Auction-based VM Allocation for Deadline-Sensitive Tasks in Distributed Edge Cloud

Guoju Gao, Mingjun Xiao, *IEEE Member*, Jie Wu, *IEEE Fellow*,
He Huang, *IEEE Member*, Shengqi Wang, and Guoliang Chen

Abstract—Edge cloud computing is a new paradigm in which the computation and storage services of remote cloud data centers are moved to Edge Cloud Nodes (ECNs) in network edges. Compared to traditional cloud data centers, ECNs are geographically close to mobile users so the communication latency is significantly reduced. In this paper, we study the problem of allocating Virtual Machine (VM) resources in geo-distributed ECNs to mobile users by using the auction theory. First, we treat mobile users and ECNs as the buyers and sellers of the VM resource auction, respectively. Then, we model the VM resource allocation problem as an n-to-one weighted bipartite graph matching problem with 0-1 knapsack constraints. Since this problem is NP-hard, we design a greedy approximation algorithm to determine the winners of the auction, based on which we propose a truthful Auction-based VM resource Allocation (AVA) mechanism to solve the problem. Moreover, we prove that the AVA mechanism not only achieves an approximately optimal solution for winner selection, but also has the properties of truthfulness, individual rationality, and computational efficiency. Finally, we conduct extensive simulations on real traces to verify the significant performances of the proposed AVA mechanism.

Index Terms—Auction mechanism, edge cloud, mobile cloud computing, virtual machine allocation.

1 Introduction

Recent years have witnessed the proliferation of mobile cloud computing, through which mobile users can migrate various applications from their smart devices to remote cloud data centers such as Amazon EC2 [4], Microsoft Azure [18], and so on. By using the resources (such as computation, storage, etc.) in clouds, these smart devices can break through resource limitations to complete complicated applications, significantly enriching their functionality. However, since mobile users are usually far away from remote cloud data centers, migrating applications will inevitably lead to a long communication delay as well as a heavy network load, severely downgrading mobile users' experience. To mitigate such negative influences, a new mobile cloud computing paradigm, called edge cloud [8, 13, 21, 24, 25], also known as fog computing or cloudlets [14, 15, 17, 32], is proposed. In the edge cloud paradigm, a number of small-scale computing and storage servers are placed at network edges to form some Edge Cloud Nodes (ECNs). Mobile users can directly access nearby ECNs so that the latency and network load can be reduced significantly.

A typical Mobile Edge Cloud (MEC) framework is composed of lots of mobile users, some ECNs, and a Centralized Cloud (CC). The ECNs, which are deployed by some cloud service providers, are distributed at network edges. The mobile users can connect to the ECNs via a wireless local area network [2, 11, 17], (also known as radio access network [2, 3, 31], wireless metropolitan area network [14, 24], etc.), while these ECNs connect to the remote CC through core network [2, 8] (also known as backbone/backhaul network [11, 13, 25], wide area network [8, 17, 31], etc.), as shown in Fig. 1. Compared to the remote CC, ECNs are geographically distributed and small-scale. Some mobile users will first send cloud service requests to an ECN. Then, if the ECN cannot provide cloud service for these mobile users, it may upload part of the requests to the CC via core network, which would inevitably incur a very long transmission delay.

Virtual Machine (VM) resource allocation is one of the most important issues in edge clouds [12, 13, 15, 22, 30]. Consider that a lot of mobile users have some cloud computing applications (i.e., *tasks*) that need to be dealt with. However, these users lack sufficient storage and computing resources. Thus, they need to rent VM resources from the ECNs. In order to ensure the quality of the cloud computing applications, the transmission delay of each task must be no larger than a deadline (called *deadline* constraints). Meanwhile, the ECNs would complete the allocated tasks only when the total VM resources requested by mobile users are no larger than the VM resource capacity of the ECNs (called *capacity* constraints). However, if the capacity constraint of an ECN cannot be satisfied, the ECN may upload part of the allocated tasks to the CC under the deadline constraints.

- *G. Gao, M. Xiao, S. Wang, and G. Chen are with the School of Computer Science and Technology / Suzhou Institute for Advanced Study, University of Science and Technology of China, Hefei, P. R. China.*
  Correspondence to: xiaomj@ustc.edu.cn
- *J. Wu is with the Department of Computer and Information Sciences, Temple University, 1805 N. Broad Street, Philadelphia, PA 19122.*
  E-mail: jiewu@temple.edu
- *H. Huang is with the School of Computer Science and Technology, Soochow University, Suzhou, China.*
  E-mail: huangh@suda.edu.cn

This research was supported in part by the National Natural Science Foundation of China (NSFC) (Grant No. 61872330, 61572457, 61379132, U1709217, 61303206, 61572342, 61502261), NSF grants CNS 1757533, CNS 1629746, CNS 1564128, CNS 1449860, CNS 1461932, CNS 1460971, IIP 1439672, the NSF of Jiangsu Province in China (Grant No. BK20131174, BK2009150), and Anhui Initiative in Quantum Information Technologies (Grant No. AHY150300). Additionally, the support provided by China Scholarship Council (CSC) during a visit of 'Guoju Gao' to Temple University (Grant CSC No. 201806340014) is acknowledged.

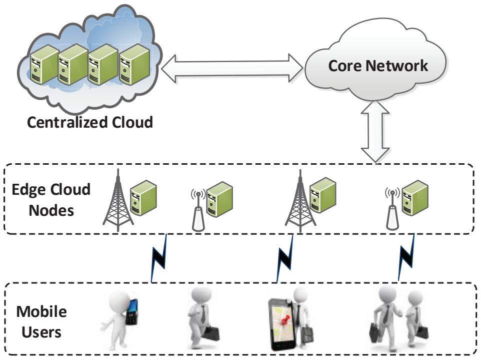


Fig. 1. An illustration of the mobile edge cloud computing framework.

In this paper, we focus on the VM resource allocation for deadline-sensitive cloud computing tasks in distributed edge cloud, attempting to avoid overwork and wasted resource in any of ECNs. Moreover, due to the competition for VM resources between users, we study the problem of allocating VM resources in ECNs to mobile users by using auction theory. Actually, it is very challenging to design the auction-based VM resource allocation mechanism in MEC. We summarize three major challenges.

Firstly, different from the traditional two-layer cloud structure, the mobile edge cloud is actually a three-layer structure, as shown in Fig. 1. When the requested VM resources exceed the capacity constraints of ECNs, the ECNs can upload part of the allocated tasks to the CC. That is to say, the CC can be seen as the ultimate resource pool for all ECNs. Thus, the VM resource allocation among distributed ECNs is a special multiple 0-1 knapsack problem. Especially, when involving the heterogeneous deadline constraints, such VM resource allocation problem in MEC is nontrivial.

Secondly, the bandwidths of ECNs are heterogeneous. Here, the bandwidths of an ECN include the bandwidth associated with the CC and the bandwidth allocated to mobile users. This indicates that the transmission delays of uploading these deadline-sensitive tasks to different ECNs are heterogeneous. Also, if a task is uploaded from an ECN to the CC, the transmission delay contains two parts: the time of transmitting the task from users to an ECN and the time of uploading the task from the ECN to the CC. Therefore, the transmission delay of a task is dynamic, which depends on the specific allocation decisions.

Thirdly, due to the heterogeneous resource configuration of ECNs including VM resource capacity, bandwidth, communication cost, etc, mobile users might have different preferences for each ECN. Thus, many users might compete for the limited resources in the same ECN. The users who lose the competition have to seek alternative resources from other ECNs, which results in competition for resources in other ECNs. Additionally, to ensure that each mobile user will not manipulate its bids (i.e., truthfulness) and each user's payoff is nonnegative (i.e., individual rationality), a proper auction mechanism for such VM resource competition across multiple geo-distributed ECNs is needed.

In fact, there have been some works devoted to the resource allocation and workload scheduling in edge clouds. For example, Tan *et al.* [24] proposed the OnDisc algorithm to dispatch and schedule jobs between ECNs and cloud data centers, so as to minimize the total response time; Jia *et al.* [14] investigated how to schedule tasks between multiple cloudlets, so as to minimize the maximum average response time. However, most of these works study the resource allocation problem in edge clouds mainly from the viewpoint of workload scheduling without considering the competition for resource requests from an economic point of view. Among the existing works, only a few studies [15, 22, 30] are close to our problem to a certain extent. More specifically, [15] proposes two truthful mechanisms to coordinate the resource auction between users and ECNs. However, this work simply assumes that all ECN resources are homogeneous, and it allows each ECN to serve only one request, which does not apply to real application scenarios. In contrast, [30] studies the problem of stimulating mobile device clouds to compete for mobile users' computation tasks. Essentially, it is a crowdsourcing-based task allocation problem, different from the resource allocation in ECNs. [22] designs an online auction mechanism for virtual cluster provisioning in geo-distributed clouds, which is modeled as an online combinational knapsack problem. This also differs from our VM allocation problem in distributed ECNs.

In this paper, we propose an Auction-based VM resource Allocation (AVA) mechanism to solve the problem of mobile users with deadline-sensitive tasks competing for VM resources in heterogeneous geo-distributed ECNs. In the AVA mechanism, mobile users and ECNs (including CC) are seen as the buyers and sellers of VM resource auction, respectively. The capacities of VM resources in ECNs are seen as heterogeneous knapsacks. Then, the VM resource allocation is modeled as *an n-to-one weighted bipartite graph matching problem with 0-1 knapsack constraints*, unlike existing cloud auction problems such as [22, 23, 34]. Since the problem is NP-hard, we adopt a greedy strategy to determine the winners of the auction so that the AVA mechanism can achieve a near-optimal social welfare performance.

More specifically, our major contributions are summarized as follows:

- We formalize the competitive VM resource allocation problem for deadline-sensitive tasks in a three-layer edge cloud structure and propose the Auction-based VM resource Allocation (AVA) mechanism, which mainly consists of a greedy winning bid selection algorithm and a payment determination algorithm.
- We prove that the winning bid selection problem of AVA is NP-hard. We first remove the deadline constraints, and then transform the three-layer edge cloud structure into a two-layer one. Based on this, we propose a greedy winning bid selection algorithm and further analyze its approximation ratio.
- We also design a truthful payment determination algorithm. Then, we prove that the AVA mechanism has the properties of truthfulness, individual rationality, and computational efficiency.
- We conduct extensive simulations on real traces to evaluate the performances of the proposed AVA mechanism. The results show that AVA not only achieves better social welfare performances than the compared algorithms, but also guarantees truthfulness, individual rationality, and computational efficiency.

The remainder of the paper is organized as follows. We first describe the auction model and problem formulation in Section 2. Then, we present the design of our mechanism and the theoretical analysis in Sections 3 and 4, respectively. In Section 5, we evaluate the performances of our mechanism through extensive simulations. After reviewing the related works in Section 6, we conclude the paper in Section 7.

2 Model & Problem Formulation

2.1 Model

Consider that some cloud service providers have deployed many ECNs and are willing to sell the extra VM resources in their ECNs. These ECNs connect to the remote CC via core network. On the other hand, a lot of mobile users wish to execute some deadline-sensitive cloud computing applications, but they cannot afford to deploy their own ECNs. In order to address such a problem with a low cost, these users can rent the VM resources of ECNs from cloud service providers. Different from the traditional distributed cloud computing scenario, ECNs can upload part of the allocated tasks to the CC if the required VM resources of tasks exceed the capacity of ECNs. However, this will inevitably incur a long transmission delay.

In this paper, we design a mechanism to support the trade of VM resources between cloud service providers and mobile users. In order to successfully run this mechanism, we first build a platform, which also acts as the auctioneer. The platform continuously receives deadline-sensitive computing requests from mobile users and collects ECNs' state information from the cloud service providers. Based on this, the platform makes the decision to allocate the deadline-sensitive computing tasks via auction. The auction is conducted periodically. The ECNs and mobile users are seen as the VM resource sellers and buyers, respectively. If a mobile user applies for VM resources that cover more than one auction cycle, it will submit multiple requests. Additionally, since the allocated VM resources will be solely occupied by the winning mobile users in each auction cycle, the execution time of a cloud computing task in the ECNs and in the centralized cloud is identical. Thus, we only consider the transmission delay in this paper.

Since the VM resource capacity of each ECN is limited, multiple mobile users might compete for the same ECN simultaneously. If a user does not win the most preferred ECN during the auction, it will be assigned to other ECNs. In this way, as many mobile users' requests as possible can be met. If the required VM resources of the allocated tasks exceed the capacity of an ECN, this ECN might upload part of the tasks to the CC, while guaranteeing that the transmission delay of each task is less than its deadline.

More specifically, the interactions between mobile users and ECNs via the platform in each round of auction, which is illustrated in Fig. 2, are presented as follows:

1) When a mobile user wishes to rent VM resources to run its deadline-sensitive cloud computing applications, it first generates a request and then submits the request to the platform. The request is composed of the user's maximum tolerable latency (i.e., deadline), the amount of required VM resources and the amount of input data. We use $r_i = \{T_i, A_i, I_i\}$ to denote the i-th mobile user's request, where T_i, A_i and I_i mean the deadline, the total VM resources and the amount of input data. Moreover, the set of all requests is denoted by $\mathcal{R}$.

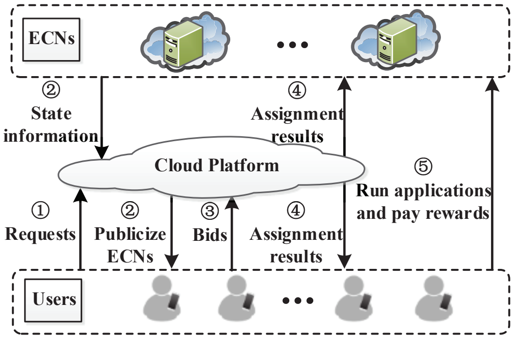


Fig. 2. The auction model for distributed edge cloud.

2) The platform will periodically collect the state information of each ECN, and then publicize it to the mobile users who submit requests. The state information includes several main parameters: VM resource capacity, bandwidths, unit cost of renting VM resources and unit cost of transmitting data to the CC. Here, the bandwidths of an ECN contain the bandwidth linked with the CC and the bandwidth associated with mobile users. We use $s_j = \{L_j, c_j^v, c_j^t, b_j^\uparrow, b_j^\downarrow\}$ to denote the state information of the j-th ECN where L_j denotes the VM resource capacity of ECN s_j. c_j^v and c_j^t denote the unit cost of VM resources and the unit cost of transmitting data to the CC, respectively. $b_j^\uparrow$ means the bandwidth between s_j and the CC, while $b_j^\downarrow$ indicates the bandwidth between s_j and the mobile users. The set of all ECNs is denoted by $\mathcal{S}$. Furthermore, we let c_0^v denote the unit cost of VM resources in the CC. Here, c_0^v is much less than c_j^v (for $\forall s_j \in \mathcal{S}$).

3) Then, the mobile user values differently to the ECNs according to the state information. At the same time, the mobile user determines a bid for each ECN. For each request r_i, we use b_{ij} and v_{ij} to denote the bid and valuation of i-th mobile user to the j-th ECN, respectively. The set of all bids is denoted by $\mathcal{B}$. The mobile users will send their bids to the platform.

Remarks: b_{ij} here is the reward that the i-th user claims to pay for renting the VM resources on the j-th ECN (i.e., s_j), while v_{ij} is the true valuation that the i-th user evaluates if it runs its application on s_j. The valuation v_{ij} is actually known to nobody except r_i itself. The value b_{ij} is not necessarily equal to v_{ij}, since the user might manipulate the claimed reward. Such strategic manipulation might cause the platform and the ECNs to get less reward. Thus, the whole mechanism needs to ensure that each mobile user will not manipulate its bids, i.e., truthfulness.

4) Based on the received bids and requests from mobile users, the platform *determines the winners of the auction*, *makes the scheduling decision for the winners*, and *computes the corresponding payments*.

Remarks: Each request will be assigned to at most one ECN, and each ECN can serve multiple requests under the capacity and deadline constraints. The scheduling decision for the winning bid means that the winning task will be first uploaded to an ECN, and then the ECN either completes the task by itself or uploads the task to the CC.

5) Mobile users upload the input data to the ECNs to run their cloud computing applications, and then pay

the corresponding rewards. According to the scheduling decisions for winning bids, ECNs decide to upload some tasks to the CC and pay the CC.

In this paper, we let a request correspond to only one deadline-sensitive cloud computing application. For simplicity, we suppose that each mobile user only submits one request in each round of auction. Actually, if a user wants to submit more than one request for multiple computing applications, we can use multiple virtual users, each of whom has only one request, to replace this mobile user. Additionally, we use d_{ij} to denote the transmission delay between the request r_i and the ECN s_j, which is calculated as follows:

$$d_{ij} = \begin{cases} \dfrac{I_i}{b_j^{\downarrow}}; & (r_i \rightarrow s_j) \\ \dfrac{I_i}{b_j^{\downarrow}} + \dfrac{I_i}{b_j^{\uparrow}}; & (r_i \rightarrow s_j \rightarrow CC) \end{cases} \tag{1}$$

where $r_i \rightarrow s_j$ means the request r_i is allocated to the ECN s_j and is completed by s_j itself, while $r_i \rightarrow s_j \rightarrow CC$ indicates that the request r_i is allocated to s_j and then is uploaded to the CC by s_j. Moreover, we suppose that I_i here is the amount of the input data plus the output data. That is, d_{ij} here has included the transmission delay of downloading output data from ECNs or CC to mobile users.

2.2 Problem Formulation

The above auction model involves two main problems: the *Winning Bid Selection (**WBS**)* problem and the *Payment Determination (**PD**)* problem.

Considering the set of requests, $\mathcal{R}$, the set of ECNs, $\mathcal{S}$ and the set of all bids, $\mathcal{B}$, we formalize the *WBS* problem as follows. First, we let Φ denote a solution to the *WBS* problem, called the winning bid set, which is composed of the bids that win the auction. Moreover, we use Φ^E and Φ^C to denote the winning bid scheduling solution. A bid $b_{ij} \in \Phi^E$ means that the request r_i will be assigned to the ECN s_j and be completed by s_j itself (i.e., $r_i \rightarrow s_j$), while $b_{ij} \in \Phi^C$ indicates that r_i will be uploaded to the CC through s_j (i.e., $r_i \rightarrow s_j \rightarrow CC$). Moreover, due to $\Phi^E \cap \Phi^C = \phi$, we get $\Phi = \Phi^E \cup \Phi^C$.

Second, we give the definition of social welfare, which is the optimization objective of the *WBS* problem.

Definition 1. [*Social Welfare*] The social welfare is the total valuations of the winning bids minus the total costs, i.e., $\sum_{b_{ij} \in \Phi^E} (v_{ij} - A_i \cdot c_j^v) + \sum_{b_{ij} \in \Phi^C} \left(v_{ij} - A_i \cdot (c_j^t + c_0^v)\right)$, where $A_i \cdot c_j^v$ denotes the cost of s_j conducting request r_i while $A_i \cdot (c_j^t + c_0^v)$ means the cost of transmitting r_i to the CC and the CC conducting r_i.

Based on this, we formalize the *WBS* problem as follows.

$$max \sum_{b_{ij} \in \Phi^E} (v_{ij} - A_i \cdot c_j^v) + \sum_{b_{ij} \in \Phi^C} \left(v_{ij} - A_i \cdot (c_j^t + c_0^v)\right) \tag{2}$$

$$s.t. \quad \Phi^E \cap \Phi^C = \phi, \tag{3}$$

$$\Phi^E \cup \Phi^C = \Phi \subseteq \mathcal{B} \tag{4}$$

$$\textstyle\sum_{j: b_{ij} \in \Phi} 1 \leq 1, \quad \forall r_i \in \mathcal{R} \tag{5}$$

$$\textstyle\sum_{i: b_{ij} \in \Phi^E} A_i \leq L_j, \quad \forall s_j \in \mathcal{S} \tag{6}$$

$$d_{ij} \leq T_i, \quad \forall r_i \in \mathcal{R} \tag{7}$$

Remarks: Here, Eq. 3 means that a request cannot be completed by ECNs and the CC simultaneously; Eq. 4 indicates that a solution to the *WBS* problem, which is a subset of the bid set, consists of two winning bid scheduling sets (i.e., Φ^E and Φ^C); Eq. 5 points out that each request is assigned to at most one ECN; Eq. 6 shows that the VM resource capacity of each ECN should be satisfied; Eq. 7 denotes that the transmission delay of each request should be less than its deadline.

Moreover, since our auction mechanism is truthful, we can use each bid to replace the corresponding true valuation, i.e., $v_{ij} = b_{ij}$, in Eq. 2. We will prove the truthfulness in Section 4 to ensure the correctness of $v_{ij} = b_{ij}$. Thus, the optimization objective of the *WBS* problem is equivalent to maximizing $\sum_{b_{ij} \in \Phi^E} (b_{ij} - A_i \cdot c_j^v) + \sum_{b_{ij} \in \Phi^C} \left(b_{ij} - A_i \cdot (c_j^t + c_0^v)\right)$. Additionally, we can assume $b_{ij} - A_i \cdot c_j^v \geq 0$ and $b_{ij} - A_i \cdot (c_j^t + c_0^v) \geq 0$ for $\forall r_i \in \mathcal{R}$ and $\forall s_j \in \mathcal{S}$. Actually, if a bid b_{ij} cannot cover the cost $A_i \cdot c_j^v$ or $A_i \cdot (c_j^t + c_0^v)$, the platform will directly delete the bid.

Next, the *PD* problem is to determine the payment for each winning bid so that the whole auction model satisfies the truthfulness and individual rationality, which are defined as follows:

Definition 2. [*Truthfulness*] [10, 20, 30, 35, 38] For each winning bid b_{ij}, we let $p_{ij}(b_{ij})$ denote the corresponding payment determined by the payment computation algorithm of an auction mechanism. Then, the i-th user's payoffs for the truthful bid and the untruthful bid are $v_{ij} - p_{ij}(v_{ij})$ and $v_{ij} - p_{ij}(b_{ij})$, respectively. The truthful mechanism means that

$$v_{ij} - p_{ij}(v_{ij}) \geq v_{ij} - p_{ij}(b_{ij}). \tag{8}$$

The truthfulness of the auction mechanism can ensure that each user reports its true valuation, since an untruthful bid will lead to a worse payoff.

Definition 3. [*Individual Rationality*] [15, 16, 23, 36, 37] In order to guarantee that each winning mobile user can receive a nonnegative payoff, the payment of one mobile user should be no more than its corresponding valuation, that is, $v_{ij} \geq p_{ij}(b_{ij})$, $\forall r_i \in \mathcal{R}$ and $\forall s_j \in \mathcal{S}$.

Here, each mobile user's true valuation must cover its corresponding payment. Otherwise, it is not motivated to participate in the edge cloud computing.

In addition, we define the computational efficiency of an auction mechanism as follows.

Definition 4. [*Computational Efficiency*] [7, 19, 34, 40, 41] An auction mechanism has the property of computational efficiency, if it can be conducted in polynomial time.

Since an auction cycle is not large, the winning bid set and the corresponding payment must be output in near-real time. In reality, an algorithm with computational efficiency is more important than an optimal algorithm with a high computational complexity.

For ease of reference, we summarize the commonly used notations throughout the paper in Table 1.

3 Design of the AVA Mechanism

In this section, we propose an Auction-based VM resource Allocation (AVA) mechanism for deadline-sensitive

TABLE 1
Description of major notations.

Notations	Description
ECN and CC	the abbreviations of Edge Cloud Node and Centralized Cloud, respectively.
i and j (i.e., r_i and s_j)	the indexes for users (i.e., requests) and ECNs, respectively.
$r_i = \{T_i, A_i, I_i\}$	the deadline, the required VM resource and the amount of input data of the i-th request.
$s_j = \{L_j, c_j^v, c_j^t, b_j^{\uparrow}, b_j^{\downarrow}\}$	the capacity, the unit cost of renting VM resources, the unit cost of transmitting data to the CC, the bandwidth linked with the CC, and the bandwidth associated with users of the j-th ECN.
b_{ij}, v_{ij}	the claimed bid and true valuation of r_i for s_j, respectively.
$\mathcal{R}$, $\mathcal{S}$ and $\mathcal{B}$	the sets of all requests, all ECNs and all bids, respectively.
c_0^v	the unit cost of renting VM resources in the CC.
d_{ij}	the transmission delay between r_i and s_j.
$r_i \to s_j$	r_i will be allocated to s_j and be completed by s_j itself.
$r_i \to s_j \to CC$	r_i will be uploaded to the CC through s_j and be completed by the CC.
Φ	a solution to the *WBS* problem (a winning bid set).
Φ^E and Φ^C	the winning bid scheduling solution: $b_{ij} \in \Phi^E$ means $r_i \to s_j$, while $b_{ij} \in \Phi^C$ indicates $r_i \to s_j \to CC$.
$\mathbb{S}$ and $\mathbb{B}$	the sets of virtual ECNs and virtual bids, respectively.
$\widehat{\mathcal{S}}$ and $\widehat{\mathcal{B}}$	the sets of ECNs and bids including virtual ECNs and bids (i.e., $\widehat{\mathcal{S}} = \mathcal{S} \cup \mathbb{S}$ and $\widehat{\mathcal{B}} = \mathcal{B} \cup \mathbb{B}$), respectively.
$p_{ij}(b_{ij})$	the payment of r_i for s_j based on the bid b_{ij}.
G, $\mathcal{E}$ and $\langle r_i, s_j \rangle$	a bipartite graph, the set of edges and the index for an edge, respectively.

computing tasks in geo-distributed edge clouds. The AVA mechanism mainly consists of a Winning Bid Selection (*WB-S*) algorithm and a Payment Determination (*PD*) algorithm. We first analyze the NP-hardness of the *WBS* problem, and then, we design a greedy winning bid selection algorithm and a truthful payment determination algorithm to solve the *WBS* and *PD* problems, respectively.

3.1 Problem Hardness Analysis

First, we prove that the *WBS* problem cannot be solved in polynomial time unless $P = NP$. More specifically, we have the following theorem:

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

Proof: We first consider a special case of the *WBS* problem, where the ECNs cannot upload tasks to the CC, the number of ECNs is equal to 1, and at the same time we let $T_i \geq d_{i1} = I_i / b_1^{\downarrow}$ for $\forall r_i \in \mathcal{R}$. Then, the special *WBS* problem is formalized as "maximize: $\sum_{b_{i1} \in \Phi^E} (v_{i1} - A_i \cdot c_1^v)$, subject to: $\sum_{i: b_{i1} \in \Phi^E} A_i \leq L_1$ for $\forall r_i \in \mathcal{R}$, $\Phi^E \subseteq \mathcal{B}$."

Now, we introduce the trivial 0-1 knapsack problem [28]: "maximize: $\sum_{i=1}^{n} w_i \cdot x_i$, subject to: $\sum_{i=1}^{n} v_i \cdot x_i \leq C$, $x_i \in \{0, 1\}$." Here, w_i and v_i denote the weight and volume of the i-th item, and C means the capacity of the knapsack.

By mapping C, w_i and v_i in the trivial 0-1 knapsack problem to L_1, v_{i1}–$A_i \cdot c_1^v$ and A_i in the special *WBS* problem, we get the two problems to be equivalent. That is to say, the special case of the *WBS* problem is a trivial 0-1 knapsack problem, which is NP-hard. Thus, the more general *WBS* problem is at least NP-hard. ■

3.2 Winning Bid Selection: Basic Solution

The *WBS* problem is how to select the winning bid set, so that we can maximize the social welfare, while ensuring that the deadline constraints of requests and the capacity constraints of ECNs can be satisfied simultaneously. Since the *WBS* problem has both deadline constraints and capacity constraints, we divide our solution into two phases. We take the deadline and capacity constraints into consideration in the two phases, respectively.

First phase: We focus on removing the deadline constraints of requests. First, we use $d_{ij}^1 = I_i / b_j^{\downarrow}$ and $d_{ij}^2 = I_i / b_j^{\downarrow} + I_i / b_j^{\uparrow}$ to denote the transmission delay of uploading r_i to s_j and the transmission delay of uploading r_i to the CC through s_j, respectively. Apparently, we have $d_{ij}^2 \geq d_{ij}^1$. For simplicity, we call d_{ij}^1 and d_{ij}^2 the good and bad transmission delay, respectively. Second, according to the relationships of T_i, d_{ij}^1 and d_{ij}^2, we update the bid set $\mathcal{B}$ and the ECN set $\mathcal{S}$. That is, we will remove the bids which cannot satisfy the deadline constraints, and add some virtual bids and ECNs if the deadline is larger than the bad transmission delay. More specifically, we have the three following cases:

- Case 1: if the deadline of r_i is less than the good transmission delay, i.e., $T_i < d_{ij}^1$, we will directly delete the bid b_{ij} from the bid set $\mathcal{B}$. This is because that the bid b_{ij} cannot satisfy the deadline constraint.
- Case 2: if T_i is less than the bad transmission delay but no less than the good transmission delay, i.e., $d_{ij}^1 \leq T_i < d_{ij}^2$, we will take no action.
- Case 3: if T_i is no less than the bad transmission delay, i.e., $T_i \geq d_{ij}^2$, we will create a virtual ECN s_{j*} and a virtual bid b_{ij*}, where $s_{j*} = \{L_{j*} = A_i, c_{j*}^v = c_j^t + c_0^v, c_{j*}^t = b_{j*}^{\uparrow} = b_{j*}^{\downarrow} = 0\}$ and the value of b_{ij*} is equal to that of b_{ij}. Here, the procedure of generating virtual ECNs and bids means removing the CC from the edge cloud computing scenario and transforming a three-layer edge cloud structure into a two-layer structure. We use $\mathbb{S}$ and $\mathbb{B}$ to denote the virtual ECN and bid sets, respectively. Then, we will add the virtual ECN s_{j*} and the virtual bid b_{ij*} into $\mathbb{S}$ and $\mathbb{B}$, respectively.

Second phase: We concentrate on the *WBS* problem with the capacity constraints. We first model the winning bid selection as an n-to-one weighted bipartite graph matching problem with 0-1 knapsack constraints. Since the problem is NP-hard due to the capacity constraints, we adopt a greedy strategy to determine a maximum matching, which has the approximately maximum weight in total. For simplicity, we let $\widehat{\mathcal{S}}$ and $\widehat{\mathcal{B}}$ denote the updated ECN and bid sets which contain the virtual ECNs and bids, respectively. That is, $\widehat{\mathcal{S}} = \mathcal{S} \cup \mathbb{S}$ and $\widehat{\mathcal{B}} = \mathcal{B} \cup \mathbb{B}$. Then, the winning bids are determined as follows.

First, we construct the weighted bipartite graph with capacity constraints, denoted as $G = \{\mathcal{R}, \widehat{\mathcal{S}}, \mathcal{E} : \widehat{\mathcal{B}}\}$, where $\mathcal{R}$

Algorithm 1 Preprocessing Algorithm

```
Require: $\mathcal{R}$, $\mathcal{S}$, $\mathcal{B}$, $c^v_0$.
Ensure: $\mathbb{S}$, $\mathbb{B}$, $\widehat{\mathcal{S}}$, and $\widehat{\mathcal{B}}$.
 1: Initialize $\mathbb{S} = \mathbb{B} = \phi$;
 2: for $r_i \in \mathcal{R}$ do
 3:     for $s_j \in \mathcal{S}$ do
 4:         $d^1_{ij} = I_i/b^\downarrow_j$,  $d^2_{ij} = I_i/b^\downarrow_j + I_i/b^\uparrow_j$;
 5:         if $T_i < d^1_{ij}$ then
 6:             Remove $b_{ij}$ from $\mathcal{B}$, i.e., $\mathcal{B} = \mathcal{B} - \{b_{ij}\}$;
 7:         else if $d^1_{ij} \le T_i < d^2_{ij}$ then
 8:             Continue;   //Case 2;
 9:         else if $d^2_{ij} \le T_i$ then
10:             Generate a virtual ECN $s_{j*}$, $\mathbb{S} = \mathbb{S} + \{s_{j*}\}$;
                //$s_{j*} = \{L_{j*} = A_i, c^v_{j*} = c^t_j + c^v_0, c^t_{j*} = b^\uparrow_{j*} = b^\downarrow_{j*} = 0\}$;
11:             Generate a virtual bid $b_{ij*}$, $\mathbb{B} = \mathbb{B} + \{b_{ij*}\}$;
12: $\widehat{\mathcal{S}} = \mathcal{S} \cup \mathbb{S}$ and $\widehat{\mathcal{B}} = \mathcal{B} \cup \mathbb{B}$;
13: return $\mathbb{S}$, $\mathbb{B}$, $\widehat{\mathcal{S}}$ and $\widehat{\mathcal{B}}$;
```

and $\widehat{\mathcal{S}}$ are two separate vertex sets, and $\mathcal{E}$ refers to the edge set between $\mathcal{R}$ and $\widehat{\mathcal{S}}$. We let $\langle r_i, s_j \rangle \in \mathcal{E}$ denote the edge between r_i and s_j for convenience. Here, each bid $b_{ij} \in \widehat{\mathcal{B}}$ corresponds to an edge $\langle r_i, s_j \rangle$. For each vertex $r_i \in \mathcal{R}$, the required VM resources A_i is seen as the volume of item in the trivial 0-1 knapsack, while for each vertex $s_j \in \widehat{\mathcal{S}}$, the VM resource capacity L_j is seen as the capacity of the knapsack. Moreover, each edge $\langle r_i, s_j \rangle$ corresponds to a weight, which is defined as the social welfare per *unit* VM resource. Let w_{ij} denote the weight of the edge $\langle r_i, s_j \rangle \in \mathcal{E}$. Then, we have

$$w_{ij} = \frac{b_{ij}}{A_i} - c^v_j, \quad \text{for } \forall \langle r_i, s_j \rangle \in \mathcal{E}. \tag{9}$$

Based on the weighted bipartite graph G, we can simplify and re-formalize the *WBS* problem. After removing the CC and adding some virtual ECNs and bids, the three-layer edge cloud structure is changed into a two-layer structure. For the two-layer edge cloud structure, each request can only be allocated to one ECN. So we get the winning bid set $\Phi^C = \phi$, and further have $\Phi = \Phi^E$. Accordingly, we re-formalize the *WBS* problem:

$$max \qquad \textstyle\sum_{b_{ij} \in \Phi} (b_{ij} - A_i \cdot c^v_j) \tag{10}$$

$$s.t. \qquad \Phi \subseteq \widehat{\mathcal{B}}, \tag{11}$$

$$\textstyle\sum_{j: b_{ij} \in \Phi} 1 \le 1, \quad \forall r_i \in \mathcal{R} \tag{12}$$

$$\textstyle\sum_{i: b_{ij} \in \Phi} A_i \le L_j, \quad \forall s_j \in \widehat{\mathcal{S}} \tag{13}$$

Second, after constructing the weighted bipartite graph with the capacity constraints, we greedily select some edges to form a maximum matching of G with the approximately maximum weight. More specifically, in each round, we select the edge with the largest weight. Without loss of generality, let this edge be $\langle r_i, s_j \rangle$. Now, we have the two following cases.

(1) $A_i \le L_j$: This means that the remaining capacity L_j of ECN s_j is not less than the amount of resources A_i. Then, we add the corresponding bid b_{ij} into the assignment solution Φ. Moreover, we directly remove the request vertex r_i from $\mathcal{R}$ and delete all edges relevant to r_i from $\mathcal{E}$ in G. We also update the value of L_j by subtracting A_i for the vertex s_j.

(2) $A_i > L_j$: We directly delete the edge $\langle r_i, s_j \rangle$ from $\mathcal{E}$ and continue to find the edge with the next largest weight.

Algorithm 2 Winning Bid Selection (*WBS*) Algorithm

```
Require: $G = \{\mathcal{R}, \widehat{\mathcal{S}}, \mathcal{E} : \widehat{\mathcal{B}}\}$.
Ensure: $\Phi$.
 1: Initialize $\Phi = \phi$;
 2: while $\mathcal{R} \neq \phi$ and $\widehat{\mathcal{S}} \neq \phi$ and $\mathcal{E} \neq \phi$ in $G$ do
 3:     Select the edge with largest weight, denoted as $\langle r_i, s_j \rangle$;
 4:     if $A_i \le L_j$ then
 5:         Add $b_{ij}$ into $\Phi$, i.e., $\Phi = \Phi + \{b_{ij}\}$;
 6:         Remove $r_i$ from $\mathcal{R}$ and delete all edges related to $r_i$;
 7:         Update $L_j \Leftarrow L_j - A_i$ for the vertex $s_j$ in $\langle r_i, s_j \rangle$;
 8:     else
 9:         Remove $\langle r_i, s_j \rangle$ from $\mathcal{E}$ and continue;
10: return $\Phi$;
```

Note that when an edge is deleted from the bipartite graph, the related weight is also deleted. This selection process is repeatedly conducted until $\mathcal{R}$, $\widehat{\mathcal{S}}$, or $\mathcal{E}$ becomes an empty set. Finally, we get an assignment solution Φ. If $b_{ij} \in \Phi$, it indicates that the bid b_{ij} wins and the request r_i is assigned to the ECN s_j.

After getting the winning bid set Φ, we then make scheduling decisions for each bid $b_{ij} \in \Phi$. That is, for a winning bid b_{ij}, we either let s_j complete the task r_i or upload r_i to the CC through s_j, i.e., we need to determine the winning bid scheduling sets Φ^E and Φ^C.

3.3 Winning Bid Selection: Detailed Algorithm

Based on the above solution, we first design a *Preprocessing Algorithm*, as shown in Algorithm 1, to remove the deadline constraints. By deleting some bids and adding some virtual bids and ECNs, Algorithm 1 will update the ECN set and the bid set. More specifically, In Step 1, we initialize the virtual ECN and bid sets (i.e., $\mathbb{S}$ and $\mathbb{B}$). In Steps 2-4, the good and bad transmission delay (i.e., d^1_{ij} and d^2_{ij}) are calculated. According to the relationships among T_i, d^1_{ij} and d^2_{ij}, we remove some bids that cannot satisfy the deadline constraints in Steps 5-6. In Steps 7-8, if $d^1_{ij} \le T_i < d^2_{ij}$, we take no action. If $d^2_{ij} \le T_i$, we first generate a virtual ECN s_{j*} and a virtual bid b_{ij*}, and then add them into $\mathbb{S}$ and $\mathbb{B}$, respectively, in Steps 9-11. In Step 12, we get the updated ECN set $\widehat{\mathcal{S}}$ and the updated bid set $\widehat{\mathcal{B}}$. At last, Algorithm 1 outputs $\mathbb{S}$, $\mathbb{B}$, $\widehat{\mathcal{S}}$ and $\widehat{\mathcal{B}}$ in Step 13..

Next, we construct a weighted bipartite graph with capacity constraints $G = \{\mathcal{R}, \widehat{\mathcal{S}}, \mathcal{E} : \widehat{\mathcal{B}}\}$. Based on this, we design a greedy Winning Bid Selection (*WBS*) algorithm, as shown in Algorithm 2. First, we initialize the assignment solution in Step 1. Then, we conduct the greedy winning bid selection procedure in Steps 2-9. More specifically, in Step 3, the edge (e.g., $\langle r_i, s_j \rangle$) with the largest weight (i.e., w_{ij}) is selected. Then, we compare the remaining capacity L_j of s_j with the amount of VM resources A_i in Step 4. If $A_i \le L_j$, the assignment solution is expanded in Step 5, i.e., $\Phi = \Phi + \{b_{ij}\}$. In Steps 6-7, all edges related to the vertex r_i in G are deleted, and the vertex mark L_j is updated by subtracting A_i. If $A_i > L_j$, Algorithm 2 directly removes $\langle r_i, s_j \rangle$ from $\mathcal{E}$ in G and continues to conduct the selection procedure in Steps 8-9. When $\mathcal{R}$, $\widehat{\mathcal{S}}$, or $\mathcal{E}$ becomes an empty set, Algorithm 2 terminates and outputs Φ in Step 10.

Algorithm 3 Winning Bid Scheduling Algorithm

```
Require: $\Phi$ and $\mathbb{S}$.
Ensure: $\Phi^E$ and $\Phi^C$.
1: Initialize $\Phi^E = \Phi^C = \phi$;
2: for $b_{ij} \in \Phi$ (corresponding to $\langle r_i, s_j \rangle$) do
3:    if $s_j \in \mathbb{S}$ then
4:       $\Phi^C = \Phi^C + \{b_{ij}\}$;
5:    else
6:       $\Phi^E = \Phi^E + \{b_{ij}\}$;
7: return $\Phi^E$ and $\Phi^C$;
```

We can straightforwardly demonstrate the correctness of Algorithm 2 in the following theorem:

Theorem 2. Algorithm 2 is correct, that is, it will terminate for sure and produce a feasible assignment solution.

Proof: Since only one edge in $\mathcal{E}$ is selected at each round where $\mathcal{E}$ is a limited set, Algorithm 2 will terminate for sure. Moreover, when a bid b_{ij} is added into the solution Φ, r_i will be removed from $\mathcal{R}$ and the vertex mark L_j of s_j is minus A_i. Based on this, the constraints of Eq. 5 and Eq. 6 in the formalized *WBS* problem can be satisfied. Thus, the produced solution must be feasible. ■

Besides, we design a winning bid scheduling algorithm, as shown in Algorithm 3. According to the winning bid set Φ and the virtual ECN set $\mathbb{S}$, Algorithm 3 can determine the winning bid scheduling sets Φ^E and Φ^C. In Step 1, we initialize Φ^E and Φ^C. Then, for each winning bid $b_{ij} \in \Phi$, we determine if the corresponding ECN s_j belongs to the virtual ECN set $\mathbb{S}$: if yes, we add b_{ij} into Φ^C; otherwise, we add b_{ij} into Φ^E, in Steps 2-6. At last, Algorithm 3 outputs the winning bid scheduling sets Φ^E and Φ^C, in Step 7.

3.4 Payment Determination: Basic Solution

The truthful payment determination computes the payment for each winning bid, ensuring that each user honestly reports its true valuation for its cloud service request. In this paper, we adopt the rule of *critical payment* introduced by Myerson [19] to determine the payment for each winning bid. The critical payment is defined as follows.

Definition 5. [*Critical Payment*] The payment for bid b_{ij}, denoted as $p_{ij}(b_{ij})$, is said to be critical value if the user declares a bid that is not smaller than $p_{ij}(b_{ij})$, the submitted bid must win; otherwise, it will not win.

According to Definition 5, in order to determine the critical payment for bid b_{ij}, we first need to determine the alternative bid of b_{ij}. Here, the alternative bid of a winning bid b_{ij} is such a bid that will replace b_{ij} to become a winning bid when we remove b_{ij} from $\widehat{\mathcal{B}}$. More specifically, we first remove the corresponding edge $\langle r_i, s_j \rangle$ from $\mathcal{E}$ in G to get a new weighted bipartite graph without b_{ij}. For convenience, we use $\mathcal{E}_{-ij}$ and G_{-ij} to denote the edge set and the new bipartite graph without the edge $\langle r_i, s_j \rangle$, that is, $G_{-ij} = \{\mathcal{R}, \widehat{\mathcal{S}}, \mathcal{E}_{-ij} : \widehat{\mathcal{B}} - \{b_{ij}\}\}$. According to G_{-ij}, we reselect a new winning bid set using Algorithm 2 and let Φ_{-ij} denote the new assignment solution. Here, the alternative bid of b_{ij} must belong to Φ_{-ij}. Accordingly, we have two cases: the request r_i is assigned to another ECN $s_{j'}$, or the ECN s_j has accepted some requests so that it has no enough remaining capacity to provide cloud service for request r_i. For the alternative bid of b_{ij} in Φ_{-ij}, we assume that bid $b_{ij'}$ is the winner related to r_i, and $b_{i_1 j}, b_{i_2 j}, \cdots, b_{i_\kappa j}$ are the winners relevant to s_j, respectively. The corresponding weights are $w_{ij'}$ and $w_{i_1 j}, w_{i_2 j}, \cdots, w_{i_\kappa j}$. Note that $b_{ij'}$ for the first case is exactly a candidate alternative bid of b_{ij}. For the second case, we can find the critical weight for the ECN s_j, denoted as $w_{i_{min}j}$ for convenience. That is, we have

Algorithm 4 Payment Determination (*PD*) Algorithm

```
Require: $G = \{\mathcal{R}, \widehat{\mathcal{S}}, \mathcal{E} : \widehat{\mathcal{B}}\}$, $\Phi = \Phi^E \cup \Phi^C$ and $\mathbb{B}$.
Ensure: $\mathcal{P} = \{p_{ij}(b_{ij}) | b_{ij} \in \Phi\}$.
1: for each $b_{ij} \in \Phi$ do
2:    $\mathcal{E}_{-ij} = \mathcal{E} - \{\langle r_i, s_j \rangle\}$ and $G_{-ij} = \{\mathcal{R}, \widehat{\mathcal{S}}, \mathcal{E}_{-ij} : \mathcal{B} - \{b_{ij}\}\}$;
3:    Execute Algorithm 2: $\Phi_{-ij} = \text{WBS}(G_{-ij})$;
4:    Determine $w_{i_{min}j} = \min\{w_{i_1 j}, w_{i_2 j}, \cdots, w_{i_\kappa j} \in \Phi_{-ij} :$
      $L_j - \sum_{w_{i_x j} \geq w_{i_y j}} A_{i_x} \geq A_i\}$;
5:    $p_{ij}(b_{ij}) = A_i \cdot (c_j^v + \max\{w_{ij'}, w_{i_{min}j}\})$;  // $b_{ij'} \in \Phi_{-ij}$
6:    if $b_{ij} \in \mathbb{B}$ then
7:       User $r_i$ pays the ECN $s_{j*}$ with the reward $p_{ij}(b_{ij})$;
         //$s_j$ here is the virtualization of $s_{j*}$, i.e.,
         //$s_j = \{L_j = A_i, c_j^v = c_{j*}^t + c_0^v, c_j^t = b_j^{\uparrow} = b_j^{\downarrow} = 0\}$;
8:    else
9:       User $r_i$ pays the ECN $s_j$ with the reward $p_{ij}(b_{ij})$;
         // $s_j$ here is a real ECN.
10: return $\mathcal{P}$;
```

$$w_{i_{min}j} = \min\{w_{i_1 j}, w_{i_2 j}, \cdots, w_{i_\kappa j} : L_j - \sum_{w_{i_x j} \geq w_{i_y j}} A_{i_x} \geq A_i\}. \quad (14)$$

Here, $L_j - \sum_{w_{i_x j} \geq w_{i_y j}} A_{i_x} \geq A_i$ indicates that s_j always selects the requests with relatively large weights until it has no enough remaining capacity for request r_i. Accordingly, the bid $b_{i_{min}j}$ is exactly another candidate alternative bid of b_{ij}. Moreover, if $w_{ij'} \geq w_{i_{min}j}$, $b_{ij'}$ will become the alternative bid of b_{ij}. Otherwise, if $w_{ij'} < w_{i_{min}j}$, the alternative bid of b_{ij} will be $b_{i_{min}j}$.

Thus, the critical payment $p_{ij}(b_{ij})$ is determined by:

$$p_{ij}(b_{ij}) = A_i \cdot (c_j^v + \max\{w_{ij'}, w_{i_{min}j}\}). \quad (15)$$

Remarks: We do not distinguish the virtual bids from the winning bids at first. In fact, whether a winning bid is virtual or real is irrelevant to the payment determination process. For a virtual winning bid $b_{ij*} \in \Phi$ in which $s_{j*} \in \mathbb{S}$, the mobile user r_i will pay the ECN s_j with the reward $p_{ij*}(b_{ij*})$ where s_{j*} is the virtualization of s_j (i.e., $s_{j*} = \{L_{j*} = A_i, c_{j*}^v = c_j^t + c_0^v, c_{j*}^t = b_{j*}^{\uparrow} = b_{j*}^{\downarrow} = 0\}$).

3.5 Payment Determination: Detailed Algorithm

Based on the above method, the payment determination algorithm is shown in Algorithm 4. For each winning bid b_{ij} in the original solution Φ (i.e., $b_{ij} \in \Phi$), we first re-construct a new bipartite graph G_{-ij} without b_{ij} in Step 2. Then, we execute the greedy *WBS* algorithm based on the input G_{-ij} and get a new assignment solution Φ_{-ij} in Step 3. Here, we use the form *"Output* = WBS(*Input*)*"* to denote the execution of the *WBS* algorithm for convenience. In Step 4, we determine the critical weight for the ECN s_j, i.e., $w_{i_{min}j}$, according to Eq. 14. In Step 5, the critical payment for the bid b_{ij} is determined according to Eq. 15. If b_{ij} is a virtual bid,

	r_1	r_2	r_3	r_4	r_5	r_6
T_i	4	3	4	5	3	2
A_i	4	3	5	2	1	6
I_i	6	4	6	7	5	8

	L_j	c_j^v	c_j^t	$b_j^{\downarrow}$	$b_j^{\uparrow}$
s_1	6	2	1	2	3
s_2	5	3	2	2	4

(a)

d_{ij}^1 \ d_{ij}^2	s_1	s_2
r_1	3 \ 5	3 \ 4.5
r_2	2 \ 3.3	2 \ 4
r_3	3 \ 5	3 \ 4.5
r_4	3.5 \ 5.8	3.5 \ 5.3
r_5	2.5 \ 5.3	2.5 \ 3.8
r_6	4 \ 6.7	4 \ 6

(b)

b_{ij}	s_1	s_2
r_1	20	48
r_2	18	33
r_3	45	20
r_4	8	16
r_5	10	9

(c)

w_{ij}	s_1	s_2
r_1	3	9
r_2	4	8
r_3	7	1
r_4	2	5
r_5	8	6

(d)

Fig. 3. An illustration of removing deadline constraints and calculating edge weights based on bids.

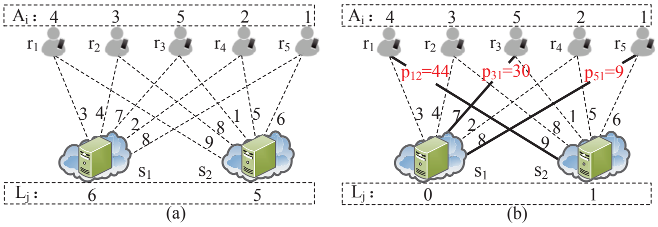


Fig. 4. Illustrations of the execution of the *WBS* and *PD* algorithms.

we identify the ECN which will obtain the payment in Steps 6-7. That is to say, the real ECN s_{j*} where s_j is the virtualization of s_{j*}, i.e.,$s_j=\{L_j=A_i, c_j^v=c_{j*}^t+c_0^v, c_j^t=b_j^{\uparrow}=b_j^{\downarrow}=0\}$, will get the payment $p_{ij}(b_{ij})$. Otherwise, when b_{ij} is a real bid, i.e., $b_{ij}\in\mathcal{B}$, the real ECN s_j will naturally obtain the payment $p_{ij}(b_{ij})$ in Steps 8-9. After computing the critical payments for all winning bids in Φ, the algorithm terminates and outputs the results in Step 10.

3.6 A Walk-Through Example

To better understand Algorithms 1-4, we present an example to show the procedure, in which 6 users request cloud services from 2 ECNs. That is, $\mathcal{S}=\{s_1,s_2\}$ and $\mathcal{R}=\{r_1,r_2,r_3,r_4,r_5,r_6\}$. The detailed parameter values are shown in Fig. 3(a). Based on Algorithm 1, we can calculate the transmission delay, which is shown in Fig. 3(b). Since $T_6<d_{61}^1$ and $T_6<d_{62}^1$, we delete the bids b_{61} and b_{62} from $\mathcal{B}$ directly. Then, we display the updated bid set in Fig. 3(c). Before conducting Algorithm 2, we first compute the weights of all edges. The results are shown in Fig. 3(d). Afterwards, Fig. 4 (a) displays the bipartite graph, including the edge weights, the amount of VM resources for each request, and the capacity constraints of ECNs. Based on Fig. 4 (a), the greedy *WBS* algorithm (i.e., Algorithm 2) is conducted as follows.

In the first round, $\Phi=\phi$. Algorithm 2 selects the edge with the largest weight, that is, $\langle r_1,s_2\rangle$. Because of $A_1\le L_2$, we add b_{12} into Φ. After removing r_1 and its corresponding edges, we update the remaining capacity of s_2, i.e., $L_2\Leftarrow L_2-A_1=1$. In the second round, $\Phi=\{b_{12}\}$. The largest weight in the new graph is $w_{22}=w_{51}=8$. Because of $A_2=3>L_2=2$, we add b_{51} into Φ. Similarly, we remove r_5 and update $L_2\Leftarrow L_2-A_1=5$. In the third round, we select b_{31} and get $L_1=0$. Now, no bids can be added into Φ due to $A_i>L_j$ for each remaining vertex r_i, and Algorithm 2 outputs $\Phi=\{b_{12},b_{51},b_{31}\}$ in Fig. 4 (b).

Due to no virtual bids in the example, we get $\Phi=\Phi^E$ by conducting Algorithm 3. Next, we describe the procedure of the truthful payment determination algorithm (i.e., Algorithm 4).

First, for bid b_{12}, we remove $\langle r_1,s_2\rangle$ from G. After executing Algorithm 2 based on G_{-12}, we get a new solution: $\Phi_{-12}=\{b_{22},b_{51},b_{31},b_{42}\}$. Then, we determine the critical weight for ECN s_2 (i.e., selecting $w_{i_{min}2}$ from $\{w_{22},w_{42}\}$). According to Eq. 14, we get $w_{i_{min}2}=w_{22}$. Thus, we get $p_{12}(b_{12})=A_1\cdot(c_2^v+\max\{0,w_{22}\})=44$ according to Eq. 15. Second, for bid b_{51}, we execute Algorithm 2 according to the new graph without the edge $\langle r_5,s_1\rangle$ and get $\Phi_{-51}=\{b_{12},b_{31},b_{52}\}$. Likewise, we have $w_{i_{min}1}=w_{31}$, and further get $p_{51}(b_{51})=A_5\cdot(c_1^v+\max\{w_{52},w_{31}\})=9$. In the same way, for the bid b_{31}, we can also obtain $\Phi_{-31}=\{b_{12},b_{51},b_{21},b_{41}\}$. After determining the critical weight for s_1 according to Eq. 14, i.e., $w_{i_{min}1}=w_{21}$, we compute the corresponding payment, i.e., $p_{31}(b_{31})=A_3\cdot(c_1^v+\max\{0,w_{21}\})=30$. The determined payments are shown in Fig. 4 (b).

We can find that requests r_3 and r_5 are assigned to ECN s_1 and r_1 is assigned to s_2, respectively. Due to $\Phi=\Phi^E$, none of the requests will be uploaded to the CC. Moreover, we find that the payment of each winning bid is no larger than its true valuation.

4 Theoretical Analysis

In this section, we present the theoretical analysis, showing that the AVA mechanism is truthful, individually rational, and computationally efficient. Also, we analyze the approximation ratio of the greedy *WBS* algorithm.

4.1 Truthfulness

To demonstrate that the AVA mechanism is truthful, we need to reveal that each user will honestly submit its real valuation when the strategies of other users are given. According to Myerson's theorem [19], our AVA mechanism is truthful if and only if the following two conditions hold: (1) the winning bid selection algorithm (i.e., allocation rule) is *monotonic*, and (2) each winning bid is paid the *critical payment* as introduced in Definition 5.

Lemma 1. The *WBS* algorithm is monotonic. More specially, for each bid b_{ij}, if b_{ij} wins according to the *WBS* algorithm, then $\tilde{b}_{ij}=b_{ij}+\theta$ will also win, where $\theta\ge 0$.

Proof: For an arbitrary request $r_i\in\mathcal{R}$ and ECN $s_j\in\widehat{\mathcal{S}}$, if the bid b_{ij} wins, then a larger bid $b'_{ij}\ge b_{ij}$ must win according to the greedy strategy used in Algorithm 2. ■

Lemma 2. Each winning bid is paid the critical payment.

Proof: For an arbitrary winning bid b_{ij}, we assume that in Φ_{-ij}, r_i is assigned to another ECN $s_{j'}$ (i.e., $b_{ij'}$ wins in Φ_{-ij}), and s_j connects to other users (i.e., the bid set $\{b_{i_1j},b_{i_2j},\cdots,b_{i_\kappa j}\in\Phi_{-ij}\}$ wins). Then, the critical payment of bid b_{ij} is determined by $p_{ij}(b_{ij})/A_i-c_j^v=\max\{w_{ij'},w_{i_{min}j}\}$ according to Eqs. 14 and 15. Here, if user r_i claims a higher bid $b\ge p_{ij}(b_{ij})$ for ECN s_j, then $b/A_i-c_j^v\ge w_{ij'}$ and $b/A_i-c_j^v\ge w_{i_{min}j}$ hold. Thus, the claimed bid b will be selected prior to $b_{ij'}$ and $b_{i_{min}j}$ according to the greedy strategy of Algorithm 2. If $b\le p_{ij}(b_{ij})$, we have three sub-cases: (1) $b/A_i-c_j^v\le w_{ij'}$ and $b/A_i-c_j^v\le w_{i_{min}j}$; (2) $w_{ij'}\le b/A_i-c_j^v\le w_{i_{min}j}$; and (3) $w_{i_{min}j}\le b/A_i-c_j^v\le w_{ij'}$. Based on this, the following conclusion holds in all three sub-cases: r_i is assigned to $s_{j'}$ or s_j corresponds to $\{r_{i_1},r_{i_2},\cdots,r_{i_\kappa}|b_{i_\kappa j}\in\Phi_{-ij}\}$

where $r_i \neq r_{i_\kappa}$. Note that when $b_{i_{min}j}$ is replaced by b_{ij}, the capacity constraint of s_j is still satisfied according to Eq. 14. This means that $b_{i_{min}j}$ is actually the alternative bid for b_{ij} when $\max\{w_{ij'}, w_{i_{min}j}\} = w_{i_{min}j}$. Thus, in this case, bid b_{ij} will lose the auction. This means that $p_{ij}(b_{ij}) = A_i \cdot (c_j^v + \max\{w_{ij'}, w_{i_{min}j}\})$ is the critical payment exactly. The lemma holds. ■

After proving the monotonicity of WBS algorithm and the critical payment for winning bids in the above lemmas, we get that the proposed AVA mechanism satisfies Myerson's theory [19]. Therefore, we can directly prove the truthfulness of the AVA mechanism as follows.

Theorem 3. The proposed auction mechanism is truthful.

Proof: Based on Lemmas 1 and 2, the theorem holds. ■

4.2 Individual Rationality

The individual rationality means that mobile user's true valuation must cover its corresponding payment according to Definition 3. Thus, we just need to prove that each winning user will receive a nonnegative payoff in the AVA mechanism. As a result, we have the following theorem.

Theorem 4. The AVA mechanism is individually rational.

Proof: In the AVA mechanism, if a user does not win the cloud service from the ECNs, its payoff is zero. Otherwise, if r_i wins the cloud service from s_j with the bid b_{ij}, its corresponding payoff is $v_{ij} - p_{ij}(b_{ij})$ according to Definition 2. We suppose $p_{ij}(b_{ij}) = A_i \cdot (c_j^v + \max\{w_{ij'}, w_{i_{min}j}\})$ where $w_{i_{min}j} = \min\{w_{i_1j}, w_{i_2j}, \cdots, w_{i_\kappa j} \in \Phi_{-ij} : L_j - \sum_{w_{i_xj} \geq w_{i_yj}} A_{i_x} \geq A_i\}$. Since b_{ij} is selected prior to $b_{ij'}$ and $b_{i_{min}j}$ in Φ, we have $b_{ij}/A_i - c_j^v = w_{ij} \geq \max\{w_{ij'}, w_{i_{min}j}\}$. That is, $b_{ij} \geq A_i \cdot (c_j^v + \max\{w_{ij'}, w_{i_{min}j}\}) = p_{ij}(b_{ij})$. Because of the truthfulness of the mechanism in Theorem 3, we get $b_{ij} = v_{ij}$. Hence, we have $v_{ij} \geq p_{ij}(b_{ij})$. The theorem is correct. ■

4.3 Computational Efficiency

To prove the computational efficiency of the AVA mechanism, we just need to prove that AVA can be conducted in polynomial time according to Definition 4.

Theorem 5. The mechanism is computationally efficient.

Proof: First, we determine that the algorithmic procedure of Algorithm 1 is in polynomial time and that the computational overhead is $O(|\mathcal{R}| \cdot |\mathcal{S}|)$, where $|\cdot|$ means the cardinality of the set. Second, we give the computational overheads of Algorithm 2 and Algorithm 3, that is, $O(|\mathcal{R}|^2 \cdot |\widehat{\mathcal{S}}|^2)$ and $O(|\Phi|)$, respectively. At last, Algorithm 4 is also in polynomial time and its computational complexity is $O(|\Phi| \cdot |\mathcal{R}|^2 \cdot |\widehat{\mathcal{S}}|^2)$. According to Definition 4, we get that the AVA mechanism is computationally efficient. ■

4.4 Approximation Ratio Analysis

The *WBS* problem is an n-to-one weighted bipartite matching problem with capacity constraints, which is a strongly NP-hard problem [28]. Algorithm 2, adopting the greedy selection strategy, can produce an approximate solution. Here, we adopt the mathematical induction method to analyze the approximation ratio.

Theorem 6. Algorithm 2 can achieve a $(\gamma+1)$-approximation of the optimal assignment solution, where $\gamma = \max\{L_j/A_i | r_i \in \mathcal{R}, s_j \in \widehat{\mathcal{S}}\}$. Moreover, $(\gamma+1)$ is an urgent bound.

Proof: Let Φ_{opt} be the optimal solution of $G = \{\mathcal{R}, \widehat{\mathcal{S}}, \mathcal{E} : \widehat{\mathcal{B}}\}$. For the simplicity of the following descriptions, we use $\alpha = \sum_{b_{ij} \in \Phi_{opt}} (b_{ij} - A_i \cdot c_j^v) / \sum_{b_{ij} \in \Phi} (b_{ij} - A_i \cdot c_j^v) = \Phi_{opt}(G)/\Phi(G)$ to denote the approximation ratio, in which we let $\Phi(G)$ and $\Phi_{opt}(G)$ denote our solution and the optimal solution based on the bipartite graph G, respectively. Then, we prove $\alpha \leq \gamma + 1$ by using the mathematical induction method. Consider that $\widehat{\mathcal{S}}$ is given and $\mathcal{R}$ changes.

First, when $|\mathcal{R}| = 1$, we directly have $\alpha = 1 \leq \gamma + 1$.

Second, we assume that $\alpha \leq \gamma + 1$ holds when $|\mathcal{R}| \leq m$. Then, we consider $|\mathcal{R}| = m+1$. We here need to establish a link between $|\mathcal{R}| = m+1$ and $|\mathcal{R}| \leq m$. Without loss of generality, we consider $b_{11}/A_1 - c_1^v = w_{11} = \max\{w_{ij} | \langle r_i, s_j \rangle \in \mathcal{E}\}$. Thus, b_{11} must belong to Φ. According to this, we analyze the two cases as follows.

(1) b_{11} also belongs to Φ_{opt}. Then, we get a new bipartite graph $G^* = \{\mathcal{R} - \{r_1\}, \widehat{\mathcal{S}}, \mathcal{E} - \{\langle r_1, s_1 \rangle\} : \widehat{\mathcal{B}} - \{b_{11}\}\}$ after removing b_{11} (i.e., the edge $\langle r_1, s_1 \rangle$). Because of $|\mathcal{R} - \{r_1\}| \leq m$, we have $\alpha = \Phi_{opt}(G^*)/\Phi(G^*) \leq \gamma + 1$ according to the inductive assumption, and we further get

$$\alpha = \frac{\Phi_{opt}(G)}{\Phi(G)} = \frac{(b_{11} - A_1 \cdot c_1^v) + \Phi_{opt}(G^*)}{(b_{11} - A_1 \cdot c_1^v) + \Phi(G^*)} \leq \gamma + 1. \tag{16}$$

(2) b_{11} does not belong to Φ_{opt}. Here, we assume that some other bids are selected to access the ECN s_1, and let the set $\Phi_{opt}^{s_1} = \{b_{i1} | b_{i1} \in \Phi_{opt}\}$ denote it. For simplicity, we also use $\widetilde{\mathcal{R}} = \{r_i | b_{i1} \in \Phi_{opt}^{s_1}\}$ to denote the request set connecting to s_1. Moreover, we suppose that $b_{1j'}$ is related to r_1 in Φ_{opt}. Based on this, we form two bipartite sub-graphs, denoted as G^* and G^+. More specifically, G^* is the sub-graph of G where the request vertices $\{r_1\} \cup \widetilde{\mathcal{R}}$ and the corresponding edges are removed, and the capacity constraints of L_1 and $L_{j'}$ are updated by using $L_1 = L_1 - \sum_{r_i \in (\{r_1\} \cup \widetilde{\mathcal{R}})} A_i$ and $L_{j'} = L_{j'} - A_1$, respectively. G^+ is the sub-graph of G where the request vertices $\{r_1\} \cup \widetilde{\mathcal{R}}$ and the corresponding edges are removed, and the capacity constraint of L_1 is updated by using $L_1 = L_1 - \sum_{r_i \in (\{r_1\} \cup \widetilde{\mathcal{R}})} A_i$. Thus, we have $G^* \subseteq G^+$.

Here, $|\mathcal{R} - \{r_1\} - \widetilde{\mathcal{R}}| \leq m$. Based on the assumption of mathematical method, we have $\Phi_{opt}(G^*)/\Phi(G^*) \leq \gamma + 1$. Since $G^* \subseteq G^+$ holds, we get

$$\Phi_{opt}(G^*) \leq (\gamma+1)\Phi(G^*) \leq (\gamma+1)\Phi(G^+). \tag{17}$$

Furthermore, we have

$$\begin{aligned} \alpha = \frac{\Phi_{opt}(G)}{\Phi(G)} &= \frac{\Phi_{opt}(G^*) + (b_{1j'} - A_1 \cdot c_{j'}^v) + \sum\limits_{b_{i1} \in \Phi_{opt}^{s_1}} (b_{i1} - A_i \cdot c_1^v)}{\Phi(G^+) + (b_{11} - A_1 \cdot c_1^v)} \\ &\leq \frac{(\gamma+1)\Phi(G^+) + (b_{1j'} - A_1 \cdot c_{j'}^v) + \sum\limits_{b_{i1} \in \Phi_{opt}^{s_1}} (b_{i1} - A_i \cdot c_1^v)}{\Phi(G^+) + (b_{11} - A_1 \cdot c_1^v)}. \end{aligned} \tag{18}$$

Now, according to the foundation of the induction (i.e., $b_{11}/A_1 - c_1^v = w_{11} = \max\{w_{ij} | \langle r_i, s_j \rangle \in \mathcal{E}\}$), we can get

$$b_{1j'} - A_1 \cdot c_{j'}^v = w_{1j'} \cdot A_1 \leq w_{11} \cdot A_1 = b_{11} - A_1 \cdot c_1^v, \tag{19}$$

TABLE 2
Statistics of Workload Logs.

Trace	Number of jobs	Number of CPUs	Average number of CPUs	Average Run-Time per job (min)
DAS-2	$1,124,772$	400	4.3	6.2
SHARCNET	$1,195,242$	$6,828$	1.5	533
NorduGrid	$781,370$	$2,000$	1.1	$1,488$
AuverGrid	$404,176$	475	1	420

and

$$\sum_{b_{i1}\in\Phi^{s_1}_{opt}} (b_{i1}-A_i\cdot c^v_1)\le L_1\cdot w_{11}=\frac{L_1}{A_1}\cdot(b_{11}-c^v_1\cdot A_1) \quad (20)$$

$$\le\max\{\frac{L_j}{A_i}|r_i\in\mathcal{R},s_j\in\mathcal{S}\}\cdot(b_{11}-c^v_1\cdot A_1) \quad (21)$$

$$=\gamma\cdot(b_{11}-c^v_1\cdot A_1). \quad (22)$$

According to Eqs. 18, 19, 20, 21, and 22, we have

$$\frac{\Phi_{opt}(G)}{\Phi(G)}\le\frac{(\gamma+1)\Phi(G^+)+(1+\gamma)(b_{11}-c^v_1\cdot A_1)}{\Phi(G^+)+(b_{11}-c^v_1\cdot A_1)}=\gamma+1. \quad (23)$$

Based on the above induction, we conclude that $\alpha\le\gamma+1$ holds for all cases.

In addition, we prove that $\gamma+1$ is an urgent bound. We consider an extreme case in which $\mathcal{R}=\{r_1,r_2,r_3\}$, $\widehat{\mathcal{S}}=\{s_1,s_2\}$, $\{L_1=L+\delta,L_2=2L\}$, $\{A_1=L+\delta,A_2=A_3=L\}$, $c^v_1=c^v_2=c$, and $\widehat{\mathcal{B}}=\{b_{11}=(w+c)\cdot A_1,b_{12}=(w+\delta+c)\cdot A_1,b_{21}=(\delta+c)\cdot A_2,b_{22}=(w+c)\cdot A_2,b_{31}=(\delta+c)\cdot A_3,b_{32}=(w+c)\cdot A_3\}$, where L, w and c denote three arbitrary positive numbers and δ means a positive number which is infinitely close to zero. Here, we have $\gamma=L_2/A_2=2$. According to the greedy strategy used in Algorithm 2, we have $\Phi=\{b_{12},b_{21}\}$ and the social welfare is $(w+\delta)(L+\delta)+\delta\cdot L$. In contrast, we have $\Phi_{opt}=\{b_{11},b_{22},b_{32}\}$ and the corresponding social welfare is $w(L+\delta)+2wL$. In the worst case, the approximation ratio $\alpha=(3wL+L\delta)/(wL+w\delta+2L\delta+\delta^2)$ is infinitely close to $\gamma+1=3$. Thus, we conclude that $\gamma+1$ is an urgent bound.

According to this, the theorem holds. ■

5 Performance Evaluation

In this section, we evaluate the performance of the AVA mechanism by using extensive trace-driven simulations, and we compare it with two heuristic algorithms. We conduct the simulations on a computer with Intel(R) Core(TM) i5-3470 CPU @3.2GHz and 8GB RAM under a Windows platform. Moreover, all simulations are implemented in C/C++ language.

5.1 Compared Algorithms

In addition to the *WBS* algorithm for the winner selection problem in the simulations, we also implement two algorithms for comparison: "Nearest" and "Selfish" [24]. "Nearest" means that the platform always dispatches the requests with the smallest latency to an ECN under the capacity constraints. While in "Selfish", each request only cares about its own performance, and it expects to be assigned to the ECNs with the largest true valuations (i.e., v_{ij}) under the capacity constraints.

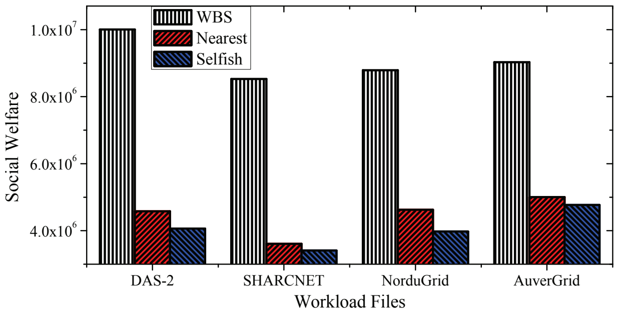


Fig. 5. Performance comparisons on the social welfare with different files (the numbers of requests and ECNs are 2000 and 20, respectively).

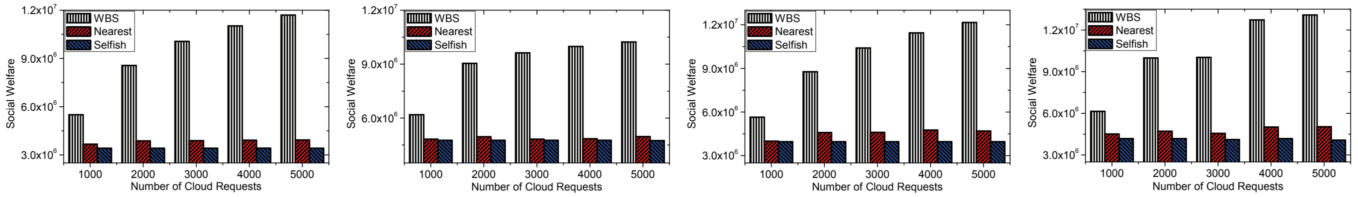
5.2 Simulation Settings

Because real user request data has not been publicly released by cloud providers yet, we adopt a widely-used dataset called Grid Workload Archive [1] in our simulations. Similar to the works [20, 34], we select four out of six available logs in the Grid Workload Archive. More specifically, these logs are called "DAS-2" (from a research grid at the Advanced School for Computing and Imaging in Netherlands), "SHARCNET" (from SHARCNET clusters installed at several academic institutions in Ontario, Canada), "NorduGrid" (from the NorduGrid system), and "AuverGrid" (from the AuverGrid system). In Table 2, we provide a brief description of the used workloads.

In the simulations, we generate a user request by extracting the information of a job from the four log files. For simplicity, we assume that the deadline of all requests can be met. Thus, we only care about the following parameters: the VM resource amount requested by each user (i.e., A_i), the capacity constraints of ECNs (i.e., L_j), and the unit cost of VM resources in ECNs (i.e., c^v_j). The amount A_i of VM resources for each request mainly depends on RunTime (the time each job needs to complete its execution) and ReqNumCPUs (the requested number of CPUs). Here, since there are no ECNs provided in the four log files, we artificially generate some ECNs in the simulations, each of which has a limited resource capacity and a unique unit cost. The number of ECNs is selected from $\{10,20,30,40,50\}$. Moreover, we generate the capacity constraints of the ECNs based on available CPUs in the four log files. The unit cost c^v_j is generated from $[1,20]$ randomly. Here, the generation process is based on the uniform distribution. Moreover, we directly generate the user bids between 1000 and 10000 for different ECNs. Since the number of jobs in the files is large, we use parts of the jobs as the user requests in our simulations, and the number of requests is selected from $\{1000,2000,3000,4000,5000\}$. In our simulations, the default numbers of ECNs and requests are 20 and 2000, respectively.

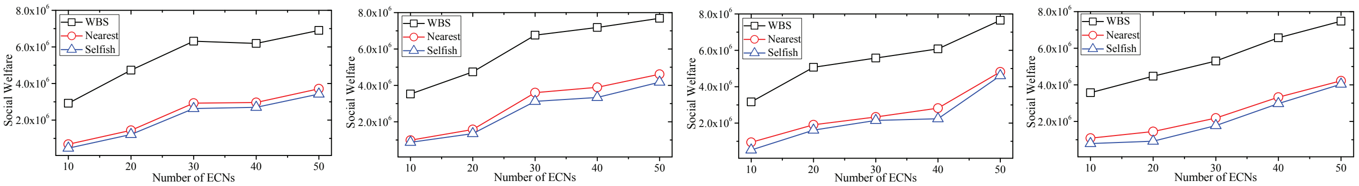
5.3 Simulation Results

In addition to the total social welfare, we also use the following metrics to evaluate the performance of our AVA mechanism: truthfulness, individual rationality, computational efficiency, and successful ratio. Truthfulness is the property that no request can improve its payoff by submitting a bid different from the real valuation; Individual rationality is the property that the payoff of each request is non-negative; Computational efficiency is the property that



(a) Workload file: DAS-2 (b) Workload file: SHARCNET (c) Workload file: NorduGrid (d) Workload file: AuverGrid

Fig. 6. Performance comparisons based on four different workload files: total social welfare vs. the number of cloud requests.



(a) Workload file: DAS-2 (b) Workload file: SHARCNET (c) Workload file: NorduGrid (d) Workload file: AuverGrid

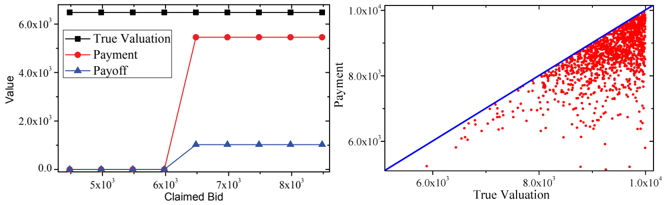
Fig. 7. Performance comparisons based on four different workload files: total social welfare vs. the number of ECNs.

the AVA mechanism can be conducted in polynomial time; Successful ratio is the ratio of the number of successfully assigned requests and the number of total requests.

Evaluation of Social Welfare: We first evaluate the performances of social welfare based on the four log files using the *WBS*, Nearest, and Selfish algorithms under the default settings. The results are shown in Fig. 5. We find that the social welfare obtained by the *WBS* algorithm is the largest, and *WBS* achieves about 126% better social welfare than the Nearest algorithm and about 148% better than Selfish. Then, we evaluate the performance (i.e., social welfare) of the three algorithms when we change the number of cloud requests based on the four used log files (DAS-2, SHARCNET, NorduGrid, AuverGrid). We present the results in Fig. 6. Also, we find that the *WBS* algorithm achieves at least twice the social welfare that the two compared algorithms achieve. Along with the increase in the number of cloud requests, the total obtained social welfare of all algorithms increases. In addition, we also evaluate the social welfare performances of the three algorithms when we change the number of ECNs based on the four used log files. The results are shown in Fig. 7. We also get that the *WBS* algorithm always outperforms the compared algorithms, along with the increase in the number of ECNs. These simulations validate our theoretical analysis results.

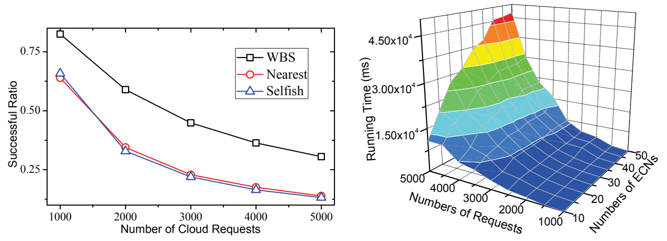
Evaluation of Truthfulness and Individual Rationality: We verify the truthfulness of our AVA mechanism by randomly picking a request and allowing it to submit a bid that is different from its true valuation. Here, the true valuation is fixed and given, which is equal to the originally submitted bid. Without loss of generality, we conduct the experiment based on the workload file "DAS-2". The result is illustrated in Fig. 8(a). We see that the payoff and payment are both zero when the bid claims a lower value, and they remain unchanged when the claimed bid is not less than the corresponding true valuation. Moreover, we verify the individual rationality of the AVA mechanism by comparing the true valuation of each bid and the corresponding payment under the default settings. The results are shown in Fig. 8(b). We get that the payment of each winning bid is less than its true valuation. These results are consistent with our theoretical analysis.

Evaluation of Successful Ratio: We also evaluate the successful ratio of the three algorithms, as shown in Fig. 9(a). When the number of requests changes from 1000 to 5000, the successful ratio decreases accordingly. This is because the number of successfully assigned requests remains nearly unchanged under the capacity constraints of the ECNs when the number of total requests increases. Moreover, the *WBS* algorithm achieves 36% and 40% higher successful ratios than the "Nearest" and "Selfish" algorithms, respectively.



(a) Truthfulness (b) Individual rationality

Fig. 8. Property verifications: truthfulness and individual rationality.



(a) Successful ratio (b) Computational efficiency

Fig. 9. Performance evaluations: successful ratio and computational efficiency.

Evaluation of Computation Efficiency: Finally, we verify the computational efficiency of the AVA mechanism. Along with the increase in the number of ECNs and the number of requests, the running time of the AVA mechanism increases, as shown in Fig. 9(b). When the number of ECNs is 50 and the number of requests is 5000, the execution time of AVA is less than 47s, which is much smaller than the auction cycle (dozens and hundreds of minutes). This means that the AVA mechanism can work efficiently in real applications. These simulation results remain consistent with our theoretical analysis.

6 Related Work

In this paper, we focus on the auction-based VM allocation problem for deadline-sensitive tasks in distributed edge cloud. So far, there has been much research on edge cloud computing, such as [2, 3, 6–8, 10, 12–17, 20–25, 29–39, 41].

The authors of [20, 36] and [23, 34] presented truthful auction mechanisms for dynamic resource provisioning and allocation in cloud computing. The authors of the former took heterogeneous user demands into account, while the latter designed combinatorial auction mechanisms. Furthermore, the authors in [12, 39] proposed auction mechanisms for dynamic VM provisioning and pricing across different geo-distributed data centers. The work [16, 37] studied the online auction mechanisms for Infrastructure-as-a-Service (IaaS) clouds, in which [37] focused on achieving the maximization of both social welfare and providers' profit, while [16] considered the unique features of an elastic model for inputting time-varying user demands and a unified model for requesting heterogeneous VMs together. Also, Zhang *et al.* [38] designed an efficient randomized auction mechanism for dynamic VM provisioning and pricing with $(1-\varepsilon)$-optimal social welfare in expectation. The authors in [35] presented a framework for truthful online auctions in cloud computing where users with heterogeneous demands can come and leave on the fly. Moreover, Anisetti *et al.* [5] modeled the multi-cloud provisioning scenario as a generalization of second price procurement e-auctions, in which the cloud service customers and providers are seen as the auctioneer and bidders, respectively. They proposed an e-auction mechanism based on matching and ranking algorithms to improve the truthfulness on the e-auction outcome. Zheng *et al.* [40] proposed the first family of strategy-proof double auctions for multi-cloud, multi-tenant bandwidth reservation, which can achieve strategy-proofness, ex-post budget balance, good social welfare, fine cloud bandwidth utilization, and high tenant satisfaction ratio, simultaneously. Different from the existing auction mechanisms, we design a truthful auction-based VM allocation mechanism for deadline-sensitive tasks in distributed ECNs, in which there exists the resource competition across multiple geographically distributed ECNs. Moreover, the deadline and capacity constraints are taken into consideration simultaneously.

On the other hand, the works [14, 24, 39] focused on balancing the workload between multiple edge cloud servers to minimize the total response time. Meanwhile, some other works [12, 13, 17] investigated how to configure edge clouds dynamically by proposing online resource placement methodologies. Additionally, [25] designed the edge cloud as a tree hierarchy of geo-distributed servers and proposed a workload placement algorithm that maximizes the amount of peak workloads. [8] studied the multi-user computation offloading problem for mobile edge cloud in a multi-channel wireless interference environment. Different from these works, which mainly study the workload scheduling and VM placement problem in edge clouds, we focus on VM allocation from an economic view.

The works most relevant to our problem are [15, 22, 30]. The authors in [15] designed two double auction mechanisms to stimulate cloudlets to serve nearby mobile devices, in which the authors simply assume that the mobile users' requests and cloudlet resources are homogeneous, and each cloudlet can serve only one request. The authors in [30] proposed two auction mechanisms for two task models, which is essentially a crowdsourcing-based task allocation problem. Neither of these works analyzes the approximation ratio of their proposed mechanisms. The authors in [22] designed an online auction mechanism for dynamic virtual cluster provisioning (including VM placement and inter-VM traffic routing) and pricing in geo-distributed clouds, which is modeled as an online combinational knapsack problem.

In contrast, we propose a truthful auction-based VM resource allocation mechanism for deadline-sensitive tasks in distributed edge cloud, which solves the problem of mobile users competing for VM resources in geographically distributed ECNs with capacity constraints. We model the problem as an n-to-one weighted bipartite graph matching problem with 0-1 knapsack constraints. Furthermore, we analyze the approximation ratio of the winner selection algorithm and further prove that the factor is urgent. Also, our problem differs from the latest research on the online knapsack problem and budgeted bipartite matching problem [27], in which two proposed online truthful algorithms can achieve competitive ratios of $2e$ and 24, respectively. Moreover, the auction method adopted in this paper is different from the four basic auction types, where we apply the rule of critical payment introduced by Myerson [19]. Meanwhile, the Vickrey-Clarke-Groves (VCG) auctions [26], based on the optimal allocation, cannot be applied in this paper, because the n-to-one weighted bipartite matching problem with 0-1 knapsack constraints cannot be solved optimally. The Generalized Second-Price auction [9], as a non-truthful auction mechanism for multiple items, is not suitable for the VM resource allocation problem, which needs to guarantee the truthfulness.

7 Conclusion and Future Work

In this paper, we have studied the problem of allocating heterogeneous VM resource requests to geo-distributed edge cloud nodes with capacity constraints in order to maximize the total social welfare. We propose a truthful auction-based VM resource allocation mechanism, i.e., AVA, which mainly consists of a greedy winning bid selection algorithm and a truthful payment determination algorithm. We prove that the AVA mechanism can ensure the properties of truthfulness, individual rationality, and computational efficiency. Moreover, we give the approximation ratio of the winner selection algorithm, and prove that it is an urgent bound. Finally, extensive simulations based on real traces verify the performance of the AVA mechanism. In the future research, we will study the VM resource allocation problem for the interaction-intensive cloud applications which involve the migration of cloud computing tasks.

References

[1] Grid workloads archive. Available [Online]: http://gwa.ewi.tudelft.nl, Aug. 2018.

[2] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie. Mobile edge computing: A survey. *IEEE Internet of Things Journal*, 5(1):450–465, 2018.

[3] A. Aissioui, A. Ksentini, A. Gueroui, and T. Taleb. On enabling 5g automotive systems using follow me edge-cloud concept. *IEEE Transactions on Vehicular Technology*, 67(6):5302–5316, 2018.

[4] Amazon. "Amazon Elastic Compute Cloud". Available [Online]: https://aws.amazon.com/cn/ec2/, Aug. 2018.

[5] M. Anisetti, C. A. Ardagna, P. A. Bonatti, E. Damiani, M. Faella, C. Galdi, and L. Sauro. E-auctions for multi-cloud service provisioning. In *2014 IEEE International Conference on Services Computing*, pages 35–42, 2014.

[6] A. Ceselli, M. Premoli, and S. Secci. Mobile edge cloud network design optimization. *IEEE/ACM Transactions on Networking*, 25(3):1818–1831, 2017.

[7] L. Chen, J. Wu, X. Zhang, and G. Zhou. TARCO: Two-stage auction for d2d relay aided computation resource allocation in hetnet. *IEEE Transactions on Services Computing*, pages 1–1, 2018.

[8] X. Chen, L. Jiao, W. Li, and X. Fu. Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Transactions on Networking*, 24(5):2795–2808, 2016.

[9] 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, 2007.

[10] G. Gao, M. Xiao, J. Wu, L. Huang, and C. Hu. Truthful incentive mechanism for nondeterministic crowdsensing with vehicles. *IEEE Transactions on Mobile Computing*, 17(12):2982–2997, 2018.

[11] L. Gao, T. H. Luan, S. Yu, W. Zhou, and B. Liu. Fogroute: Dtn-based data dissemination model in fog computing. *IEEE Internet of Things Journal*, 4(1):225–235, 2017.

[12] F. Hao, M. Kodialam, T. V. Lakshman, and S. Mukherjee. Online allocation of virtual machines in a distributed cloud. *IEEE/ACM Transactions on Networking*, 25(1):238–249, 2017.

[13] I.-H. Hou, T. Zhao, S. Wang, and K. Chan. Asymptotically optimal algorithm for online reconfiguration of edge-clouds. In *ACM MobiHoc*, pages 291–300, 2016.

[14] M. Jia, W. Liang, Z. Xu, and M. Huang. Cloudlet load balancing in wireless metropolitan area networks. In *IEEE INFOCOM*, pages 1–9, 2016.

[15] A.-L. Jin, W. Song, P. Wang, D. Niyato, and P. Ju. Auction mechanisms toward efficient resource sharing for cloudlets in mobile cloud computing. *IEEE Transactions on Services Computing*, 9(6):895–909, 2016.

[16] J. Li, Y. Zhu, J. Yu, C. Long, G. Xue, and S. Qian. Online auction for IaaS clouds: Towards elastic user demands and weighted heterogeneous VMs. In *IEEE INFOCOM*, pages 1–9, 2017.

[17] Y. Li and W. Wang. Can mobile cloudlets support mobile applications? In *IEEE INFOCOM*, pages 1060–1068, 2014.

[18] Microsoft. "Windows Azure". Available [Online]: http://www.windowsazure.com/, Aug. 2018.

[19] R. B. Myerson. Optimal auction design. *Mathematics of operations research*, 6(1):58–73, 1981.

[20] M. 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*, 26(2):594–603, 2015.

[21] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato. Hybrid method for minimizing service delay in edge cloud computing through vm migration and transmission power control. *IEEE Transactions on Computers*, 66(5):810–819, 2017.

[22] W. Shi, C. Wu, and Z. Li. An online mechanism for dynamic virtual cluster provisioning in geo-distributed clouds. In *IEEE INFOCOM*, pages 1–9, 2016.

[23] W. Shi, L. Zhang, C. Wu, Z. Li, and F. C. M. Lau. An online auction framework for dynamic resource provisioning in cloud computing. *IEEE/ACM Transactions on Networking*, 24(4):2060–2073, 2016.

[24] H. Tan, Z. Han, X.-Y. Li, and F. C. Lau. Online job dispatching and scheduling in edge-clouds. In *IEEE INFOCOM*, pages 1–9, 2017.

[25] L. Tong, Y. Li, and W. Gao. A hierarchical edge cloud architecture for mobile computing. In *IEEE INFOCOM*, pages 1–9, 2016.

[26] H. R. Varian and C. Harris. The VCG auction in theory and practice. *American Economic Review*, 104(5):442–45, 2014.

[27] R. Vaze. Online knapsack problem and budgeted truthful bipartite matching. In *IEEE INFOCOM*, pages 1–9, 2017.

[28] V. V. Vazirani. *Approximation algorithms*. Springer Science & Business Media, 2013.

[29] H. Wang, Z. Kang, and L. Wang. Performance-aware cloud resource allocation via fitness-enabled auction. *IEEE Transactions on Parallel and Distributed Systems*, 27(4):1160–1173, 2016.

[30] X. Wang, X. Chen, and W. Wu. Towards truthful auction mechanisms for task assignment in mobile device clouds. In *IEEE INFOCOM*, pages 1–9, 2017.

[31] X. Wang, K. Wang, S. Wu, D. Sheng, H. Jin, K. Yang, and S. Ou. Dynamic resource scheduling in mobile edge cloud with cloud radio access network. *IEEE Transactions on Parallel and Distributed Systems*, pages 1–1, 2018.

[32] Z. Xu, W. Liang, W. Xu, M. Jia, and S. Guo. Efficient algorithms for capacitated cloudlet placements. *IEEE Transactions on Parallel and Distributed Systems*, 27(10):2866–2880, 2016.

[33] H. Yin, X. Zhang, H. H. Liu, Y. Luo, C. Tian, S. Zhao, and F. Li. Edge provisioning with flexible server placement. *IEEE Transactions on Parallel and Distributed Systems*, 28(4):1031–1045, 2017.

[34] S. Zaman and D. Grosu. A combinatorial auction-based mechanism for dynamic VM provisioning and allocation in clouds. *IEEE Transactions on Cloud Computing*, 1(2):129–141, 2013.

[35] 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.

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

[37] X. Zhang, Z. Huang, C. Wu, Z. Li, and F. C. M. Lau. Online auctions in IaaS clouds: Welfare and profit maximization with server costs. *IEEE/ACM Transactions on Networking*, 25(2):1034–1047, 2017.

[38] X. Zhang, C. Wu, Z. Li, and F. C. Lau. A truthful (1-ε)-optimal mechanism for on-demand cloud resource provisioning. In *IEEE INFOCOM*, pages 1053–1061, 2015.

[39] J. Zhao, H. Li, C. Wu, Z. Li, Z. Zhang, and F. C. M. Lau. Dynamic pricing and profit maximization for the cloud with geo-distributed data centers. In *IEEE INFOCOM*, pages 118–126, 2014.

[40] Z. Zheng, Y. Gui, F. Wu, and G. Chen. Star: Strategy-proof double auctions for multi-cloud, multi-tenant bandwidth reservation. *IEEE Transactions on Computers*, 64(7):2071–2083, 2015.

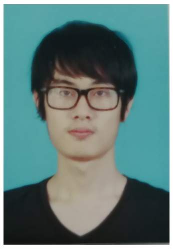
[41] Z. Zhou, F. Liu, Z. Li, and H. Jin. When smart grid meets geo-distributed cloud: An auction approach to datacenter demand response. In *IEEE INFOCOM*, pages 2650–2658, 2015.

Guoju Gao received his B.S. degree in information security from the University of Science and Technology of Beijing, Beijing, China, in 2014. He is currently working toward a PhD degree on computer science and technology with the School of Computer Science and Technology, the University of Science and Technology of China, Hefei, China. His research interests include mobile cloud computing, mobile crowdsourcing, privacy preservation, and incentive mechanism.

Mingjun Xiao is an associate professor in the School of Computer Science and Technology at the University of Science and Technology of China (USTC). He received his Ph.D. from USTC in 2004. His research interests include mobile crowdsensing, mobile social networks, vehicular ad hoc networks, mobile cloud computing, auction theory, data security and privacy. He has published more over 60 papers in referred journals and conferences, including TMC, TON, TPDS, TC, INFOCOM, ICNP, etc. He served as the TPC member of INFOCOM'18, ICDCS'15, Mobihoc'14, etc. He is on the reviewer board of several top journals such as TMC, TON, TPDS, TSC, TVT, TCC, etc.

Jie Wu is the Director of the Center for Networked Computing and Laura H. Carnell professor at Temple University. He also serves as the Director of International Affairs at College of Science and Technology. He served as Chair of Department of Computer and Information Sciences from the summer of 2009 to the summer of 2016 and Associate Vice Provost for International Affairs from the fall of 2015 to the summer of 2017. Prior to joining Temple University, he was a program director at the National Science Foundation and was a distinguished professor at Florida Atlantic University. His current research interests include mobile computing and wireless networks, routing protocols, cloud and green computing, network trust and security, and social network applications. Dr. Wu regularly publishes in scholarly journals, conference proceedings, and books. He serves on several editorial boards, including IEEE Transactions on Services Computing and the Journal of Parallel and Distributed Computing. Dr. Wu was general co-chair for IEEE MASS 2006, IEEE IPDPS 2008, IEEE ICDCS 2013, ACM MobiHoc 2014, IEEE ICPP 2016, and IEEE CNS 2016, as well as program co-chair for IEEE INFOCOM 2011 and CCF CNCC 2013. He was an IEEE Computer Society Distinguished Visitor, ACM Distinguished Speaker, and chair for IEEE Technical Committee on Distributed Processing (TCDP). Dr. Wu is a CCF Distinguished Speaker and a Fellow of the IEEE. He is the recipient of the 2011 China Computer Federation (CCF) Overseas Outstanding Achievement Award.

He Huang is an associate professor in the School of Computer Science and Technology at Soochow University, Soochow, China. He received the PhD degree in Department of Computer Science and Technology from University of Science and Technology of China in 2011. His current research interests include spectrum auction, privacy preserving in auction, wireless sensor networks, and algorithmic game theory. He is a member of IEEE computer society, and a member of ACM.



Shengqi Wang received his B.S. degree in Materials Science from the University of Science and Technology of China, Hefei, China in 2018. He is now working toward a Master Degree on Materials Science and Computer Science with the School of Engineering, Cornell University, Ithaca, USA. His research interests include mobile cloud computing, algorithmic game theory, machine learning and device-to-device communication.

Guoliang Chen received the B.S. degree from Xi'an Jiaotong University, China, in 1961. Since 1973, he has been with the University of Science and Technology of China, Hefei, China, a professor with the Department of Computer Science and Technology, and the director of the School of Software Engineering. From 1981 to 1983, he was a visiting scholar at Purdue University, West Lafayette, IN, USA. He is currently also the director of the National High Performance Computing Center at Hefei. He has published nine books and more than 200 research papers. His research interests include parallel algorithms, computer architectures, computer networks, and computational intelligence. Prof. Chen is an Academician of Chinese Academy of Sciences. He was the recipient of the National Excellent Teaching Award of China in 2003.