Proc. IEEE Int. Conf. Commun.*, 2018, pp. 1–6.
- [29] L. Li, M. Siew, T. Q. Quek, J. Ren, Z. Chen, and Y. Zhang, "Learning-based priority pricing for job offloading in mobile edge computing," 2019, *arXiv:1905.07749*.
- [30] B. Baek, J. Lee, Y. Peng, and S. Park, "Three dynamic pricing schemes for resource allocation of edge computing for IoT environment," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4292–4303, May 2020.
- [31] Y. Chen, Z. Li, B. Yang, K. Nai, and K. Li, "A stackelberg game approach to multiple resources allocation and pricing in mobile edge computing," *Future Gener. Comput. Syst.*, vol. 108, pp. 273–287, 2020.
- [32] A. Kiani and N. Ansari, "Toward hierarchical mobile edge computing: An auction-based profit maximization approach," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 2082–2091, Dec. 2017.
- [33] Z. Kong, C.-Z. Xu, and M. Guo, "Mechanism design for stochastic virtual resource allocation in non-cooperative cloud systems," in *Proc. IEEE Int. Conf. Cloud Comput.*, 2011, pp. 614–621.
- [34] L. Mashayekhy, M. M. Nejad, and D. Grosu, "A PTAS mechanism for provisioning and allocation of heterogeneous cloud resources," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 9, pp. 2386–2399, Sep. 2015.
- [35] Y. Zhang, M. Li, K. Bai, M. Yu, and W. Zang, "Incentive compatible moving target defense against VM-colocation attacks in clouds," in *Proc. IFIP Int. Inf. Secur. Conf.*, 2012, pp. 388–399.
- [36] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, *Algorithmic Game Theory*. New York, NY, USA: Cambridge Univ. Press, 2007.
- [37] B. Edelman, M. Ostrovsky, and M. Schwarz, "Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords," *Amer. Econ. Rev.*, vol. 97, no. 1, pp. 242–259, Mar. 2007.
- [38] T. Roughgarden, "Algorithmic game theory," *Commun. ACM*, vol. 53, no. 7, pp. 78–86, 2010.
- [39] M. R. Garey and D. S. Johnson, *Computers and Intractability*. San Francisco, CA, USA: Freeman, 1979.
- [40] C. Chekuri and S. Khanna, "A polynomial time approximation scheme for the multiple knapsack problem," *SIAM J. Comput.*, vol. 35, no. 3, pp. 713–728, 2005.
- [41] A. Caprara, H. Kellerer, U. Pferschy, and D. Pisinger, "Approximation algorithms for knapsack problems with cardinality constraints," *Eur. J. Oper. Res.*, vol. 123, no. 2, pp. 333–345, Jun. 2000.
- [42] L. Khachiyan, "A polynomial algorithm for linear programming," *Doklady Akademii Nauk SSSR*, vol. 244, no. 5, pp. 1093–1096, 1979.
- [43] IBM ILOG CPLEX V12.1 User's Manual. 2009. [Online]. Available: <ftp://ftp.software.ibm.com/software/websphere/ilog/docs/>



Tayebah Bahreini (Student Member, IEEE) received the BSc degree in computer science from the University of Isfahan, Iran, in 2010 and the MSc degree in computer engineering from Shahed University, Iran, in 2014. She is currently working toward the PhD degree with the Department of Computer Science, Wayne State University. Her research interests include distributed systems, approximation algorithms, parallel computing, and game theory. She was the recipient of 2019 National Center for Women and Information Technology Collegiate National Award. She was selected as a 2019 Top Ten Women in Edge by the Edge Computing World Organization for her contribution to research in edge computing. She is a student member of the ACM.



Hossein Badri (Student Member, IEEE) received the BSc degree in industrial engineering from the Sharif University of Technology and the MSc degree in industrial engineering from Shahed University. He is currently working toward the MSc degree in computer science and the PhD degree in industrial and systems engineering with Wayne State University. His research interests include parallel approximation algorithms, stochastic optimization, game theory, data-driven optimization in distributed systems, logistics and transportation systems, and health-care systems.



Daniel Grosu (Senior Member, IEEE) received the Diploma degree in engineering (automatic control and industrial informatics) from the Technical University of Iași, Romania, in 1994, and the MSc and PhD degrees in computer science from the University of Texas at San Antonio in 2002 and 2003, respectively. He is currently an associate professor with the Department of Computer Science, Wayne State University, Detroit. He has authored or coauthored more than 100 peer-reviewed papers in the areas of his research interests. His research interests include parallel and distributed computing, approximation algorithms, and topics at the border of computer science, game theory, and economics. He was on the program and steering committees of several international meetings in parallel and distributed computing, including the ICDCS, CLOUD, ICPP, and NetEcon. He is a senior member of the ACM, and IEEE Computer Society.

► For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.