2018 Third ACM/IEEE Symposium on Edge Computing

An Envy-Free Auction Mechanism for Resource Allocation in Edge Computing Systems

Tayebeh Bahreini
Department of Computer Science
Wayne State University
Email: tayebeh.bahreini@wayne.edu

Hossein Badri
Department of Computer Science
Wayne State University
Email: hossein.badri@wayne.edu

Daniel Grosu
Department of Computer Science
Wayne State University
Email: dgrosu@wayne.edu

***Abstract*—One of the major challenges in Mobile Edge Computing (MEC) systems is to decide how to allocate and price edge/cloud resources so that a given system's objective, such as revenue or social welfare, is optimized. One promising approach is to allocate these resources based on auction models, in which users place bids for using a certain amount of resources. In this paper, we address the problem of resource allocation and pricing in a two-level edge computing system. We consider a system in which servers with different capacities are located in the cloud or at the edge of the network. Mobile users compete for these resources and have heterogeneous demands. We design an auction-based mechanism that allocates and prices edge/cloud resources. The proposed mechanism is novel in the sense that it handles the allocation of resources available at the two-levels of the system by combining features from both position and combinatorial auctions. We show that the proposed mechanism is individually-rational and produces envy-free allocations. The first property guarantees that users are willing to participate in the mechanism, while the second guarantees that when the auction is finished, no user would be happier with the outcome of another user. We evaluate the performance of the proposed mechanism by performing extensive experiments. The experimental results show that the proposed mechanism is scalable and obtains efficient solutions.**

***Index Terms*—edge computing, resource allocation, pricing**

I. Introduction

Despite the tremendous developments in the technology of mobile devices, they still suffer from limitations with respect to storage, computation power, and more importantly, battery life, which is the most crucial for mobile users [14]. These drawbacks of mobile devices could be alleviated by running applications remotely on a static infrastructure that does not suffer from these limitations. Mobile Cloud Computing (MCC) has been introduced to allow applications to perform their computation on the cloud servers. MCC is a centralized paradigm for mobile applications in which data, computation, and storage are migrated from mobile devices to the resource-rich and powerful servers located in the clouds. In fact, MCC is an extension of cloud computing which considers the mobility of users and the ad-hoc structure of mobile devices [9]. MCC brings about many benefits for mobile users. It extends the battery life of mobile devices, reduces the execution time of applications, and improves the data storage capacity and processing power. However, in MCC, centralized data centers that are used by cloud services are usually far from end-users, and therefore, communication between mobile devices and datacenters involves many network hops and results in high latencies. Thus, MCC may not be efficient for some applications that need a very quick response or require a large amount of data transmission [23]. In recent years, several paradigms have been developed to resolve the inefficiencies of MCC by sending a portion of data/computation to the edge of the network instead of running on cloud data centers [24].

Mobile Edge Computing (MEC) has been recently introduced to enable processing data at the edge of the network, where edge can be any computing resource of the network [25]. MEC has received significant attention from researchers who have mainly focused on designing offloading strategies in order to minimize the energy consumption, average delay, and total cost. Compared to cloud data centers, edge systems have much more limited resources leading to increased competition among users who desire to acquire resources within their proximity. Therefore efficiency of resource allocation, incentives, and monetization are major challenges in MEC systems. A recent NSF report on grand challenges in edge computing [19], identified incentives and monetization (i.e., developing incentive schemes for mobile users and edge providers) as one of the five main challenges in the development of edge computing systems.

In this paper, we address this challenge by developing an auction-based mechanism for resource allocation and pricing in edge computing systems. 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 [20]. 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.

Our proposed resource allocation mechanism is novel since it combines features from both position [26] and combinatorial auctions [5] 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) [8] and prices that lead to close to optimal social welfare for the users, where the *social welfare* is the sum of the users' valuations. In addition, the mechanism requires very small execution time even for very large problem instances with hundreds of users. We evaluate the performance of the proposed mechanism through an extensive experimental analysis.

The rest of the paper is organized as follows. In Section II, we review the recent work on pricing and allocation in edge computing systems. In Section III, we formally define and formulate the problem of allocation and pricing. In Section IV, we present the proposed allocation and pricing mechanism and prove its properties. In Section V, we describe the experimental setup and discuss the experimental results. In Section VI, we conclude the paper and suggest possible directions for future work.

II. Related work

Researchers approached the resource allocation problem in mobile computing systems from different perspectives. Several efforts have been devoted to optimizing computation offloading and designing intelligent decision making procedures for task migration that considers the dynamism of the MEC environment, characteristics of applications and mobile devices, network conditions, and user's preferences [4], [6], [10], [21]. Service placement is also an important problem in edge computing which is associated with deciding where the services should be run. In the placement problem, users' services can be run either on the core cloud, or on the edge of the network [2], [3], [7].

Considerable efforts have been devoted to the design of mechanisms for resource allocation and pricing in cloud computing. Several researchers approached the problem of designing market-based resource allocation and pricing from different perspectives [17], [28], [30], [31]. When it comes to MCC systems, there are only few papers that addressed the resource allocation problem by considering competitive environments. Zhang et al. [29] proposed a mechanism based on a multi-round sealed bid sequential combinatorial auction to allocate limited wireless and computational resources from multiple service providers to multiple users. The resource allocation mechanisms designed for cloud and MCC systems are not directly applicable to the edge computing settings where resources are distributed over multiple levels and users have different valuations for the services provided from different levels. We directly address these issues in the design of our proposed mechanism by incorporating features that are characteristic to position auctions [26].

Developing resource allocation mechanisms for mobile blockchain applications has been an important challenge addressed by several researchers. Xiong et al. [27] proposed a pricing mechanism for running the mining process 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. [11] developed a truthful mechanism that maximizes the social welfare. Users participating in a *truthful mechanism* do not have incentives to report valuations that are different from their true valuations for their requests. 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. [15] 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.

In this paper, we develop an efficient pricing and resource allocation mechanism for a two-level edge computing system with multiple resources and heterogeneous demands. To the best of our knowledge, the closest work to ours is by Kiani et al. [12] 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 Amazons Elastic Compute Cloud (EC2) spot pricing. In their algorithm, the price of each type of VMs in each access point is determined based on the amount of demands and available resources at the access point. However, in their setting, requests are placed for only one VM instance of a single type, that is, if a user needs to request a bundle of VM instances of different types, the user has to request each of the VM instances separately because requests for bundles of VM instances are not allowed. Furthermore, if a user needs to request multiple VM instances of the same type, the user has to submit multiple requests (bids), one for each instance. Submitting multiple bids for individual VM instances involves the risk of ending up obtaining only a subset of the requested set of VM instances. As an example, consider a computation intensive application for which a user needs a specific bundle of VM instances in order to execute and complete it by a given deadline. Satisfying only a portion of the request would lead to missing the deadline and to higher costs.

However, our mechanism allows requests for bundles of VM instances and for multiple VM instances of the same type. Our algorithm determines the price of VM instances based on the bids of the users, while in [12], the price is determined based on the amount of demands and the available resources. In addition, our algorithm produces envy-free allocations.

Several researchers have applied some of the traditional auction models for solving resource allocation problems in computing systems [13], [16], [32]. The Vickrey-Clarke-Groves (VCG) auction has been one of the most popular truthful auctions [18]. 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 unfeasible 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 system 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. Our proposed mechanism is unique in the sense that it handles the allocation of resources available at the two-levels of the system by combining features from both position and combinatorial auctions. The proposed mechanism is individually-rational and produces envy-free allocations which are two important and desirable properties of a mechanism. The first property guarantees that users are willing to participate in the mechanism, while the second guarantees that when the auction is finished, no user would be happier with the outcome of another user.

III. Edge Resource allocation and pricing problem

We consider an edge computing provider that offers a limited set of VM (Virtual Machine) instances at the edge and cloud level. Edge resources are closer to users, therefore, users prefer to run their applications on the edge servers to obtain better performance. However, the capacity of edge resources is more restricted than that of the cloud servers. Therefore, there is competition between users to obtain VM instances on the edge servers. In our setting, VM instances are sold by the provider based on an auction model in which the price of resources is not pre-defined but determined by employing an auction mechanism.

We consider that 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. The two levels offer their resources to users as m types of VM instances, where each type of VM instance is characterized by three types of resources: vCPU ($t = 1$), memory ($t = 2$), and storage ($t = 3$). A VM instance of type k provides q_{kt} units of resource of type t, where $k = 1, \ldots, m$ and $t = 1, 2, 3$. The total number of VM instances of type k available at level l is denoted by C_{kl}. As an example, a 'small' VM instance of type $k = 1$ may consist of 1 vCPU, 2GB of memory, and 20GB of storage. That is, the instance of type $k = 1$ is characterized by $q_{11} = 1$, $q_{12} = 2$, and $q_{13} = 20$.

User i requests a bundle of VM instances and submits a bid for the bundle. The request of user i is denoted by $\theta_i = (b_i, r_i) = (b_{i1}, \ldots, b_{im}; r_{i1}, \ldots, r_{im})$, where b_{ik} is the bid for a VM instance of type k, and r_{ik} is the number of VM instances of type k requested by user i. Users' requests are submitted to a mechanism that determines the allocation of VM instances to users and the price they have to pay for their allocations.

Our proposed mechanism (presented in Section 4) can be viewed as a hybrid of a multi-unit combinatorial auction [5] and a position auction [26]. It is a position auction because, (i) the auctioneer never allocates the bundle request of a user to more than one level; (ii) users have different preferences for each level (position) of resources, and resources at the edge are more preferred. To characterize the user's preference for the edge and cloud level, we define α_j as the *preference factor* for level l, which is determined based on the average distance between users and resources at level l (obviously, $\alpha_1 > \alpha_2$). 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 single minded. A user is single-minded if he/she 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 an allocation mechanism, where $a_i^l = (a_{i1}^l, \ldots, a_{im}^l)$ is the allocation of resources at level l for user i, and a_{ik}^l is the number of VM instances of type k allocated to this user on level l. Thus, the valuation function for user i is as follows,

$$v_i(a_i) = \begin{cases} \alpha_1 \sum_{k=1}^{m} b_{ik} r_{ik} & \text{if } r_i \preceq a_i^1 \text{ and } a_i^2 = 0 \\ \alpha_2 \sum_{k=1}^{m} b_{ik} r_{ik} & \text{if } r_i \preceq a_i^2 \text{ and } a_i^1 = 0 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where $r_i \preceq a_i^l$, if $r_{ik} \leq a_{ik}^l$ for $i = 1, \ldots n$, $k = 1, \ldots, m$, and $l = 1, 2$. The users' valuation for each type of VM instance is additive, that is, the valuation of a user for a set of VM instances of the same type is proportional to the number of VM instances in the set. The total valuation of a user is the sum of the valuations for each of the types of requested VM instances. Since $\alpha_1 > \alpha_2$, users prefer the allocation at the edge level instead of that at the cloud level.

A mechanism determines the prices for VM instance bundles based on the users' declarations and the available resources in the system. It decides a base price for resources allocated at the edge level, denoted by π_1, and a base price for resources allocated at the cloud level, denoted by π_2. Based on these prices, the price of a unit of resource of type t on level l is defined as $\pi_l \cdot w_t$, where w_t is the weight of the resource of type t. Here, w_t is used to differentiate the values of a unit of different types of resources. Therefore, the price of a VM instance of type k at level l is $\sum_{t=1}^{3} \pi_l w_t q_{kt}$. In other words, the price per unit of each VM type is the same for winners of the same level, but winners of different levels should pay differently for the same type of VM instance.

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. Now, we formulate the *Edge Resource Allocation and Pricing* (ERAP) problem. In this problem formulation, the cloud/edge provider decides how to allocate VM instances to users such that the *welfare* is maximized, where the welfare is the sum of users' valuations,

$$V = \sum_{l=1}^{2} \sum_{i=1}^{n} \alpha_l x_{il} (\sum_{k=1}^{m} b_{ik} r_{ik}) \tag{2}$$

Considering this objective, the mixed-integer linear program (MILP) formulation of ERAP is as follows:

ERAP-MILP:

$$\text{maximize} \quad \sum_{l=1}^{2} \sum_{i=1}^{n} \sum_{k=1}^{m} \alpha_l b_{ik} r_{ik} x_{il} \tag{3}$$

subject to:

$$\sum_{i=1}^{n} r_{ik} y_{ilk} \le C_{kl} \quad \forall l, \forall k \tag{4}$$

$$m x_{il} \le \sum_{k=1}^{m} y_{ikl} \quad \forall i, \forall l \tag{5}$$

$$\sum_{l=1}^{2} y_{ilk} \le 1 \quad \forall i, \forall k \tag{6}$$

$$x_{il} \in \{0, 1\} \quad \forall i, \forall l \tag{7}$$

$$y_{ikl} \in \{0, 1\} \quad \forall i, \forall l, \forall k \tag{8}$$

where x_{il} is a binary decision variable associated with the allocation of the request of user i to level l. Variable x_{il} is 1 if the bundle requested by user i is allocated to level l; and 0, otherwise. The objective function (3) is to maximize the welfare. Constraints (4) ensure that the total allocated requests of each type of VM instances on each level does not exceed the available capacity for that level. Here, y_{ikl} is a binary variable equal to 1 if the request of user i for type k VM instances is allocated at level l, and 0, otherwise. Constraints (5) ensure that $x_{il} = 1$ only if the requested bundle of user i is completely allocated to level l. Constraints (6) guarantee that the request of user i for type k is not allocated to more than one level. Finally, constraints (7) and (8) guarantee the integrality of the decision variables. ERAP-MILP is used in the experimental results section to obtain the optimal allocation for the ERAP problem and compare it with that obtained by the proposed mechanism.

The payment that user i has to pay to the provider is defined as,

$$p_i = \sum_{l=1}^{2} \sum_{k=1}^{m} \sum_{t=1}^{3} \pi_l w_t q_{kt} r_{ik} x_{il} \tag{9}$$

If a user is not allocated the requested bundle or a superset of it, then she pays 0. Table I shows the notation used in the paper.

IV. Allocation and pricing mechanism

Since it is unlikely to find polynomial time optimal mechanisms for ERAP, we give up on optimality and rely on polynomial time heuristics. In this section, we design a greedy mechanism for ERAP and investigate its properties.

TABLE I: Notation

Notation	Description
n	Number of users.
m	Types of VM instances.
α_l	Preference factor for level l.
b_{ik}	Bid of user i for a VM instance of type k.
r_{ik}	Number of VM instances of type k requested by user i.
C_{kl}	Number of VM instances of type k available at level l.
q_{kt}	Amount of resource of type t for a VM instance of type k.
B_i	Average bid of user i.
π_l	Base price for level l resources.
p_i	Payment of user i.
w_t	Weight of resource of type t.
x_{il}	Decision variable equal to 1 if the request of user i is allocated at level l, and 0, otherwise
y_{ikl}	Decision variable equal to 1 if the request of user i for type k VM instances is allocated at level l, and 0, otherwise

A. *G-ERAP mechanism*

The proposed mechanism, called G-ERAP (Greedy Edge Resource Allocation and Pricing), is given in Algorithm 1. 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. The input to G-ERAP consists of the vector of requests (θ_i) from users, and the vector of VM capacities (C). G-ERAP determines how these resources are assigned to users. The output of the mechanism consists of the social welfare V, the price for each unit of resources on the first level and the second level (π_1, π_2), and the allocation matrix X, where $X = [x_{il}]$, $i = 1, \ldots, n$, and $l = 1, 2$. First, the mechanism determines the average bid per unit of resource for each user (lines 6-7). The average bid of user i is defined as follows,

$$B_i = \frac{\sum_{k=1}^{m} b_{ik} r_{ik}}{\sum_{k=1}^{m} \sum_{t=1}^{3} w_t q_{kt} r_{ik}} \tag{10}$$

Then, it sorts the users in non-increasing order of their average bids (line 8), and allocates VM instances to users starting from the first level (i.e., the edge level), accordingly. For the current user, it checks if there are enough resources at the current level (lines 13-16). If there are, it assigns the requested VM instances to the user, and updates the social welfare and the capacity (lines 17-20). If there are not enough resources to allocate the requested bundle at the first level, it increases the index of the level by one (i.e., it starts allocating VM instances on the second level), and stores the index of the user as the first allocated user in the second level, user denoted by u (lines 24-28). In order to guarantee envy-freeness and individual rationality, G-ERAP stops once it reaches to a user for which there are not enough resources to satisfy the requested bundle in the second level.

Next, G-ERAP determines the base payments for each unit of resource on the first level and the second level (lines 29-33). Let us assume that 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

Algorithm 1 G-ERAP Mechanism

```
Input: Vector of requests; $\theta_i = (b_{i1}, \dots, b_{im}; r_{i1}, \dots, r_{im})$
Input: Vector of VMs' capacities at each level;
        $C = \{C_{11}, \dots, C_{1m}; C_{21}, \dots, C_{2m}\}$
 1: $V \leftarrow 0$
 2: $X \leftarrow 0$
 3: for $k = 1, \dots, m$ do
 4:     $\tilde{C}_{1k} \leftarrow C_{1k}$
 5:     $\tilde{C}_{2k} \leftarrow C_{2k}$
 6: for $i = 1, \dots, n$ do
 7:     $B_i \leftarrow \frac{\sum_{k=1}^{m} b_{ik} r_{ik}}{\sum_{k=1}^{m} \sum_{t=1}^{3} w_t q_{kt} r_{ik}}$
 8: Sort users in non-increasing order of their $B_i$
 9: $l \leftarrow 1$
10: $i \leftarrow 1$
11: while $i \le n$ do
12:     $available \leftarrow true$
13:     for $k = 1, \dots, m$ do
14:         if $\tilde{C}_{lk} < r_{ik}$ then
15:             $available \leftarrow false$
16:             break
17:     if $available$ then
18:         for $k = 1, \dots, m$ do
19:             $\tilde{C}_{lk} \leftarrow \tilde{C}_{lk} - r_{ik}$
20:             $V \leftarrow V + \alpha_l b_{ik} r_{ik}$
21:         $x_{il} \leftarrow 1$
22:         $i \leftarrow i + 1$
23:     else
24:         if $l = 1$ then
25:             $u \leftarrow i - 1$
26:             $l \leftarrow l + 1$
27:         else
28:             break
29: if $i < n$ then
30:     $\pi_2 \leftarrow \alpha_2 B_i$
31: else
32:     $\pi_2 \leftarrow \alpha_2 (B_i - \epsilon)$
33: $\pi_1 \leftarrow \pi_2 + \frac{(\alpha_1 - \alpha_2)}{2} (B_u + B_{u+1})$
Output: $X = \{x_{11}, \dots, x_{n1}; x_{12}, \dots, x_{n2}\}$
Output: $V$
Output: $(\pi_1, \pi_2)$
```

level. The payments for each unit of resources on the first and second level are determined as follows,

$$\pi_1 = \alpha_2 B^* + \frac{(\alpha_1 - \alpha_2)}{2}(B_u + B_{u+1}) \tag{11}$$

$$\pi_2 = \alpha_2 B^* \tag{12}$$

where

$$B^* = \begin{cases} B_{u'+1} & \text{if } u' < n \\ B_{u'} - \epsilon & \text{otherwise} \end{cases} \tag{13}$$

In this formulation, 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 of the first loser 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 little less than the weighted bid of the last allocated user (i.e., $B_{u'} - \epsilon$, where ϵ is a very small positive real number).

B. Properties of the proposed mechanism

The proposed mechanism combines features from both position [26] and combinatorial auctions [5] and is not truthful. Here, we do not target truthfulness for our mechanism but instead, we guarantee that it produces envy-free allocations. This allows us to design a computationally efficient mechanism suitable for selling resources in two-level edge computing systems, where the edge resources are at the first level and the cloud resources at the second. A famous example of a non-truthful position auction mechanism used in practice is the generalized second price auction employed by Google to sell online advertising [8].

In the following we show that our mechanism is individually-rational and produces envy-free allocations which are two important and desirable properties of a mechanism [18]. The first property guarantees that users are willing to participate in the mechanism, while the second guarantees that when the auction is finished, 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, $v_i - p_i \ge 0$ (i.e., a user reporting her true valuation for the bundle never incurs a loss).*

Lemma 1. *The base price at the edge level π_1, satisfies $\pi_1 \le \alpha_1 B_u$, where u is the last user allocated at the edge level ($l = 1$).*

Proof. The base price for the edge level (level 1) resources is given by:

$$\begin{aligned}\pi_1 &= \alpha_2 B^* + \frac{(\alpha_1 - \alpha_2)}{2}(B_u + B_{u+1}) \\ &\le \alpha_2 B^* + \frac{(\alpha_1 - \alpha_2)}{2} 2B_u \le \alpha_2 B^* + (\alpha_1 - \alpha_2) B_u \\ &\le \alpha_1 B_u + \alpha_2 (B^* - B_u)\end{aligned}$$

Because user u is the last user allocated at the edge level, $B^* \le B_u$, and therefore, $\pi_1 \le \alpha_1 B_u$. □

Theorem 1. *G-ERAP is individually-rational.*

Proof. To prove this, 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 h:

Case I. User h is allocated to the edge level ($B_h \ge B_u$).

$$\begin{aligned}v_h - p_h &= \alpha_1 B_h \sum_{k=1}^{m} \sum_{t=1}^{3} w_t q_{kt} r_{hk} - \pi_1 \sum_{k=1}^{m} \sum_{t=1}^{3} w_t q_{kt} r_{hk} \\ &\ge \alpha_1 B_u \sum_{k=1}^{m} \sum_{t=1}^{3} w_t q_{kt} r_{hk} - \pi_1 \sum_{k=1}^{m} \sum_{t=1}^{3} w_t q_{kt} r_{hk}\end{aligned}$$

Using the result of Lemma 1, we have, $v_h - p_h \ge 0$.

Case II. User h is allocated to the cloud level ($B_u \ge B_h \ge B^*$).

$$v_h - p_h = \alpha_2 B_h \sum_{k=1}^{m} \sum_{t=1}^{3} w_t q_{kt} r_{hk} - \pi_2 \sum_{k=1}^{m} \sum_{t=1}^{3} w_t q_{kt} r_{hk}$$

$$\geq \alpha_2 B^* \sum_{k=1}^{m}\sum_{t=1}^{3} w_t q_{kt} r_{hk} - \pi_2 \sum_{k=1}^{m}\sum_{t=1}^{3} w_t q_{kt} r_{hk}$$

According to Equation (12), $\pi_2 = \alpha_2 B^*$. Thus, $v_h - p_h \geq 0$.

Case III. User h loses the auction ($B^* \geq B_h$). According to Equation (13), $v_h = 0$. Because user h is not allocated any bundle of VMs, $p_h = 0$. Thus, $v_h - p_h = 0$. □

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 2. *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,

$$v_i(a_i) - p_i \geq v_i(a_{i'}) - p_{i'} \tag{14}$$

and,

$$v_{i'}(a_{i'}) - p_{i'} \geq v_{i'}(a_i) - p_i \tag{15}$$

These equations are equivalen to,

$$\alpha_1 \sum_{k=1}^{m} b_{ik} r_{ik} - \pi_1 \sum_{k=1}^{m}\sum_{t=1}^{3} w_t q_{kt} r_{ik} \geq \alpha_2 \sum_{k=1}^{m} b_{ik} r_{ik} - \pi_2 \sum_{k=1}^{m}\sum_{t=1}^{3} w_t q_{kt} r_{ik} \tag{16}$$

and,

$$\alpha_2 \sum_{k=1}^{m} b_{i'k} r_{i'k} - \pi_2 \sum_{k=1}^{m}\sum_{t=1}^{3} w_t q_{kt} r_{i'k} \geq \alpha_1 \sum_{k=1}^{m} b_{i'k} r_{i'k} - \pi_1 \sum_{k=1}^{m}\sum_{t=1}^{3} w_t q_{kt} r_{i'k} \tag{17}$$

According to the definition of B_i, the average bid for a user (Equation (10)), we rewrite Equation (16) as,

$$\alpha_1 B_i \sum_{k=1}^{m}\sum_{t=1}^{3} w_t q_{kt} r_{ik} - \pi_1 \sum_{k=1}^{m}\sum_{t=1}^{3} w_t q_{kt} r_{ik} \geq \alpha_2 B_i \sum_{k=1}^{m}\sum_{t=1}^{3} w_t q_{kt} r_{ik} - \pi_2 \sum_{k=1}^{m}\sum_{t=1}^{3} w_t q_{kt} r_{ik}$$
$$\implies \alpha_1 B_i - \pi_1 \geq \alpha_2 B_i - \pi_2 \tag{18}$$

Similarly, Equation 17 is equivalent to,

$$\alpha_2 B_{i'} \sum_{k=1}^{m}\sum_{t=1}^{3} w_t q_{kt} r_{i'k} - \pi_2 \sum_{k=1}^{m}\sum_{t=1}^{3} w_t q_{kt} r_{i'k} \geq \alpha_1 B_{i'} \sum_{k=1}^{m}\sum_{t=1}^{3} w_t q_{kt} r_{i'k} - \pi_1 \sum_{k=1}^{m}\sum_{t=1}^{3} w_t q_{kt} r_{i'k}$$
$$\implies \alpha_2 B_{i'} - \pi_2 \geq \alpha_1 B_{i'} - \pi_1 \tag{19}$$

Based on equations (18) and (19), we only need to show that,

$$B_{i'}(\alpha_1 - \alpha_2) \leq \pi_1 - \pi_2 \leq B_i(\alpha_1 - \alpha_2) \tag{20}$$

Using the definition of B_u and B^* from Equations (11) and (13) we obtain,

$$B^* \leq B_{i'} \leq B_{u+1} \leq B_u \leq B_i \tag{21}$$

Also, based on Equations (11) and (12),

$$\pi_1 - \pi_2 = \frac{(\alpha_1 - \alpha_2)}{2}(B_u + B_{u+1}) \tag{22}$$

thus,

$$\frac{(\alpha_1 - \alpha_2)}{2}(B_{i'} + B_{i'}) \leq \pi_1 - \pi_2 \leq \frac{(\alpha_1 - \alpha_2)}{2}(B_i + B_i)$$
$$(\alpha_1 - \alpha_2)B_{i'} \leq \pi_1 - \pi_2 \leq (\alpha_1 - \alpha_2)B_i \tag{23}$$

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,

$$v_{i''}(a_{i''}) - p_{i''} \geq v_{i''}(a_i) - p_i \tag{24}$$

This is equivalent to show that,

$$0 \geq \alpha_1 B_{i''} \sum_{k=1}^{m}\sum_{t=1}^{3} w_t q_{kt} r_{i''k} - \pi_1 \sum_{k=1}^{m}\sum_{t=1}^{3} w_t q_{kt} \cdot r_{i''k} \tag{25}$$

which is equivalent to

$$0 \geq \alpha_1 B_{i''} - \pi_1 \tag{26}$$

Also, based on Equation (11),

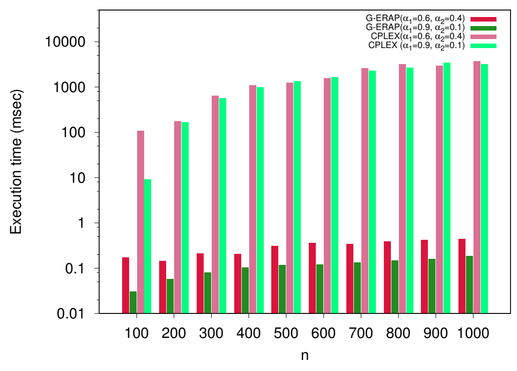
$$\pi_1 \geq \alpha_2 B^* + \frac{(\alpha_1 - \alpha_2)}{2}(2B^*) \implies \pi_1 \geq \alpha_1 B^* \geq \alpha_1 B_{i''} \tag{27}$$

which satisfies Equation (26).

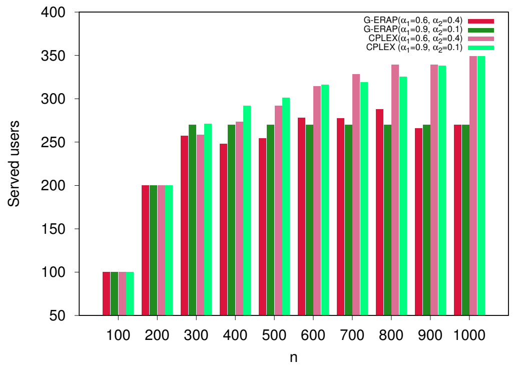
Case III. User i' and user i'' cannot improve their utility by exchanging their bids. The proof is similar to Case II. □

V. Experimental Results

In this section, we perform an extensive experimental analysis to evaluate the performance of the proposed mechanism, G-ERAP. We compare the performance of G-ERAP with that of the optimal solution obtained by solving the ERAP-MILP problem using CPLEX.

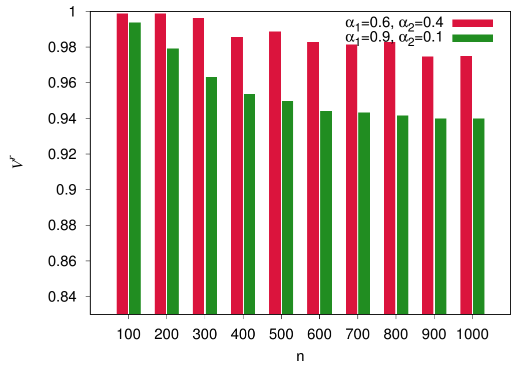


(a) Execution time

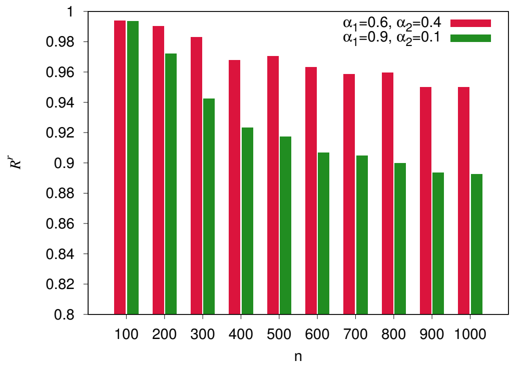


(b) Number of served users

Fig. 1: The effect of the total number of users, n, on the execution time and the number of served users.



(a) Social welfare



(b) Revenue

Fig. 2: The effect of the total number of users, n, on the social welfare and the revenue.

A. Experimental setup

We generate several problem instances with different number of users, demands, and capacities for the edge and cloud. We consider that the provider offers four types of VM instances as shown in Table II. 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. The distribution of the other parameters that are used to generate problem instances in our simulation experiments are shown in Table III. In this table, we denote by $U[a, b]$, the uniform distribution within interval $[a, b]$, and by $N(\mu, \sigma)$, the normal distribution with mean μ and variance σ. To generate the bids b_{ik} of user i, we first draw from $U[1, 10]$, the bid b_{i1} for the first type of VM instances requested by user i. Then, we obtain the bids for the other types of VM instances requested by user i according to the first row in the table. We also use a similar technique to generate r_{ik}, the number of VM instances of each type that are requested by user i. The vector of resource weights w, for the three types of resource is given in the last row of the table.

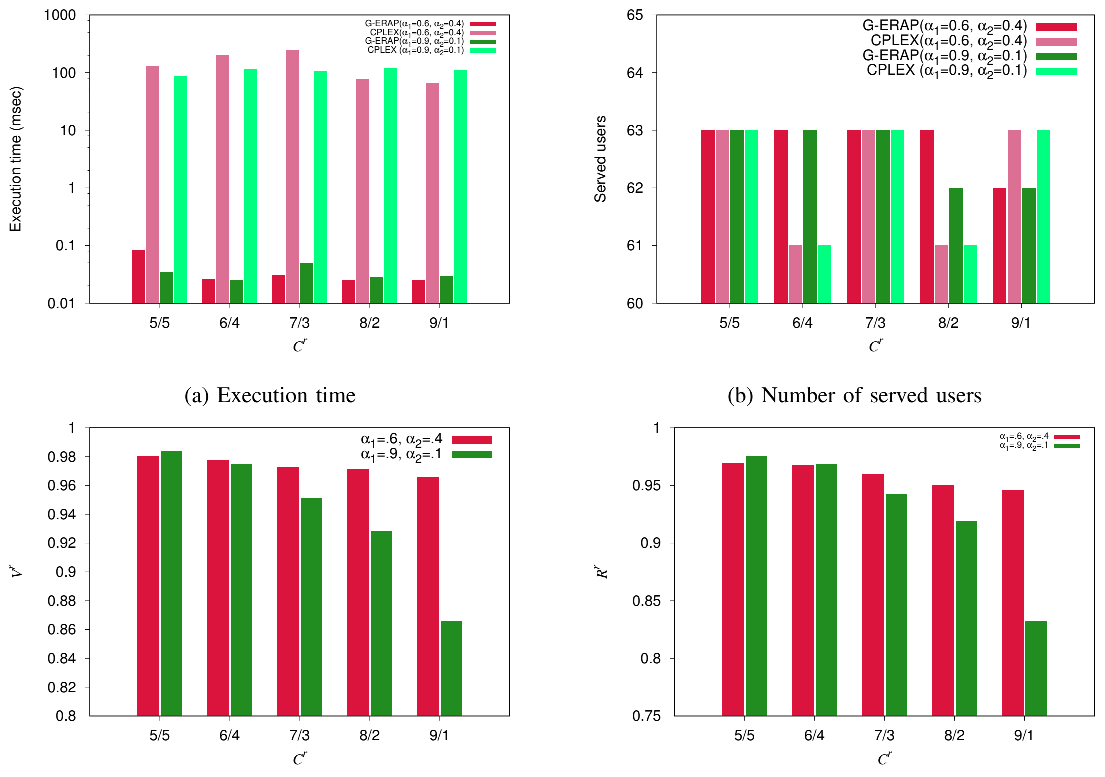
TABLE II: Types of VM instances used in the experiments.

Type	vCPU	Memory(GB)	SSD Storage (GB)
medium	1	3.75 (1 unit)	1×4 (4 units)
Large	2	7.5 (2 units)	1×32 (32 units)
Xlarge	4	15 (4 units)	2×40 (80 units)
2xlarge	8	30 (8 units)	2×80 (160 units)

The G-ERAP mechanism is 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 [1].

B. Analysis of results

First, we investigate the effects of the number of users on the performance of G-ERAP. For this purpose, we consider a fixed amount of total capacity for each type of VM instances at both the edge and cloud level. For each type of VM instance, the total capacity is drawn from the uniform distribution within the interval $[0, 10000]$. We define a parameter, called *cloud-*



(a) Execution time

(b) Number of served users

(c) Social welfare

(d) Revenue

Fig. 3: The effect of the cloud-edge capacity ratio, $\mathcal{C}^r$, on the execution time, number of served users, social welfare, and revenue (n =100).

TABLE III: Simulation parameters

Parameter	Distribution
b_{ik}	$b_{i1} + N(b_{i1}, 0.5b_{i1})$
b_{i1}	$U[0, 10]$
r_{ik}	$N(r_{i1}, 0.7r_{i1})$
r_{i1}	$U[0, 20]$
w	$[8, 4, 1]$

edge capacity ratio, $\mathcal{C}^r = \sum_{k=1}^{m}(\frac{C_{k2}}{C_{k1}})$, which is the sum of the ratios of the capacity for each VM instance type at the cloud and the capacity at the edge level. In our first set of experiments, we set the capacity ratios for each type of VM instances at 7/3, by assigning 70% of the total capacity of each type of VM instance on the cloud side, and 30% on the edge side. Thus, $\mathcal{C}^r = \frac{7}{3}$. We perform experiments with two sets of problem instances. For the first set, we consider the preference factors $\alpha_1 = 0.6$, and $\alpha_2 = 0.4$, while for the second set, we consider $\alpha_1 = 0.9$, $\alpha_2 = 0.1$. We consider different number of users n, ranging from 100 to 1000. We compare the performance of G-ERAP against that of the solutions obtained by solving ERAP-MILP using CPLEX.

Figure 1a shows the execution time of G-ERAP and that of CPLEX for the two sets of problem instances. In all the cases, the execution time of G-ERAP is less than one millisecond, significantly smaller than the time required by CPLEX to obtain the solution. The execution time of G-ERAP is linear in the number of users, while that of CPLEX is exponential.

Figure 1b shows the number of served users (i.e., the number of users who are successfully allocated at either the cloud level or the edge level) when employing the solution determined by G-ERAP and that determined by CPLEX. In most cases, both CPLEX and G-ERAP serve more users for the second set ($\alpha_1 = 0.9, \alpha_2 = 0.1$). For instances with a small number of users, there is no significant difference between the solution of our mechanism and that of CPLEX in terms of the number of served users. When the number of users increases, we observe that the CPLEX solution serves more users than our mechanism. The reason is that CPLEX finds the optimal allocation, while G-ERAP finds an allocation that is not optimal. Despite the fact that G-ERAP is not optimal, our experiments show that the social welfare and the total revenue obtained by G-ERAP are close to those obtained by CPLEX (Figure 2).

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, we define two relative metrics: (i) the *social welfare ratio*, $\mathcal{V}^r = \frac{\mathcal{V}_{\text{G-ERAP}}}{\mathcal{V}_{\text{CPLEX}}}$, where $V_{\text{G-ERAP}}$ is the social welfare obtained by G-ERAP, and V_{CPLEX} is the social welfare obtained by the CPLEX solution; and, (ii) the *revenue ratio*, $\mathcal{R}^r = \frac{R_{\text{G-ERAP}}}{R_{\text{CPLEX}}}$, where $R_{\text{G-ERAP}}$ is the revenue obtained by G-ERAP, and R_{CPLEX} is the revenue obtained by the CPLEX solution. R_{CPLEX} is calculated using Equation 9 and the allocation matrix X determined by CPLEX. Figure 2a shows the social welfare ratio for the two sets of problem instances. The results obtained by G-ERAP for both sets of problem instances are very close to those of CPLEX. The performance of G-ERAP in terms of social welfare is very close to that obtained by the CPLEX solution (with a social welfare ratio above 0.94). However, G-ERAP is more sensitive to the number of users for the second set of instances ($\alpha_1 = 0.9, \alpha_2 = 0.1$). Figure 2b shows the revenue ratios. G-ERAP exhibits similar behavior as that observed in the case of social welfare. There is no significant gap between the performance of G-ERAP and that of CPLEX. Again, G-ERAP seems to be more sensitive to the number of users for the second set of instances ($\alpha_1 = 0.9, \alpha_2 = 0.1$).

In the next set of experiments, we investigate the effects of the cloud-edge capacity ratio, $\mathcal{C}^r$, on the performance of G-ERAP. We draw the total capacity of each VM instance type from the uniform distribution within the interval $[0, 2500]$ and keep it fixed. We allocate the capacity to each level according to the following cloud-edge capacity ratios, $\mathcal{C}^r = \frac{5}{5}, \frac{6}{4}, \frac{7}{3}, \frac{8}{2}$, and $\frac{9}{1}$. These ratios will allow us to investigate the performance of G-ERAP for systems with plenty of available resources at the edge level ($\mathcal{C}^r = \frac{5}{5}$), and systems with very few resources at the edge level ($\mathcal{C}^r = \frac{9}{1}$). We perform our experiments with two sets of problem instances. In the first set, we consider the preference factors $\alpha_1 = 0.6$, and $\alpha_2 = 0.4$, while in the second set, we consider $\alpha_1 = 0.9$, $\alpha_2 = 0.1$. Figure 3a shows the effects of $\mathcal{C}^r$ on the execution time of both G-ERAP and CPLEX, for the two sets of problem instances. The execution time of G-ERAP is less than 0.1 milliseconds for all the four values of $\mathcal{C}^r$. Also, the execution time of both G-ERAP and CPLEX are not significantly sensitive to variations in $\mathcal{C}^r$. Figure 3b shows the effects of $\mathcal{C}^r$ on the number of served users when considering the solution obtained by the CPLEX solver and by G-ERAP. Similar to the execution time, we observe that $\mathcal{C}^r$ does not have significant effects on the number of served users for both G-ERAP and CPLEX solutions. Figure 3c shows the effect of $\mathcal{C}^r$ on the social welfare for the two sets of problem instances. G-ERAP is able to maintain a welfare ratio above 0.86 even for the worst cases, where resources are very scarce at the edge level ($\mathcal{C}^r = \frac{9}{1}$). A similar behavior is observed when we consider the effects of $\mathcal{C}^r$ on the revenue (Figure 3d), where the revenue ratio is kept above 0.83 even when the resources at the edge level are very scarce. We also observe that when the edge level is more preferred ($\alpha_1 \gg \alpha_2$), the efficiency of the proposed algorithm is more significantly 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. Overall, G-ERAP requires very small execution time and yields social welfare and revenue close to those obtained by the CPLEX solution.

VI. Conclusion

We proposed G-ERAP, a greedy resource allocation and pricing mechanism for MEC systems which considers heterogeneous servers and resources of different types. We proved the properties of the proposed mechanism and evaluated its efficiency by performing an extensive experimental analysis on several MEC resource allocation problem instances. The experimental results showed that the proposed mechanism has a very small execution time and yields revenue close to the optimal. The small execution time and the near-optimality of allocations makes the proposed mechanism a very suitable candidate for deployment in current and future edge computing systems. As a future research, we plan to design allocation mechanisms for settings in which users have non-uniform preferences over edge/cloud servers. We can define preferences based on the location of users and the type of applications that they need to execute. For example, if users have to execute computation-intensive applications, they would prefer the cloud servers, while if they have to execute small applications requiring a large amount of communication, they would prefer the edge servers.

Acknowledgment

This research was supported in part by NSF under grant IIS-1724227. We would like to thank our shepherd, John Kubiatowicz and the anonymous reviewers for their helpful and constructive suggestions, which considerably improved the quality of the paper.

References

[1] IBM ILOG CPLEX V12.1 user's manual, 2009.
[2] T. Bahreini and D. Grosu. Efficient placement of multi-component applications in edge computing systems. In *Proc. 2nd ACM/IEEE Symp. on Edge Computing*, pages 5:1–5:11, October 2017.
[3] M. Chowdhury, M. R. Rahman, and R. Boutaba. Vineyard: Virtual network embedding algorithms with coordinated node and link mapping. *IEEE/ACM Trans. on Networking*, 20(1):206–219, 2012.
[4] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti. Clonecloud: elastic execution between mobile device and cloud. In *Proc. of the 6th Conf. on Computer Systems*, pages 301–314. ACM, 2011.
[5] P. Cramton, Y. Shoham, and R. Steinberg. *Combinatorial Auctions*. MIT Press, 2006.
[6] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl. Maui: making smartphones last longer with code offload. In *Proc. 8th ACM International Conf. on Mobile Systems, Applications, and Services*, pages 49–62, 2010.
[7] D. Dutta, M. Kapralov, I. Post, and R. Shinde. Embedding paths into trees: Vm placement to minimize congestion. In *European Symposium on Algorithms*, pages 431–442. Springer, 2012.
[8] B. Edelman, M. Ostrovsky, and M. Schwarz. Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords. *American Economic Review*, 97(1):242–259, March 2007.
[9] X. Fan, J. Cao, and H. Mao. A survey of mobile cloud computing. *zTE Communications*, 9(1):4–8, 2011.
[10] X. Gu, K. Nahrstedt, A. Messer, I. Greenberg, and D. Milojicic. Adaptive offloading inference for delivering applications in pervasive computing environments. In *Proc. 1st IEEE Int. Conf. on Pervasive Computing and Communications*, pages 107–114, 2003.
[11] Y. Jiao, P. Wang, D. Niyato, and Z. Xiong. Social welfare maximization auction in edge computing resource allocation for mobile blockchain. *arXiv preprint arXiv:1710.10595*, 2017.

[12] A. Kiani and N. Ansari. Toward hierarchical mobile edge computing: An auction-based profit maximization approach. *IEEE Internet of Things Journal*, 4(6):2082–2091, 2017.

[13] 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. on Cloud Computing*, pages 614–621, 2011.

[14] K. Kumar and Y.-H. Lu. Cloud computing for mobile users: Can offloading computation save energy? *Computer*, 43(4):51–56, 2010.

[15] 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. *arXiv preprint arXiv:1711.02844*, 2017.

[16] L. Mashayekhy, M. M. Nejad, and D. Grosu. A PTAS mechanism for provisioning and allocation of heterogeneous cloud resources. *IEEE Transactions on Parallel and Distributed Systems*, 26(9):2386–2399, 2015.

[17] L. Mashayekhy, M. M. Nejad, D. Grosu, and A. V. Vasilakos. An online mechanism for resource allocation and pricing in clouds. *IEEE Transactions on Computers*, 65(4):1172–1184, 2016.

[18] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, New York, NY, USA, 2007.

[19] NSF. NSF Workshop Report on Grand Challenges in Edge Computing. 2016.

[20] M. Patel, B. Naughton, C. Chan, N. Sprecher, S. Abeta, A. Neal, et al. Mobile-edge computing - introductory technical white paper. European Telecommunications Standards Institute (ETSI), Mobile-edge Computing (MEC) industry initiative, September 2014.

[21] 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. on Mobile Systems, Applications, and Services*, pages 43–56, 2011.

[22] T. Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial intelligence*, 135(1-2):1–54, 2002.

[23] M. Satyanarayanan. A brief history of cloud offload: A personal journey from odyssey through cyber foraging to cloudlets. *GetMobile: Mobile Computing and Communications*, 18(4):19–23, 2015.

[24] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing*, 8(4):14–23, 2009.

[25] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646, 2016.

[26] H. R. Varian. Position auctions. *International Journal of Industrial Organization*, 25(6):1163 – 1178, 2007.

[27] Z. Xiong, S. Feng, D. Niyato, P. Wang, and Z. Han. Edge computing resource management and pricing for mobile blockchain. *arXiv preprint arXiv:1710.01567*, 2017.

[28] S. Zaman and D. Grosu. Combinatorial auction-based allocation of virtual machine instances in clouds. *Journal of Parallel and Distributed Computing*, 73(4):495–508, 2013.

[29] H. Zhang, F. Guo, H. Ji, and C. Zhu. Combinational auction-based service provider selection in mobile edge computing networks. *IEEE Access*, 5:13455–13464, 2017.

[30] H. Zhang, H. Jiang, B. Li, F. Liu, A. V. Vasilakos, and J. Liu. A framework for truthful online auctions in cloud computing with heterogeneous user demands. *IEEE Transactions on Computers*, 65(3):805–818, 2016.

[31] L. Zhang, Z. Li, and C. Wu. Dynamic resource provisioning in cloud computing: A randomized auction approach. In *Proc. IEEE INFOCOM*, pages 433–441, 2014.

[32] Y. Zhang, M. Li, K. Bai, M. Yu, and W. Zang. Incentive compatible moving target defense against vm-colocation attacks in clouds. In *IFIP International Information Security Conference*, pages 388–399. Springer, 2012.