An Edge Computing Matching Framework with Guaranteed Quality of Service

Nafiseh Sharghivand, Farnaz Derakhshan, Lena Mashayekhy, *Senior Member, IEEE,* and Leyli Mohammadkhanli

Abstract—Edge computing is a new computing paradigm, which aims at enhancing user experience by bringing computing resources closer to where data is produced by Internet of Things (IoT). Edge services are provided by small data centers located at the edge of the network, called cloudlets. However, IoT users often face strict Quality of Service (QoS) constraints for a proper remote execution of their applications on edge. Each user has specific resource requirements and budget limitations for her IoT application, while each cloudlet offers a limited number and types of resources, each with a specific cost. Therefore, a key challenge is how to efficiently match cloudlets to IoT applications and enable a convenient any-time access to edge computing services considering preferences and incentives of users and cloudlets. In this paper, we address this problem by proposing a novel two-sided matching solution for edge services considering QoS requirements in terms of service response time. In addition, we determine dynamic pricing of edge services based on the preferences and incentives of cloudlets, IoT users, and the system. The proposed matching is incentive compatible, individually rational, weakly budget balanced, asymptotically allocative efficient, and computationally efficient. We perform a comprehensive assessment through extensive performance analysis experiments to evaluate our proposed matching and pricing solutions.

Index Terms—edge computing; cloudlet; Internet of Things; Quality of Service; two-sided matching; combinatorial matching.

1 Introduction

The ubiquitous penetration of smart connected devices (Internet of Things) into everyday life is projected to reach 75 billion by 2025 [1]. The growth of IoT will continue as users enjoy the convenience of mobility and with the emergence and progress of new technologies such as wearable devices, autonomous vehicles/drones, and augmented/virtual reality. These IoT devices are restricted by weight, size, battery life, and heat dissipation imposing limitations on their computing capabilities to execute applications. To empower the resource shortage of IoT devices, cloud computing offers many services, allowing these devices to offload their tasks to a more powerful computing infrastructure. However, these devices require heterogeneous computing resources and quality of services (QoS) depending on their applications [2], and offloading to a conventional centralized cloud is infeasible for applications mandating low-latency communications and real-time responses due to physical distance between the cloud and IoT users. As a result, computing systems have undergone a fundamental transformation from homogeneous or semi-homogeneous centralized high-performance systems to a heterogeneous geo-distributed edge computing environment in order to fulfill today's IoT computational needs.

Edge computing is a new computing paradigm that enables processing data closer to IoT users, where data has been produced locally [3], [4]. Instead of moving a large amount of raw data from users to a distant cloud, it decentralizes processing power to ensure real-time ultra-low-latency processing. The distribution of computing resources throughout edge computing is accomplished by having many small-sized heterogeneous clusters of servers referred to as "cloudlets" or "micro data centers" at the edge of the network. Cloudlets help to mitigate the overload of IoT devices by accepting offloaded computation and offer edge services in the form of virtual machine (VM) instances. However, to enable a convenient any-time access to edge computing resources, a key challenge is how to efficiently match each IoT application to a serving cloudlet and determine the associated edge service pricing.

A cloudlet has limited available resources, and it can provide specific QoS guarantees (such as service response time) for the offered edge services. Moreover, cloudlets incur cost in providing edge services. On the other hand, each IoT user (application owner) has specific resource requirements for her application with limited budget and may need a strict guarantee for the edge service quality. Both sides (IoT users and cloudlets) may not declare their true preferences, but to misreport them in order to increase their utilities (e.g., higher quality of experience and/or lower payment for a user). We formulate this problem as a two-sided matching game between IoT users and cloudlets. A two-sided matching game is an assignment problem between the sets of users and cloudlets (players), where the players of each set have preferences over the players of the other set. The preference relations allow every IoT user/cloudlet to be best matched for the edge services while maximizing its own benefit (utility). Finding the best matching of a two-sided matching game is an NP-hard problem.

- *N. Sharghivand, F. Derakhshan, and L. Mohammadkhanli are with the Department of Computer Engineering, University of Tabriz, Tabriz, Iran. L. Mashayekhy is with the Department of Computer and Information Sciences, University of Delaware, Newark, DE 19716. E-mail: n.sharghivand@tabrizu.ac.ir, derakhshan@tabrizu.ac.ir, mlena@udel.edu, l-khanli@tabrizu.ac.ir*

We propose an efficient QoS-aware matching mechanism maximizing social welfare, which is defined as the sum of utilities of users, cloudlets, and the system. Our proposed mechanism gives incentives to IoT users and cloudlets to be truthful and to reveal their true preferences. This is an important property of our mechanism, making it robust against strategic users and cloudlets who try to change their allocation and/or other users'/cloudlets' allocations. Moreover, the pricing of the matched edge computing services are dynamically determined by our mechanism based on IoT demand and cloudlet supply. Our proposed matching mechanism improves quality of experience (QoE) and user satisfaction by considering QoS metrics in the matching. In addition, it is computationally efficient.

1.1 Our Contribution

We model the matching between cloudlets and IoT users considering QoS as an Integer Programming optimization model, and propose a novel two-sided matching and pricing mechanism, Combinatorial QoS-aware Matching of Edge Computing Services (C-QMECS). C-QMECS is based on padding that intentionally creates imbalances between VMs availability and demand by introducing a phantom user enabling it to achieve incentive compatibility, individually rationality, weakly budget balanced, asymptotically allocative efficiency, and computational tractability. We devise a grouping approach for the users to reduce the computation required by C-QMECS. In addition, C-QMECS improves users' satisfaction and QoE by considering QoS metrics in the matchings. As a result, our mechanism avoids user service rejection after their assignment to a cloudlet due to unacceptable QoE. C-QMECS is multi attributes that considers QoS requirements of IoT users and QoS guarantees of cloudlets in addition to their pricing preferences. Moreover, considering the heterogeneous nature of required and offered services, C-QMECS allows users and cloudlets to respectively request for and offer a variety of different VMs. Therefore, C-QMECS is also assumed to be combinatorial. We provide a comprehensive assessment through extensive performance analysis experiments to evaluate C-QMECS.

1.2 Related Work

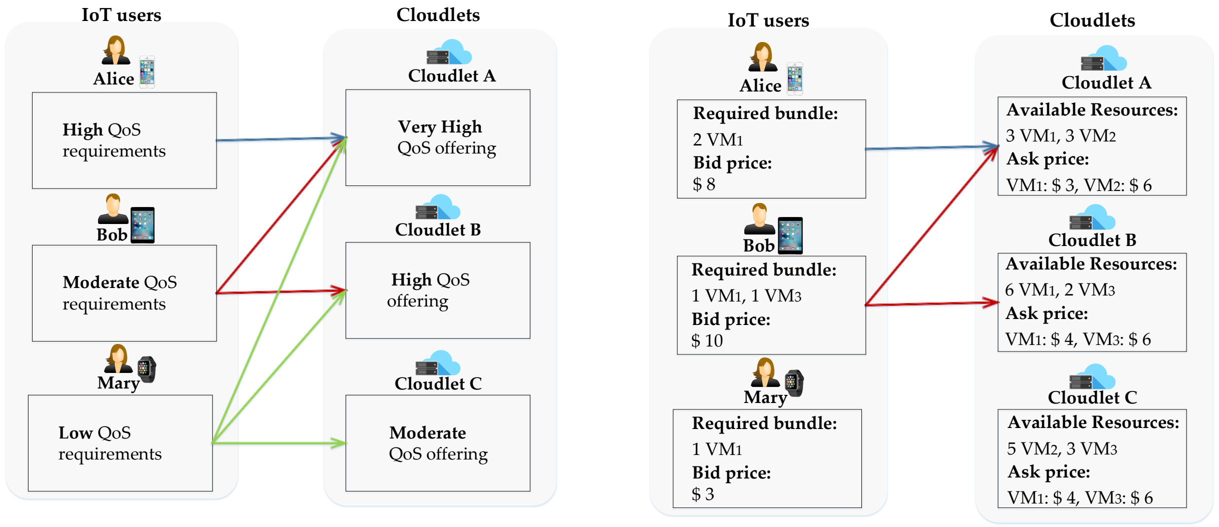
A large body of research has focused on VM provisioning, allocation, and placement in clouds by designing centralized or semi-centralized (i.e., in-site distributed) approaches [5], [6]. However, they require global information and often centralized controllers, yielding significant overhead and complexity especially when dealing with combinatorial integer programming problems to be solved. Moreover, they do not consider the possibility of interactions among clouds and/or users. These studies employ optimization techniques without considering users and clouds as decision makers. The aforementioned limitations of optimization have led to an interesting body of literature that deals with the use of game theory and mechanism design in cloud resource management [7], [8]. Users' demands and preferences, and cloudlets' offerings and preferences can directly affect the computing system. Therefore, providing edge computing services for each user depends not only on properties of the user, but also on properties of other users and cloudlets, leading to a game among users and cloudlets.

Most of game-theoretical studies in cloud computing focus solely on one side of the market, where users interact with only one provider. Main stream cloud provider powerhouses such as Amazon have been offering cloud services in a one-sided auction market (Amazon Spot Market) for several years [9]. In our previous studies [10]–[13], we proposed truthful offline and online one-sided mechanisms for allocation and pricing of heterogeneous VMs in clouds. Several researchers have studied resource management in cloud federations to facilitate big data processing [14], [15]. However, these studies only focus on interactions among a group of cloud providers to provide a single big data processing service, and they are not suitable to be adapted in multi-user edge computing environment. Efficient resource management mechanisms need to consider both supply and demand sides together. Designing double-auction mechanisms for cloud computing has been studied by considering homogeneous or heterogeneous VMs [16], [17]. However, existing cloud-based solutions do not address the challenges of "being at the edge" [18], and are not appropriate for edge computing as QoE of users are critical.

Studies on various facets of edge computing have come from both academia and industry. Amazon IoT Greengrass, Azure IoT Edge, Google Cloud IoT Core, and Google Edge TPU (Tensor Processing Unit) are some of the commercial implementations of edge computing to extend cloud capabilities to the proximity of users. Most studies in academia focus on decisions on efficient caching [19] and fully/partially computation offloading to cloudlets considering one user application [20], [21] or multiple applications [14], [22].

Several studies in edge computing investigate the use of game theory and mechanism design. Kiani et al. [23] proposed a hierarchical mobile edge computing (HI-MEC) architecture, where the computing resources are offered using an auction-based profit maximization model. Bahreini et al. [24] investigated the problem of resource allocation and pricing in a two-level edge computing system, where servers are located in the cloud or at the edge. They designed an envy-free auction-based mechanism to allocate available resources at these two levels. Gao et al. [25] modeled the VM allocation problem as an n-to-one weighted bipartite graph matching problem, and designed a greedy approximation algorithm to determine the winners of the auction. However, they did not consider QoS guarantees. Moreover, they considered only one VM type. Li et al. [26] designed a computing resource trading market for edge-cloud-assisted IoT in a blockchain network, employing an iterative double-sided auction scheme. Zavodovski et al. [27] developed a double auction mechanism by employing a distributed ledger trust model to match heterogeneous cloud and edge resources to users using a heuristic approach and a group-based pricing. However, none of the above mentioned studies consider the characteristics of IoT applications and the objectives of edge computing in providing QoS guarantees for users.

To the best of our knowledge, this is the first work that models QoS in the matchings of edge services between users and cloudlets in a heterogeneous edge computing


IoT users
Cloudlets
Alice
High QoS requirements
Bob
Moderate QoS requirements
Mary
Low QoS requirements
Cloudlet A
Very High QoS offering
Cloudlet B
High QoS offering
Cloudlet C
Moderate QoS offering
IoT users
Cloudlets
Alice
Required bundle: 2 VM1 Bid price: $ 8
Bob
Required bundle: 1 VM1, 1 VM3 Bid price: $ 10
Mary
Required bundle: 1 VM1 Bid price: $ 3
Cloudlet A
Available Resources: 3 VM1, 3 VM2 Ask price: VM1: $ 3, VM2: $ 6
Cloudlet B
Available Resources: 6 VM1, 2 VM3 Ask price: VM1: $ 4, VM3: $ 6
Cloudlet C
Available Resources: 5 VM2, 3 VM3 Ask price: VM1: $ 4, VM3: $ 6

(a) Efficient matching according to QoS requirements (b) Efficient matching according to bundle requirements

Fig. 1: Two-Sided Matching

environment to guarantee service time requirements. We model their preferences and interactions as decision makers using game theory and matching theory, and design a novel QoS-aware matching in edge computing.

1.3 Organization

The rest of the paper is organized as follows. In Section 2, we discuss the necessity of using QoS-aware matching in edge computing. In Section 3, we describe the system model. In Section 4, we propose our mechanism to solve the cloudlet-IoT matching problem. In Section 5, we evaluate our proposed mechanism by extensive experiments. Finally, in Section 6, we summarize our results and present possible directions for future research.

2 Necessity of QoS-Aware Matching

In order to describe the advantages of a QoS-aware matching mechanism in a possible real-world situation, we consider the following example. We assume there are three IoT users called Alice, Mary, and Bob, who have high, moderate, and low QoS requirements, respectively. Moreover, there are three cloudlets A, B, and C in the environment, providing very high, high, and moderate QoS, respectively. Therefore, in order for the users to have high QoE, they should be matched to a cloudlet that meets their QoS requirements. Fig. 1a illustrates the best possible matching in this case, where Alice can be matched to cloudlet A, Bob can be matched to either cloudlets A or B, and finally, Mary can be matched to either cloudlets A, B, or C.

Considering the existing constraints on required and offered QoS, the next step is to find the best matching according to the resource requirements and budget limitations of users, and available resources and prices of cloudlets. To satisfy truthfulness, a typical mechanism-design goal is to maximize social welfare, which is defined as the sum of utilities of users, cloudlets, and the broker. We will describe social welfare in detail in Section 3, and show that the sum of utilities is equal to the difference between users' bids and the cloudlets' asks.

Fig. 1b shows the final best matching, where the obtained social welfare is maximized considering QoS. In this example, Alice and Bob pay a total of $18, while cloudlets A and B receive a payment of $15. Therefore, the obtained social welfare is equal to $3. However, existing approaches find the matching that improves the social welfare of all participants without considering the QoS requirements and guarantees of users and cloudlets in matching, respectively. For instance, matching Bob to cloudlet C provides a lower priced VM_3. However, this matching cannot satisfy the QoS requirements of Bob's application. Therefore, he will have a poor QoE and thus dissatisfaction.

3 System Model

In this section, we introduce the system model, where J is the set of cloudlets and I is the set of IoT users. Each user requests an indivisible bundle of VMs (one or several VM instances of the same type or different types) for her IoT application (i.e., edge service), while it requires some QoS metrics to be satisfied. Each cloudlet offers a set of VMs that can be allocated to different users, and it guarantees several QoS metrics for the offered services. We define $VM = \{vm_1, vm_2, ..., vm_K\}$ as the set of K different types of VM instances offered by the cloudlets. The vector L defines the QoS metrics including Average Response Time (ART), Maximum Response Time (MRT), and Response Time Failure (RTF). ART is defined as the average time that each user has to wait for her requested bundle of VMs to be allocated. MRT shows the maximum promised response time by the cloudlet. RTF shows the percentage of the times when the cloudlet has violated its promised MRT. All these QoS metrics are defined as quantified metrics to quantify service response time, which can be measured using software and hardware monitoring tools and simply declared in numeric values [28]. Vector L can be extended to include other quantified QoS metrics.

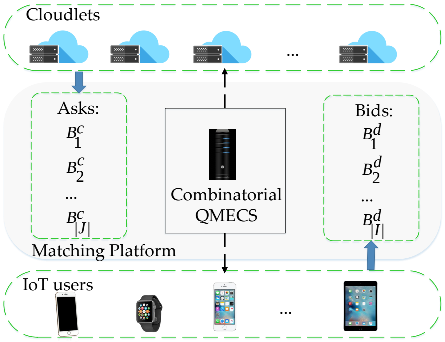


Fig. 2: System Model.

Each cloudlet $j \in J$ declares the quantity of available VMs for each type, $VM^c_j = \{q^c_{j1}, q^c_{j2}, ..., q^c_{jK}\}$, where q^c_{jk} is the number of available VMs of type vm_k at cloudlet j. The asking prices or costs for providing one VM instance of different types are specified by $P^c_j = \{p^c_{j1}, p^c_{j2}, ..., p^c_{jK}\}$. Finally, the QoS guaranteed by cloudlet j is denoted by $\vec{QoS}^c_j =< QoS^c_{j1}, \ldots, QoS^c_{j|L|} >$. Hence, the specification of each cloudlet is denoted by ask $B^c_j = (VM^c_j, P^c_j, \vec{QoS}^c_j)$. The superscript c refers to cloudlets in all the notations. For example, assuming four types of VMs offered by the cloudlets, for cloudlet j with 4, 3, 0, and 2 available VMs of types vm_1 to vm_4, respectively, the reserve prices of \$1, \$2, \$3, \$4 for each instance, and guarantees of $ART = 1.5$ ms, $MRT = 2$ ms, and $RTF = 0.01\%$, its specification is denoted by ask $B^c_j = ([4, 3, 0, 2], [\$1, \$2, \$3, \$4], [1.5ms, 2ms, 0.01\%])$.

Each IoT user requests several VM instances of different types, sets a preferred price for the requested bundle, and specifies QoS requirements for the requested edge service. Hence, the specification of an IoT request of user i consists of three parts. First, $VM^d_i = (q^d_{i1}, q^d_{i2}, ..., q^d_{iK})$ represents the number of requested VMs of each type, where q^d_{ik} denotes the number of VMs of type vm_k requested by user i. Second, p^d_i represents bidding price for the whole requested bundle, that is the maximum price user i is willing to pay for the requested bundle of VMs. Finally, $\vec{QoS}^d_i$ denotes the QoS requirements of user i's IoT application. These QoS values suggest the least acceptable qualities for the edge service requested by user i. As a result, each user request is denoted by bid $B^d_i = (VM^d_i, p^d_i, \vec{QoS}^d_i)$. The superscript d refers to users in all the notations. For instance, the bid of user i, who is willing to pay upto \$7 for two VMs of type vm_1 and one VM of type vm_2 with $ART = 2$ ms, $MRT = 2.5$ ms, and $RTF = 0.05\%$ is denoted by $B^d_i = ([2, , 1, 0, 0], \$7, [2ms, 2.5ms, 0.05\%])$. Fig. 2 shows the supply and demand sides in the edge computing system.

A two-sided market also requires a broker as a mediator and trusted third party to facilitate the matching. A broker is responsible for receiving IoT user requests and cloudlet offers, determining matchings and prices, billing the users, receiving payments from users, and paying the participating cloudlets. In edge computing, we can consider several brokers, each managing a specific small region. To avoid a single point of failure, we can also consider a backup broker for each one, similar to Standby ResourceManager of YARN.

The utility of user i is defined as $U^d_i = p^d_i - \pi^d_i$, the difference between her valuation and her payment if her request is matched; and zero otherwise. Similarly, the utility of cloudlet j is defined as $U^c_j = \pi^c_j - \sum_{vm_k \in VM} q^m_{jk} p^c_{jk}$, the difference between the cloudlet's payment and its cost for the matched VMs; and zero otherwise. We use q^m_{jk} to represent the number of matched VMs of type vm_k of cloudlet j. The broker's monetary payoff π^b is calculated as $\pi^b = \sum_{i \in I} \pi^d_i - \sum_{j \in J} \pi^c_j$, which is the total payments received from the users minus the revenues of the cloudlets.

The objective of participants is to maximize their own utilities. This is defined by social welfare as the summation of utility of each IoT user, cloudlet, and the broker as follows:

$$
\begin{aligned}
V(I, J) &= \sum_{i \in I} U^d_i + \sum_{j \in J} U^c_j + \pi^b \\
&= \sum_{i \in I^m} (p^d_i - \pi^d_i) + \sum_{j \in J^m} (\pi^c_j - \sum_{vm_k \in VM} q^m_{jk} p^c_{jk}) \\
&\quad + \sum_{i \in I^m} \pi^d_i - \sum_{j \in J^m} \pi^c_j \\
&= \sum_{i \in I^m} p^d_i - \sum_{j \in J^m} \sum_{vm_k \in VM} q^m_{jk} p^c_{jk}, \qquad (1)
\end{aligned}
$$

where I^m and J^m are the sets of matched IoT users and cloudlets, respectively. Therefore, social welfare is equal to the difference between users' bids and the cloudlets' asks.

To find the best matching of the services between IoT users and cloudlets maximizing social welfare, we formulate the problem optimally as an Integer Program (IP). First, we define the decision variables as follows: α_{ijk} is the number of allocated VMs of type vm_k by cloudlet j to IoT user i; $x_i \in \{0, 1\}$ shows whether IoT user i has received her requested bundle or not; and y_{jk} denotes the number of VMs of type vm_k allocated by cloudlet j. In addition, we define $Z(i, j) = \vec{QoS}^d_i \succeq \vec{QoS}^c_j$ as an indicator function for the matching between cloudlet j and user i based on the offered and requested QoS, which returns 1 if $QoS^d_{il} \geq QoS^c_{jl}, \forall l \in L$, that is the offered QoS by cloudlet j meets the requirements of user i, and 0 otherwise. The value of $Z(i, j)$ indicates the feasibility of matching the VMs provided by cloudlet j to IoT user i. In our example, cloudlet j with $B^c_j = ([4, 3, 0, 2], [\$1, \$2, \$3, \$4], [1.5ms, 2ms, 0.01\%])$ is qualified to provide the requested service of user i with $B^d_i = ([2, , 1, 0, 0], \$7, [2ms, 2.5ms, 0.05\%])$, since it guarantees a better service response time; in this case $Z(\vec{QoS}^d_i \succeq \vec{QoS}^c_j) = 1$. We now formulate the problem as an IP as follows:

$$
\begin{aligned}
\texttt{Matching}(I, J) :\ \text{Max}\ V(I, J) &= \sum_{i \in I} p^d_i x_i \\
&- \sum_{j \in J} \sum_{vm_k \in VM} p^c_{jk} y_{jk} \qquad (2)
\end{aligned}
$$

Subject to:

$$
\sum_{j \in J} \alpha_{ijk} = q^d_{ik} x_i, \quad \forall i \in I, vm_k \in VM, \qquad (3)
$$

$$\sum_{i \in I} \alpha_{ijk} = y_{jk}, \quad \forall j \in J, vm_k \in VM, \tag{4}$$

$$\alpha_{ijk} \leq q^c_{jk}\, Z(i,j), \quad \forall i \in I, j \in J, vm_k \in VM, \tag{5}$$

$$y_{jk} \leq q^c_{jk}, \quad \forall j \in J, vm_k \in VM, \tag{6}$$

$$\alpha_{ijk} \in \mathbb{Z}^*, \quad \forall i \in I, j \in J, vm_k \in VM, \tag{7}$$

$$y_{jk} \in \mathbb{Z}^*, \quad \forall j \in J, vm_k \in VM. \tag{8}$$

$$x_i \in \{0,1\}, \quad \forall i \in I. \tag{9}$$

The objective function (Eq. (2)) is to maximize the social welfare function $V(I,J)$. Constraint (3) ensures that each IoT user receives her whole requested bundle or nothing. Constraint (4) guarantees that each cloudlet supplies VMs based on their availability. Constraint (5) ensures that each cloudlet can serve an IoT user if only it can provide the requested VMs and meet the QoS requirements of that user. Constraint (6) ensures that the allocated VMs do not exceed the available VMs of the cloudlets. Constraints (7) and (8) guarantee the integrality of decision variables α_{ijk} and y_{jk}. We use $\mathbb{Z}^*$ to represent the set of nonnegative integers. Constraint (9) ensures decision variable x_i is binary.

Desirable properties. Cloudlets and IoT users are self-interested and rational, meaning that their objectives are to maximize their own utilities. To promote transactions and attract both users and cloudlets, we design a QoS-aware matching and pricing mechanism for edge computing considering the desirable properties that such a mechanism needs to satisfy in order to be applicable in real world.

- *Incentive Compatibility (IC).* If each participant can achieve the best outcome by acting truthfully. This property preserves the system from strategic manipulations by participants. Therefore, there is no need for participants to analyze the behavior of other participants in order to choose their best strategy and they can simply declare their true preferences.
- *Individual Rationality (IR).* If each participant does not incur a negative utility by participating. This property guarantees participation of IoT users and cloudlets in the mechanism.
- *Weakly Budget Balanced (BB).* If the total payments of users always exceeds or equals the revenues of cloudlets. This property avoids the broker to suffer a loss by running the mechanism.
- *Allocative Efficiency (AE).* If the matching maximizes the social welfare.
- *Computational Efficiency (CE).* If the matching is performed in polynomial time.

Although it is desired to adhere all properties, according to the Myerson-Sattherwithe impossibility theorem [29], no mechanism can achieve IC, IR, BB, AE properties simultaneously. Therefore, since IC, IR, and BB are essential for a mechanism to be applicable in real world, we can relax AE property, while still achieving an acceptable level of social welfare.

4 A Combinatorial Two-Sided Matching Mechanism

In this section, we describe our proposed mechanism, C-QMECS, that satisfies IC, IR, BB, asymptotic AE, and CE. C-QMECS introduces a phantom IoT user to the system with a fixed request (equal to the maximum quantity of offered resources from all cloudlets), unlimited budget, and specific QoS. This is because the cloudlet with the largest supply has the greatest power in manipulating prices, and adding this phantom user avoids such a manipulation. Adding a phantom user also creates an imbalance between VMs availability and demand, which provides an efficient way to improve market price equilibrium leading to a budget surplus, that motivates the broker to run the mechanism.

In order to have a computationally efficient mechanism, the matching and pricing decisions of C-QMECS are determined based on our proposed linear programming-based (LP) approach described in the following subsection. These LPs can be solved in polynomial time in the worst case.

4.1 Combinatorial QoS-aware Matching of Edge Computing Services (C-QMECS)

In this section, we define our proposed C-QMECS mechanism in five steps as follows:

Step 1: Preference submission. Collect one sealed bid/ask preference from each IoT user/cloudlet.

Step 2: User set reduction. Solve the following LP $\texttt{SELECT}(I,J)$, with IoT set I and cloudlet set J:

$$\texttt{SELECT}(I,J): \ \text{Max}\ \tilde{V}(I,J) = \sum_{i \in I} p^d_i\, x_i - \sum_{j \in J} \sum_{vm_k \in VM} p^c_{jk}\, y_{jk}$$

Subject to:

$$\sum_{j \in J} \alpha_{ijk} = q^d_{ik}\, x_i, \quad \forall i \in I, vm_k \in VM,$$

$$\sum_{i \in I} \alpha_{ijk} = y_{jk}, \quad \forall j \in J, vm_k \in VM,$$

$$0 \leq \alpha_{ijk} \leq q^c_{jk}\, Z(i,j), \quad \forall i \in I, j \in J, vm_k \in VM,$$

$$0 \leq y_{jk} \leq q^c_{jk}, \quad \forall j \in J, vm_k \in VM$$

$$0 \leq x_i \leq 1, \quad \forall i \in I.$$

For any IoT user $i \in I$, if $x_i = 1$, then i remains for the matching. All other users who could not acquire their whole requested bundles are eliminated. Let $\tilde{I}$ denote the set of remaining IoT users for matching at this step.

Step 3: Grouping users and setting padding. We group the users in $\tilde{I}$, such that the QoS requirements of all users in each group are satisfied by the same set of cloudlets. The set of grouped users is denoted by $G = \{g_1, g_2, ..., g_M\}$, where M is the number of groups. In order to set a padding for each group, we need to define a phantom user with a specific bid (bundle, bidding price, QoS requirements). The size of the padding is determined by the size of the bundle of the phantom user.

We randomly select a user u from each group g_u as its delegate since all users in the same group are satisfied by the same set of cloudlets in terms of QoS requirements. We then define a $|\tilde{I}| \times K$ padding matrix $\boldsymbol{Q}^u$, such that for user i, we have $\boldsymbol{Q}^u_i = \{q^u_{i1}, q^u_{i2}, ..., q^u_{iK}\}$. If $i = u$ then $\boldsymbol{Q}^u_i = \boldsymbol{Q}^p$; otherwise $\boldsymbol{Q}^u_i = \boldsymbol{0}$. We define $\boldsymbol{Q}^p$ as a general padding vector equals to $\{q^p_1, q^p_2, ..., q^p_K\}$, where $q^p_k = \max_{j\in J}\{q^c_{jk}\}$ for each $vm_k \in VM$ representing the number of required VMs of each type by the phantom user. Meaning that, the size of the padding for each VM type is equal to the highest amount of supply for that VM type. After determining the bundle of the phantom user for IoT group g_u, we allocate unlimited budget and set the same QoS constraints as user u's. In other words, the request of the phantom user for group g_u is defined as $(\boldsymbol{Q}^p, \$\infty, \vec{QoS}^d_u)$.

Note that only one phantom user is added to the system. Depending on a group that we perform the matching, the QoS requirements of the phantom user is adjusted. Adding the phantom user to the system is needed to guarantee incentive compatibility, while it may avoid serving a limited number of users. However, in edge computing with high number of cloudlets with limited resources, the padding size would be very small. Therefore, the number of unserved users is significantly low. This is also in alignment with asymptotic AE of C-QMECS (Theorem 7).

Step 4: Matching. For each $g_u \in G$, where user u is the delegate of this group, we solve the following LP $\texttt{PADDING}(\tilde{I}, J, \boldsymbol{Q}^u)$ to determine the final set of IoT users to be matched for the edge services:

$$\texttt{PADDING}(\tilde{I}, J, \boldsymbol{Q}^u): \ \text{Max}\ \hat{V}(\tilde{I}, J, \boldsymbol{Q}^u) = \sum_{i\in\tilde{I}} p^d_i\, x_i - \sum_{j\in J}\sum_{vm_k\in VM} p^c_{jk}\, y_{jk}$$

Subject to:

$$\sum_{j\in J} \alpha_{ijk} = q^d_{ik}\, x_i + q^u_{ik}, \quad \forall i \in \tilde{I}, vm_k \in VM,$$

$$\sum_{i\in\tilde{I}} \alpha_{ijk} = y_{jk}, \quad \forall j \in J, vm_k \in VM,$$

$$0 \le \alpha_{ijk} \le q^c_{jk}\, Z(i,j), \quad \forall i \in \tilde{I}, j \in J, vm_k \in VM,$$

$$0 \le y_{jk} \le q^c_{jk}, \quad \forall j \in J, vm_k \in VM,$$

$$0 \le x_i \le 1, \quad \forall i \in \tilde{I}.$$

For any IoT user $i \in g_u$, if $x_i = 1$, then i is considered as one of final matched users. Let $\hat{I}$ denote the set of users selected in this step. Grouping the users can significantly reduce the number of LPs needed to be solved in this step. Then, we solve $\texttt{SELECT}(\hat{I}, J)$ to determine the set of matching cloudlets and their allocation of edge services considering the trading set of IoT users $\hat{I}$.

Step 5: Determining payment. The payment of each user i is calculated using the following equation:

$$\pi^d_i = \begin{cases} \hat{p}^d_i & \text{if } i \in \hat{I}, \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

If user i wins, she pays $\hat{p}^d_i$. This is the critical payment of the user, that is the minimum bid price to remain in the winning user set in the optimal solution to $\texttt{PADDING}(\tilde{I}, J, \boldsymbol{Q}^u)$. This price can be viewed as a shadow price and can be calculated through sensitivity analysis. If user i does not win, she pays nothing and obtains a utility of zero.

The revenue for each cloudlet is defined based on the marginal contribution of the cloudlet, that is the amount it adds to the social welfare by participating. The payment to each cloudlet j is calculated using the following equation:

$$\pi^c_j = \sum_{vm_k\in VM} p^c_{jk} y_{jk} + \tilde{V}(\hat{I}, J) - \tilde{V}(\hat{I}, J/\{j\}) \tag{11}$$

In other words, if winning cloudlet j had not been participating, other qualified cloudlets would have won. Therefore, cloudlet j receives a payment equal to the offered price of new winning cloudlets.

All users in $\hat{I}$ are also selected by the optimal solution of $\texttt{SELECT}(\hat{I}, J/\{j\})$ for any $j \in J$ (Lemma 1). Suppose $\hat{I}^j (\hat{I}^j \subset \hat{I})$ denotes all the users that cloudlet j is qualified to be matched, and $\hat{J}^j$ denotes the set of new winning cloudlets which get to allocate their VMs to users in $\hat{I}$ in the absence of cloudlet j. Then, the total number of VMs of type vm_k allocated by all the cloudlets in $\hat{J}^j$ is $Y_k = \sum_{i\in\hat{I}^j} q^d_{ik}$. All these VMs are ranked according to their offered prices from low to high. Hence, we de-

Algorithm 1 C-QMECS Mechanism

```
/*Step 1: Preference submission*/
Input: $B^d_i = (VM^d_i, p^d_i, \vec{QoS}^d_i); \forall i \in I$
Input: $B^c_j = (VM^c_j, P^c_j, \vec{QoS}^c_j); \forall j \in J$
/*Step 2: User set reduction*/
$\tilde{I} = \{i | i \in I, x_i = 1$ in the optimal solution to $\texttt{SELECT}(I, J)\}$
/*Step 3: Grouping users and setting padding*/
for all $vm_k \in VM$ do
    $q^p_k = \max_{j\in J}\{q^c_{jk}\}$
end for
$\boldsymbol{Q}^p = \{q^p_1, q^p_2, ..., q^p_K\}$
Find $G$ based on QoS
for all $g_u \in G$ do
    for all $i \in g_u$ do
                ▷ $u$ is the delegate user belonging to group $g_u$
        if $i = u$ then
            $\boldsymbol{Q}^u_i = \boldsymbol{Q}^p$
        else if $i \neq u$ then
            $q^u_{ik} = 0, \forall vm_k \in VM$
        end if
    end for
    A phantom user with request $(\boldsymbol{Q}^p, \$\infty, \vec{QoS}^d_u)$ is added to $g_u$
end for
/*Step 4: Matching*/
for all $g_u \in G$ do
    $\hat{I} = \{i | i \in g_u, x_i = 1$ in the optimal solution to $\texttt{PADDING}(\tilde{I}, J, \boldsymbol{Q}^u)\}$   ▷ $\hat{I}$ is the set of matched IoT users
end for
Solve $\texttt{SELECT}(\hat{I}, J)$ to determine $\alpha, y$
/*Step 5: Determining payment*/
$\Pi^d = \{\pi^d_i : \min \text{price in } \texttt{PADDING}(\tilde{I}, J, \boldsymbol{Q}^u) | \forall i \in \hat{I}\}$
$\Pi^c = \{\pi^c_j = \sum_{vm_k\in VM}\sum_{r=1}^{y_{jk}} c^k_{-j}[Y_k - r + 1] | \forall j \in J\}$
Output: $x; y; \alpha; \Pi^d; \Pi^c$
```

fine $c^k_{-j}[r]$ as the price of rth cheapest instance of VM type vm_k. We can rewrite the Equation (11) as follows:

$$\pi^c_j = \sum_{vm_k \in VM} \pi^c_{jk} = \sum_{vm_k \in VM} \sum_{r=1}^{y_{jk}} c^k_{-j}[Y_k - r + 1], \quad (12)$$

which calculates the cost when cloudlet j is not participating. We will prove in Lemma 2 that Equation (11) is equivalent to Equation (12).

C-QMECS mechanism. The C-QMECS mechanism is given in Algorithm 1. It has two inputs: the preferences of IoT users and cloudlets, and it returns the matching solution and payments as outputs. The matching solution α shows the number of allocated VMs of each type by each cloudlet to each IoT user. The set $\Pi^d = \{\pi^d_i | i \in \hat{I}\}$ determines the buying prices of requested bundles for the matched IoT users. Finally, the last output is the selling prices of the matched cloudlets for each type of VM, i.e., $\Pi^c = \{\pi^c_j | \forall j \in J\}$, where $\pi^c_j = \{\pi^c_{jk} | \forall vm_k \in VM\}$.

4.2 An Illustrative Example

We provide an example to better demonstrate how our proposed mechanism, C-QMECS, works. Consider four IoT users and three cloudlets with the preferences shown in Tables 1 and 2, respectively. Now, we go through the steps of the mechanism.

In step 1, as it is shown in Fig.3a, the preferences of all participants are collected.

In step 2, $\texttt{SELECT}(I, J)$ is solved to eliminate users who do not have any chance to win their required bundles and thus reducing the user set. All cloudlets are qualified based on their offered QoS to provide services for users u_1 and u_2. Moreover, cloudlets c_1 and c_2 are only qualified to provide services for user u_3 based on her required QoS. Finally, none of the cloudlets can satisfy the QoS requirements of services requested by user u_4. Thus, as Fig.3b illustrates, u_4 is simply eliminated in this step.

In step 3, C-QMECS groups the users. Fig.3c shows that two groups g_1 and g_2 are formed. Then, C-QMECS determines the padding vector, such that each of its elements is equal to the highest supply in the market. In Fig.3c, the highest supply of each VM type is highlighted by a red rectangle. Then, a phantom user is added with a bundle equivalent to the highest supply.

In step 4, first $\texttt{PADDING}(\tilde{I}, J, \boldsymbol{Q}^u)$ is solved for each of the two groups to determine the set of winning users, i.e., $\hat{I}$. With the added phantom user and its unlimited budget in the optimal solution to $\texttt{PADDING}(\tilde{I}, J, \boldsymbol{Q}^{u_1})$, u_1 acquires 1/3 of her requested bundle, while u_2 acquires his whole requested bundle. Solving $\texttt{PADDING}(\tilde{I}, J, \boldsymbol{Q}^{u_3})$, u_3 can also win her requested VM. Thus, $\hat{I} = \{u_2, u_3\}$. Then, the mechanism solves $\texttt{SELECT}(\hat{I}, J)$ to determine the set of matching cloudlets. Fig.3d shows the final matching between the IoT users and cloudlets.

In step 5, the last step, the payments and revenues of users and cloudlets are determined, respectively. As for winning user u_2, his critical payment to require vm_1 is $2.5, while his critical payment for vm_2 is $4. Thus, he pays $6.5 in total, which is less than his own bid ($7). Similarly, the critical payment for user u_2 to win her required VM is $2. As for the winning cloudlets, cloudlet c_1 receives $1.1, as the lowest-priced vm_1 in its absence is $1.1. Similarly, cloudlet c_2 receives $2 for its vm_2 and $4 for its vm_3. Thus, cloudlet c_2 receives $6 in total. Cloudlet c_3 does not receive any revenue as it does not allocate any of its VMs. Note that, the received revenues of all winning cloudlets are higher than their own asking prices. Finally, the broker receives $8.5 from users and pays $7.1 to the cloudlets, achieving a non-negative utility of $1.4. Fig.3e shows the payment transactions.

4.3 Theoretical Analysis

In this section, we evaluate our proposed C-QMECS mechanism theoretically. We prove that our proposed mechanism demonstrates the properties of incentive compatibility (IC) (Theorems 1 and 4), individual rationality (IR) (Theorems 2 and 5), budget balanced (BB) (Theorem 6), and asymptotically allocative efficiency (Theorem 7), which are critical for the addressed problem. Moreover, C-QMECS has a polynomial time complexity, which makes it applicable in real world. Meanwhile, it induces an integral and feasible matching (Theorem 3), as each user acquires her whole requested bundle, while each cloudlet allocates discrete units of VMs.

Theorem 1. C-QMECS mechanism is truthful for all users, i.e., bidding truthfully is the dominant strategy for all users for any nonnegative padding.

Proof: A truthful mechanism needs to satisfy the critical payment property and monotonicity.

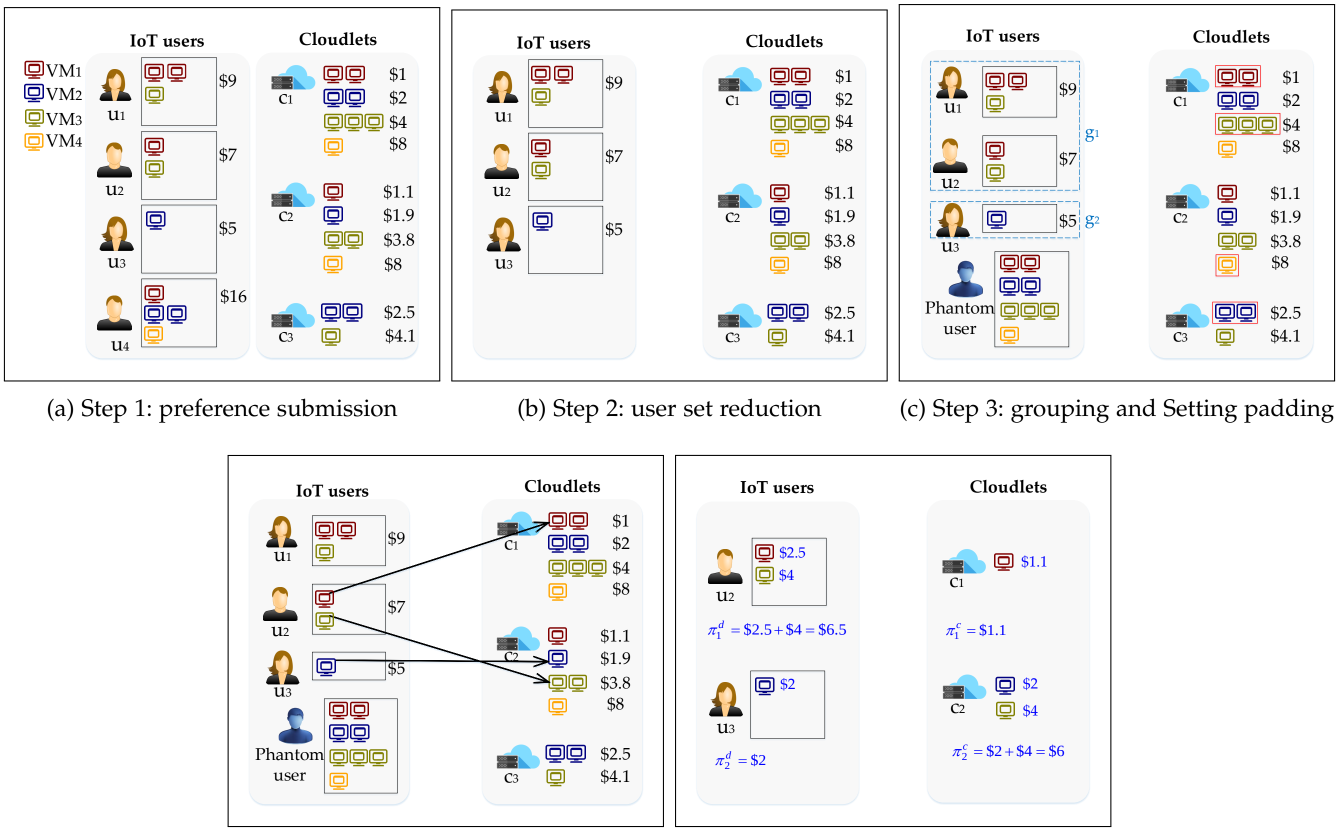
To satisfy critical payment, we need to show that there exists a unique value $\hat{p}^d_i$ for each matched user i, where

TABLE 1: Bid Preferences of IoT Users

IoT users	No. of Asked VMs				Bundle	Asked QoS		
	VM$_1$	*VM*$_2$	*VM*$_3$	*VM*$_4$	price ($)	*ART*	*MRT*	*RTF*
u_1	2	0	1	0	9	2.4	2.7	0.02
u_2	1	0	1	0	7	2.5	2.8	0.03
u_3	0	1	0	0	5	1.9	2.2	0.02
u_4	1	2	0	1	16	1.0	1.8	0.01

TABLE 2: Ask Preferences of Cloudlets

Cloudlets	No. of available VMs				Price per instance ($)				Guaranteed QoS		
	VM$_1$	*VM*$_2$	*VM*$_3$	*VM*$_4$	*VM*$_1$	*VM*$_2$	*VM*$_3$	*VM*$_4$	*ART*	*MRT*	*RTF*
c_1	2	2	3	1	1	2	4	8	1.8	2.0	0.01
c_2	1	1	2	1	1.1	1.9	3.8	8	1.7	1.9	0.02
c_3	0	2	1	0	1	2.5	4.1	8	2.2	2.5	0.01



(a) Step 1: preference submission

(b) Step 2: user set reduction

(c) Step 3: grouping and Setting padding

(d) Step 4: matching

(e) Step 5: determining payment

Fig. 3: Matching steps in C-QMECS mechanism through an example

bidding higher than (or equal to) this critical payment ($p_i^d \geq \hat{p}_i^d$) is a wining declaration and bidding lower than that ($p_i^d < \hat{p}_i^d$) is a losing declaration. In our proposed C-QMECS mechanism, the payment of each matched user is calculated using Equation (10), where each winning user pays $\hat{p}_i^d$ calculated using sensitivity analysis of the optimal solution to $\texttt{PADDING}(\tilde{I}, J, \boldsymbol{Q}^u)$. We need to show that $\hat{p}_i^d$ is the minimum price that matched user i should pay in order to acquire her requested bundle in the optimal solution of $\texttt{PADDING}(\tilde{I}, J, \boldsymbol{Q}^u)$. Note that the cost of vm_k for user i is determined by the lowest $\lceil \sum_{i \in \tilde{I}} q_{ik}^d x_i + q_{ik}^u \rceil$th asking price of vm_k among qualified cloudlets. Similarly, in the optimal solution of $\texttt{SELECT}(\hat{I}, J)$, the cost of vm_k for user i is equal to $\lceil \sum_{i \in \hat{I}} q_{ik}^d x_i \rceil$th asking price of vm_k among qualified cloudlets. Since $\lceil \sum_{i \in \tilde{I}} q_{ik}^d x_i + q_{ik}^u \rceil \geq \lceil \sum_{i \in \hat{I}} q_{ik}^d x_i \rceil$, therefore, the costs under $\texttt{SELECT}(\hat{I}, J)$ are lower than those under $\texttt{PADDING}(\tilde{I}, J, \boldsymbol{Q}^u)$. Thus, we can conclude that when user i bids higher than $\hat{p}_i^d$ and acquires her requested bundle in $\texttt{PADDING}(\tilde{I}, J, \boldsymbol{Q}^u)$, she can also acquire her requested bundle in $\texttt{SELECT}(\hat{I}, J)$. Therefore, when user i bids higher than (or equal to) $\hat{p}_i^d$, she is selected for trading. Conversely, for any bid price p_i^d lower than $\hat{p}_i^d$ (i.e., $p_i^d < \hat{p}_i^d$), user i does not win in the optimal solution to $\texttt{PADDING}(\tilde{I}, J, \boldsymbol{Q}^u)$, and subsequently cannot participate in the trading. Therefore, payment $\hat{p}_i^d$ by each winning user i can be simply considered as her critical payment.

To satisfy monotonicity, we need to show that any matched user i by declaring a bidding price higher than its submitted bidding price (p_i^d) will still be matched. This can be shown easily since user i wins in the matching by bidding any price higher than her critical price. If any user i has already won in the matching, we can conclude that $p_i^d \geq \hat{p}_i^d$. Any price higher than p_i^d will still satisfy this constraint, and therefore user i remains a winning user in the matching.

Therefore, bidding truthfully is always the dominant strategy for all users in C-QMECS.

□

Theorem 2. C-QMECS mechanism is individual rational (IR) for all users.

Proof: If user i wins, she pays her critical payment $\hat{p}_i^d$, which is always less than or equal to her own valuation. Conversely, if user i loses, she pays nothing. Therefore, in both cases, the utility of a participating user is non-negative. We thus can conclude that C-QMECS is IR for all users. □

Theorem 3. C-QMECS mechanism induces an integral and feasible allocation.

Proof: To prove that the outcome of C-QMECS is an integral and feasible allocation, we need to show that each user either acquires her whole requested bundle or nothing. In addition, each cloudlet allocates discrete units of VMs. According to Theorem 1, if user $i \in \hat{I}$, then she wins and acquires her requested bundle. However, if $i \notin \hat{I}$, she does not acquire anything. Therefore, a discrete number of VMs are allocated to users. To determine the VMs provided by cloudlets, the payment procedure of C-QMECS is based on perturbation technique, where there is always a unique way to sort cloudlets from low to high according to their offered prices. Thus, a discrete number of VMs are allocated by each cloudlet. This concludes the feasibility of the obtained solutions by C-QMECS. □

To prove that C-QMECS is truthful for all cloudlets, we first need to prove the following lemmas.

Lemma 1. For any user $i \in \hat{I}$, $x_i = 1$ in the optimal solution to $\texttt{SELECT}(\hat{I}, J/\{j\})$ for any $j \in J$, considering the defined padding $\boldsymbol{Q}^p$.

Proof: We have the following two possible cases:

Case 1: Cloudlet j does not allocate any VMs in the optimal solution to $\texttt{SELECT}(\hat{I}, J)$. In this case, it is clear that removing cloudlet j will not affect the final matching result.

Case 2: Cloudlet j allocates some or all of its VMs in the optimal solution to $\texttt{SELECT}(\hat{I}, J)$. Again, removing cloudlet j cannot affect the users whose QoS requirements are not satisfied by cloudlet j. However, we show that for other winning users, we still have $x_i = 1$ in the optimal solution to $\texttt{SELECT}(\hat{I}, J/\{j\})$. To prove this, we can consider removing cloudlet j as introducing a padding equal to the number of VMs offered by cloudlet j. We denote the imposed padding by $\boldsymbol{Q}^j$. According to our proposed C-QMECS mechanism, we have $\boldsymbol{Q}^j \leq \boldsymbol{Q}^p$. Therefore, each user $i \in \hat{I}$ will also win in the optimal solution to $\texttt{SELECT}(\hat{I}, J/\{j\})$. This is due to the fact that all users in $\hat{I}$ have been matched under a more competitive environment, where $\boldsymbol{Q}^p$ was employed. Therefore, they can definitely be matched in a less competitive environment, where a padding of size $\boldsymbol{Q}^j$ is imposed. □

Lemma 2. Equation (11) is equivalent to Equation (12).

Proof: According to Lemma 1, any user $i \in \hat{I}$ will always win in the matching, no matter if any cloudlet (j') participates or not. Thus, for any $vm_k \in VM$, we always have $Y_k = \sum_{j\in J} y'_{jk} = \sum_{j\in J/\{j'\}} y''_{jk} = \sum_{i\in \hat{I}} q^d_{ik}$, where y'_{jk} and y''_{jk} belong to the optimal solutions of $\texttt{SELECT}(\hat{I}, J)$ and $\texttt{SELECT}(\hat{I}, J/\{j'\})$, respectively. Now, suppose cloudlet j' enters the matching again. As a results, the offered VMs by cloudlet j' will replace VMs of other cloudlets with higher prices in the matching. To further explain this, we consider the following two possible cases for each $vm_k \in VM$:

Case 1: If $p^c_{j'k} > c^k_{-j'}[Y_k]$, then we have $y_{j'k} = 0$. In this case no VMs will be replaced by cloudlet j' because its prices are higher compared to previously matched VMs.

Case 2: If $p^c_{j'k} < c^k_{-j'}[Y_k - r + 1]$, then we have $y_{j'k} = r$, where $r \in \{1, 2, ..., q^c_{j'k}\}$. In this case, r VM instances of type vm_k offered by cloudlet j' are cheaper than r instances of previously matched VMs. Therefore, r VM instances of type vm_k will be replaced by cloudlet j'.

Remember that Y_k is the number of allocated VMs by other qualified cloudlets in the absence of j'. Therefore, the change of social welfare can be calculated as follows:

$$\tilde{V}(\hat{I}, J) - \tilde{V}(\hat{I}, J/\{j'\}) = \sum_{vm_k\in VM} \sum_{r=1}^{y_{j'k}} c^k_{-j'}[Y_k - r + 1] - \sum_{vm_k\in VM} p^c_{j'k} y_{j'k}.$$

Based on Equation (11), we simply conclude the following equation, which is Equation (12):

$$\pi^c_{j'} = \sum_{vm_k\in VM} \pi^c_{j'k} = \sum_{vm_k\in VM} \sum_{r=1}^{y_{j'k}} c^k_{-j'}[Y_k - r + 1]$$

□

Theorem 4. C-QMECS mechanism is truthful (IC) for all cloudlets.

Proof: A truthful mechanism needs to satisfy the critical payment property and monotonicity. We consider one single type of VM (e.g., vm_k), and the proof can be simply generalized to all types of VMs.

To satisfy critical payment, we need to show that there exists a unique value π^c_{jk} for each matched cloudlet j and vm_k, where submitting an asking price lower than (or equal to) this critical price ($p^c_{jk} \leq \pi^c_{jk}$) is a winning declaration, and higher than that ($p^c_{jk} > \pi^c_{jk}$) is a losing deceleration. In our proposed C-QMECS mechanism, the revenue of each winning cloudlet is calculated using equation (12). We consider the following two cases:

Case 1: If $p^c_{jk} < c^k_{-j}[Y_k - r + 1]$, where $r \in \{1, 2, ..., q^c_{jk}\}$, the asking price of cloudlet j for VM type vm_k is less than r VM instances of type vm_k that are allocated in its absence. Therefore, as $\texttt{SELECT}(\hat{I}, J)$ aims to maximize the obtained social welfare by matching low priced VMs to higher bids, cloudlet j succeeds to allocate r VM instances of type vm_k.

Case 2: If $p^c_{jk} > c^k_{-j'}[Y_k]$, the price of all allocated VMs of type vm_k in the absence of cloudlet j, is less than the asking price of cloudlet j. Thus, in this extreme case, r is equal to zero, and cloudlet j cannot allocate any of its VMs.

Therefore, π^c_{jk} is considered as the critical price for cloudlet j.

To satisfy monotonicity, as long as $p^c_{jk} < c^k_{-j}[Y_k - r + 1]$, cloudlet j can allocate r instances of its VMs of type vm_k. Thus, if cloudlet j can allocate r VM instances by its current asking price of p^c_{jk}, it will also win at least r instances of VMs by asking any price less than that.

Therefore, truthfulness is always the dominant strategy for all cloudlets.

□

Theorem 5. C-QMECS mechanism is individual rational (IR) for all cloudlets.

Proof: According to Equation (12), the utility of any cloudlet j can be calculated as follows:

$$U^c_j = \sum_{vm_k\in VM} \sum_{r=1}^{y_{jk}} (c^k_{-j}[Y_k - r + 1] - p^c_{jk}) \qquad (13)$$

Since the revenue of any matched cloudlet is always higher than its asking price, it can always achieve a non-negative utility. In other words, any winning cloudlet will receive a payment equal to the asking price of winning cloudlets in its absence, which are higher than its own asking price. Obviously, a losing cloudlet receives nothing, and thus its utility is equal to zero. Therefore, it is guaranteed that all cloudlets achieve non-negative utility. Thus, we can conclude that C-QMECS is IR for all cloudlets. □

TABLE 3: Statistics of datasets with different workloads.

Dataset	# of users	# of cloudlets	# of requested VMs by each user				# of offered VMs by each cloudlet				Description
			vm_1	vm_2	vm_3	vm_4	vm_1	vm_2	vm_3	vm_4	
Dataset-0	[180,200]	[70,80]	[1,10]	[1,10]	[1,10]	[1,10]	[1,50]	[1,50]	[1,50]	[1,50]	Very high supply
Dataset-1	[180,200]	[60,70]	[1,10]	[1,10]	[1,10]	[1,10]	[1,50]	[1,50]	[1,50]	[1,50]	High supply
Dataset-2	[180,200]	[30,40]	[1,10]	[1,10]	[1,10]	[1,10]	[1,50]	[1,50]	[1,50]	[1,50]	High demand
Dataset-3	[180,200]	[20,30]	[1,10]	[1,10]	[1,10]	[1,10]	[1,50]	[1,50]	[1,50]	[1,50]	Very high demand
Dataset-4	[180,200]	[80,100]	[4,5]	[4,5]	[4,5]	[4,5]	[9,10]	[9,10]	[9,10]	[9,10]	Moderate supply-demand (small padding size)
Dataset-5	[180,200]	[16,20]	[4,5]	[4,5]	[4,5]	[4,5]	[45,50]	[45,50]	[45,50]	[45,50]	Moderate supply-demand (medium padding size)
Dataset-6	[180,200]	[8,10]	[4,5]	[4,5]	[4,5]	[4,5]	[90,100]	[90,100]	[90,100]	[90,100]	Moderate supply-demand (large padding size)
Dataset-7	[180,200]	[4,5]	[4,5]	[4,5]	[4,5]	[4,5]	[180,200]	[180,200]	[180,200]	[180,200]	Moderate supply-demand (very large padding size)

TABLE 4: Statistics of datasets with different QoS requirements and guarantees.

	requested by each user			guaranteed by each cloudlet			
Dataset	ART	MRT	RTF	ART	MRT	RTF	Description
Dataset-8	[0.6,1]	[1.1,1.5]	[0.11%,0.15%]	[0.25,0.65]	[0.75,1.15]	[0.075%,0.115%]	Good QoS guarantees
Dataset-9	[0.6,1]	[1.1,1.5]	[0.11%,0.15%]	[0.33,0.73]	[0.83,1.23]	[0.083%,0.123%]	Fair QoS guarantees
Dataset-10	[0.6,1]	[1.1,1.5]	[0.11%,0.15%]	[0.4,0.8]	[0.9,1.3]	[0.09%,0.13%]	Bad QoS guarantees

TABLE 5: Bid and ask prices distribution.

	m5.2xlarge	m5.4xlarge	m5.12xlarge	m5.24xlarge
IoT users	[0.416,0.48]	[0.832,0.96]	[2.496,2.88]	[4.992,5.76]
Cloudlets	[0.384,0.448]	[0.768,0.896]	[2.304,2.688]	[4.608,5.376]

Theorem 6. C-QMECS is weakly budget balanced (BB).

Proof: In order to prove that C-QMECS is weakly BB, we need to show that the payments of users are always greater than or equal to the payments received by cloudlets. We first determine the lower bound on the payments of the users, and then show that it never goes beyond the upper bound on the payments received by the cloudlets.

For the user side, based on the definition of $\hat{p}_i^d$, as long as $p_i^d \geq \hat{p}_i^d$ holds for any user $i \in I$, she remains in the winning user set and subsequently $\hat{I}$ remains unchanged. Therefore, according to the mechanism, for each winning user i we have $p_i^d \geq \sum_{vm_k \in VM} q_{ik}^d \times c^k[\sum_{i \in I}(q_{ik}^d x_i + q_{ik}^u)] \geq \sum_{vm_k \in VM} q_{ik}^d \times c^k[\sum_{i \in \hat{I}}(q_{ik}^d + q_{ik}^u)]$. Clearly, as long as p_i^d approaches $\hat{p}_i^d$, term $\sum_{vm_k \in VM} q_{ik}^d \times c^k[\sum_{i \in \hat{I}}(q_{ik}^d + q_{ik}^u)]$ remains unchanged, as $\hat{I}$ remains the same. Consequently, we have $\hat{p}_i^d \geq \sum_{vm_k \in VM} q_{ik}^d \times c^k[\sum_{i \in \hat{I}}(q_{ik}^d + q_{ik}^u)]$. Finally, in the optimal solution to the matching problem, the payment of all users is not less than the total payments, when $\sum_{i \in \hat{I}} q_{ik}^d$ instances of VM type $vm_k \in VM$ are traded at unit price $c^k[\sum_{i \in \hat{I}}(q_{ik}^d + q_{ik}^u)]$ for any $vm_k \in VM$.

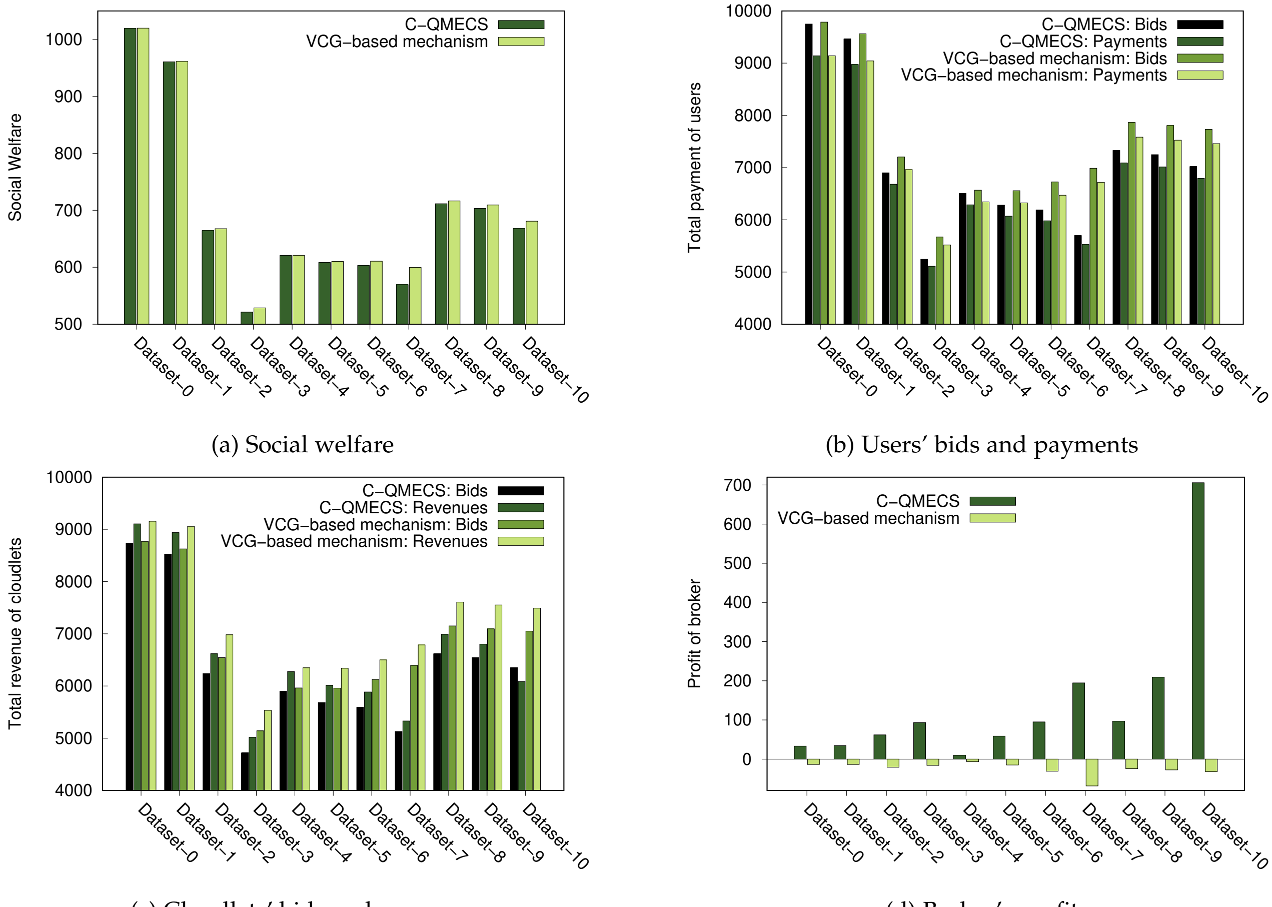
For the cloudlet side, according to Lemma 1, the maximum payment per VM instance of type vm_k is equal to $c_{-j}^k[\sum_{i \in \hat{I}} q_{ik}^d]$. Since c_{-j}^k and c^k are both monotonically increasing functions and $q_{ik}^u = \max_{j \in J}\{q_{jk}^c\}$ for the delegate user, therefore $c_{-j}^k[\sum_{i \in \hat{I}} q_{ik}^d] \leq c^k[\sum_{i \in \hat{I}}(q_{ik}^d + q_{ik}^u)]$. Therefore, the payments received by all cloudlets are not more than the total payment when $\sum_{i \in \hat{I}} q_{ik}^d$ instances of VM type $vm_k \in VM$ are allocated at unit price of $c^k[\sum_{i \in \hat{I}}(q_{ik}^d + q_{ik}^u)]$. As a result, the payments received by the cloudlets never exceed the payments paid by the users, which consequently results into a nonnegative payoff for the broker. Therefore, the proposed C-QMECS mechanism is weakly budget-balance. □

Theorem 7. C-QMECS is asymptotically allocative efficient as the number of available VMs of each type grows to infinity, given bounded cost distributions for all VM types.

Proof: We investigate the efficiency of the allocation as the capacity of cloudlets increases (higher number of VMs becomes available). Without loss of generality, we assume that the demand remains fixed to show when the ratio of supply to demand grows to infinity, C-QMECS is asymptotically allocative efficient.

We assume that the distribution of the reported asking prices of cloudlets for each $vm_k \in VM$ is $[\underline{p}_k, \overline{p}_k]$. Then, for any user i, while $p_i^d < \sum_{vm_k \in VM} q_{ik}^d \underline{p}_k$, user i can never win in neither C-QMECS nor the optimal solution, no matter how much the supply (number of VMs) increases. However, if $p_i^d > \sum_{vm_k \in VM} c^k[\sum_{i \in I}(q_{ik}^d + q_{ik}^u)]q_{ik}^d > \sum_{vm_k \in VM} q_{ik}^d \underline{p}_k$, user i wins in both C-QMECS and the optimal solution. This is due to the fact that $p_i^d > \sum_{vm_k \in VM} c^k[\sum_{i \in I}(q_{ik}^d + q_{ik}^u)]q_{ik}^d$ and $p_i^d > \sum_{vm_k \in VM} c^k[\sum_{i \in I} q_{ik}^d]q_{ik}^d$ hold, respectively. As the supply for all VM types increases, the number of VMs with cheaper costs increases as well. Therefore, $c^k[\sum_{i \in I}(q_{ik}^d + q_{ik}^u)]$ will approach $\underline{p}_k$ for all $vm_k \in VM$. Therefore, the obtained social welfare will converge to the optimal social welfare as the number of VMs increases, and thus C-QMECS can asymptotically achieve allocative efficiency. □

Finally, C-QMECS is computationally efficient (CE) and highly scalable. The implementation of C-QMECS involves polynomial-time computations that depend on $|I|$, $|J|$, K, and L.



(a) Social welfare

(b) Users' bids and payments

(c) Cloudlets' bids and revenues

(d) Broker's profit

Fig. 4: C-QMECS versus the VCG-based mechanism in terms of economic efficiency

5 Experimental Results

Our proposed C-QMECS mechanism is IC, IR, BB, CE, and asymptotically AE. In addition, we perform extensive experiments to evaluate the performance of our proposed mechanism.

5.1 Experimental Setup

We created several datasets, each with specific features, to evaluate the performance of our proposed C-QMECS mechanism under different workloads (Table 3) and QoS requirements and guarantees (Table 4). We considered four types of VMs and generated the bidding and asking prices based on the prices of m5.2xlarge, m5.4xlarge, m5.12xlarge, and m5.24xlarge VM instances of Amazon EC2 at different regions in the U.S. Table 5 shows the uniform distribution of the prices for each VM type.

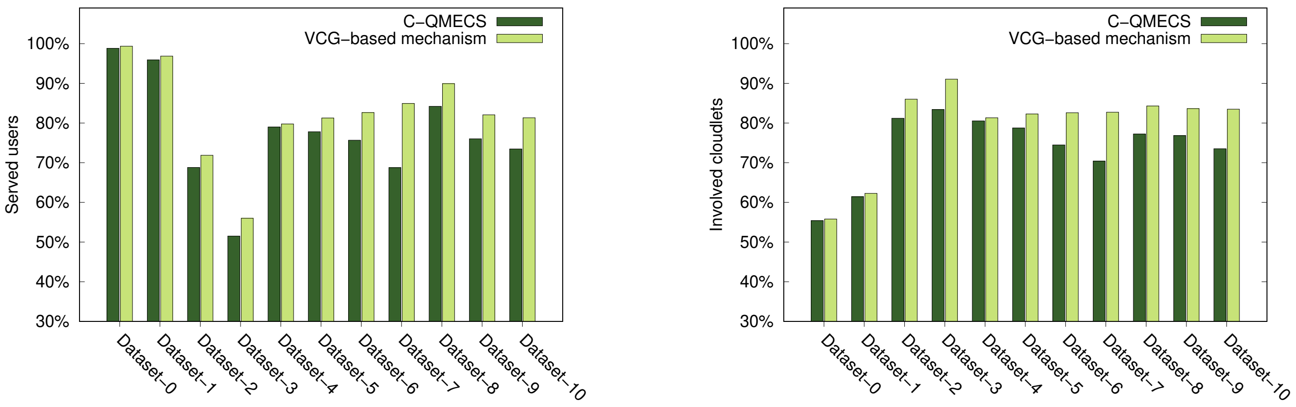
We investigate the performance of our proposed mechanism versus the socially-optimal strategy-proof VCG-based mechanism in terms of social welfare, percentage of matched users, percentage of utilized resources, users' payments, cloudlets' revenues, broker's profit, and run time efficiency. In the VCG-based mechanism, the winner determination is obtained by solving $\texttt{Matching}(I, J)$ MIP optimally, and the trading prices are computed based on participant's marginal contribution to social welfare.

In addition, we evaluate the performance of our proposed C-QMECS mechanism under different QoS requirements and guarantees (Dataset-8 to Dataset-10) versus Combinatorial MECS (C-MECS) mechanism in terms of matched users. The C-MECS mechanism is based on C-QMECS mechanism, where C-MECS applies the matchings without considering QoS requirements and guarantees. Therefore, there may be users whose QoS requirements are not satisfied by the matched cloudlets. In this particular experiment, supply and demand are fixed (assuming 250 users and 10 cloudlets) to better evaluate the performance of C-QMECS under variant QoS requirements and guarantees. Each user requests for 4 VMs of each type, while each cloudlet offers 100 VMs of each type.

We used Java and ILOG Concert Technology for the simulations. The experiments are conducted on an Intel 2.3GHz Core i3 system with 4 GB RAM.

5.2 Analysis of Results

Fig. 4a shows the realized social welfare by C-QMECS and the optimal social welfare obtained by the VCG-based mechanism. The results show when supply is higher than demand (e.g., Dataset-0), the obtained social welfare is almost equal to the optimal value. However, when the ratio of supply to demand is low (e.g., Dataset-3), the difference between the obtained social welfare by C-QMECS and the



(a) Percentage of matched users

(b) Percentage of utilized resources

Fig. 5: C-QMECS versus the VCG-based mechanism in terms of matching efficiency

VCG-mechanism increases. This is due to the fact that when supply is high, there will be still enough resources for the users after adding the phantom user. However, when supply is low, the addition of the phantom user may not leave enough resources for the IoT users. Therefore, more users are eliminated due to adding the phantom user with a proportional padding size. As for the datasets with moderate supply-demand ratios, in the case of small, medium, and even large padding sizes (Dataset-4 to Dataset-6), the obtained social welfare is very close to the optimal social welfare. However, as the padding size approaches to the extreme case (Dataset-7) where only few cloudlets with very large amount of available resources exist, more users are eliminated in the matching step. This in turn results in a decrease in the obtained social welfare. Under variant QoS requirements and guarantees, the obtained social welfare is higher when the percentage of qualified cloudlets for the users is higher (e.g., Dataset-8). One important point that can be seen from Fig. 4a is the asymptotic AE of C-QMECS. As the number of available VMs grows (e.g., Dataset-3 to Dataset-0), the obtained social welfare of C-QMECS reaches to the optimal social welfare (Theorem 7).

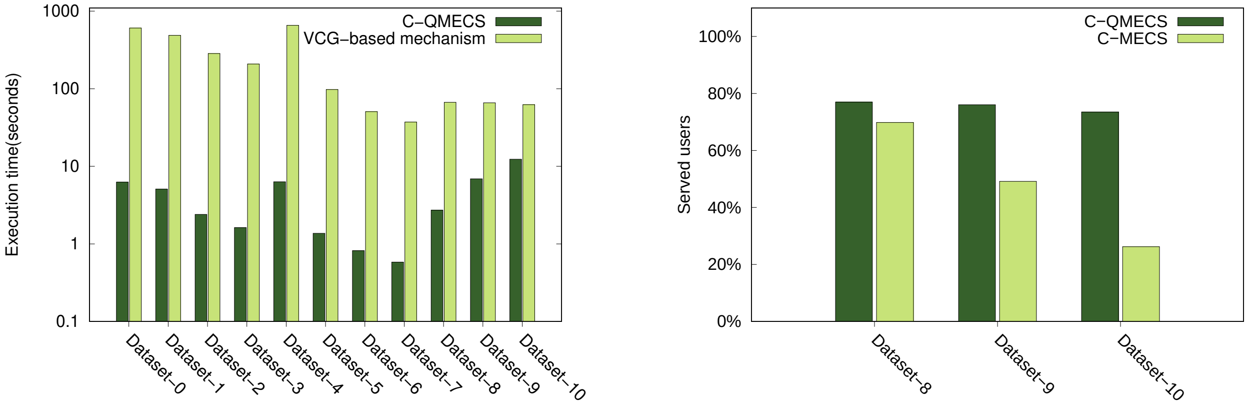
According to Fig. 4b, the total payments and bids strongly depend on the number of successful IoT-cloudlet matchings, and thus these values are higher when supply to demand ratio is high (e.g., Dataset-0). The difference between payments in C-QMECS and VCG-based mechanism in the case of low supply-demand ratio is because of the smaller number of matchings in C-QMECS (e.g., Dataset-3). Similarly, when the padding size grows in the moderate supply-demand cases, the total payments of users decreases (e.g., Dataset-7) due to decreasing number of successful matchings in C-QMECS. Moreover, a reduction in QoS guarantees of cloudlets decreases the number of matchings as well. Therefore, the total payments of users decreases as the number of qualified cloudlets decreases in the system (e.g., Dataset-10). The obtained results are consistent with our theoretical analysis in Theorem 2, which shows that C-QMECS is individually rational for users as the payments are always less than bids.

Similarly, as it can be seen in Fig. 4c, the higher involvement rate of cloudlets in the matching step (e.g., Dataset-0) leads to more revenue for the cloudlets. This figure also validates our theoretical results of Theorem 5.

Fig. 4d illustrates the broker's utility under various workloads and QoS requirements and guarantees. The results show that compared to the VCG-based mechanism, the broker's utility in C-QMECS is always greater than zero, which practically shows that C-QMECS is budget balanced (Theorem 6). Note that the VCG-based mechanism is not budget balance and thereby cannot ensure the profitability for the broker.

According to Fig. 5a, for cases with higher supply to demand ratio (e.g., Dataset-0), users have higher chances to acquire their requested services, and thus the percentage of served users is higher as well. However, when the supply is low, several users are eliminated due to the addition of the phantom user, and they miss their chances for winning their requested bundles. Therefore, a lower percentage of users are served in C-QMECS compared to the VCG-based mechanism (e.g, Dataset-3). As for datasets with moderate supply to demand ratio, the percentage of matched users highly depends on the padding size. That is, the higher the padding size, the lower the matching rates of users (e.g., Dataset-7). This is also because more users are eliminated in the matching step due to the larger padding size. For datasets with variant QoS requirements and guarantees, as we expect the percentage of matched users decreases with the reduction of QoS guarantees by the cloudlets (e.g., Dataset-10).

Fig. 5b demonstrates the percentage of utilized resources of cloudlets. When demand is low, only a part of available resources is sufficient for serving the existing demand (e.g. Dataset-0). However, as demand increases, more resources are used to fulfill the demand (Dataset-0 to Dataset-3). Moreover, when ratio of supply to demand is low (e.g, Dataset-3), the difference between the amount of resource utilization in C-QMECS and VCG-based mechanism increases. It is because some of the resources are assumed to be allocated to the phantom user, and the remaining users have less available resources. This also holds for the case of a large padding size in the moderate supply-demand case (e.g., Dataset-7), where more resources are allocated to the phantom user. This consequently results in the elimination of a part of



(a) Execution time (matching + payments)

(b) Percentage of satisfied users after matching

Fig. 6: C-QMECS other properties

users in the matching step, leading to reduced resource utilization. Finally, when the number of existing qualified cloudlets for users decreases (Dataset-8 to Dataset-10), a lower number of cloudlets can satisfy the QoS requirements of users. Therefore, a lower number of cloudlets can allocate their resources leading to decreased resource utilization.

Fig. 6a shows the running time of the mechanisms in logarithmic scale. The results show that our proposed C-QMECS mechanism has a lower execution time compared to the VCG-based mechanism, especially when the number of cloudlets is higher (e.g., Dataset-0).

Finally, Fig. 6b shows the efficiency of our proposed mechanism versus C-MECS in edge computing due to considering QoS metrics in the matchings. The results show that the consideration of QoS metrics in the matchings of C-QMECS improves user satisfaction and quality of experience significantly. The results also show that considering QoS in the matching is more critical when the QoS guarantees are low (Dataset-9 and Dataset-10).

To sum up, our proposed C-QMECS mechanism shows a better performance as the number of cloudlets increases, the padding size decreases, and the QoS guarantees improve. It also satisfies all the critical properties, including IC, IR, BB, and CE. It demonstrates an acceptable level of allocative efficiency as it ensures asymptotic AE. We thereby conclude that C-QMECS is a suitable QoS-aware two-sided matching mechanism for edge computing considering the incentives and preferences of IoT users, cloudlets, and the broker.

6 Conclusion

With rapid changes in the QoS requirements of IoT devices and emerging new paradigms such as edge computing, efficient QoS-aware matching algorithms are needed to match cloudlets to IoT applications when providing edge computing services. In this paper, we addressed this challenge by proposing a well-designed two-sided matching solution, C-QMECS, for edge services considering QoS requirements in terms of service response time. C-QMECS enhances QoE of users by matching IoT users to those cloudlets which can guarantee their required QoS. In addition, we proposed payment determination solutions considering preferences and incentives of cloudlets, IoT users, and the system. C-QMECS is also combinatorial as it allows users and cloudlets to request for and offer a variety of VMs. The proposed matching mechanism is incentive compatible, individually rational, weakly budget balanced, asymptotically allocative efficient and computationally efficient. The experimental results showed that C-QMECS is suitable for edge computing services by providing close to optimal matching, efficient pricing, and guaranteed QoS. For future work, we plan to extend our model and mechanism to handle mobility of IoT users and cloudlets and to support dynamic changes such as online changes in QoS requirements and pricings. Finally, we will investigate distributed solutions to design a federated incentive-compatible matching mechanism with incomplete information.

Acknowledgments

This paper is a revised and extended version of [30] presented at the the 37th IEEE International Performance Computing and Communications Conference. This research was supported in part by INSF and University of Tabriz grant 97013453, and NSF grant CNS-1755913.

References

[1] Statista. [Online]. Available: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/

[2] E. Balevi and R. D. Gitlin, "Unsupervised machine learning in 5g networks for low latency communications," in *Proc. of the 36th IEEE International Performance Computing and Communications Conference (IPCCC)*, 2017, pp. 1–2.

[3] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[4] W. Shi and S. Dustdar, "The promise of edge computing," *Computer*, vol. 49, no. 5, pp. 78–81, 2016.

[5] S. Singh and I. Chana, "QoS-aware autonomic resource management in cloud computing: a systematic review," *ACM Computing Surveys*, vol. 48, no. 3, p. 42, 2016.

[6] G. Portaluri, D. Adami, A. Gabbrielli, S. Giordano, and M. Pagano, "Power consumption-aware virtual machine placement in cloud data center," *IEEE Transactions on Green Communications and Networking*, vol. 1, no. 4, pp. 541–550, 2017.

[7] 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*, vol. 65, no. 3, pp. 805–818, 2016.

[8] W. Shi, L. Zhang, C. Wu, Z. Li, F. Lau, W. Shi, L. Zhang, C. Wu, Z. Li, and F. Lau, "An online auction framework for dynamic resource provisioning in cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 4, pp. 2060–2073, 2016.

[9] Amazon, "Amazon ec2 spot instances," [Online] Available: https://aws.amazon.com/ec2/spot/pricing/, 2019, accessed: 2019-06-24.

[10] L. Mashayekhy, M. Nejad, and D. Grosu, "A PTAS mechanism for provisioning and allocation of heterogeneous cloud resources," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 9, pp. 2386–2399, 2015.

[11] M. Nejad, L. Mashayekhy, and D. Grosu, "Truthful greedy mechanisms for dynamic virtual machine provisioning and allocation in clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 2, pp. 594 – 603, 2015.

[12] L. Mashayekhy, M. Nejad, and D. Grosu, "Physical machine resource management in clouds: A mechanism design approach," *IEEE Transactions on Cloud Computing*, vol. 3, no. 3, pp. 247–260, 2015.

[13] 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*, vol. 65, no. 4, pp. 1172–1184, 2016.

[14] W. Ma, X. Liu, and L. Mashayekhy, "A strategic game for task offloading among capacitated UAV-mounted cloudlets," in *Proceedings of the IEEE International Congress on Internet of Things*, 2019, pp. 1–8.

[15] L. Mashayekhy, M. Nejad, and D. Grosu, "A trust-aware mechanism for cloud federation formation," *IEEE Transactions on Cloud Computing*, pp. 1–15, 2019.

[16] D. Kumar, G. Baranwal, Z. Raza, and D. P. Vidyarthi, "A systematic study of double auction mechanisms in cloud computing," *Journal of Systems and Software*, vol. 125, pp. 234–255, 2017.

[17] B. Shi, J. Wang, Z. Wang, and Y. Huang, "Trading web services in a double auction-based cloud platform: A game theoretic analysis," in *Proc. of the IEEE International Conference on Services Computing*, 2017, pp. 76–83.

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

[19] K. Zhang, S. Leng, Y. He, S. Maharjan, and Y. Zhang, "Cooperative content caching in 5g networks with mobile edge computing," *IEEE Wireless Communications*, vol. 25, no. 3, 2018.

[20] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *Proc. of the 2016 IEEE Intl. Symp. on Information Theory*, 2016, pp. 1451–1455.

[21] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Transactions on Wireless Communications*, vol. 12, no. 9, pp. 4569–4581, 2013.

[22] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, no. 5, pp. 2795–2808, 2016.

[23] A. Kiani and N. Ansari, "Toward hierarchical mobile edge computing: An auction-based profit maximization approach," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 2082–2091, 2017.

[24] T. Bahreini, H. Badri, and D. Grosu, "An envy-free auction mechanism for resource allocation in edge computing systems," in *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, Oct 2018, pp. 313–322.

[25] G. Gao, M. Xiao, J. Wu, H. Huang, S. Wang, and G. Chen, "Auction-based vm allocation for deadline-sensitive tasks in distributed edge cloud," *IEEE Transactions on Services Computing*, pp. 1–1, 2019.

[26] Z. Li, Z. Yang, and S. Xie, "Computing resource trading for edge-cloud-assisted internet of things," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2019.

[27] A. Zavodovski, S. Bayhan, N. Mohan, P. Zhou, W. Wong, and J. Kangasharju, "DeCloud: Truthful decentralized double auction for edge clouds," in *Proc. of the IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, 2019, pp. 2157–2167.

[28] S. K. Garg, S. Versteeg, and R. Buyya, "A framework for ranking of cloud computing services," *Future Generation Computer Systems*, vol. 29, no. 4, pp. 1012–1023, 2013.

[29] R. B. Myerson and M. A. Satterthwaite, "Efficient mechanisms for bilateral trading," *Journal of economic theory*, vol. 29, no. 2, pp. 265–281, 1983.

[30] N. Sharghivand, F. Derakhshan, and L. Mashayekhy, "QoS-aware matching of edge computing services to internet of things," in *Proceedings of the 37th IEEE International Performance Computing and Communications Conference*, 2018, pp. 1–8.

BIOGRAPHIES

Nafiseh Sharghivand received the B.Sc. and M.Sc. degrees in computer engineering from University of Tabriz, Tabriz, Iran. She is currently pursuing her Ph.D. degree in computer engineering at University of Tabriz. Her research interests include cloud and edge computing, Internet of Things, game theory, mechanism design, and multiagent systems.

Farnaz Derakhshan is an assistant professor in the Department of Computer Engineering at University of Tabriz. She received her PhD degree in Computer Engineering from Liverpool University in 2008. Her research interests include multiagent systems, game theory, and Internet of Things.

Lena Mashayekhy is an assistant professor in the Department of Computer and Information Sciences at the University of Delaware. Her research interests include edge/cloud computing, data-intensive computing, Internet of Things, and algorithmic game theory. Her doctoral dissertation received the 2016 IEEE TCSC Outstanding PhD Dissertation Award. She is also a recipient of the 2017 IEEE TCSC Award for Excellence in Scalable Computing for Early Career Researchers. She has published more than forty peer-reviewed papers in venues such as IEEE Transactions on Parallel and Distributed Systems and IEEE Transactions on Cloud Computing. She is a member of the IEEE and ACM.

Leyli Mohammadkhanli is an associate professor in the Department of Computer Engineering at University of Tabriz. She received her PhD degree in Computer Engineering from Iran University of Science and Technology in 2007. Her research interests include cloud computing, computer networks, and big data.