Pricing Model for Dynamic Resource Overbooking in Edge Computing

Zhiqing Tang, Fuming Zhang, Xiaojie Zhou, Weijia Jia, *Fellow, IEEE*, and Wei Zhao, *Fellow, IEEE*

Abstract—Edge Computing (EC) with cloud-like Quality of Service (QoS) can find its wide applications in various resource-constrained smart cities where the resource requirements can be different during peak and off-peak periods. During off-peak periods, there are often many resources that have been requested but not used, which can be reused to obtain higher profit. However, to the best of our knowledge, there is no effective pricing model or overbooking mechanism in EC. To fill in this gap, a novel pricing model for dynamic resource overbooking is proposed in this paper, specifically: 1) To meet the needs of different users in EC, methods of on-demand, daily, auction, and the new spot billing are designed, in which resources can be overbooked. 2) An auction approach with pricing rule and winner determination rule is designed for auction billing, which is proved to guarantee individual rationality, computational efficiency, and truthfulness. 3) To make more use of the auction approach to utilize idle resources, a dynamic resource overbooking mechanism is introduced, including a cancellation policy and a resource prediction method. The mechanism is validated with real-world data-trace. Experimental results show that the dynamic resource overbooking mechanism maximizes the profit of edge nodes with a high QoS Satisfaction ratio of on-demand and daily billing.

Index Terms—Auction, edge computing, pricing model, resource overbooking

1 Introduction

In recent years, due to the rapidly increasing number of mobile devices, cloud computing, which is relatively far from these devices, cannot meet applications that have strict requirements for delay or mobility, such as vehicle networks and wireless access networks [1], [2]. To compensate for these weaknesses in cloud computing, the Edge Computing (EC) [3] paradigm can play an important role. In EC, the computation resources of cloud data centers are partially offloaded to the decentralized edge nodes by deploying the edge nodes at the edge of the network [2]. Compared with cloud computing, decentralized edge nodes can not only support the mobility of tasks [4], but also significantly reduce delay and transmission cost while meeting the resource requirements of mobile tasks [5]. Besides, fulfilling delay requirements, EC can effectively support domain-specific large-scale distributed decision-making systems [6], such as the intelligent transportation system in cities [7], etc.

However, the delay requirements of resource-consuming applications may vary significantly in EC. Take the intelligent transportation system as an example: real-time traffic information processing, which has a distinct peak period, requires a strict guarantee of low delay. On the other hand, the emergency only takes a short time to be processed with the exact delay requirement. Both of the above delay-sensitive tasks are ideal scenarios for applying EC [7]. Therefore, how to efficiently allocate limited resources among multiple types of tasks is a critical issue in EC [8]. The dynamic overbooking mechanism and pricing model are proposed in this paper to solve the above problem. The practice of renting idle resources again is called overbooking [9], [10], which aims to minimize resource waste during off-peak periods. Moreover, the pricing model provides proper billing methods for regular rental and overbooking.

Existing research on the pricing model in EC mainly focuses on homogeneous tasks and single billing model. Bittencourt *et al.* [11] describe the pricing model in EC and propose a general architecture. They discuss its components, interfaces, and interactions but do not give a practical algorithm. Zhang *et al.* [12] propose a hierarchical Stackelberg game based pricing strategy to achieve high utility with a 3-layer model in EC. However, the task request submitted by each user is the same. To fill in these gaps, a pricing model including on-demand, daily, auction, and spot billing methods is proposed. The first two billing methods are designed for regular rental, while the latter two are designed for overbooking. Compared with our previous work [13], a new spot billing method is proposed.

- *Zhiqing Tang is with the Institute of Artificial Intelligence and Future Networks, Beijing Normal University, Zhuhai 519087, China. E-mail: domain@sjtu.edu.cn.*
- *Fuming Zhang and Xiaojie Zhou are with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China. E-mail: {zhangfuming-alex, szxjzhou}@sjtu.edu.cn.*
- *Weijia Jia is with the Institute of Artificial Intelligence and Future Networks, Beijing Normal University, Zhuhai 519087, China, and also with Guangdong Key Lab of AI and Multi-Modal Data Processing, BNU-HKBU United International College, Zhuhai 519087, China. E-mail: jiawj@bnu.edu.cn.*
- *Wei Zhao is with the CAS Shenzhen Institute of Advanced Technology, Shenzhen 518055, China. E-mail: zhao8686@gmail.com.*

Manuscript received 15 July 2020; revised 19 Mar. 2022; accepted 12 May 2022. Date of publication 20 May 2022; date of current version 7 June 2023.
This work was supported in part by Guangdong Key Lab of AI and Multi-modal Data Processing, United International College (UIC), Zhuhai under Grant 2020KSYS007, and in part by Chinese National Research Fund (NSFC) under Grant 61872239, in part by the Engineering & Tech Center of Artificial Intelligence and Future Educations of Beijing Normal University, Zhuhai, Guangdong, China; Science and Technology Development Fund of Macau SAR under Grant 0060/2019/A1, and in part by Zhuhai Science-Tech Innovation Bureau under Grants ZH22017001210119PWC and 28712217900001.
(Corresponding author: Weijia Jia.)
Recommended for acceptance by J. Wang.
Digital Object Identifier no. 10.1109/TCC.2022.3175610

Unlike the auction billing method, it has no maximum runtime limit. As long as the bid is higher than the current server's adjusted fee, users' tasks will be executed continuously, complementing the auction billing method.

Among these four billing methods, the auction theory has been extensively studied [14]. Jin *et al.* [15], [16] design two auction approaches in EC. However, both of them are based on homogeneous tasks. Wang *et al.* [17] consider heterogeneous tasks, but they aim to minimize the expenditure of users without considering the profit. Compared with the current work, the proposed auction approach mainly has the following improvements: 1) The heterogeneity of tasks (buyers) and servers (sellers) is considered. 2) The servers of one edge node are grouped to receive the bids from tasks, and each server can accept multiple winning tasks. 3) Multiple resource requirements of tasks are taken into consideration.

For overbooking, unlike in cloud computing, which has been widely used to improve resource utilization [9], [10], in EC, due to the heterogeneity of edge nodes and task types, an overbooking mechanism that can meet multiple needs is still in its infancy. Barbarossa *et al.* [18] propose a strategy to overbook the computation and communication resource based on the statistics of blocking events in mmW-mobile edge computing. Slim *et al.* [19] propose a costless service offloading strategy for distributed edge cloud considering resource overbooking. To efficiently overbook idle resources while ensuring high Quality of Service (QoS) for normal rental resource, the available resource needs to be dynamically determined as accurately as possible. Moreno *et al.* [9] and Imam *et al.* [20] predict the resource utilization through a neural network and overbook the resource based on predicted results. However, they do not regard Service Level Agreement Violations (SLAV). To solve this problem, Long Short-Term Memory (LSTM) [21] and residual network [22] based predictors are used to predict resource utilization supplemented by dynamic resource adjustment mechanisms similar to TCP congestion control [23]. To further improve the QoS satisfaction ratio, adaptive padding is added to the resource predictors, and a cancellation policy is introduced. Our experimental results show that the average QoS satisfaction ratio with the enhanced method can reach over 99.95%, and the total profit is increased by 7.82%.

Notably, this paper is an extended version of [13]. The following changes have been made in this extended version: 1) A new billing method, spot billing method, has been designed. The pricing model is further improved, and the system can provide users with more flexible services, as introduced in Section 2.2.2) Algorithm 1 Pricing Rule has been revised. Besides, Algorithm 1 is divided into two parts as shown in Section 3.1, in which the candidate assignment determination is divided into Algorithm 2.3) A new resource utilization predictor based on the residual network has been adopted, which can solve the degradation problem, making it easier for deep networks to train higher accuracy. 4) An adaptive padding mechanism has been added, further improving the prediction accuracy as shown in Section 4.1.5) A new algorithm, Algorithm 4 Cancellation Policy, has been added as described in Section 4.2 to restore the QoS satisfaction ratio from the next moment by canceling some auction tasks, releasing and recycling the resource they occupy. 6) A larger-scale simulation experiment has been conducted to help us better select hyperparameters while verifying the algorithm's effectiveness and redrew all the experimental results. 7) More detailed data processing and experimental settings have been added. And the main notations have been listed in Table 1 for better readability.

TABLE 1
Notations

n	Edge node ($n \in \mathbf{N}$)
$s_{n,j}$	jth server of edge node n
$s_{n,j}^{c_x}$	Resource capacity of $s_{n,j}$
t	Time slot
$\mathbf{B}(t)$	Mobile task set at time t
$b_i(t)$	ith task at time t ($b_i \in \mathbf{B}(t)$)
$t_i^s(t), t_i^e(t)$	Estimated start and end time of $b_i(t)$
$r_i(t)$	Resource request of $b_i(t)$
$v_{i,n}^c(t), v_{i,n}^m(t)$	CPU and memory valuation for node n of $b_i(t)$
$e_i(t)$	Expected billing method of $b_i(t)$
$L_{n,j}(t)$	QoS satisfaction level of $s_{n,j}$
$s_{n,j}^b(t), s_{n,j}^s(t)$	Resource overbooked by auction and spot billing
$s_{n,j}^o(t), s_{n,j}^d(t)$	Resource rented by on-demand and daily billing
$\sigma_i(t)$	Task assignment of $b_i(t)$
T_b	Time limit for tasks using auction billing
$s_{n,j}^a(t), \mathbf{A}(t)$	(Adjusted) Asking price of $s_{n,j}$
$w_{total}^{n,j}(t)$	Total revenue of $s_{n,j}$
$d_L^{n,j}(t)$	Discount rate of $s_{n,j}$
$C_{n,j}(t)$	Cost of $s_{n,j}$
$s_{n,j}^u(t), \mathbf{U}(t)$	Unused resource of $s_{n,j}$
$s_{n,j}^p(t)$	Predicted resource usage (on-demand, daily)
R	Total profit of the system
$\mathbf{C}_b(t), \mathbf{C}_s(t)$	Candidate task set and server set at time t
$\sigma_c(t)$	Candidate assignment at time t
$\mathbf{P}_c^b(t), \mathbf{P}_c^s(t)$	Payment of candidate task set and server set
$\mathbf{W}_b(t), \mathbf{W}_s(t)$	Winning task set and server set at time t
$\sigma_w(t)$	Winning assignment at time t
$\mathbf{P}_w^b(t), \mathbf{P}_w^s(t)$	Payment of winning task set and server set
th_{ex}	Extra threshold of available resource
$\mathbf{B}_a(t)$	Assigned auction tasks with end time after $t+1$
$s_{n,j}^f(t)$	Resource will be freed before $t+1$ of $s_{n,j}$
$s_{n,j}^r(t)$	Resource need to be recycled before $t+1$ of $s_{n,j}$
$\mathbf{B}_c(t)$	Auction tasks to be cancelled at time t
$s_{n,j}^{u'}(t)$	Adjusted unused resource of $s_{n,j}$ at time t
th_{up}, th_{lo}	Upper and lower thresholds of available resource
$s_{n,j}^{a_0}$	Base asking price of $s_{n,j}$

To summarize, an efficient pricing model for dynamic resource overbooking is proposed. The contributions are as follows:

1) A pricing model including on-demand, daily, auction, and spot billing methods is proposed, in which the resource can be overbooked according to different QoS requirements. A novel auction approach is designed for auction billing by applying pricing and winner determination rules and proving that the approach guarantees individual rationality, computational efficiency, and truthfulness.
2) Novel resource prediction methods based on LSTM and residual network are adopted, where an adaptive padding method and a threshold are used to improve the prediction accuracy. Furthermore, the dynamic resource overbooking mechanism, including a cancellation policy and QoS satisfaction ratio feedback based on the resource prediction, is introduced.

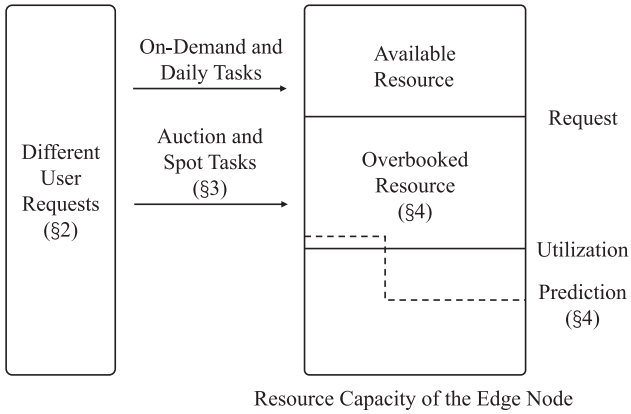


Fig. 1. Overview of the pricing model.

3) The algorithms are validated with real-world data-trace, and the experiment is scaled five times compared with [13]. The experimental results show that the auction approach can achieve desirable properties. In the meantime, the dynamic overbooking mechanism improves the profit by 51.58% under the premise of high QoS satisfaction ratio.

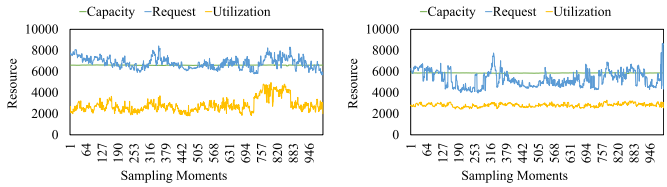
The remainder of this paper is organized as shown in Fig. 1. First, in Section 2, the system model is introduced, which primarily includes edge nodes and user tasks. Different user tasks are categorized into four billing methods, i.e., the on-demand, daily, auction, and spot billing methods. Second, for the auction billing in the pricing model, an online auction approach is proposed and analyzed in Section 3, which includes a pricing rule and a winner determination rule. Third, based on the four billing methods, the dynamic resource overbooking mechanism is illustrated in Section 4 to overbook resources as much as possible, where the resource utilization prediction method and the cancellation policy are used to improve prediction accuracy. Finally, the experimental settings and results are described in Section 5 and the paper is concluded in Section 6.

2 Modeling and Problem Formulation

In this section, the system model for EC is illustrated in 2.1, which includes mobile tasks and edge nodes. Then, the pricing model consisting of on-demand, daily, auction, and spot billing methods is introduced in 2.2. Finally, the problem is formulated in 2.3.

2.1 System Model

A three-layer Mobile-Edge-Cloud architecture is considered [2], [24]. Edge service providers gain revenue by leveraging the computation resources of edge nodes to tasks from mobile users [25]. Agencies with various requirements of delay and computation resources can rent these resources. Take the edge-assisted intelligent transportation system as an example [26], re-planning of traffic routes is a delay-sensitive task that requires long-term computation resources. Unexpected traffic accident information processing is also delay-sensitive but requires short-term computation resources. Moreover, road surveillance video processing is not delay-sensitive but needs more computation resources.



(a) CPU (b) Memory

Fig. 2. Data set overview.

The main notations have been listed in Table 1 for better readability compared with our previous work [13].

It is assumed that there is a set of heterogeneous and distributed edge nodes $\mathbf{N}$. For each edge node $n \in \mathbf{N}$, it consists of a set of physical servers $\mathbf{S}_n$, and $\mathbf{S} = \{\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_{|\mathbf{N}|}\}$ is used to denote all servers. The number of servers per edge node and the resource capacity of each server can be different. The resource capacity of server $s_{n,j} \in \mathbf{S}_n$ is denoted as $s_{n,j}^{cx} = \{s_{n,j}^{cc}, s_{n,j}^{cm}\}$, where $x \in \{c, m\}$ refers to the capacity of a certain resource. When $x = c$, it indicates the CPU resource. Otherwise when $x = m$, it indicates the memory resource.

In EC, mobile users generate tasks and offload them to the edge nodes. The set of tasks generated at time t is denoted as $\mathbf{B}(t)$. These tasks are divided into delay-sensitive and computation-oriented tasks. The former needs to be processed in time, while the latter requires a lot of computation resources. Specifically, the ith task is denoted as $b_i(t) = \{t_i^s(t), t_i^e(t), r_i(t), v_i(t), e_i(t)\}$, where $t_i^s(t)$ is the estimated start time, $t_i^e(t)$ is the estimated end time, $r_i(t)$ is the resource request, $v_i(t)$ is the valuation, and $e_i(t)$ is the expected billing method. For each $r_i(t) = \{r_i(t).c, r_i(t).m\}$, $r_i(t).c$ and $r_i(t).m$ are the requests of CPU and memory resource, respectively. $v_i(t) = \{v_{i,n}^c(t), v_{i,n}^m(t) | n \in \mathbf{N}\}$ contains the valuation of CPU and memory resource for each edge node. The valuation of each task is different due to the heterogeneity of the edge nodes [27]. Besides, the task will choose not to bid on too distant edge nodes by setting the valuation of the corresponding edge nodes to -1. The details of the expected billing method $e_i(t)$ are explained in the next subsection.

2.2 Pricing Model

As mentioned in the previous subsection, the edge nodes could profit by renting out computation resources. Generally, the edge nodes can rent out as many resources as the servers own. However, as shown in Fig. 2 from the data set [28], [29], the actual resource utilization is lower than the resource allocated and far lower than the resource capacity. To make more use of the resource, the allocated but unused resource is submitted for a second sale, called resource overbooking [9]. As the intelligent transportation system in cities mentioned above, most of the resources allocated to traffic information processing tasks are unused during off-peak periods, which can be used for a second sale through the auction. Many cloud service providers currently provide different billing methods, including Amazon AWS [30], Microsoft Azure [31], etc., but flexible billing methods are only adopted in cloud computing. In EC, most service providers currently provide billing by the number of requests. Therefore, applying the pricing model proposed in this paper to EC can effectively solve the problem of resource utilization of edge nodes and improve revenue.

The pricing model is designed for users to offload tasks to the edge nodes, which includes four billing methods, defined as $e_i(t) \in \{\text{on-demand, daily, auction, spot}\}$. The details are described as follows.

- *On-Demand(OD)*: This billing method is primarily for those delay-sensitive tasks with uncertain start time or processing time, e.g., the unexpected traffic accident information processing in EC. It charges a considerable fee while providing a fairly high QoS satisfaction ratio.
- *Daily*: This billing is designed for tasks that are delay-sensitive but with determined and relatively long processing time, e.g., the re-planning of traffic routes in EC. In such a billing method, computation resources are reserved, and bills are paid daily. Like the above billing method, it provides a high QoS satisfaction ratio for high fees.
- *Auction*: This billing serves to delay insensitive tasks which require many computation resources, e.g., road surveillance video processing in EC. The advantage is that it can provide the tasks' resources at a lower price. However, tasks billed in this way have runtime limits and will be evicted when the resource is depleted.
- *Spot*: It has some similarities with the auction billing method. The main difference is that tasks can be executed as long as the price does not surpass its bid. Under the dynamic price adjustment based on QoS satisfaction ratio feedback, this is a proper complement to the auction billing.

Compared with our previous work [13], the new billing method, spot billing, has been designed to further improve the pricing model and the system can provide users with more flexible services. Users can flexibly choose the billing methods according to different preferences of tasks' requirements. As mentioned in the above billing methods, a high QoS satisfaction ratio is provided for tasks which adopt on-demand or daily billing method. Since tasks in EC have an expected computation time or are expected to be completed as soon as possible, the QoS of tasks is closely related to the computation time. Besides, the computation time is proportional to SLAV [4], [32]. Then, the QoS satisfaction ratio $L(t) = \{L_{n,j}(t) | s_{n,j} \in \mathbf{N}\}$ of each server can be defined as [4], [32], [33]:

$$L_{n,j}(t) = \min\left\{\frac{s_{n,j}^{c_x} - s_{n,j}^{\text{ob}_x}(t)}{s_{n,j}^{\text{nr}_x}(t)}, 1 | x \in \{c, m\}\right\}, \tag{1}$$

where $s_{n,j}^{c_x}$ is the resource capacity, $s_{n,j}^{\text{ob}_x}(t)$ is the total overbooked resource and $s_{n,j}^{\text{nr}_x}(t)$ is the normally rented resource, i.e., $s_{n,j}^{\text{ob}_x}(t) = s_{n,j}^{b_x}(t) + s_{n,j}^{s_x}(t)$ and $s_{n,j}^{\text{nr}_x}(t) = s_{n,j}^{o_x}(t) + s_{n,j}^{d_x}(t)$, where $s_{n,j}^{b_x}(t)$ and $s_{n,j}^{s_x}(t)$ denote the resource overbooked by auction and spot billing of $s_{n,j}$, respectively. $s_{n,j}^{o_x}(t)$ and $s_{n,j}^{d_x}(t)$ are the resources rented by on-demand and daily billing with QoS satisfaction of $s_{n,j}$, respectively. $s_{n,j}^{b_x}(t)$ is obtained as:

$$s_{n,j}^{b_x}(t) = \sum_{t' \in \{t' | (b_i(t') | t_i^s(t') \le t \le t_i^e(t'), \sigma_i(t') = \{n,j\})\}} r_i^x(t'),$$

where $\sigma_i(t)$ denote the assignment of the task $b_i(t)$ at time t, e.g., $\sigma_i(t) = \{n, j\}$ means task $b_i(t)$ is assigned to $s_{n,j}$. Therefore, the above formula indicates that at time t, the resource of server $s_{n,j}$ overbooked by auction is obtained by summing the resource requests of those tasks assigned to the server $s_{n,j}$ and whose start time is less than or equal to t and end time is greater than or equal to t. As mentioned in the auction billing method, if $e_i(t)$ is auction, the task can last at most T_b time slots, which is defined as:

$$t_i^e(t) = \begin{cases} t_i^s + T_b, & \text{tie(t) - tis(t)} > \text{Tb} \\ t_i^e(t), & \text{tie(t) - tis(t)} \le \text{Tb} \end{cases}.$$

2.3 Problem Formulation

Our goal is to maximize the profit of the edge nodes through overbooking with a high QoS satisfaction ratio for those on-demand and daily tasks in EC. Moreover, the profit is defined as the revenue minus the cost intuitively.

The revenue of edge nodes is the sum of the payments of all tasks. For task $b_i(t)$, if the conventional billing method $e_i(t)$ is on-demand billing, its payment depends on its CPU and memory demands, which is defined as:

$$w_i^o(t) = \left(\sum_{x \in \{c,m\}} r_i^x(t) \times w_x^o\right) \times |t_i^e(t) - t_i^s(t)|,$$

where $w_x^o, x \in \{c, m\}$ is the on-demand price of corresponding resource. $t_i^s(t)$ and $t_i^e(t)$ are the start time and the end time of $b_i(t)$, respectively.

In addition, if $b_i(t)$ is a daily task, the payment is defined as:

$$w_i^d(t) = \left(\sum_{x \in \{c,m\}} r_i^x(t) \times w_x^d\right) \times t_d,$$

where $w_x^d, x \in \{c, m\}$ is the price of daily billing of the corresponding resource, and t_d is the amount of days. Generally, the prices of on-demand and daily tasks are fixed as constants [30].

Furthermore, if $b_i(t)$ is an auction task, the payment is defined as:

$$w_i^a(t) = \int_{t_i^s(t)}^{t_i^e(t)} \left(\sum_{x \in \{c,m\}} r_i^x(t) \times p_i^x(t)\right) dt,$$

where $p_i^x(t), x \in \{c, m\}$ is the price of corresponding resource which is determined through the auction approach described in Section 3.

Otherwise, if $b_i(t)$ is a spot task, the payment is defined as:

$$w_i^s(t) = \int_{t_i^s(t)}^{t_i^e(t)} \left(\sum_{x \in \{c,m\}} r_i^x(t) \times w_x^s \times 1[w_x^s > s_{n,j}^{a_x}(t)]\right) dt, \tag{2}$$

where $w_x^s, x \in \{c, m\}$ is the price of corresponding resource in the spot billing method. The ratio of user's bid to the asking price of the server is fixed. Besides, $1[\cdot]$ is Iverson bracket, whose value is equal to 1 when the condition in the bracket is satisfied. Otherwise it is 0. $s_{n,j}^{a_x}(t)$ is the asking price of corresponding resource. $\mathbf{A}(t) = \{s_{n,j}^a(t) | s_{n,j} \in \mathbf{S}\}$ is

used to represent the set of asking price for all servers, where $s^a_{n,j}(t) = \{s^{a_x}_{n,j}(t)|x \in \{c,m\}\}$ is the asking price of $s_{n,j}$ at t. Eq (2) states that the user will be billed only if the user's bid is higher than the asking price of the server.

Therefore, the total revenue of $s_{n,j}$ is defined as:

$$w^{n,j}_{total}(t) = d^{n,j}_L(t) \times \left(\sum_{t\in\{t|e_i(t)=\text{OD}\}} w^o_i(t) + \sum_{t\in\{t|e_i(t)=\text{daily}\}} w^d_i(t) \right) + \sum_{t\in\{t|e_i(t)=\text{auction}\}} w^a_i(t) + \sum_{t\in\{t|e_i(t)=\text{spot}\}} w^s_i(t), \sigma_i(t) = \{n,j\} \quad (3)$$

where $d^{n,j}_L$ is a discount rate. While aiming to maximize the profit with overbooking, the edge nodes may violate the QoS of on-demand and daily tasks. To ensure the high QoS satisfaction ratio of on-demand and daily billing, the revenue has to be reduced to punish such violation with a discount, which is defined as the SLA [34]:

$$d^{n,j}_L(t) = \begin{cases} 1, & L_{n,j}(t) \geq 99.95\% \\ 0.9, & 99\% \leq L_{n,j}(t) < 99.95\% \\ 0.75, & 95\% \leq L_{n,j}(t) < 99\% \\ 0, & L_{n,j}(t) < 95\% \end{cases}.$$

Besides, the cost of the servers mainly consists of the energy consumption of CPU and memory utilization [35], which is defined as:

$$C_{n,j}(t) = p_e \times \left(\sum_{x\in\{c,m\}} (s^{c_x}_{n,j} - s^{u_x}_{n,j}(t)) \times h_x \right),$$

where p_e is the unit price of electricity, $h_x, x \in \{c,m\}$ is the amount of power consumed per unit. Unused resources are denoted by $s^{u_x}_{n,j}(t)$, and $\mathbf{U}(t) = \{s^u_{n,j}(t)|s_{n,j} \in \mathbf{S}\}$ is used to represent the set of unused resources for all servers, where $s^u_{n,j}(t) = \{s^{u_x}_{n,j}(t)|x \in \{c,m\}\}$ is the unused resource for the server $s_{n,j}$ at t, which is obtained as:

$$s^{u_x}_{n,j}(t) = s^{c_x}_{n,j} - s^{p_x}_{n,j}(t) - (s^{b_x}_{n,j}(t) - s^{s_x}_{n,j}(t)), \quad (4)$$

where $s^{p_x}_{n,j}(t)$ is the predicted resource utilization of the tasks billed in on-demand and daily methods. The details of the resource utilization prediction methods are explained in Section 4.1.

To summarize, the problem is formulated as:

Problem 1.

$$\max R = \sum_{s_{n,j}\in\mathbf{S}} \sum_{t=0}^{T} \Big(w^{n,j}_{total}(t) - C_{n,j}(t) \Big),$$
$$s.t. \quad s^{o_x}_{n,j}(t) \leq s^{u_x}_{n,j}(t) \quad \forall s_{n,j} \in \mathbf{S}, \forall x \in \{c,m\}. \quad (5)$$

In Problem 1, the prices of on-demand and daily billing are fixed. The bid price of spot billing is related to the asking price of the server. However, the price of an auction is dynamically adjusted, and the overbooking ratio is also determined online. To solve Problem 1, an online auction approach and a dynamic overbooking mechanism are needed to overbook the resource as much as possible with a high QoS satisfaction ratio of on-demand and daily billing. Details of the online auction approach and dynamic overbooking mechanism are described in Section 3 and 4, respectively.

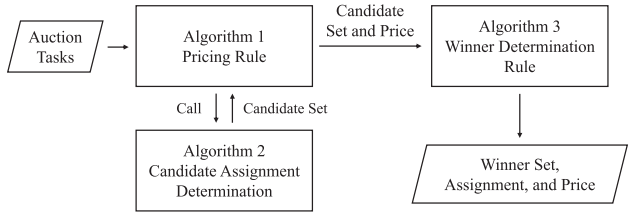


Fig. 3. Overview of the auction.

3 Online Auction Approach

In this section, the online auction approach is introduced in 3.1, which includes the pricing rule and the winner determination rule. Then, the approach is theoretically analyzed in 3.2. The overview of the auction is shown in Fig. 3. When the auction tasks arrive, the auctioneer first collects the tasks and servers' bids and then calls Algorithm 1 to get the pricing and candidate sets. Algorithm 1 first obtains the candidate sets of the tasks and the servers by calling Algorithm 2 according to the task bids and the remaining resources of servers. Then, Algorithm 1 further determines the auction price. After that, Algorithm 3 obtains the final winner sets of tasks and servers according to the candidate sets and pricing.

Compared with our previous work [13], Algorithm 1 Pricing Rule has been revised. After determining the candidate assignments among the servers and the tasks with the corresponding prices, the resource occupied by the pre-allocated auction tasks should be restored, which is not well considered in the previous version. Besides, Algorithm 1 is divided into two parts, in which the candidate assignment determination is divided into Algorithm 2.

3.1 Pricing Rule and Winner Determination Rule

A trusted third party, referred to as the auctioneer, must administer the auction between mobile tasks and servers in EC. The auctioneer first collects bids and the asking prices from the tasks and servers, respectively. Then it uses the pricing rule to determine the candidate assignments among the servers and the tasks with the corresponding prices. After that, the winner determination rule determines the winning bids for each server from the candidate assignments with the corresponding prices.

The auction approach should satisfy the following three properties [15], [17]:

- *Individual rationality*: No winning buyer is charged more than its bid, and no winning seller is rewarded less than its asking price.
- *Computational efficiency*: The auction outcome is tractable within polynomial time complexity.
- *Truthfulness*: The bid submitted by each mobile device should be truthful, i.e., no buyer can improve its utility by submitting a bid different from its true valuation.

Algorithm 1. Pricing Rule

Input: $\mathbf{A}(t), \mathbf{B}(t), \mathbf{U}(t)$
Output: $\mathbf{C}_b(t), \mathbf{C}_s(t), \mathbf{P}_c^b(t), \mathbf{P}_c^s(t)$
1: Set $\mathbf{C}_b(t), \mathbf{C}_s(t), \sigma_c(t), \mathbf{P}_c^b(t), \mathbf{P}_c^s(t), \mathbf{C}_b^p(t) = \emptyset$
2: **for** $n \in \mathbf{N}$ **do**
3: Set $\mathbf{V}_n = \{b_i(t) | v_{i,n}^x(t) \neq -1\}, \mathbf{W}_n = \{s_{n,j}\}$,
4: **for** $x \in \{c, m\}$ **do**
5: Call Algorithm 2 to get candidate assignment
6: **if** $|\mathbf{C}_n^{b_x}(t)| < |\mathcal{V}_i|$ and $|\mathbf{C}_n^{s_x}(t)| < |\mathcal{W}_i|$ **then**
7: Calculate $P^x(t)$ by Eq. (7)
8: **end if**
9: **if** $S_{n,|\mathbf{C}_n^{s_x}(t)|}^{a_x}(t) < P^x(t) < v_{|\mathbf{C}_n^{b_x}(t)|,n}^x$ **then**
10: $P_n^{b_x}(t) = P_n^{s_x}(t) = P^x(t)$
11: **else**
12: **for** $b_i(t) \in \mathbf{C}_n^{b_x}(t)$ **do**
13: **if** $\sigma_c^i(t) = \{n, |\mathbf{C}_n^{s_x}(t)|\}$ **then**
14: $\mathbf{C}_n^{b_x}(t) = \mathbf{C}_n^{b_x}(t) / \{b_i(t)\}$ and $\mathbf{C}_b^{p_x}(t) = \mathbf{C}_b^{p_x}(t) \bigcup \{b_i(t)\}$
15: **end if**
16: **end for**
17: $\mathbf{C}_n^{s_x}(t) = \mathbf{C}_n^{s_x}(t) / \{s_{n,|\mathbf{C}_n^{s_x}(t)|}\}$
18: $P_n^{b_x}(t) = \max\{v_{i,n}^x(t) | b_i(t) \in \mathbf{C}_b^{p_x}(t)\}$ and $P_n^{s_x}(t) = P_n^{b_x}(t)$
19: **end if**
20: **end for**
21: $\mathbf{C}_n^b(t) = \mathbf{C}_n^{b_c}(t) \bigcap \mathbf{C}_n^{b_m}(t)$ and $\mathbf{C}_n^s(t) = \mathbf{C}_n^{s_c}(t) \bigcap \mathbf{C}_n^{s_m}(t)$
22: **for** $s_{n,j} \in \mathbf{S}_n$ **do**
23: Reset $s_{n,j}^{u_c}(t)$ and $s_{n,j}^{u_m}(t)$
24: **end for**
25: **end for**
26: $\mathbf{C}_b(t) = \mathbf{C}_1^b(t) \bigcup \mathbf{C}_2^b(t) \ldots \bigcup \mathbf{C}_{|\mathbf{N}|}^b(t)$ and $\mathbf{C}_s(t) = \mathbf{C}_1^s(t) \bigcup \mathbf{C}_2^s(t) \ldots \bigcup \mathbf{C}_{|\mathbf{N}|}^s(t)$
27: $\mathbf{P}_c^b(t) = \{\mathbf{P}_n^b(t) | n \in \mathbf{N}\}$ and $\mathbf{P}_c^s(t) = \{\mathbf{P}_n^s(t) | n \in \mathbf{N}\}$
28: **end**

To satisfy these properties, the online auction approach is designed based on McAfee's mechanism [36], which achieves individual rationality and truthfulness. In McAfee's mechanism, one seller can only accept one buyer, which is not suitable in the EC scenario, while our auction approach, consisting of a pricing rule and a winner determination rule, can support one seller trading with multiple buyers. Similar to McAfee's mechanism, our auction approach is individually rational and truthful, and the theoretical analysis is shown in Section 3.2. In addition, the pseudocode of the pricing rule and winner determination rule can be found in Algorithms 1 and 3, respectively.

In Algorithm 1, the auctioneer first sorts the bids and the asking prices. Then the auctioneer determines the candidate assignments according to the sorted results and the remaining capacity of each server. After that, the price of each task and server is determined. Some notations used in Algorithm 1 are introduced as follows. $\mathbf{U}(t)$ is the unused resource, and $\mathbf{A}(t)$ is the set of asking prices for all servers. $\mathbf{C}_b(t)$ and $\mathbf{C}_s(t)$ are the sets of candidate tasks and servers, respectively. $\sigma_c(t)$ is the candidate assignment. $\mathbf{P}_c^b(t)$ is the set of payments of the buyer, and $\mathbf{P}_c^s(t)$ is the set of income of the seller. $\mathbf{C}_b^p(t)$ is the set of prices for candidate tasks.

As shown in Algorithm 1, servers of each edge node are grouped to receive bids. $\mathbf{V}_n$ and $\mathbf{W}_n$ are the sets of received bids and servers for edge node n, respectively. As shown in line 5, Algorithm 2 is called to determine candidate assignments. In Algorithm 2, first, the received bids and asking prices of each resource type are sorted in descending and ascending orders with results $\mathcal{V}_n$ and $\mathcal{W}_n$, respectively. For each task $b_i(t)$, the $b_i^{\{n,j\}}(t)$ is defined as

$$b_i^{\{n,j\}}(t) = \prod_{x \in \{c,m\}} 1[v_{i,n}^x(t) \geq s_{n,j}^{a_x}(t)] \times 1[r_k^x(t) \leq s_{n,j}^{u_x}(t)]. \tag{6}$$

As shown in lines 5 - 9, if $b_i^{\{n,j\}}(t) = 1$, it means that the bid of task $b_i(t)$ is larger than the asking price of server $s_{n,j}$, and the remaining capacity of $s_{n,j}$ is also larger than the request resource of $b_i(t)$ for both CPU and memory resource, then $b_i(t)$ can be assigned to $s_{n,j}$. The task $b_i(t)$ and server $s_{n,j}$ are added to the candidate assignment sets $\mathbf{C}_n^{b_x}(t)$ and $\mathbf{C}_n^{s_x}(t)$ of n with resource type x, respectively. Then the assignment $\sigma_c(t)$ is updated, task $b_i(t)$ is removed from the set of sorted tasks $\mathcal{V}_n$, and the available resource $s_{n,j}^u(t)$ is updated.

Algorithm 2. Candidate Assignment Determination

Input: $\mathbf{V}_n, \mathbf{W}_n, \mathbf{C}_n^b(t), \mathbf{C}_n^s(t), \sigma_c(t)$
Output: $\mathbf{C}_n^b(t), \mathbf{C}_n^s(t), \sigma_c(t)$
1: Sort $\mathbf{V}_n$ to $\mathcal{V}_n$ in descending order of $v_{i,n}^x(t)$, and sort $\mathbf{W}_n$ to $\mathcal{W}_n$ in ascending order of $s_{n,j}^{a_x}(t)$
2: **for** $s_{n,j} \in \mathcal{W}_n$ **do**
3: **for** $b_i(t) \in \mathcal{V}_n$ **do**
4: Calculate $b_i^{\{n,j\}}(t)$ by Eq. (6)
5: **if** $b_i^{\{n,j\}}(t) = 1$ **then**
6: $\mathbf{C}_n^{b_x}(t) = \mathbf{C}_n^{b_x}(t) \bigcup \{b_i(t)\}$ and $\mathbf{C}_n^{s_x}(t) = \mathbf{C}_n^{s_x}(t) \bigcup \{s_{n,j}\}$
7: $\sigma_c^i(t) = \{n, j\}$
8: $\mathcal{V}_n = \mathcal{V}_n / \{b_i(t)\}$
9: $s_{n,j}^{u_x}(t) = s_{n,j}^{u_x}(t) - r_i^x(t)$
10: **else**
11: **break**
12: **end if**
13: **end for**
14: **end for**
15: **end**

After this step, the price $\mathbf{P}(t)$ is determined in lines 7 - 20 in Algorithm 1. If there are unassigned bids and servers, the pricing of each resource type $\mathbf{P}(t) = \{P^x(t) | x \in \{c, m\}\}$ is calculated based on McAfee's mechanism as [36]

$$P^x(t) = \frac{v_{|\mathbf{C}_n^{b_x}(t)|+1,n}^x(t) + s_{n,|\mathbf{C}_n^{s_x}(t)|+1}^{a_x}(t)}{2}. \tag{7}$$

If $P^x(t)$ is between $S_{n,|\mathbf{C}_n^{s_x}(t)|}^{a_x}(t)$ and $v_{|\mathbf{C}_n^{b_x}(t)|,n}^x$, the price charged for bid $P_n^{b_x}(t)$ and the price rewarded to the server $P_n^{s_x}(t)$ are set to $P^x(t)$ [36]. Otherwise, all the assigned bids of the $|C_s^i(t)|$th server are cancelled, and the price is set as the highest bid of the $|C_s^n(t)|$th server. As shown in line 12, if task $b_i(t)$ is in the candidate assignment sets of n, it means that $b_i(t)$ meets both the requirements of CPU and memory resource, and then $b_i(t)$ is added to the assignment set $\mathbf{C}_n^b(t)$. After restoring the resource occupied by the pre-allocated auction tasks, the output is obtained as shown in lines 27–28.

Based on the pricing rule, the candidate bids with the corresponding prices $P_n^{b_x}$ and $P_n^{s_x}$ are determined. Then the winner determination is introduced in Algorithm 3. In Algorithm 3, $\mathbf{W}_b(t)$, $\mathbf{W}_s(t)$ are the sets of winning tasks and

servers, respectively. $\sigma_w(t)$ is the winning match between tasks and servers. In the winner determination, as shown in line 2 and line 5, the auctioneer first sorts servers and bids by a weighted sum of prices as follows:

$$w_p^{n,j}(t) = \alpha_p \times s_{n,j}^{a_c}(t) + \beta_p \times s_{n,j}^{a_m}(t), \tag{8}$$

$$w_p^{i}(t) = \alpha_p \times v_n^{k_c}(t) + \beta_p \times v_n^{k_m}(t), \tag{9}$$

where α_p and β_p control the weights. Then, the winning bid is determined from the candidate sets as shown in lines 7–14. If the server $s_{n,j}$ still has enough resource capacity, then task $b_i(t)$ is assigned to it. Besides, when the winning match $\sigma_w(t)$ is updated, the sets of winning tasks $\mathbf{W}_b(t)$ and servers $\mathbf{W}_s(t)$ are updated with the corresponding prices. After that, as shown in line 12, the $s_{n,j}^u(t)$ is updated.

Algorithm 3. Winner Determination Rule

Input: $\mathbf{A}(t), \mathbf{B}(t), \mathbf{U}(t), \mathbf{C}_b(t), \mathbf{C}_s(t), \mathbf{P}_c^b(t), \mathbf{P}_c^s(t)$
Output: $\mathbf{W}_b(t), \mathbf{W}_s(t), \sigma_w(t), \mathbf{P}_w^b(t), \mathbf{P}_w^s(t)$
1: Set $\mathbf{W}_b(t), \mathbf{W}_s(t), \sigma_w(t), \mathbf{P}_w^b(t), \mathbf{P}_w^s(t) = \emptyset$, and calculate $w_p^{n,j}(t)$ by Eq. (8)
2: Sort $\mathbf{C}_s(t)$ to $\mathcal{S}$ in ascending order of $w_p^{n,j}$
3: **for** $s_{n,j} \in \mathcal{S}$ **do**
4: Calculate $w_p^i(t)$ by Eq. (9)
5: Sort $\mathbf{C}_n^b(t)$ to $\mathcal{C}_b^n$ in descending order of $w_p^i(t)$
6: Set $\mathcal{C}_b^n = \mathcal{C}_b^n - W_b(t)$
7: **for** $b_i(t) \in \mathcal{C}_b^n$ **do**
8: **if** $s_{n,j}^{u_c}(t) \geq r_i^c(t)$ and $s_{n,j}^{u_m}(t) \geq r_i^m(t)$ **then**
9: $\sigma_w^i(t) = \{n, j\}$ and $\sigma_w(t) = \sigma_w(t) \bigcap \{\sigma_w^i(t)\}$
10: $\mathbf{W}_b(t) = \mathbf{W}_b(t) \bigcap \{b_i(t)\}$ and $\mathbf{W}_s(t) = \mathbf{W}_s(t) \bigcap \{s_{n,j}\}$
11: $\mathbf{P}_w^b(t) = \mathbf{P}_w^b(t) \bigcap \{\mathbf{P}_i^b(t) | \mathbf{P}_i^b(t) \in \mathbf{P}_c^b(t)\}$ and $\mathbf{P}_w^s(t) = \mathbf{P}_w^s(t) \bigcap \{\mathbf{P}_n^s(t) | \mathbf{P}_n^s(t) \in \mathbf{P}_c^s(t)\}$
12: $s_{n,j}^{u_c}(t) = s_{n,j}^{u_c}(t) - r_i^c$ and $s_{n,j}^{u_m}(t) = s_{n,j}^{u_m}(t) - r_i^m$
13: **end if**
14: **end for**
15: **end for**
16: **end**

3.2 Theoretical Analysis

In this subsection, the auction approach is proved to hold the properties of computational efficiency, individual rationality, and truthfulness.

Theorem 1. *The proposed auction approach achieves the individual rationality for each bid.*

Proof. In Algorithm 1, there are two cases for task $b_i(t)$ to be assigned as a buyer candidate and for server $s_{n,j}$ to become a seller candidate.

- $S_{n,|\mathbf{C}_n^{s_x}(t)|}^{a_x}(t) < P^x(t) < v_{|\mathbf{C}_n^{b_x}(t)|,n}^x(t)$: In this case, task $b_i(t)$ must have an actual bid price $v_i(t)$ that $v_{i,n}^x(t) \geq v_{|\mathbf{C}_n^{b_x}(t)|,n}^x(t)$. due to the descending order in the set of sorted tasks $\mathcal{V}_n$. So that $v_{i,n}^x(t) \geq v_{|\mathbf{C}_n^{b_x}(t)|,n}^x(t) > P^x(t)$. Moreover, the server must have an asking price $s_{n,j}^{a_x}(t) \leq s_{n,|\mathbf{C}_n^{s_x}(t)|}^{a_x}(t)$ due to the ascending order in the set of sorted servers $\mathcal{W}_i$, so each asking price of servers satisfies that $s_{n,j}^{a_x}(t) \leq s_{n,|\mathbf{C}_n^{s_x}(t)|}^{a_x}(t) < P^x(t)$.
- Otherwise, $P^x(t) \notin [s_{i,|\mathbf{C}_n^{s_x}(t)|}^{a_x}(t), v_{|\mathbf{C}_n^{b_x}(t)|,n}^x(t)]$. In this case, the price is set to $\max\{v_{i,n}^x(t) | b_i(t) \in \mathbf{C}_b^p\}$. For each task $b_i(t)$, it can be easily obtained that $v_{i,n}^x(t) \geq v_{|\mathbf{C}_n^{b_x}(t)|,n}^x(t) > P^x(t)$. And for each server $s_{n,j}$, it can be obtained that $s_{n,j}^{a_x}(t) \leq s_{n,|\mathbf{C}_n^{s_x}(t)|}^{a_x}(t)$. In addition, it is obvious that $s_{n,|\mathbf{C}_n^{s_x}(t)|}^{a_x}(t) \leq \max\{v_{i,n}^x(t) | b_i(t) \in \mathbf{C}_b^p\}$. So $s_{n,j}^{a_x}(t) \leq \max\{v_{i,n}^x(t) | b_i(t) \in \mathbf{C}_b^p\}$.

Therefore, each buyer assigned in Algorithm 1 is never charged a price higher than its bid. In contrast, each seller assigned is rewarded a payment no less than its asking price, ensuring individual rationality for buyers and sellers. □

Theorem 2. *The proposed auction approach is computationally efficient.*

Proof. To analyze the time complexity, $|\mathbf{S}|$, $|\mathbf{B}(t)|$, and $|\mathbf{N}|$ are used to denote the total number of servers, the number of tasks, and the number of edge nodes, respectively. To implement Algorithm 1, for each edge node and each resource type, first, the tasks and the servers are sorted with a time complexity $O(|\mathbf{B}(t)| \log |\mathbf{B}(t)|)$ and $O(|\mathbf{S}| \log |\mathbf{S}|)$, respectively. Then, the time complexity for Algorithm 2 is $O(|\mathbf{S}||\mathbf{B}(t)|)$. The time complexities of obtaining $P.x$ and pricing are $O(1)$ and $O(|\mathbf{B}(t)|)$, respectively. In total, the time complexity of Algorithm 1 is $O(|\mathbf{N}||\mathbf{B}(t)| \log |\mathbf{B}(t)|) + O(|\mathbf{N}||\mathbf{S}| \log |\mathbf{S}|) + O(|\mathbf{N}||\mathbf{S}||\mathbf{B}(t)|)$.

In Algorithm 3, the servers are sorted with time complexity $O(|\mathbf{S}| \log |\mathbf{S}|)$. Then, within the first for-loop, the bids are sorted with time complexity $O(|\mathbf{B}(t)| \log |\mathbf{B}(t)|)$, and the time complexity of winning assignment is $O(|\mathbf{N}||\mathbf{B}(t)|)$. In total, Algorithm 3 has a time complexity of $O(|\mathbf{S}|(|\mathbf{S}| \log |\mathbf{S}| + |\mathbf{B}(t)| \log |\mathbf{B}(t)| + |\mathbf{N}||\mathbf{B}(t)|))$. Therefore, the overall time complexity is polynomial. □

Theorem 3. *The proposed auction approach is truthful.*

We should prove that each mobile task will honestly submit all of its actual costs to demonstrate the truthfulness. The proposed mechanism is truthful if and only if the following two conditions are satisfied [17], [37]: 1) the winner determination algorithm is monotonic, and 2) each winning bid pays the critical value. The definitions of monotonicity and critical value are described as follows:

Definition 1. Monotonicity*: For each task $b_{i_1}(t)$, if $b_{i_1}(t)$ wins, then $b_{i_2}(t)$ also wins, where the corresponding bids of $b_{i_1}(t)$ and $b_{i_2}(t)$ are $v_{i_1}(t)$ and $v_{i_2}(t) = v_{i_1}(t) + \epsilon (\epsilon > 0)$, respectively.*

Definition 2. Critical Value*: For each task $b_i(t)$, there is a critical value $P_i^b(t)$. If the bid of $b_i(t)$ declares a cost that is not larger than $P_i^b(t)$, it must win. Otherwise, it will lose.*

Lemma 1. *The winner determination process in Algorithm 3 is monotonic.*

Proof. Assume that $b_{i_k}(t)$ is one of the winning tasks determined in the kth step of Algorithm 3, which means $k-1$ tasks have won in the previous $k-1$ steps. Let $(b_{i_1}(t), b_{i_2}(t), \ldots, b_{i_k}(t))$ be the list of the winning tasks that have been determined in the first k steps. If $b_{i_k}(t)$ was replaced by another task, e.g., $b_{i_j}(t)$, where the corresponding bids of

$b_{i_k}(t)$ and $b_{i_j}(t)$ are $v_{i_k}(t)$ and $v_{i_j}(t) = v_{i_k(t)} + \sigma(\sigma > 0)$, respectively. According to Algorithm 3, $b_{i_j}(t)$ must win in the kth step or even earlier step. As a result, the auction approach is monotonic. □

Lemma 2. *The winning bid in Algorithm 3 pays the critical value.*

Proof. It is assumed that task $b_{i_l}(t)$ wins its bid for server $s_{n,j}$ in the lth step of Algorithm 3. In this case, the payment of $b_{i_l}(t)$ is set to $P_n^b(t)$. For $\gamma > 0$, another bid with submitted price $v_{i_l}(t) = P_n^b(t) + \gamma$ would win, because its cost per unit resource must be higher than the valuation of $b_{i_l}(t)$. But the bid $v_{i_l}(t) = P_n^b(t) - \gamma$ will not win, as its valuation must be lower than the valuation of task $b_{i_l}(t)$. Hence, the above lemma is proved. □

According to the above analysis, the following theorem can be easily obtained through Lemmas 1, and 2 [17], [37]. Hence, the theorem is proved.

The auction approach solves the pricing of the resource of auction billing while guaranteeing the individual rationality, computational efficiency, and truthfulness. Furthermore, based on the pricing model consisting of the auction and three other billing methods, the unused resource can be overbooked to achieve more profit. The dynamic overbooking mechanism is described in the next section.

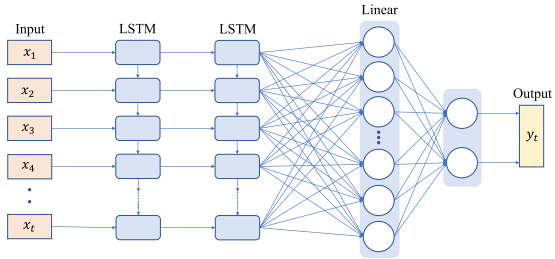
4 Dynamic Overbooking Mechanism

In this section, the dynamic overbooking mechanism based on resource utilization prediction and QoS satisfaction ratio feedback is presented. The resource utilization prediction with deep neural networks, cancellation policy, and the dynamic overbooking mechanism is presented in 4.1, 4.2 and 4.3, respectively.

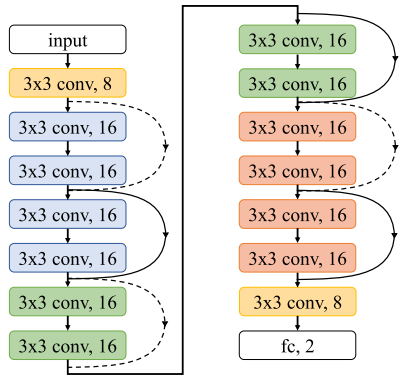
4.1 Resource Utilization Prediction

The neural network is used to make resource utilization predictions, including the RNN and CNN. The motivation behind the RNN is to make full use of the sequentiality of the information, i.e., the time-sequential resource utilization. An LSTM [21] based resource utilization predictor is proposed whose architecture is shown in Fig. 4a. The network is composed of two LSTM layers and one output layer.

Furthermore, inspired by the residual network framework [22], another predictor is implemented [22], containing 14 convolution layers, as shown in Fig. 4b. Each convolution layer is followed by batch normalization and a non-linear activation layer. There are three residual blocks in this network architecture, marked in blue, green, and red from top to bottom. A dropout layer follows each of these three residual blocks. The network ends with a 2-way fully-connected layer. As Fig. 4b shows, shortcut connections are inserted in the network. The identity shortcuts can be directly used when the input and output are of the same dimensions (solid line shortcuts in the figure). An additional convolution is performed when the dimensions increase or decrease (dotted line shortcuts in the figure), followed by a batch normalization. Compared with the previous LSTM based network [13], the shortcut connections introduced by the residual network can solve the degradation problem, making it easier for deep networks to train higher accuracy.



(a) LSTM based Predictor



(b) Residual Network based Predictor

Fig. 4. Architecture of different predictors.

However, when calculating the number of unused resources based on the predicted resource utilization and then overbooking this part of the resource, some inaccurate prediction result will cause overbooking of too much resource, which will lead to the decline of the QoS satisfaction ratio for those on-demand and daily tasks. To solve this problem, the following two solutions are proposed. On the one hand, the adaptive padding method is used to calculate the deviation between the predicted value and the actual value in the past few time slots [38]. We add these deviations together and compensate for the current prediction result. Since the goal is to avoid excessive overbooking as much as possible, when accumulating deviations, only those cases where the predicted value is less than the actual value are counted. On the other hand, an extra hyper-parameter th_{ex} is set, which is used to adequately reduce the predicted unused resource for each server at each time slot. The new adaptive padding mechanism can further improve the prediction accuracy. Since the neural network predictor is pre-trained, it cannot correct the wrong predictions. The introduction of this mechanism can appropriately correct the prediction error compared with our previous work [13].

With the adaptive padding method and the hyper-parameter th_{ex}, we still cannot ensure there is no excessive overbooking. To restore the declined QoS Satisfaction ratio of on-demand and daily tasks caused by the inaccurate prediction of resource utilization, the cancellation policy for auction tasks is designed in Algorithm 4 based on our previous work [13]. The details are described in the next section.

4.2 Cancellation Policy

In the preliminary version of this paper [13], without cancellation policy, the decline in QoS satisfaction ratio due to the

excessive overbooking will not be restored in a short period of time. Therefore, the cancellation policy is proposed to restore the QoS satisfaction ratio from the next moment by canceling some auction tasks, releasing and recycling the resource they occupy. Some notations used in Algorithm 4 are introduced as follows. $\mathbf{B}_a(t)$ is the set of assigned auction tasks of $s_{n,j}$ with end time after $t+1$ and $\mathbf{B}_c(t)$ is the set of auction tasks that will be canceled at the next time slot. $s^f_{n,j}(t)$ denotes the resource used by auction tasks that will be released before the next time slot, while $s^r_{n,j}(t)$ denotes the resource that needs to be recycled.

Algorithm 4. Cancellation Policy

Input:$\mathbf{B}(t)$
Output:$\mathbf{B}_c(t)$
1: Set $\mathbf{B}_c(t) = \emptyset$
2: **for** $s_{n,j} \in \mathbf{N}$ **do**
3: Calculate $s^f_{n,j}(t)$ and $s^r_{n,j}(t)$ by Eqs. (10) and (11), respectively
4: Set $\mathbf{B}_a(t) = \{b_i(t') | t^e_i(t') > t+1, \sigma_{t'}\{i\} = \{n,j\}\}$
5: Calculate $w^i_r(t)$ by Eq. (12), and sort $\mathbf{B}_a(t)$ to $\mathcal{B}_a$ in ascending order of $w^i_r(t)$
6: **for** $b_i(t') \in \mathcal{B}_a$ **do**
7: **if** $s^{r_c}_{n,j}(t) < 0$ and $s^{r_m}_{n,j}(t) < 0$ **then**
8: $s^{r_c}_{n,j}(t) = s^{r_c}_{n,j}(t) + r^c_i(t')$ and $s^{r_m}_{n,j}(t) = s^{r_m}_{n,j}(t) + r^c_i(t')$
9: $\mathbf{B}_a = \mathbf{B}_a / \{b_I(t')\}$ and $\mathbf{B}_c(t) = \mathbf{B}_c(t) \bigcup \{b_i(t')\}$
10: **else**
11: **break**
12: **end if**
13: **end for**
14: **for** $x \in \{c, m\}$ **do**
15: Sort $\mathbf{B}_a(t)$ to $\mathcal{B}_a$ in ascending order of $r^x_i(t)$
16: **for** $b_i(t') \in \mathcal{B}_a$ **do**
17: **if** $s^{r_x}_{n,j}(t) < 0$ **then**
18: $s^{r_x}_{n,j}(t) = s^{r_x}_{n,j}(t) + r^x_i(t')$
19: $\mathbf{B}_a = \mathbf{B}_a / \{b_i(t')\}$ and $\mathbf{B}_c(t) = \mathbf{B}_c(t) \bigcup \{b_i(t')\}$
20: **else**
21: **break**
22: **end if**
23: **end for**
24: **end for**
25: **end for**
26: **end**

In Algorithm 4, as shown in lines 3 - 5, $s^f_{n,j}(t)$, $s^r_{n,j}(t)$, $\mathbf{B}_a(t)$, and $w^i_r(t)$ are calculated according to Eqs. (10) - (12). $s^f_{n,j}(t) = \{s^{f_x}_{n,j}(t) | x \in \{c,m\}\}$ is used to denote the resource need to be released, which counts the resource occupied by those auction tasks whose estimated end time is less than or equal to the next time slot, and is defined as follows:

$$s^{f_x}_{n,j}(t) = \sum_{t' \in \{t' | (b_i(t') | t^e_i(t') \le t+1, \sigma_{t'}\{i\} = \{n,j\})\}} r^x_i(t'). \tag{10}$$

$s^r_{n,j}(t) = \{s^{r_x}_{n,j}(t) | x \in \{c,m\}\}$ is used to indicate the resource need to be recycled and it is defined as:

$$\begin{aligned} s^{r_x}_{n,j}(t) = {} & s^{c_x}_{n,j} - s^{o_x}_{n,j}(t) - s^{d_x}_{n,j}(t) \\ & - s^{b_x}_{n,j}(t) - s^{s_x}_{n,j}(t) + s^{f_x}_{n,j}(t). \end{aligned} \tag{11}$$

As shown in Eq. (11), if $s^r_{n,j}(t)$ is negative, it means that the current resource usage does not exceed the server capacity. Therefore, no auction tasks need to be canceled.

The weighted sum of auction tasks' CPU and memory usage is defined as:

$$w^i_r(t) = \alpha_r \times r_i(t).c + \beta_r \times r_i(t).m. \tag{12}$$

$w^i_r(t)$ is taken as a sorting criterion to sort $\mathcal{B}_a$ in ascending order. Here $\mathcal{B}_a$ denotes those auction tasks whose estimated end time is larger than the next time slot. After that, from lines 6 to 13, if both CPU and memory have resource need to be recycled, then the resource of the current task will be reclaimed, and this task is removed from the candidate task set and added to $\mathbf{B}_c(t)$, which is the set of tasks to be canceled. Finally, the operations performed within the loop from lines 14 to 24 are similar to the above, except for a specific type of resource.

4.3 Dynamic Overbooking Mechanism

To maximize the profit of the edge nodes through overbooking with a high QoS satisfaction ratio for the on-demand and daily tasks, the dynamic overbooking mechanism is described as follows.

Algorithm 5. Dynamic Overbooking

Input:$\mathbf{A}(t), \mathbf{B}(t), \mathbf{U}(t), \mathbf{L}(0) = \{1\}$
Output:R
1: **for** $t \in [1, T]$ **do**
2: **for** $s_{n,j} \in \mathbf{N}$ **do**
3: Get $s^p_{n,j}(t)$ from neural network
4: Calculate $s^u_{n,j}(t)$ by Eq. (4)
5: **if** $L_{n,j}(t-1) < 1$ **then**
6: $s^{u'}_{n,j}(t) = 0$
7: **else**
8: Calculate $s^{u'}_{n,j}(t)$ by Eq. (13)
9: **end if**
10: Calculate $s^a_{n,j}(t)$ by Eq. (14)
11: **end for**
12: Call Algorithm 1 Pricing Rule and Algorithm 3 Winner Determination Rule
13: Calculate $\mathbf{L}(t) = \{L_{n,j}(t) | s_{n,j} \in \mathbf{N}\}$ by Eq. (1)
14: Call Algorithm 4 Cancellation Policy
15: **end for**
16: Calculate R by Eq. (5)
17: **end**

The procedure of the dynamic overbooking mechanism is shown in Algorithm 5. For each time slot t, the servers first get the predicted resource utilization from the neural network and calculate the available resource for overbooking. Then, the servers adjust the available resource according to the QoS satisfaction ratio feedback and update the asking price. After that, the auctioneer collects the necessary information and calls Algorithms 1 and 2. Finally, the QoS satisfaction ratio is updated, Algorithm 4 is called, and the profit is calculated.

As shown in Algorithm 5, the predicted resource utilization of each server $s^{p_x}_{n,j}(t)$ is obtained from the neural network in lines 3–4. Then, the available CPU and memory resources are obtained according to Eq. (4).

Then, as shown in lines 5–10, the available resource for overbooking is adjusted and the asking price is updated. To improve the QoS satisfaction ratio, a TCP congestion control like dynamic available resource adjustment is adopted [23]. When the QoS satisfaction ratio $L_{n,j}(t-1) < 1$, the available resource for auction billing $s^{u'}_{n,j}(t)$ of server $s_{n,j}$ is set to 0. Otherwise, $s^{u'}_{n,j}(t) = \{s^{u'x}_{n,j}(t) | x \in \{c, m\}\}$ is obtained as:

$$s^{u'x}_{n,j}(t) = \begin{cases} \min\{s^{u'x}_{n,j}(t-1) + \epsilon) \times \alpha_u, s^{ux}_{n,j}(t)\}, \\ \qquad\qquad \mathrm{s^{ux}_{n,j}(t)} > \mathrm{th_{up}} \\ \min\{s^{u'x}_{n,j}(t-1) + \beta_u, s^{ux}_{n,j}(t)\}, \\ \qquad\qquad \mathrm{th_{lo}} < \mathrm{s^{ux}_{n,j}(t)} < \mathrm{th_{up}} \\ \min\{s^{u'x}_{n,j}(t-1), s^{ux}_{n,j}(t)\}, \\ \qquad\qquad \mathrm{s^{ux}_{n,j}(t)} < \mathrm{th_{lo}}, \end{cases} \tag{13}$$

where ϵ is a small positive constant that ensures $s^{u'}_{n,j}(t) \neq 0$, α_u and β_u are the parameters controlling the increment speed of available resource. Moreover, th_{up} and th_{lo} are the over and under threshold of available resource, respectively. When the QoS satisfaction ratio is less than 1, the available resource for auction is set to 0. Then the available resource is updated according to Eq. (13). After that, the asking price $s^a_{n,j}(t)$ is calculated by Eq. (14), which should be larger than the base asking price $s^{a_0}_{n,j}$ of each server. $s^a_{n,j}(t) = \{s^{a_x}_{n,j}(t) | x \in \{c, m\}\}$ is obtained as:

$$s^{a_x}_{n,j}(t) = \begin{cases} \frac{1}{L_{n,j}(t-1)} \times \frac{1-s^{u_x}_{n,j}t}{1-th_{up}.x} \times s^{a_0x}_{n,j}, \ \mathrm{s^{ux}_{n,j}t} > \mathrm{th^x_{up}} \\ \frac{1}{L_{n,j}(t-1)} \times \frac{1-s^{u_x}_{n,j}t}{1-th_{lo}.x} \times s^{a_0x}_{n,j}, \ \mathrm{s^{ux}_{n,j}t} < \mathrm{th^x_{lo}} \\ \frac{1}{L_{n,j}(t-1)} \times s^{a_0x}_{n,j}, \qquad \text{Otherwise.} \end{cases} \tag{14}$$

After the update of the available resource and the asking price, as shown in line 12, the auctioneer performs the auction by calling Algorithms 1 and 2. Next, in line 13, the QoS satisfaction ratio $L(t)$ is updated and used for the next time slot. Then the cancellation policy is called, and those assigned auction tasks in $B_c(t)$ will be canceled. Finally, the total profit is calculated.

5 Performance Evaluation

In this section, the experimental data set and the data preprocessing method are first described in 5.1. Then, the parameter settings are introduced in 5.2. Finally, the performance of the online auction approach and the dynamic overbooking mechanism is illustrated in 5.3 and 5.4, respectively.

A larger-scale simulation experiment has been conducted to help us better select hyperparameters while verifying the effectiveness of the algorithm compared with the previous work [13]. To select the appropriate hyperparameters, trade-offs from the two perspectives of QoS satisfaction and profit are made. In the experiment, three hyperparameters are adjusted, and a total of six figures are added. The experimental results show that with the above improvements, the QoS satisfaction ratio has been increased to 99.95%, meeting the SLA requirements and increasing the revenue by 7.82%. The details are as follows.

5.1 Data Preprocessing

Overview. The data set used in the experiment is the Google cluster trace [28], [29]. With proper preprocessing, this data trace can be used in the cloud, EC, etc. [4], [39], [40]. The raw data contained cluster statistics of about 12.5 k servers for 29 days in May 2011, and the size is larger than 40 GB. In the Google cluster, work arrives in the form of jobs, and a job is comprised of one or several tasks. In total, six kinds of data tables are provided in this cluster trace, which are job events, task events, machine events, machine attributes, task constraints, and task usage [1]. Since the CPU and memory-related information of tasks are extracted in the cluster trace, and the types of tasks in the cluster in recent years are basically two types of computation-intensive tasks and delay-sensitive tasks. So the cluster trace is very representative. Even data from 2011 is still widely used in recent years [41], [42], [43].

Data for Different Billing Methods. The preprocessing of task data mainly includes the normal rental tasks and overbooking tasks. The former is used for daily and on-demand billing and the latter for auction and spot billing. For normal rental tasks, first, the data is loaded from the files. Then traverse the data and check the consistency of each data, including whether the start and end times of the tasks are reasonable, whether there are contradictions between the tasks, etc. After preprocessing, consecutive cluster tasks with consistent time sequences are obtained. In addition, for the data that does not meet the consistency requirements, its duration, number, etc., are extracted as auction and spot tasks. Since the data-trace is enormous, it would be challenging to analyze the whole data set at once. Thus, the entire data set is sampled, and the approximate distribution of the data is observed. One thousand sampling time slots are randomly generated from which the statistics are collected.

Data for Resource Utilization Prediction. As introduced in Section 4.1, LSTM and residual network-based neural networks are used to predict resource utilization in the dynamic overbooking mechanism. Extensive training data with useful features need to be prepared to train these neural networks. First, the tables of task usage, task events, machine events, and machine attributes are joined together from the original data set. Then, the useless attributes (such as user name) are removed, and the items containing null or illegal values are deleted.

After these steps, a data set containing millions of data with 15 features is obtained, which are machine type, machine platform, CPU capacity, memory capacity, task count, CPU request, memory request, mean CPU usage, sampled CPU usage, maximum CPU usage, canonical memory usage, assigned memory usage, maximum memory usage, scheduling class, and priority. Finally, the records for five consecutive periods are combined as the network's input. Besides, the mean CPU usage and canonical memory usage in the next period are taken as training targets.

5.2 Parameter Settings

The parameter settings are introduced as follows.

1. The word *machine* is the term in the data set. In this paper, the term *server* is used.

TABLE 2
Number of Servers With Different Resource

CPU	Memory	Number
0.5	0.2493	32
0.5	0.4995	50
0.5	0.1241	1
0.5	0.749	9
0.25	0.2498	1
1	1	7

Edge Nodes and Servers. It is assumed that 100 servers are randomly assigned to 5 edge nodes. These servers can be divided into six types according to their resource capacity, and each type is configured according to the machine attributes described in the data set. The number of servers with the different resources is listed in Table 2. The experiment duration is set to 100-time slots with a total of 500 minutes. The scale of the experiment is expanded by five times compared with [13].

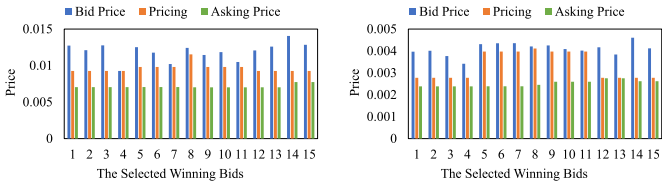
Price. In the experiment, the resource usage from the data set is distinguished according to the priority of the task and regarded as the resource usage of on-demand and daily tasks, respectively. The prices of different billing methods are set as follows:

1) On-Demand: The unit prices of on-demand billing are set to \$0.0182 and \$0.0060 for CPU and memory, respectively.
2) Daily: The unit price of daily billing is 80% of the on-demand billing.
3) Spot: As for spot billing, up to 20% of the resource capacity of the server is sold to them, and their bid price w_x^s is set as 120% of $s_{n,j}^{a_0 x}$, where $s_{n,j}^{a_0 x}$ is the base asking price of server S_i^j and is set to \$0.0068 and \$0.0023 for CPU and memory, respectively [30]. Furthermore, the power consumed per unit CPU h_c and memory h_m are set as 0.008 and 0.00014, respectively, and the unit power fare p_e is \$0.2 [35].
4) Auction: The price of auction billing is described specifically as follows.

Auction. 1×10^6 tasks are chosen from the data set as auction tasks. The time of each task that chooses to bid is randomly generated. During each time slot of 5 minutes, there are approximately 2000 tasks in each auction. For each task b_i, its start time t_i^s, end time t_i^e, and resource request r_i are extracted from the data set. Considering the limitations on the maximum duration of auction tasks, T_b is set to 20. In other words, t_i^e is at most $t_i^s + 20$. The bid price v_i of each task is randomly generated as:

$$v_i^x = \left(\max\left(\frac{1}{6} \times Sc_i + \frac{3}{22} \times Pr_i, 1\right) + \frac{1}{2} \times rand\right) \times p_b^x,$$

where $Sc_i \in \{0, 1, 2, 3\}$ and $Pr_i \in \{0, 1, \ldots, 11\}$ are the scheduling class and priority of task b_i derived from the data set, respectively. $rand$ is a random number generated from a uniform distribution over $[0, 1)$. And p_b^x is the base bid price, which is set to \$0.0068 and \$0.0023 for CPU and memory, respectively [30]. Furthermore, the values of the parameters α_p and β_p used to control the weights of $w_p^{n,j}(t)$ and $w_p^i(t)$ in Algorithm 3 are set to 3 and 1, respectively.



(a) CPU (b) Memory

Fig. 5. Individual rationality.

Dynamic Overbooking. The values of α_u, β_u, and ϵ are set to 2, 0.05 and 0.005, respectively. The values of the thresholds which control the available resource in the experiment are set as $th_{lo} = 0.25$ and $th_{up} = 0.75$, respectively. The details of choosing the values of these three parameters are discussed in Section 5.4.

Resource Utilization Prediction. The structures of the prediction networks are described in Section 4.1. The data set has an input format of $n \times 5 \times 15$ and an output format of $n \times 2$ (n denotes the number of samples). Besides, 2×10^6 pieces of data are extracted from 1×10^4 servers and used as the training set. Moreover, 1×10^4 pieces of data are used as the testing set.

Adaptive Padding & Cancellation Policy. The default value of the hyper-parameter th_{ex} is set to 0.8. In addition, the values of the parameters α_r and β_r used to control the weights of $w_r^i(t)$ in Algorithm 5 Cancellation Policy are both set to 1.

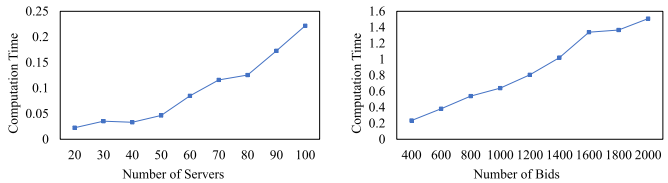
5.3 Auction

The proposed online auction approach is experimentally verified in this subsection to satisfy the following three properties: individual rationality, computational efficiency, and truthfulness.

Individual Rationality. The bid price, asking price, and the pricing of some winning tasks are shown in Fig. 5. In Fig. 5, each winning task is charged a price no higher than its bid price, while each winning server receives a payment no less than its asking price. Therefore, the proposed online auction approach achieves individual rationality.

Computational Efficiency. The algorithm is tested on a Linux Server with 2.20 GHz Intel Xeon CPU E5-2630 v4 and 16 GB memory. The number of auction tasks and the number of servers are fixed to 2000 and 100, respectively, while adjusting the other one to verify the computation time of the algorithm. The results are shown in Fig. 6a and 6b, respectively. From Fig. 6, it can be seen that the proposed auction approach is subject to polynomial computation time concerning the number of servers and bids.

Truthfulness. As for truthfulness, the verification results are shown in Fig. 7. The value in the x-axis is defined as the ratio of the submitted price $v_i'^x$ to the truthful valuation v_i^x. When the ratio equals 1, the submitted price is the truth valuation. The value in the y-axis is the utility, which is defined as the truthful valuation v_i^x minus the pricing p_i^x. From Fig. 7, it can be concluded that the maximum utility is achieved when the task submits the truthful information. As a result, the task cannot improve its utility through other bids, guaranteeing its truthfulness.



(a) Different Number of Servers (b) Different Number of Bids

Fig. 6. Computational efficiency.

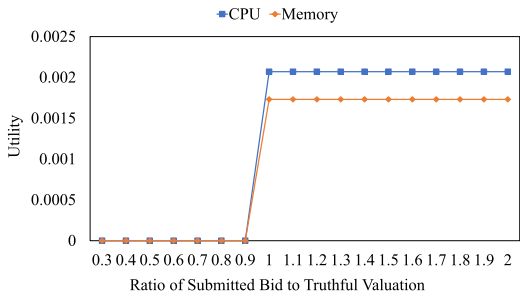
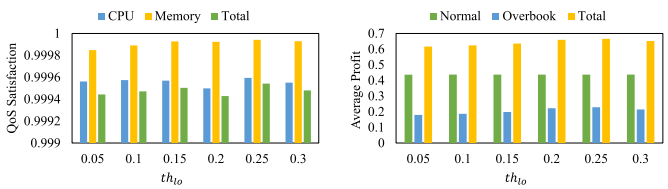
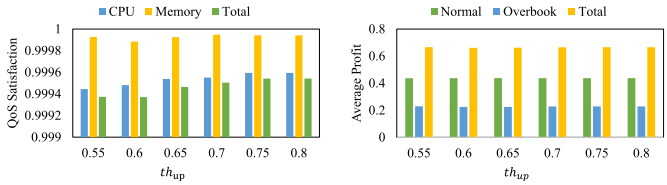


Fig. 7. Truthfulness.



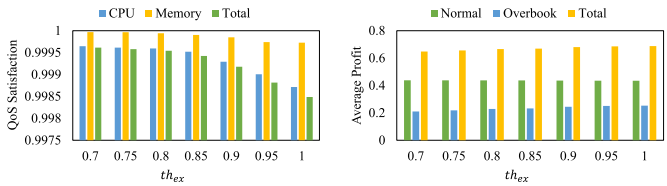
(a) QoS Satisfaction Ratio (b) Average Profit

Fig. 8. Performance with different th_{lo}.



(a) QoS Satisfaction Ratio (b) Average Profit

Fig. 9. Performance with different th_{up}.



(a) QoS Satisfaction Ratio (b) Average Profit

Fig. 10. Performance with different th_{ex}.

TABLE 3
The Prediction Results

Prediction Method	Accuracy Rate
Final State-based Method	0.674
Simple Moving Average Method ($n_w = 5$)	0.753
Exponential Moving Average Method ($\alpha_r = 0.9$)	0.645
Exponential Moving Average Method ($\alpha_r = 0.95$)	0.657
Exponential Moving Average Method ($\alpha_r = 1$)	0.680
Proposed LSTM based Predictor	**0.824**
Proposed Residual Network based Predictor	**0.841**

5.4 Dynamic Overbooking

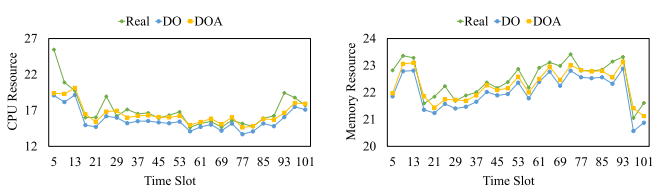
In this subsection, the performance of the dynamic overbooking mechanism is demonstrated. First, the selection of the threshold parameters is discussed, then the prediction accuracy of the neural network predictors is presented. Finally, the QoS satisfaction and profit of the dynamic overbooking mechanism and the comparison with baseline mechanisms are introduced.

Parameter Selection. Figs. 8, 9, and 10 show the results of the parameter selection. First, th_{up} and th_{ex} are fixed and the value of th_{lo} is adjusted. From Fig. 8, it can be seen that when the value of th_{lo} equals 0.25, the QoS Satisfaction ratio is the highest, reaching a maximum value of 0.99954. At the same time, the average profit reaches a maximum of 0.66583. As a result, the value of th_{lo} is set to 0.25.

Next, th_{lo} and th_{ex} are fixed and the value of th_{up} is adjusted. In Fig. 9, the QoS Satisfaction ratio increases with the increase of th_{up} and remains unchanged after 0.75. This is because even if there is a certain error in the prediction value, the predicted unused resource will not exceed the threshold of 75% of the server's resource capacity, and increasing this threshold will not impact the experimental results. Considering that when th_{up} equals to 0.75, the average profit of the edge nodes also achieves a larger value of 0.66583, then the value of th_{up} is set to 0.75.

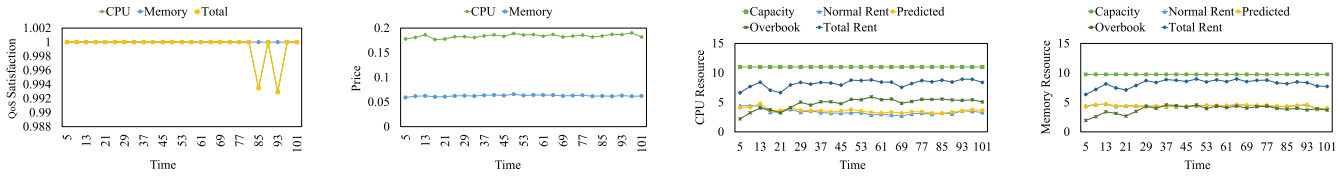
Finally, Fig. 10 shows the results of adjusting the value of th_{ex}. As th_{ex} increases, the QoS Satisfaction ratio gradually decreases, while the average profit of edge nodes continues to increase. The reason is that with excessive overbooking, the profit of the normal rental mode gradually decreases due to the SLA discount. However, the extra income from overbooking offsets the loss of the previous part and leads to an increase in total profit. So how to choose the value of th_{ex} is a trade-off between QoS satisfaction and profit. To ensure that the QoS Satisfaction ratio of the normal rental mode reaches above 0.9995, the value of th_{ex} is set to 0.8.

From Figs. 8, 9, and 10, it can be seen that the QoS satisfaction ratio and the profit are very correlated. When the resource is idle, the QoS satisfaction ratio is 1, and the profit is relatively small. When the resource is overbooked, the degree of resource utilization increases. If the overbooking is controlled within a reasonable range, then the QoS satisfaction ratio is still 1, which is the most reasonable state and the goal of this paper. When resources are overbooked too much, although the profit may increase, the QoS satisfaction ratio drops rapidly, and the computation time of daily and on-demand tasks is prolonged, which is not the goal of this paper.



(a) CPU Utilization Prediction (b) Memory Utilization Prediction

Fig. 11. Resource utilization prediction comparison.



(a) QoS Satisfaction Ratio (b) Price (c) CPU Utilization (d) Memory Utilization

Fig. 12. Performance of the overbooking mechanism of the edge node.

Resource Utilization Prediction. To effectively demonstrate the accuracy rate of predicted resource utilization of the network, the Final State-based (FS) method, Simple Moving Average (SMA) method, and Exponential Moving Average (EMA) method are used as the baselines [44]. In the FS method, the information of tasks during the last time slot is used to predict, while the information of tasks during n_w last time slot is used in the SMA method, where n_w is the size of the windows. In the EMA method, the prediction is based on the weighted sum of the previous tasks, which is obtained as:

$$Pre(t) = \alpha_r \times J_1 + (1 - \alpha_r) \times Pre(t-1),$$

where J_1 is the value of the tasks during the last time slot, α_r is the decay parameter to optimize the accuracy and adjusted by experience. The results of the predicted resource utilization of different methods are shown in Table 3. It can be concluded that the accuracy rate of the proposed LSTM and residual network-based network is much higher than the baselines.

Besides, Fig. 11 compares the resource utilization prediction with or without adaptive padding methods, which are denoted as Dynamic Overbooking with Adaptive padding (DOA) and Dynamic Overbooking (DO), respectively. From Fig. 11, it can be easily obtained that the DOA mechanism further narrows the gap between the predicted value and the actual value, which is very helpful for us to maintain high QoS during dynamic overbooking.

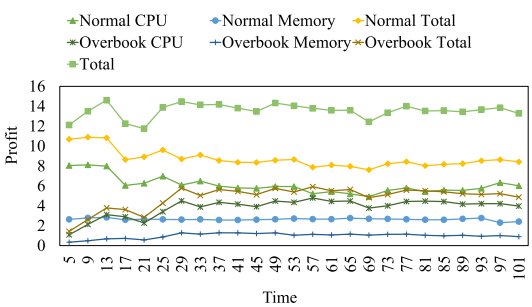
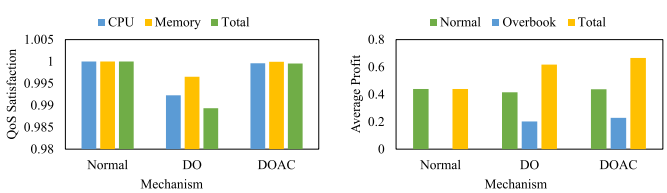


Fig. 13. Average profit of the edge node.



(a) QoS Satisfaction Ratio (b) Average Profit

Fig. 14. Performance with different mechanisms.

Edge Node Performance. The performance of a randomly selected edge node is shown in Fig. 12. As shown in Fig. 12a, this edge node provides 100% QoS satisfaction for most of the time, and the QoS Satisfaction ratio is reduced due to excessive overbooking in a short period, with a minimum value of 0.99290. The affected QoS was quickly recovered with the adaptive padding, cancellation policy, and appropriately selected threshold parameters. The average price of the edge node is shown in Fig. 12b, where the price is dynamically adjusted according to the QoS Satisfaction ratio. The CPU and memory utilization are shown in Fig. 12c and 12d, respectively. It is easy to conclude that predicted utilization matches well with real utilization, and overall utilization is significantly improved.

The average profit of the former edge node is shown in Fig. 13. The normal CPU and memory profits represent the income obtained by renting resources according to on-demand and daily billing methods, and overbooking CPU and memory profits consist of the income from auction and spot billing. From Fig. 13, it can be concluded that through overbooking, the total profit of the edge node can be significantly improved. To sum up, the dynamic resource overbooking mechanism effectively overbooks the resource with a high QoS Satisfaction ratio.

Overall Performance. Fig. 14 presents the comparison among the Dynamic Overbooking with Adaptive padding and Cancellation policy (DOAC) proposed in this paper, the Dynamic Overbooking (DO) mechanism proposed in our previous paper [13], and the Normal mechanism without overbooking. Fig. 14a shows the performance of the QoS Satisfaction ratio of these three mechanisms. The Normal mechanism without overbooking achieves 100% QoS satisfaction. Compared with the DO mechanism, the DOAC mechanism has dramatically improved the performance of QoS satisfaction, and the average QoS Satisfaction ratio can reach above 0.9995, which satisfies the SLA standard [34]. As for the average profit, compared with the Normal mechanism, the DOAC mechanism only reduces the profit on the normal rental mode by 0.47%, while the total revenue increase by 51.58% in Fig. 14b. Besides, compared with the DO mechanism, the profit of the DOAC mechanism has increased by 5.25%, 13.10%, and 7.82% on the normal rental, overbooking, and combined mode, respectively. In short, the DOAC mechanism can significantly increase revenue while achieving a high QoS satisfaction ratio.

6 Conclusion

This paper proposes a pricing model for the dynamic resource overbooking mechanism in EC. First, the system model, pricing model, and problem formulation of the

overbooking problem are described. Second, an online auction approach is proposed with individual rationality, computational efficiency, and truthfulness properties. Third, the dynamic overbooking mechanism is described based on resource utilization prediction and QoS satisfaction ratio feedback. The experiments are conducted with real-world data-trace, and the experimental results show that the auction approach and dynamic overbooking mechanism are efficient. Under the premise of 99.95% QoS satisfaction, it can increase the profit by 51.58% compared with the mechanism without overbooking. Future work will consider resource overbooking across edge nodes and cloud data centers.

REFERENCES

[1] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st Ed. MCC Workshop Mobile Cloud Comput.*, 2012, pp. 13–16.

[2] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[3] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[4] Z. Tang, X. Zhou, F. Zhang, W. Jia, and W. Zhao, "Migration modeling and learning algorithms for containers in fog computing," *IEEE Trans. Services Comput.*, vol. 12, no. 5, pp. 712–725, Sep./Oct. 2019.

[5] Y.-H. Kao, B. Krishnamachari, M.-R. Ra, and F. Bai, "Hermes: Latency optimal task assignment for resource-constrained mobile computing," *Trans. Mobile Comput.*, vol. 16, no. 11, pp. 3056–3069, 2017.

[6] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1171–1181, Dec. 2016.

[7] O. C. A. W. Group *et al.*, "Openfog reference architecture for fog computing," *OPFRA001*, vol. 20817, no. 1, pp. 1–162, 2017.

[8] M. Aazam and E.-N. Huh, "Fog computing micro datacenter based dynamic resource estimation and pricing model for IoT," in *Proc. IEEE 29th Int. Conf. Adv. Inf. Netw. Appl.*, 2015, pp. 687–694.

[9] I. S. Moreno and J. Xu, "Neural network-based overallocation for improved energy-efficiency in real-time cloud environments," in *Proc. IEEE Int. Symp. Object/Compon./Serv.-Oriented Real-Time Distrib. Comput.*, 2012, pp. 119–126.

[10] I. S. Moreno and J. Xu, "Customer-aware resource overallocation to improve energy efficiency in realtime cloud computing data centers," in *Proc. IEEE Int. Conf. Serv.-Oriented Comput. Appl.*, 2011, pp. 1–8.

[11] L. F. Bittencourt, M. M. Lopes, I. Petri, and O. F. Rana, "Towards virtual machine migration in fog computing," in *Proc. IEEE Int. Conf. P2P Parallel Grid Cloud Internet Comput.*, 2015, pp. 1–8.

[12] H. Zhang, Y. Xiao, S. Bu, D. Niyato, R. Yu, and Z. Han, "Fog computing in multi-tier data center networks: A hierarchical game approach," in *Proc. IEEE Int. Conf. Commun.*, 2016, pp. 1–6.

[13] F. Zhang, Z. Tang, M. Chen, X. Zhou, and W. Jia, "A dynamic resource overbooking mechanism in fog computing," in *Proc. 15th IEEE Int. Conf. Mobile Ad Hoc Sensor Syst.*, 2018, pp. 89–97.

[14] V. Krishna, *Auction Theory*, vol. 1. New York, NY, USA: Academic Press, 2009.

[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," *Trans. Serv. Comput.*, vol. 9, no. 6, pp. 895–909, 2016.

[16] A.-L. Jin, W. Song, and W. Zhuang, "Auction-based resource allocation for sharing cloudlets in mobile cloud computing," *IEEE Trans. Emerg. Topics Comput.*, vol. 6, no. 1, pp. 45–57, Jan.–Mar. 2018.

[17] X. Wang, X. Chen, and W. Wu, "Towards truthful auction mechanisms for task assignment in mobile device clouds," in *Proc. IEEE Int. Conf. Comput. Commun.*, 2017, pp. 1–9.

[18] S. Barbarossa, E. Ceci, and M. Merluzzi, "Overbooking radio and computation resources in mmW-mobile edge computing to reduce vulnerability to channel intermittency," in *Proc. IEEE Eur. Conf. Netw. Commun.*, 2017, pp. 1–5.

[19] F. Slim, F. Guillemin, and Y. Hadjadj-Aoul, "Close: A costless service offloading strategy for distributed edge cloud," in *Proc. IEEE Consum. Commun. Netw. Conf.*, 2018, pp. 1–6.

[20] M. T. Imam, S. F. Miskhat, R. M. Rahman, and M. A. Amin, "Neural network and regression based processor load prediction for efficient scaling of grid and cloud resources," in *Proc. IEEE Int. Conf. Comput. Inf. Technol.*, 2011, pp. 333–338.

[21] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[23] M. Allman, V. Paxson, and E. Blanton, "TCP Congestion Control," RFC5681, Sep. 2009.

[24] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *Proc. IEEE 35th Annu. Int. Conf. Comput. Commun.*, 2016, pp. 1–9.

[25] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, Jul.–Sep. 2017.

[26] J. Zhang *et al.*, "Data-driven intelligent transportation systems: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 4, pp. 1624–1639, Dec. 2011.

[27] N. Raveendran, H. Zhang, Z. Zheng, L. Song, and Z. Han, "Large-scale fog computing optimization using equilibrium problem with equilibrium constraints," in *Proc. IEEE Glob. Commun. Conf.*, 2017, pp. 1–6.

[28] J. Wilkes, "More Google cluster data," *Google Research Blog*, 2011. [Online]. Available: http://googleresearch.blogspot.com/2011/11/more-google-cluster-data.html

[29] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: Format schema," Google Inc., Mountain View, CA, USA, Technical Report, Nov. 2011, revised 2014–11-17 for version 2.1. Posted at https://github.com/google/cluster-data

[30] A. W. Services, "Aws cloud pricing principles," 2022. [Online]. Available: https://aws.amazon.com/pricing/

[31] Microsoft, "Azure pricing," 2022. [Online]. Available: https://azure.microsoft.com/en-us/pricing/

[32] X. Zhou, K. Wang, W. Jia, and M. Guo, "Reinforcement learning-based adaptive resource management of differentiated services in geo-distributed data centers," in *Proc. IEEE/ACM 25th Int. Symp. Qual. Serv.*, 2017, pp. 1–6.

[33] F. Farahnakian, P. Liljeberg, and J. Plosila, "Energy-efficient virtual machines consolidation in cloud data centers using reinforcement learning," in *Proc. IEEE 22nd Euromicro Int. Conf. Parallel Distrib. Netw. Based Process.*, 2014, pp. 500–507.

[34] M. Azure, "Sla for virtual machines," 2017. [Online]. Available: https://azure.microsoft.com/en-us/support/legal/sla

[35] 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 *Proc. IEEE Int. Conf. Comput. Commun.*, 2017, pp. 1–9.

[36] R. P. McAfee, "Mechanism design by competing sellers," *Econometrica, J. Econometric Soc.*, vol. 1, no. 1, pp. 1281–1312, 1993.

[37] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, *Algorithmic Game Theory*, vol. 1. Cambridge, UK: Cambridge Univ. Press, 2007.

[38] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes, "Cloudscale: Elastic resource scaling for multi-tenant cloud systems," in *Proc. 2nd ACM Symp. Cloud Comput.*, 2011, Art. no. 5.

[39] X. Zhu, L. T. Yang, H. Chen, J. Wang, S. Yin, and X. Liu, "Real-time tasks oriented energy-aware scheduling in virtualized clouds," *IEEE Trans. Cloud Comput.*, vol. 2, no. 2, pp. 168–180, Apr.–Jun. 2014.

[40] Z. Tang, J. Lou, F. Zhang, and W. Jia, "Dependent task offloading for multiple jobs in edge computing," in *Proc. IEEE 29th Int. Conf. Comput. Commun. Netw.*, 2020, pp. 1–9.

[41] L. Versluis *et al.*, "The workflow trace archive: Open-access data from public and private computing infrastructures," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 9, pp. 2170–2184, Sep. 2020.

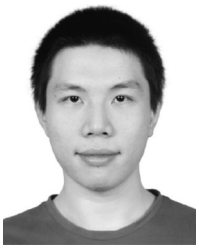
[42] D. Fernández-Cerero, Á. J. Varela-Vaca, A. Fernández-Montes, M. T. Gómez-López, and J. A. Alvárez-bermejo, "measuring data-centre workflows complexity through process mining: The google cluster case," *J. Supercomput.*, vol. 76, no. 4, pp. 2449–2478, 2020.

[43] J. Gao, H. Wang, and H. Shen, "Machine learning based workload prediction in cloud computing," in *Proc. IEEE 29th Int. Conf. Comput. Commun. Netw.*, 2020, pp. 1–9.

[44] S. Zhao, H. Chen, R. Zhao, Y. Zhao, and G. Chen, "A Big Data processing-oriented prediction method of cloud computing service request," *J. Appl. Sci. Eng.*, vol. 19, no. 4, pp. 497–504, 2016.

Zhiqing Tang received the BS degree from the School of Communication and Information Engineering, University of Electronic Science and Technology of China, China, in 2015. He is currently working toward the PhD degree in the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. His current research interests include edge computing, resource allocation, and reinforcement learning.

Fuming Zhang received the BS degree from the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, China, in 2018, where he is currently working toward the MS degree. His current research interests include edge computing, resource scheduling, and machine learning.



Xiaojie Zhou received the BS degree from the School of Data and Computer Science, Sun Yat-sen University, China, in 2016. He is currently working toward the master degree in the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. His current research interests include edge computing, resource scheduling, and reinforcement learning.

Weijia Jia (Fellow, IEEE) received the BSc/MSc degrees from Center South University, China, in 1982/1984, and the master of applied science/PhD degrees from the Polytechnic Faculty of Mons, Belgium, in 1992/1993, respectively, all in computer science. He is currently a chair professor, director of the BNU-UIC Institute of Artificial Intelligence and Future Networks, Beijing Normal University (Zhuhai) and VP for Research of BNU-HKBU United International College (UIC) and has been the Zhiyuan chair professor of Shanghai Jiao Tong University, China. He was the chair professor and the deputy director of the State Kay Laboratory of Internet of Things for Smart City, University of Macau. From 1993–1995, he joined German National Research Center for Information Science (GMD) in Bonn (St. Augustine) as a research fellow. From 1995–2013, he worked with the City University of Hong Kong as a professor. His contributions have been recognized as optimal network routing and deployment, anycast and QoS routing, sensors networking, AI (knowledge relation extractions; NLP, etc.), and edge computing. He has more than 600 publications in the prestige international journals/conferences and research books, and book chapters. He has received the best product awards from the International Science & Tech. Expo (Shenzhen) in 2011/2012 and the 1st Prize of Scientific Research Awards from the Ministry of Education of China in 2017 (list 2). He has served as area editor for various prestige international journals, chair and PC member/skeynote speaker for many top international conferences. He is the distinguished member of the CCF.

Wei Zhao (Fellow, IEEE) received the undergraduate degree in physics from Shaanxi Normal University, China, in 1977, and the MSc and PhD degrees in computer and information sciences from the University of Massachusetts at Amherst, in 1983 and 1986, respectively. He has served important leadership roles in academic including the chief research officer with the American University of Sharjah, the chair of Academic Council at CAS Shenzhen Institute of Advanced Technology, the eighth Rector of the University of Macau, the dean of science with Rensselaer Polytechnic Institute, the director for the Division of Computer and Network Systems in the U.S. National Science Foundation, and the senior associate vice president for research with Texas A&M University. He has made significant contributions to cyber-physical systems, distributed computing, real-time systems, and computer networks. He led the effort to define the research agenda of and to create the very first funding program for cyber-physical systems, in 2006. His research results have been adopted in the standard of Survivable Adaptable Fiber Optic Embedded Network. He was awarded the Lifelong Achievement Award by the Chinese Association of Science and Technology in 2005.