Reverse Auction-based Computation Offloading and Resource Allocation in Mobile Cloud-Edge Computing

Huan Zhou, *Member, IEEE*, Tong Wu, *Student Member, IEEE*, Xin Chen, *Student Member, IEEE*, Shibo He, *Senior Member, IEEE*, Deke Guo, *Senior Member, IEEE*, and Jie Wu, *Fellow, IEEE*

Abstract—This paper proposes a novel Reverse Auction-based Computation Offloading and Resource Allocation Mechanism, named RACORAM for the mobile Cloud-Edge computing. The basic idea is that the Cloud Service Center (CSC) recruits edge server owners to replace it to accommodate offloaded computation from nearby resource-constraint Mobile Devices (MDs). In RACORAM, the reverse auction is used to stimulate edge server owners to participate in the offloading process, and the reverse auction-based computation offloading and resource allocation problem is formulated as a Mixed Integer Nonlinear Programming (MINLP) problem, aiming to minimize the cost of the CSC. The original problem is decomposed into an equivalent master problem and subproblem, and low-complexity algorithms are proposed to solve the related optimization problems. Specifically, a Constrained Gradient Descent Allocation Method (CGDAM) is first proposed to determine the computation resource allocation strategy, and then a Greedy Randomized Adaptive Search Procedure based Winning Bid Scheduling Method (GWBSM) is proposed to determine the computation offloading strategy. Meanwhile, the CSC's payment determination for the winning edge server owners is also presented. Simulations are conducted to evaluate the performance of RACORAM, and the results show that RACORAM is very close to the optimal method with significantly reduced computational complexity, and greatly outperforms the other baseline methods in terms of the CSC's cost under different scenarios.

Index Terms—Computation Offloading; Resource Allocation; Reverse Auction; Mobile Cloud-Edge Computing.

1 Introduction

With the technological advancements and the popularity of Mobile Devices (MDs), Internet of Things (IoTs) has opened up a number of attractive application types with computation-intensive features, such as intelligent transportation, health care, Augmented/Virtual Reality (AR/VR), etc [1], [2], [3]. Indeed, the flourishing IoT applications always have intense requirements for users' Quality of Service (QoS) and computation resources, which leads to higher computation loads and energy consumption than traditional applications. Furthermore, considering the physical size and cost constraints, current MDs have suffered from the limitation of computation resources, which may become an inevitable bottleneck to support these computation-intensive applications in the future IoTs [4], [5], [6].

Recently, cloud computing has greatly relieved the conflict between IoT applications and resource-constraint MDs, which enables convenient access to a shared resource pool in the Cloud Service Center (CSC) [7], [8], [9]. However, for delay-sensitive IoT services, cloud computing may not be feasible as the long transmission distance in remote cloud-based processing contributes to extra transmission cost and delay [10]. Furthermore, according to Cisco's recent report [11], the number of Mobile Devices (MDs) is expected to increase from 21.5 billion in 2019 to 28.5 billion by 2022. The CSC's existing infrastructure is difficult to provide high-quality services for so many resource-constrained devices in the future [12]. Fortunately, Mobile Edge Computing (MEC) enables MDs to offload workloads to nearby edge servers [13]. Such capability not only meets the expansion requirements of computation capabilities of MDs, but also improves the QoS of IoT applications with considerably reduced delay and cost [14], [15], [16]. By now, computation offloading in MEC networks has been well investigated in the area of system architecture [17], [18], energy efficiency [19], [20], computation resources optimization [21], [22], etc.

Nevertheless, implementing computation offloading in MEC networks still faces many critical issues. The computation offloading process inevitably consumes a lot of computation and communication resources. From the economical perspective, given that edge server owners (in this paper we refer to Small Base Stations (SBSs) equipped with edge servers) are commonly rational and selfish as they are owned by third-party companies, they have no responsibility to participate in the computation offloading process without receiving any economic reimbursement [23], [24]. Therefore, it is imperative to develop incentive mechanisms, which can encourage idle SBSs with edge servers to assist the CSC in computation offloading by offering them proper rewards for their resource consumptions.

- *This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grants 62172255 and U1909207. (Corresponding author: Huan Zhou.)*
- *H. Zhou, T. Wu, and X. Chen are with the College of Computer and Information Technology, China Three Gorges University, Yichang 443002, China. E-mail: zhouhuan117@gmail.com, wutong.asd@gmail.com, sexychenxin@gmail.com.*
- *S. He is with the College of Control Science and Technology, Zhejiang University, Hangzhou 310027, China. E-mail: s18he@zju.edu.cn.*
- *D. Guo is with the Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha, 410073, China. E-mail:dekeguo@nudt.edu.cn.*
- *J. Wu is with the Department of Computer and Information Sciences, Temple University, Philadelphia PA19122, USA. E-mail: jiewu@temple.edu.*

Some researches have put forward different incentive mechanisms to resolve the incentive issues of computation offloading in MEC [25]. However, to our knowledge, the design of incentive mechanisms in mobile Cloud-Edge computing still lacks due attention. In mobile Cloud-Edge computing, we assume that MDs requesting computation offloading services have subscribed to CSC for computation offloading services and paid related fees monthly or yearly. The CSC has a lot of tasks offloaded by resource-constraint subscribed MDs to implement, while SBSs usually have idle computation resources. Long transmission distance and resource shortage caused by the increasing number of offloaded MDs may make CSC be unable to satisfy the delay requirements of MDs. Hence, CSC can recruit SBSs to replace it to accommodate offloaded computation from nearby MDs. Therefore, from the perspective of CSC, it remains open to encourage the participants of SBSs while reducing the cost of CSC. Furthermore, since the same SBS may cover multiple MDs at the same time, and the communication and computation resources of SBSs are limited, it is necessary to jointly consider the computation offloading strategy and resource allocation strategy.

Based on the above analysis, we consider the following issues in this paper: (i) How to develop the reasonable strategy of computation offloading and resource allocation with low-complexity solutions? (ii) How to stimulate SBSs to participate in the offloading process and what is the corresponding payment of the CSC to each SBS? (iii) How to minimize the cost of CSC while satisfying some specific constraints? We answer these issues by proposing a novel Reverse Auction-based Computation Offloading and Resource Allocation Mechanism, named RACORAM. In RACORAM, we use the reverse auction to stimulate SBSs to participate in the offloading process, where CSC acts as the auctioneer and SBSs act as the bidders [26]. Then, we formulate the reverse auction-based computation offloading and resource allocation problem as a Mixed Integer Nonlinear Programming (MINLP) problem, aiming to minimize the cost of CSC. We further modify the constraints of the original problem appropriately and decompose the original problem into an equivalent master problem and subproblem, respectively. After that, we propose low-complexity algorithms to solve the related optimization problems. Finally, simulations are conducted to demonstrate the effectiveness of our proposed method. The key contributions are summarized as follows:

1) We propose the system architecture of RACORAM and use the reverse auction to stimulate SBSs equipped with edge servers to participate in the offloading process.
2) The reverse auction-based computation offloading and resource allocation problem is formulated as a Mixed-Integer Non-Linear Programming (MINLP) problem, aiming to minimize the cost of CSC.
3) We decompose the original problem into an equivalent *master-problem* and *sub-problem*, and propose low-complexity algorithms to solve the problems. Specifically, we first propose a Constrained Gradient Descent Allocation Method (CGDAM) to determine the resource allocation strategy, and then propose a Greedy Randomized Adaptive Search Procedure based Winning Bid Selection Method (GWBSM) to determine the computation offloading strategy. Meanwhile, the CSC's payment determination for the winning SBSs is also presented.
4) Simulation results show that RACORAM is very close to the optimal method with significantly reduced computational complexity, and greatly outperforms the other baseline methods in terms of the CSC's cost under different scenarios.

The remainder of this paper is organized as follows. In Section 2, we review the related work. We introduce the system model in Section 3. In Section 4, we formulate the problem as a MINLP problem with the objective of minimizing the cost of the CSC. Then, we decompose the original problem into an equivalent master problem and sub-problem in Section 5, and present two low-complexity algorithms to resolve the optimization problem in Section 6. Furthermore, we introduce and analyze the simulation results in Section 7. Finally, Section 8 gives the conclusion.

2 Related Work

Recently, MEC has attracted extensive attention and research efforts, which effectively mitigates the conflict between high-resource application demands and MDs with limited resources. Especially, since the computation task processing in MEC is usually based on distributed collaboration, its core is to achieve dynamic task scheduling by effectively allocating computation, storage, and communication resources in the edge environment [27]. Some studies have investigated the resource allocation in MEC from the aspect of system utility [28]-[32]. Chen *et al.* in [28] proposed an efficient three-step algorithm to minimize the cost of computation offloading in MEC for all MDs in the system. In [29], the authors established a non-convex optimization problem to minimize the delay of all nodes under the framework of collaborative computing, and transformed the problem into convex optimization through the classical successive convex approximation method for processing. Tran *et al.* in [30] considered a multi-MEC server system that assists mobile users in computation offloading, and proposed a heuristic algorithm to maximize MDs utility from the perspective of task delay and energy consumption. In [31], the authors studied the problem of computation resource allocation based on price, and established a two-layer Stackelberg game model to maximize the utility of both the server and the user in the system. In [32], the instance provisioning of a reliability-aware Network Function Virtualization (NFV) in MEC was investigated to maximize the network throughput, which is achieved by providing a primary NFV and a secondary NFV for each MD. Some studies have also investigated the resource allocation in MEC from the aspect of energy consumption [33]-[35]. Yang et al. in [33] proposed a multi-access MEC servers system based on Orthogonal Frequency Division Multiplexing Access (OFDMA), so as to minimize the computation energy consumption of MDs. Chen *et al.* in [34] constructed an energy-saving resource allocation scheme while considering the constraints of delay, channel quality and transmission power, which aims to minimize the energy consumption of task offloading. However, these studies assume that edge server owners are cooperative.

Some studies have focused on the incentive mechanism design for MEC. In [36], the authors proposed an online incentive-driven task allocation scheme in the context of the Industrial Internet of Things, and jointly considered energy consumption, execution time, and available resources to maximize system utility. In [37], the authors utilized an optimal price-based scheme to provide computation offloading services for MDs by charging an appropriate fee, and proved that the proposed scheme can balance individual interests with the overall system interests. Liu et al. in [38] proposed a two-stage Stackelberg game between the

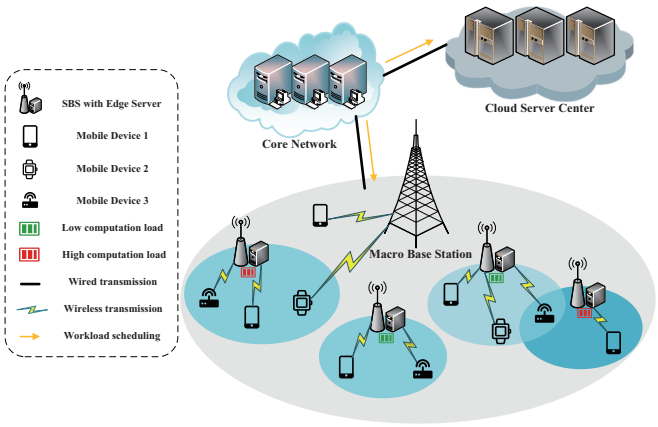

Fig. 1. System Architecture of Mobile Cloud-Edge Computation Offloading.

Cloud Service Operator (CSO) and Edge Server Owners (ESO), which enables CSO to allocate computation tasks based on ESO status information. In [39], the authors proposed a fully distributed and partially distributed multi-hop network incentive mechanism, which can significantly reduce the node's computation cost by avoiding the unreal behavior of the node. In [40], the author proposed a profit maximization-based incentive mechanism, which can maximize the profit of edge servers while ensuring the Quality of Experiment (QoE) of MDs. In [41], the authors proposed an online incentive mechanism that allows the base station to perform task scheduling, resource allocation and pricing decisions without knowing any future information, aiming to maximize the utility of the system.

Furthermore, some auction-based incentive mechanisms have been studied and applied to resource allocation, task scheduling, and other applications in MEC. Ma *et al.* in [42] proposed a truthful combined double auction mechanism, which can stimulate edge servers to provide services to nearby mobile users while ensuring budget balance. In [43], the authors introduced two double auction-based dynamic pricing strategies, named BDA and DPDA respectively, to determine the matched pairs between MDs and edge servers, as well as resource allocation schemes which meet the economic properties. The authors in [44] proposed an auction-based mechanism to stimulate edge nodes to allocate their virtual machine resources to MDs for computation offloading in MEC, aiming to maximize the total social welfare. Sun *et al.* in [45] modeled the interaction between edge servers and MDs as a breakeven and breakeven-free based double auction, so as to maximize the efficiency of the system. Le *et al.* in [46] proposed a two-stage incentive mechanism combined with the auction game to minimize the total network delay under the background of the vehicle network. In [47], the authors regarded the resource allocation in MEC as an auction problem and proposed a multi-task resource allocation algorithm based on double auction to improve system utility. He *et al.* in [48] proposed an auction-based online incentive mechanism that can optimize the long-term utility of the system without knowing the future information.

Previous studies on the auction-based incentive mechanism in MEC mainly formulate the auction from the perspective of SBSs, where some SBSs with computation capabilities act as sellers, and MDs requesting computation offloading services act as buyers. In contrast with the previous studies, this paper assumes that MDs requesting computation offloading services have subscribed to the CSC for computation offloading services and paid related fees monthly or yearly. From the perspective of the CSC, it remains open to encourage the participants of SBSs while reducing the cost of the CSC. Therefore, this paper proposes a reverse auction-based incentive mechanism to stimulate the participation of SBSs, where the CSC acts as the auctioneer and SBSs act as bidders. Then, this paper formulates the reverse auction-based computation offloading and resource allocation problem as a MINLP problem, which aims to minimize the CSC's cost by jointly considering the computation offloading decision and resource allocation. In addition, since emerging delay-sensitive applications have strong requirements for QoS, MDs' QoS is also considered in the optimization problem.

3 System Model

This section first presents the system architecture and elaborates the model involved in the system architecture. Then, a detailed definition of the wireless communication model and the model of reverse auction are given.

3.1 System Architecture

As shown in Fig. 1, this paper considers a three-tier mobile Cloud-Edge computation offloading structure, including a single-cell network, a core network, and a Cloud Service Center (CSC). The core network can communicate with the CSC in the cloud and the base stations in the cellular network through wired links, which is responsible for scheduling the computation task load. For the single-cell network, it consists of a Macro Base Station (MBS), several Small Base Stations (SBSs) equipped with edge servers, and a set of Mobile Devices (MDs). Each MD can communicate with the edge server through the wireless channel provided by the corresponding SBS, and can also communicate with CSC through the wireless channel provided by the MBS. In addition, we assume that each MD has subscribed to CSC for computation offloading services and paid related fees monthly or yearly (including traffic fees caused by data transmission, and service fees caused by occupying server computation resources, etc.).

Since the computation and communication resources of CSC are limited, the long transmission distance and resource shortages caused by the increasing number of offloaded MDs will make CSC be unable to satisfy the delay requirements of MDs. Therefore, CSC can request SBSs near MDs to assist in providing computation offloading services to reduce delay and cut down the cost, especially when the CSC is under high load. Similarly, the communication and computation resources of SBSs are also limited, so multiple MDs might compete for the same SBS simultaneously. Thus, we should jointly consider the computation offloading strategy and resource allocation strategy.

Moreover, assisting the CSC in computation offloading will inevitably generate additional resource consumptions (including communication resources, computation resources, etc.), so SBSs owned by third-party companies will not be willing to participate in computation offloading without economic compensation. In order to encourage SBSs to assist the CSC in computation offloading, the CSC should compensate for the additional resources consumed by SBSs. The specific compensation scheme will be introduced in the model of reverse auction.

We assume that the set of MDs and SBSs in the single-cell network are $\mathcal{N} = \{1, 2, ..., N\}$ and $\mathcal{S} = \{1, 2, ..., S\}$,

TABLE 1. Notations and Explanation

Notation	Explanation
CSC	The abbreviation of Cloud Service Center
MD	The abbreviation of Mobile Device
SBSs	The abbreviation of Small Base Stations
QoS	The abbreviation of Quality of Service
$\mathcal{N}$	The set of all MDs
$\mathcal{S}$	The set of all SBSs
q_n	The information of MD n's computation task
t_n^{max}	The maximum tolerable delay of MD n
c_n	The amount of required computation capacity of MD n
d_n	The amount of input data of MD n
e_s	The state information of SBS s
f_s	The idle computation resources of SBS s
ϕ_s	SBS s' bid of per unit computation resource
v_s	SBS s' true value of per unit computation resource
B	The uplink bandwidth of each SBS
ε	The CSC's cost of per unit computation resource
$\mathcal{M}$	The set of M orthogonal sub-bands
$a_{n,s}^i$	Binary variable that indicates whether MD n will offload its computation task to SBS s on sub-band i
$\mathcal{A}$	The set of offloading strategy
$\mathcal{N}_s$	The set of MDs offloading their task to SBS s
$\mathcal{N}_c$	The set of MDs offloading their task to the CSC
$r_{n,s}$	The transmission rate from MD n to SBS s
$\mathcal{F}$	The computation resource allocation strategy for each SBS
$f_{n,s}$	The computation resource allocated by SBS s to task of MD n
$t_{n,s}^{up}$	The delay of MD n uploading its task to SBS s
$t_{n,s}^{exe}$	The execute delay of task of MD n execution on SBS s
$r_{n,c}$	The transmission rate from MD n to the CSC
$f_{n,c}$	The computation resource allocated by the CSC to task of MD n
$t_{n,c}^{up}$	The delay of MD n uploading its task to the CSC
$t_{n,c}^{exe}$	The execute delay of computation task of MD n execution on the CSC
$t_n(\mathcal{A},\mathcal{F})$	The total delay of MD n
λ_n	The CSC's preference towards MD n
ω	The revenue conversion coefficient

respectively. More specifically, the relevant information of MDs, SBSs and the CSC are introduced in detail as follows:

- Since each MD has subscribed to the CSC's computation offloading service, when the MD runs a deadline-sensitive application, it first generates a computation task and then submits the task information to the CSC. We use a three-tuple $q_n = (t_n^{max}, c_n, d_n)$ to denote the information of MD n's computation task, where t_n^{max} $[seconds]$ represents the maximum tolerable delay, c_n $[Megacycles]$ represents the amount of required computation capacity, and d_n $[MBs]$ represents the amount of input data. In addition, we assume that the computation task is atomic and cannot be divided into subtasks.
- We use a three-tuple $e_s = (f_s, \phi_s, v_s)$ to represent the state information of SBS $s \in \mathcal{S}$, where f_s $[cycles/s]$ denotes idle computation resources of SBS s, ϕ_s denotes SBS s's bid of per unit computation resource, and v_s denotes SBS s's true value of per unit computation resource. Note that v_s is the private information belonging to SBS s.
- The CSC provides computation services to subscribed MDs and charges service fees. However, long transmission distance and resource shortage caused by the increasing number of offloaded MDs may make the CSC unable to satisfy the delay requirements of MDs. Hence, the CSC can purchase a portion of computation resources from SBSs to assist MDs in computation offloading to reduce its cost. We use ε to represent the CSC's cost of per unit computation resource. In this paper, we clarify that ε includes not only the execution cost of the task, but also the procurement and deployment cost of cloud servers. Therefore, we claim that ε is much larger than ϕ_s, and the CSC will prioritize offloading computation tasks to SBSs for execution.

For ease of reference, we have listed the notations used in this paper and provided corresponding explanations in Table 1.

3.2 The Model of Wireless Transmission

For the channel model between each SBS and MDs, we consider using OFDMA as the multiple access scheme in the uplink, in which the bandwidth of each SBS is B [MHz]. The bandwidth of each SBS is divided into M equal sub-bands, so the bandwidth of each sub-band is $W = B/M$.

We define the set of available sub-bands at each SBS as $\mathcal{M} = \{1, 2, ..., M\}$. Let $\mathcal{A} = \{a_{n,s}^i | n \in \mathcal{N}, s \in \mathcal{S}, i \in \mathcal{M}\}$ denote the set of offloading strategies, in which $a_{n,s}^i = 1$ indicates that SBS s is selected to provide computation offloading service for MD n on sub-band i, otherwise $a_{n,s}^i = 0$. Furthermore, we denote $\mathcal{N}_s = \{n \in \mathcal{N} | \sum_{i \in \mathcal{M}} a_{n,s}^i = 1\}$ as the set of MDs offloading their computation tasks to SBS s, and $\mathcal{N}_c = \{n \in \mathcal{N} | \sum_{s \in \mathcal{S}} \sum_{i \in \mathcal{M}} a_{n,s}^i = 0\}$ as the set of MDs offloading their computation tasks to the CSC.

The Signal-to-Interference-plus-Noise Ratio (SINR) from MD n to SBS s on sub-band i is expressed as follows:

$$\mathrm{SINR}_{n,s}^i = \frac{a_{n,s}^i p_n g_{n,s}^i}{\sum_{v \in \mathcal{S} \setminus \{s\}} \sum_{l \in \mathcal{N}_v} a_{l,v}^i p_l g_{l,s}^i + \sigma^2}, \quad (1)$$

where σ^2 is the Gaussian noise of channel, $g_{n,s}^i$ is the uplink channel gain between MD n and SBS s on sub-band i and p_n is the transmission power of MD n when uploading its task to the SBS. The first term at the denominator is the accumulated inter-cell interference. Then, the achievable rate of MD n when uploading data to SBS s is given as:

$$r_{n,s} = W \log_2 \left(1 + \sum_{i \in \mathcal{M}} \mathrm{SINR}_{n,s}^i\right), \forall s \in \mathcal{S}, n \in \mathcal{N}_s. \quad (2)$$

Thus, the transmission time of MD n when sending its task input data d_n to SBS s in the uplink can be calculated as:

$$t_{n,s}^{up} = \frac{d_n}{r_{n,s}}. \quad (3)$$

Let $\mathcal{F} = \{f_{n,s} | n \in \mathcal{N}, s \in \mathcal{S}\}$ denote the computation resource allocation strategy for each SBS, in which $f_{n,s} > 0$ is the amount of computation resources that SBS s allocates to the computation task of MD n. Obviously, $f_{n,s} = 0, \forall n \notin \mathcal{N}_s$. When MD n offloads the computation task to SBS s, its execution delay $t_{n,s}^{exe}$ can be calculated as follows:

$$t_{n,s}^{exe} = \frac{c_n}{f_{n,s}}, \forall n \in \mathcal{N}_s. \quad (4)$$

3.3 The Model of Reverse Auction

This paper uses the reverse auction to motivate SBSs to take part in the computation offloading process, where the CSC acts as the auctioneer and SBSs act as the bidders. SBSs lease their

idle resources as commodities and report their status information (bids) to the CSC.

Specifically, the auction procedure includes the following three steps:

- Each MD submits the task information q_n to the CSC for task scheduling. Meanwhile, each SBS reports its bid vector $[f_s, \phi_s]$ to the CSC.
- Based on the received task information from MDs and the bids from SBSs respectively, the CSC determines the winners of the auction from the perspective of saving its cost, and then makes the scheduling decision for the winners.
- After receiving offloaded tasks from MDs, winning SBSs will complete the tasks and return the output results back to MDs. When MDs receive the returned results, the CSC will be informed that the tasks are completed, and then the CSC will pay the corresponding rewards to SBSs.

4 Problem Formulation

In this section, we formulate the objective function of the CSC, and model the reverse auction-based computation offloading and resource allocation problem as a MINLP problem, aiming to minimize the cost of the CSC.

According to Eq. (3) and Eq. (4), the total delay of MD n offloading its task to SBS s can be calculated as follows:

$$t_{n,s} = t_{n,s}^{up} + t_{n,s}^{exe} = \frac{d_n}{r_{n,s}} + \frac{c_n}{f_{n,s}}, \forall n \in \mathcal{N}_s. \tag{5}$$

Note that, it is common to ignore the backhual delay of task execution result as we also assume that the task execution result data is very small, e.g., number or text [1], [2], [4]. Without loss of generality, we assume that the average upload rate of MD uploading input data to the CSC via MBS is $r_{n,c}$. Let $f_{n,c} > 0$ denote the amount of computation resources that the CSC allocates to MD n, and $f_{n,c} = 0, \forall n \notin \mathcal{N}_c$. Then, we can get the total delay for MD n to offload the computation task to the CSC as:

$$t_{n,c} = t_{n,c}^{up} + t_{n,c}^{exe} = \frac{d_n}{r_{n,c}} + \frac{c_n}{f_{n,c}}, \forall n \in \mathcal{N}_c, \tag{6}$$

where $t_{n,c}^{up}$ and $t_{n,c}^{exe}$ are the transmission delay and execution delay of the MD respectively. Thus, let $t_n(\mathcal{A}, \mathcal{F})$ denote the total delay of MD n, which is given as:

$$t_n(\mathcal{A}, \mathcal{F}) = \begin{cases} t_{n,s} = t_{n,s}^{up} + t_{n,s}^{exe} = \frac{d_n}{r_{n,s}} + \frac{c_n}{f_{n,s}}, & n \in \mathcal{N}_s \\ t_{n,c} = t_{n,c}^{up} + t_{n,c}^{exe} = \frac{d_n}{r_{n,c}} + \frac{c_n}{f_{n,c}}, & n \in \mathcal{N}_c \end{cases} \tag{7}$$

Considering that in a realistic scenario, MDs' QoS has a crucial impact on whether MDs will continue to subscribe to CSC's computation offloading service in the future. In our system model, MDs' QoS is mainly characterized by the computation task completion time. Therefore, while considering the actual cost generated in the system, we also consider the impact of MDs' QoS on the expected revenue of CSC in the future. Then, the CSC's cost can be given as:

$$\begin{aligned} Y(\mathcal{A}, \mathcal{F}) =& \varepsilon \sum_{n \in \mathcal{N}_c} f_{n,c} + \sum_{s \in \mathcal{S}} \phi_s \sum_{n \in \mathcal{N}_s} f_{n,s} \\ & - \sum_{s \in \mathcal{S}} \sum_{n \in \mathcal{N}_s} \lambda_n \omega \left(t_n^{\max} - t_n(\mathcal{A}, \mathcal{F}) \right), \end{aligned} \tag{8}$$

where the first term is the cost of resources consumed by CSC to assist MDs in computation offloading, the second item is the cost generated by the CSC to compensate for SBSs participating in computation offloading, and the third item represents the expected total revenue brought by QoS improvement when MDs offload computation tasks to SBSs, in which $t_n^{\max} - t_n(\mathcal{A}, \mathcal{F})$ represents the total saved delay of MD n. In addition, ω represents the revenue conversion coefficient, and $\lambda_n \in (0, 1]$ represents the CSC's preference towards MD n. For example, depending on the service level subscribed by MDs, the CSC will set a higher value of λ_n for MD n who has paid a higher subscription fee to provide higher-quality services (such as VIP MDs). In other words, λ_n is used to indicate the service level of MDs. MDs with higher subscribed service levels will bring higher expected revenue to CSC. In return, CSC will provide higher-quality services.

Remark 1. *Compared with SBSs, the CSC has much more computation resources. Therefore, to guarantee MDs' QoS and simplify the problem, we assume that the computation resources allocated by the CSC to MD n ($f_{n,c}$) is a constant, which is positively related to the CSC's preference towards MD n (λ_n), i.e., if λ_n is larger, then $f_{n,c}$ is greater, and should meet the maximum delay requirement of MD n, i.e., $f_{n,c} \geq f_{n,c}^{min}, \forall n \in \mathcal{N}_c$. To obtain $f_{n,c}^{min}$, we first set $t_{n,c}$ to:*

$$t_{n,c} = t_n^{max}, \forall n \in \mathcal{N}_c. \tag{9}$$

Then, according to Eq. (6)*, $f_{n,c}^{min}$ can be calculated as:*

$$f_{n,c}^{min} = \frac{c_n}{t_n^{\max} - \frac{d_n}{r_{n,c}}}, \forall n \in \mathcal{N}_c. \tag{10}$$

Now, we formulate the optimization problem as a system cost minimization problem, which can be expressed as:

$$\min_{\mathcal{A}, \mathcal{F}} \quad Y(\mathcal{A}, \mathcal{F}) \tag{11}$$

$$\text{s.t.} \quad a_{n,s}^i \in \{0, 1\}, \qquad \forall n \in \mathcal{N}, s \in \mathcal{S}, i \in \mathcal{M}, \tag{12}$$

$$f_{n,s} > 0, \qquad \forall n \in \mathcal{N}_s, s \in \mathcal{S}, \tag{13}$$

$$\sum_{n \in \mathcal{N}_s} f_{n,s} \leq f_s, \qquad \forall s \in \mathcal{S}, \tag{14}$$

$$\sum_{s \in \mathcal{S}} \sum_{i \in \mathcal{M}} a_{n,s}^i \leq 1, \qquad \forall n \in \mathcal{N}, \tag{15}$$

$$\sum_{n \in \mathcal{N}} a_{n,s}^i \leq 1, \qquad \forall s \in \mathcal{S}, i \in \mathcal{M}, \tag{16}$$

$$t_n(\mathcal{A}, \mathcal{F}) \leq t_n^{max}, \qquad \forall n \in \mathcal{N}. \tag{17}$$

With $Y(\mathcal{A}, \mathcal{F})$ given in Eq. (8), the meaning of the constraints can be explained as follows: constraints (12) and (15) indicate that each MD can offload to the CSC or at most one SBS on one sub-band; constraint (16) implies that each SBS can serve at most one MD per sub-band; constraints (13) and (14) imply that each SBS must allocate a positive computation resource to the associated MDs and the total computation resources allocated to all the associated MDs must not excess the SBS's idle computation resources; constraint (17) guarantees that the total delay of the offloaded computation task does not exceed its maximum tolerable delay. Moreover, the optimization problem has both binary variables and continuous non-integer variables. Therefore, the proposed optimization problem is a complex non-convex MINLP problem, which has been demonstrated to be a typical NP-hard problem [43], [51], [52], [53].

5 Problem Transformation and Decomposition

In this section, we first modify the constraints of the original problem appropriately so that the constraints on the offloading strategy $\mathcal{A}$ and the resource allocation strategy $\mathcal{F}$ can be separated from each other. Then, we decompose the original problem into an equivalent *master-problem* and *sub-problem*.

To meet the delay requirement of Constraint (17), we first have to obtain the minimum computation resources $f_{n,s}^{min}$ that need to be provided if MD n is assisted by SBS s to perform computation offloading. According to Eq. (7), Constraints (13) and (17) can be rewritten as follows:

$$\sum_{n\in\mathcal{N}_s} f_{n,s}^{\min} \le f_s, \qquad \forall s\in\mathcal{S}, \tag{18}$$

$$f_{n,s} \ge f_{n,s}^{\min}, \qquad \forall n\in\mathcal{N}_s, s\in\mathcal{S}, \tag{19}$$

$$f_{n,s}^{min} > 0, \qquad \forall n\in\mathcal{N}_s, s\in\mathcal{S}, \tag{20}$$

where $f_{n,s}^{min} = \frac{c_n}{t_n^{\max}-\frac{d_n}{r_{n,c}}}$ denotes the minimum computation resources that need to be provided if MD n is assisted by SBS s to perform computation offloading. Obviously, $f_{n,s}^{min}=0, \forall n\notin\mathcal{N}_s$.

Hence, if $\mathcal{A}$ and $\mathcal{F}$ satisfy the Constraints (18)(19)(20), the Constraints (13) and (17) are also satisfied. Thus, we can rewrite the original problem as follows:

$$\min_{\mathcal{A}}\left(\min_{\mathcal{F}} Y(\mathcal{A},\mathcal{F})\right) \tag{21}$$

$$\text{s.t.} \quad (12),(15),(16),(18),(20), \tag{22}$$

$$(14),(19). \tag{23}$$

It is worth noting that the constraints on the offloading strategy $\mathcal{A}$ in (22), and the resource allocation strategy $\mathcal{F}$ in (23), are decoupled from each other. Now, solving the problem in (21) is equivalent to solving the following problem:

$$\min_{\mathcal{A}} Y^*(\mathcal{A}) \tag{24}$$

$$\text{s.t.} \quad (12),(15),(16),(18),(20), \tag{25}$$

in which $Y^*(\mathcal{A})$ corresponds to the optimal-value function when the resource allocation strategy $\mathcal{F}$ takes the optimal solution under the fixed offloading strategy $\widetilde{\mathcal{A}}$, written as:

$$Y^*(\mathcal{A}) = \min_{\mathcal{F}} Y(\widetilde{\mathcal{A}},\mathcal{F}) \tag{26}$$

$$\text{s.t.} \quad (14),(19). \tag{27}$$

Moreover, decomposing problem (21) into problems (24) and (26) can still keep the optimality of the solution. In the next section, we will propose two low-complexity algorithms to obtain the suboptimal resource allocation strategy $\mathcal{F}$ and offloading strategy $\mathcal{A}$.

6 Main Approach

This section presents our low-complexity approaches to solve the above optimization problem. We first propose a computation resource allocation method, and then a winning bid selection method. Finally, we present the CSC's payment determination for the winning SBSs.

6.1 Computation Resource Allocation

This part proposes a Constrained Gradient Descent Allocation Method to allocate computation resources for each MD that offloads tasks to the SBS, named CGDAM.

Once a feasible offloading strategy $\widetilde{\mathcal{A}}$ that satisfies constraint (22) is given, then according to Eq. (8), the objective function in (26) can be expressed as follows:

$$Y(\widetilde{\mathcal{A}},\mathcal{F}) = \varepsilon\sum_{n\in\mathcal{N}_c} f_{n,c} + Z(\widetilde{\mathcal{A}},\mathcal{F}), \tag{28}$$

$$\begin{aligned}\text{where } Z(\widetilde{\mathcal{A}},\mathcal{F}) = &\sum_{s\in\mathcal{S}}\phi_s\sum_{n\in\mathcal{N}_s} f_{n,s} - \\ &\sum_{s\in\mathcal{S}}\sum_{n\in\mathcal{N}_s}\lambda_n\omega\left(t_n^{\max}-t_{n,s}\right).\end{aligned} \tag{29}$$

The first item in Eq. (28) represents the actual cost generated by the CSC, while the second item represents the expected cost generated by SBSs. Obviously, the first item of Eq. (28) is constant when the offloading strategy $\widetilde{\mathcal{A}}$ is fixed, so $Z(\widetilde{\mathcal{A}},\mathcal{F})$ can be regarded as a new objective function. Therefore, problem (26) can be rewritten as:

$$\min_{\mathcal{F}} Z(\widetilde{\mathcal{A}},\mathcal{F}) \tag{30}$$

$$\text{s.t.} \quad (14),(19). \tag{31}$$

Next, according to Eq. (29), the second-order derivative of $Z(\widetilde{\mathcal{A}},\mathcal{F})$ is calculated to get w.r.t $f_{n.s}$, which is expressed as:

$$\begin{aligned}&\frac{\partial^2 Z}{\partial^2 f_{n,s}} = \frac{2c_n\lambda_n\omega}{f_{n,s}^3} > 0, \quad \forall s\in\mathcal{S} n\in\mathcal{N}_s,\\ &\frac{\partial^2 Z}{\partial f_{n,s}\partial f_{v,w}} = 0, \qquad\quad \forall (n,s)\neq(v,w),\end{aligned} \tag{32}$$

where the Hessian matrix of $Z(\widetilde{\mathcal{A}},\mathcal{F})$ is diagonal with strictly positive elements, so it is positive-definite. Hence, problem (30) is convex and can be solved with the Lagrangian duality and Karush-Kuhn-Tucker (KKT) conditions [49]. However, solving a multi-variable system of non-linear equations brings high complexity [50].

In order to reduce the algorithm complexity, we design Algorithm 1 to solve this *sub-problem*. We first divide the computation resources of each SBS into many tiny atomic pieces, denoted by φ, and then allocate these computation resources pieces to each MD one by one. The key of the Algorithm 1 is to allocate computation resources pieces to the MD with the fastest change (which means that the gradient is negative and smallest) in each assignment. It is worth noting that the smaller the division of computation resources, the closer the final solution is to the optimal solution. More details are shown in Algorithm 1.

Remark 2. *(Complexity Analysis of Algorithm 1)*

In Algorithm 1, we denote f_s^{rem} as the remaining computation resources of SBS s under a fixed offloading strategy $\widetilde{\mathcal{A}}$, and $\mathcal{S}^{win}=\{s\in\mathcal{S}|\sum_{n\in\mathcal{N}}\sum_{i\in\mathcal{M}} a_{n,s}^i>0\}$ as the set of winning bids in the reverse auction. The complexity of Algorithm 1 depends on $|\mathcal{S}^{win}|$ and the number of computation resources pieces ν_s (which is calculated by φ and f_s^{rem}). In the worst case, the loop complexity of the inner layer is up to $O(\nu_s)$, while the complexity of the outer layer is $O(|\mathcal{S}^{win}|)$. In summary, we can know that the maximum complexity of Algorithm 1 is $O(|\mathcal{S}^{win}|\times\nu)$.

Algorithm 1 Constrained Gradient Descent Allocation Method (*CGDAM*).

Require: $\widetilde{\mathcal{A}}$, φ;
Ensure: $\mathcal{F}$;
function CGDAM($\widetilde{\mathcal{A}}$)
  for each SBS $s \in \mathcal{S}^{win}$ **do**
    for each MD $n \in \mathcal{N}_s$ **do**
      $f_{n,s} \leftarrow f_{n,s}^{min}$;
    end for
    $f_s^{rem} \leftarrow f_s - \sum_{n\in\mathcal{N}_s} f_{n,s}$;
    $\nu_s \leftarrow \lfloor f_s^{rem}/\varphi \rfloor, \forall s \in \mathcal{S}^{win}$;
  end for
  for each SBS $s \in \mathcal{S}^{win}$ **do**
    $\sigma \leftarrow \frac{\partial Z}{\partial f_{n,s}}$, $count \leftarrow 0$;
    $G = \{g_n = \sigma(f_{n,s}) \mid n \in \mathcal{N}_s\}$;
    while $count < \nu_s$ **and** $\exists g_n \in G : g_n < 0$ **do**
      $n \leftarrow \text{argmin}_{n\in\mathcal{N}_s}\{\text{G}\}$;
      $f_{n,s} \leftarrow f_{n,s} + \varphi$; $g_n \leftarrow \sigma(f_{n,s})$;
      $count \leftarrow count + 1$;
    end while
  end for
  return $\mathcal{F}$
end function

6.2 Winning Bid Selection

With Algorithm 1, we can obtain the solution of computation resource allocation strategy $\mathcal{F}^*$ under the fixed offloading strategy. Then, according to Eqs. (26), (28) and (30), we have:

$$Y^*(\mathcal{A}) = \varepsilon \sum_{n\in\mathcal{N}_c} f_{n,c} + Z(\mathcal{A}, \mathcal{F}^*). \tag{33}$$

According to Eq. (33), we can rewrite the problem in (24) as:

$$\min_{\mathcal{A}} \ \varepsilon \sum_{n\in\mathcal{N}_c} f_{n,c} + Z(\mathcal{A}, \mathcal{F}^*) \tag{34}$$

$$\text{s.t.} \quad (12), (15), (16), (18), (20). \tag{35}$$

Obviously, it is very difficult to solve this combinatorial optimization problem in polynomial time. The simplest way to solve this problem is by using the exhaustive method, but it takes $2^{N\times S\times M}$ times to search all the solutions, which is obviously impractical. Therefore, we design a Greedy Randomized Adaptive Search Procedure based Winning Bid Selection Method, named GWBSM to solve this *master-problem*, which can find approximate solutions in polynomial time. We first give some related definitions.

Definition 1. *We define $cost_{n,s}^i$ to indicate the minimum actual cost generated by MD n offloading the computation task to SBS s on sub-band i, which is given as:*

$$cost_{n,s}^i = \phi_s \hat{f}_{n,s}^{min} = \frac{\phi_s c_n}{t_n^{\max} - \frac{d_n}{\hat{r}_{n,s}}}, \tag{36}$$

where $\hat{f}_{n,s}^{min} = \frac{c_n}{t_n^{\max} - \frac{d_n}{\hat{r}_{n,c}}}$ denotes the minimum computation resource required when SBS s provides offloading services to MD n without considering inter-cell interference.

Algorithm 2 Construct Greedy Randomized Solution (*CGRS*).

Require: θ;
Ensure: $\mathcal{A}$;
1: **function** CGRS(θ)
2: $\mathcal{A} = \{a_{n,s}^i = 0 | \forall n \in \mathcal{N}, s \in \mathcal{S}, i \in \mathcal{M}\}$;
3: $\mathcal{P} = \{profit_{n,s}^i\}$; ▷ According to **Definition 3**
4: **while** $\mathcal{P} \neq \emptyset$ **do**
5: $profit^{\min} \leftarrow \min\{\mathcal{P}\}, profit^{\max} \leftarrow \max\{\mathcal{P}\}$;
6: $RCL \leftarrow \{profit_{n,s}^i \in T \mid profit_{n,s}^i \geq profit^{\min} + \theta\left(profit^{\max} - profit^{\min}\right)\}$;
7: Randomly select an element $profit_{h,k}^j$ from RCL;
8: $\Psi = \{profit_{n,s}^l | n = h, s = k, \forall l \in \mathcal{M}\}$;
9: **for** each $profit_{h,k}^l \in \Psi$ **do**
10: $\widetilde{\mathcal{A}} \leftarrow \mathcal{A}\backslash\{a_{h,k}^l\}$, $a_{h,k}^l \leftarrow 1$;
11: $\widetilde{\mathcal{A}} \leftarrow \widetilde{\mathcal{A}} \cup \{a_{h,k}^l\}$;
12: **if** $\widetilde{\mathcal{A}}$ satisfies constraints (27),(28) **then**
13: $\mathcal{A} \leftarrow \widetilde{\mathcal{A}}$, $j \leftarrow l$;
14: Skip to step 17;
15: **end if**
16: **end for**
17: $\mathcal{P} \leftarrow \mathcal{P}\backslash\left\{profit_{n,s}^i \in \mathcal{P} \mid n = h, \forall s \in \mathcal{S}, i \in \mathcal{M}\right\}$;
18: $\mathcal{P} \leftarrow \mathcal{P}\backslash\left\{profit_{n,s}^i \in \mathcal{P} \mid s = k, i = j, \forall n \in \mathcal{N}\right\}$;
19: **end while**
20: **return** $\mathcal{A}$
21: **end function**

Definition 2. *Similar to Definition 1, we define $cost_{n,c}^i$ to indicate the minimum actual cost for MD n to offload the computation task to the CSC, which is given as:*

$$cost_{n,c} = \varepsilon f_{n,c}^{min} = \frac{\varepsilon c_n}{t_n^{\max} - \frac{d_n}{r_{n,c}}}. \tag{37}$$

Definition 3. *According to Definition 1 and Definition 2, we define $profit_{n,s}^i$ to represent the cost saved (also called the increased profit) for the CSC when the computation task of MD n is offloaded to SBS s on sub-band i, which is given as:*

$$profit_{n,s}^i = cost_{n,c} - cost_{n,s}^i. \tag{38}$$

As shown in Algorithm 4, each iteration of GWBSM mainly includes two stages: construct greedy randomized solution and local search. The construction stage is mainly used to generate a feasible solution, and then its solution will be put into the local search for neighborhood search until a local optimal solution is found. Next, we introduce these two stages in detail.

6.2.1 Construct Greedy Randomized Solution

A simple way to construct the greedy solution is to select the offloading strategy that can bring the largest profit to the CSC according to Definition 3. However, the solution obtained in this way will make GWBSM fall into a local optimum, so we introduce a random factor that is the greedy value denoted by $\theta \in (0, 1)$.

According to Definition 3, we first evaluate the profits of offloaded tasks to SBSs by each MD. Then, we build a Restricted Candidate List (RCL) based on the greedy value θ (Lines 5-6), and randomly select an element from RCL to determine whether it can be added to the current offloading strategy (Lines 7-16). In each while-loop of the selection of the element from RCL, regardless of

Algorithm 3 Local Search (*LS*).

Require: $\mathcal{A}$;
Ensure: $\mathcal{A}, \mathcal{F}$;
1: **function** LS($\mathcal{A}$)
2: $\mathcal{F} \leftarrow CGDAM(\mathcal{A})$;
3: **for** each SBS $s \in \mathcal{S}^{win}$ **do**
4: For MDs assisted by SBS s for computation offloading, we use 2-opt operations to exchange MDs between different sub-bands to obtain a new offloading strategy $\widetilde{\mathcal{A}}$.
5: **if** $\widetilde{\mathcal{A}}$ satisfies constraints (24)-(28) **then**
6: $\widetilde{\mathcal{F}} \leftarrow CGDAM(\widetilde{\mathcal{A}})$
7: **if** $Y(\mathcal{A}, \mathcal{F}) > Y(\widetilde{\mathcal{A}}, \widetilde{\mathcal{F}})$ **then**
8: $\mathcal{A} \leftarrow \widetilde{\mathcal{A}}, \mathcal{F} \leftarrow \widetilde{\mathcal{F}}$;
9: **end if**
10: **end if**
11: **end for**
12: **for** each sub-band $m \in \mathcal{M}$ **do**
13: For MDs that offload tasks on sub-band m of SBSs, we use 2-opt operation to exchange MDs between different SBSs to obtain a new offloading strategy $\widetilde{\mathcal{A}}$.
14: **if** $\widetilde{\mathcal{A}}$ satisfies constraints (24)-(28) **then**
15: $\widetilde{\mathcal{F}} \leftarrow CGDAM(\widetilde{\mathcal{A}})$
16: **if** $Y(\mathcal{A}, \mathcal{F}) > Y(\widetilde{\mathcal{A}}, \widetilde{\mathcal{F}})$ **then**
17: $\mathcal{A} \leftarrow \widetilde{\mathcal{A}}, \mathcal{F} \leftarrow \widetilde{\mathcal{F}}$;
18: **end if**
19: **end if**
20: **end for**
21: **return** $\mathcal{A}, \mathcal{F}$
22: **end function**

whether this element is added to the current offloading strategy or not, the relevant element needs to be removed from the remaining optional elements to ensure that the constraints (15)(16) are satisfied (Lines 17-18). More details are shown in Algorithm 2.

6.2.2 Local Search

In the first stage of construction, the feasible solution obtained may not be optimal, so a local search should be performed in the neighborhood of the feasible solution. The basic idea of this stage is to search the neighborhood of the feasible solution generated in the construction stage, and replace the current feasible solution with the optimal adjacent solution to find the current optimal solution (Lines 7-9 and Lines 16-18). The design of the specific neighborhood operator is introduced in Line 4 and Line 13 of Algorithm 3. More details are shown in Algorithm 3.

It can be found that GWBSM has two main parameters: greedy value θ and **Max-Iteration**. The number of elements in RCL depends on θ. $\theta = 0$ means that any feasible element will be randomly selected in the construction stage, which will make the solution search range too wide and difficult to find the optimal solution. On the contrary, $\theta = 1$ means that only the elements with the largest profits will be selected, which makes GWBSM fall into a local optimum. For the **Max-Iteration**, it is the condition for the termination of GWBSM. In each iteration, GWBSM will find a new solution to replace the current optimal solution (Lines 4-8 in Algorithm 4), but the probability that the new solution is better than the current solution will decrease as the number of iterations increases. Therefore, the larger the value of **Max-Iteration**, the better the quality of the final solution. However, the complexity of GWBSM is linearly related to **Max-Iteration**, so how to set a proper **Max-Iteration** is also crucial. The detail of the proposed GWBSM is shown in Algorithm 4.

Algorithm 4 Greedy Randomized Adaptive Search Procedure based Winning Bid Selection Method (*GWBSM*).

Require: q_n, e_n, φ, θ, **Max-Iteration**;
Ensure: $\mathcal{A}$, $\mathcal{F}$;
1: $\mathcal{A}, \mathcal{F} \leftarrow$ Set to offload all MDs' tasks to the CSC;
2: **for** iter to **Max-Iteration do**
3: $\widetilde{\mathcal{A}} \leftarrow CGRS(\theta)$;
4: $\widetilde{\mathcal{A}}, \widetilde{\mathcal{F}} \leftarrow LS(\widetilde{\mathcal{A}})$;
5: **if** $Y(\mathcal{A}, \mathcal{F}) > Y(\widetilde{\mathcal{A}}, \widetilde{\mathcal{F}})$ **then**
6: $\mathcal{A} \leftarrow \widetilde{\mathcal{A}}, \mathcal{F} \leftarrow \widetilde{\mathcal{F}}$;
7: **end if**
8: **end for**
9: **return** $\mathcal{A}, \mathcal{F}$

Remark 3. *(Complexity Analysis of Algorithm 4)*

*Obviously, the complexity of Algorithm 4 depends on Algorithm 2, Algorithm 3, and **Max-Iteration**. For Algorithm 2, only one layer of while-loop is included, and its maximum complexity is $O(N)$. For Algorithm 3, its complexity depends on Algorithm 1 and the neighborhood size of the current feasible solution. Then, according to Remark 2, we can get the complexity of Algorithm 3 as $O(\frac{M|\mathcal{S}^{win}|(|\mathcal{S}^{win}|+M)}{2} \times |\mathcal{S}^{win}| \times \nu)$. We use ξ to represent **Max-Iteration** and let $|\mathcal{S}^{win}|$ be equal to W. Since the complexity of Algorithm 2 is too small to be ignored, the complexity of Algorithm 4 can be obtained as $O(\frac{MW^2(W+M) \times \nu \times \xi}{2})$. Furthermore, since GWBSM is a multi-start heuristic algorithm, and the solution in each iteration can be computed independently, the runtime can be greatly reduced by utilizing parallel computing.*

6.3 Payment Determination

For each winning bid, the CSC needs to make a reasonable payment determination to ensure that each SBS will honestly report the value of its resources. In this paper, we adopt a payment rule based on the standard Vickrey-Clarke-Groves (VCG) scheme for GWBSM [26]. The proposed payment rule can encourage SBSs to participate in computation offloading, while ensuring the individual rationality and truthfulness properties.

In the standard VCG scheme, each winner will pay the "opportunity cost" caused to other participants. The "opportunity cost" of bidder s is defined as the total bids of all the other bidders that would win without the participation of bidder s, minus the sum of bids of all the other actual winning bidders. Next, we introduce the following definitions:

Definition 4. *We define β_s to represent the increment in CSC's revenue after choosing SBS s to participate in the computation offloading, which is given as:*

$$\beta_s = \varepsilon \sum_{n \in \mathcal{N}_s} f_{n,c}^{min} + \sum_{n \in \mathcal{N}_s} \lambda_n \omega \left(t_n^{\max} - t_n(\mathcal{A}, \mathcal{F}) \right). \quad (39)$$

Definition 5. *We define α_s to represent the sum cost of SBS s by serving its associated MDs, which is given as:*

$$\alpha_s = \phi_s \sum_{n \in \mathcal{N}_s} f_{n,s}. \quad (40)$$

Algorithm 5 Payment Determination in GWBSM.

Require: $\mathcal{A}, \mathcal{F}$
Ensure: p_s
1: **for** each SBS $s \in \mathcal{S}$ **do**
2: $p_s \leftarrow 0$;
3: **end for**
4: **for** each SBS $s \in \mathcal{S}^{win}$ **do**
5: $Y_{\mathcal{S}}^{-s}(\mathcal{A}, \mathcal{F}) = Y_{\mathcal{S}}(\mathcal{A}, \mathcal{F}) + (\beta_s - \alpha_s)$;
6: $\mathcal{S} \leftarrow \mathcal{S} \backslash \{s\}$
7: Update $\mathcal{S}^{win}$ according to Algorithm 4;
8: Calculate p_s according to Eq. (42);
9: $\mathcal{S} \leftarrow \mathcal{S} \cup \{s\}$;
10: **end for**
11: **return** $p_s, \forall s \in \mathcal{S}^{win}$

According to Definition 4 and Definition 5, we define $Y_{\mathcal{S}}^{-s}(\mathcal{A}, \mathcal{F})$ as the optimal solution without considering the profit brought by SBS s, which can be expressed as:

$$Y_{\mathcal{S}}^{-s}(\mathcal{A}, \mathcal{F}) = Y_{\mathcal{S}}(\mathcal{A}, \mathcal{F}) + (\beta_s - \alpha_s). \tag{41}$$

Furthermore, we use $Y_{\mathcal{S}\backslash\{s\}}(\mathcal{A}, \mathcal{F})$ to denote the new optimal solution without considering the participation of SBS s. Then, the payment paid to SBS s is given as:

$$p_s = \beta_s - \left(Y_{\mathcal{S}}^{-s}(\mathcal{A}, \mathcal{F}) - Y_{\mathcal{S}\backslash\{s\}}(\mathcal{A}, \mathcal{F})\right). \tag{42}$$

Let $\sigma_s = v_s \sum_{n \in \mathcal{N}_s} f_{n,s}$ denote the sum of the true valuation consumed by SBS s in the computation offloading process. Then, the utility of each SBS $s \in \mathcal{S}^{win}$ is defined as:

$$\delta_s = p_s - \sigma_s. \tag{43}$$

We define the payment of those SBSs $s \notin \mathcal{S}^{win}$ as 0, then the details of the proposed Payment Determination are shown in Algorithm 5.

Remark 4. *(Complexity Analysis of Algorithm 5)*
According to Remark 3, we know that the complexity of Algorithm 5 is $O(\frac{MW^3(W+M) \times \nu \times \xi}{2})$.

6.4 Proof of Properties

In this part, we prove that the payment rule satisfies two crucial properties: individual rationality and truthfulness. The individual rationality guarantees that each winner can get a non-negative utility, which is essential for SBSs to participate in the computation offloading process. The truthfulness prevents SBSs obtaining higher utility by bidding untruthfully.

Theorem 1. *(Individual Rationality). The payment rule defined in Eq.* (42) *satisfies the individual rationality property, i.e.,* $\forall s \in \mathcal{S}, \delta_s = p_s - \sigma_s \geq 0$.

Proof. Based on Eq. (42), we can get:

$$\begin{aligned} p_s &= \beta_s - \left(Y_{\mathcal{S}}^{-s}(\mathcal{A}, \mathcal{F}) - Y_{\mathcal{S}\backslash\{s\}}(\mathcal{A}, \mathcal{F})\right) \\ &= \beta_s - \left(Y_{\mathcal{S}}(\mathcal{A}, \mathcal{F}) - Y_{\mathcal{S}\backslash\{s\}}(\mathcal{A}, \mathcal{F}) + \beta_s - \alpha_s\right) \\ &= Y_{\mathcal{S}\backslash\{s\}}(\mathcal{A}, \mathcal{F}) - Y_{\mathcal{S}}(\mathcal{A}, \mathcal{F}) + \alpha_s. \end{aligned}$$

When each SBS $s \in \mathcal{S}$ bids truthfully, i.e., $\alpha_s = \sigma_s$, we can obtain:

$$\begin{aligned} \delta_s &= p_s - \sigma_s \\ &= Y_{\mathcal{S}\backslash\{s\}}(\mathcal{A}, \mathcal{F}) - Y_{\mathcal{S}}(\mathcal{A}, \mathcal{F}) \\ &\geq 0. \end{aligned}$$

Hence, the individual rationality property is satisfied. □

Theorem 2. *(Truthfulness). The payment rule defined in Eq.* (42) *satisfies the truthfulness property, i.e., it is a weakly dominant strategy for each SBS to set the bid* $\phi_s = v_s$.

Proof. We assume that a certain SBS s declares the bid $\phi_s^{'}$ untruthfully, i.e., $\phi_s' > v_s$. According to Eq. (43), the utility that SBS s receives becomes:

$$\begin{aligned} \delta_s' &= p_s' - \sigma_s \\ &= \beta_s' - \left(Y_{\mathcal{S}}^{-s}(\mathcal{A}', \mathcal{F}') - Y_{\mathcal{S}\backslash\{s\}}(\mathcal{A}, \mathcal{F})\right) - v_s \sum_{n \in \mathcal{N}_s} f_{n,s}. \end{aligned}$$

Then, the difference of SBS $s \in \mathcal{S}^{win}$'s utility after submitting the untruthful bid and the truthful bid is given by:

$$\begin{aligned} \hat{\delta}_s &= \delta_s' - \delta_s \\ &= \beta_s' - \left(Y_{\mathcal{S}}^{-s}(\mathcal{A}', \mathcal{F}') - Y_{\mathcal{S}\backslash\{s\}}(\mathcal{A}, \mathcal{F})\right) - v_s \sum_{n \in \mathcal{N}_s} f_{n,s} \\ &\quad - \left(Y_{\mathcal{S}\backslash\{s\}}(\mathcal{A}, \mathcal{F}) - Y_{\mathcal{S}}(\mathcal{A}, \mathcal{F}) + \alpha_s - v_s \sum_{n \in \mathcal{N}_s} f_{n,s}\right) \\ &= \beta_s' - Y_{\mathcal{S}}^{-s}(\mathcal{A}', \mathcal{F}') + Y_{\mathcal{S}}(\mathcal{A}, \mathcal{F}) - \alpha_s \\ &= \beta_s' - (Y_{\mathcal{S}}(\mathcal{A}', \mathcal{F}') + \beta_s' - \alpha_s') + Y_{\mathcal{S}}(\mathcal{A}, \mathcal{F}) - \alpha_s \\ &= Y_{\mathcal{S}}(\mathcal{A}, \mathcal{F}) - \alpha_s - (Y_{\mathcal{S}}(\mathcal{A}', \mathcal{F}') - \alpha_s'). \end{aligned}$$

Since $\mathcal{S}^{win}$ is the solution that minimizes the objective function Eq. (8), we have:

$$Y_{\mathcal{S}}(\mathcal{A}, \mathcal{F}) - \alpha_s \leq Y_{\mathcal{S}}(\mathcal{A}', \mathcal{F}') - \alpha_s'.$$

Therefore, $\hat{\delta}_s \leq 0$, which means SBSs cannot increase their utility by bidding untruthfully. □

7 Performance Evaluation

The optimization problem in this paper belongs to NP-hard, and it is impossible to obtain the optimal solution in polynomial time. Therefore, we cannot obtain the optimality gap of the proposed algorithm theoretically. We analyze the optimality gap of the proposed algorithm through simulation. This section evaluates the performance of RACORAM as well as compares it with other baseline methods, and studies the impact of parameters on the performance of RACORAM.

7.1 Simulation Settings

In the simulation, we consider that in a single-cell scenario, there are several SBSs uniformly distributed and multiple MDs randomly distributed. Parameters in the simulation are mainly set according to the parameters in [1], [2], [4], [5]. For each SBS, unless otherwise specified, we assume that the uplink bandwidth is set to $B = 20/40$ MHz, and the idle computation resources f_s is in the range of $[10, 20]$ GHz. For the uplink channel gain, we use the distance-dependent path-loss model to calculate, which is given as $L[\mathrm{dB}] = 140.7 + 36.7 \log_{10} d_{[\mathrm{km}]}$ [54], and the log-normal shadowing standard deviation is set to 8 dB. The background noise power is set to $\sigma^2 = -100$ dBm, and the MDs' transmission power is set to $p_n = 20$ dBm. In addition, each SBS's bid of per unit computation resource ϕ_s is normally distributed over $[0.1, 0.2]$ monetary units (e.g., US dollars, or RMB)$/(Megacycle)$. Similarly, the CSC's cost of per unit computation resource ε is set to $0.3/(Megacycle)$. We assume that

TABLE 2. Computation Task Type

Type	Data Size [MBs]	CPU Cycle [Megacycles]	Maximum Tolerable Delay [s]
Type 1	0.1	3000	1
Type 2	0.2	2000	1.5
Type 3	0.3	1500	2

there are three types of computation tasks generated by MDs, and the specific information is shown in Table 2. In the simulations, unless stated otherwise, the computation tasks generated by each MD will randomly be one of the three types.

7.2 Compared Baselines

The proposed RACORAM is compared with the following baselines.

- *Exhaustive Offloading and Joint CGDAM* **(EOJC)**: As we mentioned before, the exhaustive method takes $2^{N\times S\times M}$ times to find out the optimal offloading strategy. Because of the enormous computational complexity of this method, we only evaluate its performance in the sparse network. Furthermore, CGDAM is used to allocate computation resources.
- *Random Offloading and Joint CGDAM* **(ROJC)**: The computation tasks of MDs are randomly assigned to the nearby SBSs, and tasks which cannot be handled by the SBSs will be uploaded to the CSC. Meanwhile, CGDAM is also used to allocate computation resources.
- *Greedy Offloading and Joint Minimum Resource Allocation* **(GOJMRA)**: In GOJMRA, offloaded tasks prioritize selecting the offloading strategy that can bring the largest profits to the CSC according to Definition 3. Moreover, the computation resources allocated to each task offloaded to SBSs only meet their maximum delay requirements.
- *ALL Cloud Execution* **(ACE)**: All computation tasks are served by the CSC. Certainly, this method performs worst, and will not be analyzed in the following performance evaluation.
- *Winning Bid Selection* **(WBS)** [44]: WBS is divided into two phases. In phase 1, the bid set and the SBS set are updated, and the bids which cannot satisfy the delay constraints are removed; In phase 2, an n-to-one weighted bipartite graph with capacity constraints is constructed, the CSC selects suitable SBSs to form a maximum matching with the approximately maximum weight. The computation resources for each MD are allocated according to Algorithm 1, and phases 1 and 2 are repeated until the suitable SBS set is empty.

For fairness, the payment rules of the above five methods are the same as that in RACORAM, which has been shown in Algorithm 5.

7.3 Impact of Greedy Value and Max-Iteration

This part evaluates the impact of greedy value θ and **Max-Iteration** on the performance of the proposed RACORAM. We design four types of RACORAM with different greedy values, which are $\{0.2, 0.4, 0.6, 0.8\}$ respectively. Meanwhile, the value of **Max-Iteration** is in the range of $[1, 30]$. We set the number of

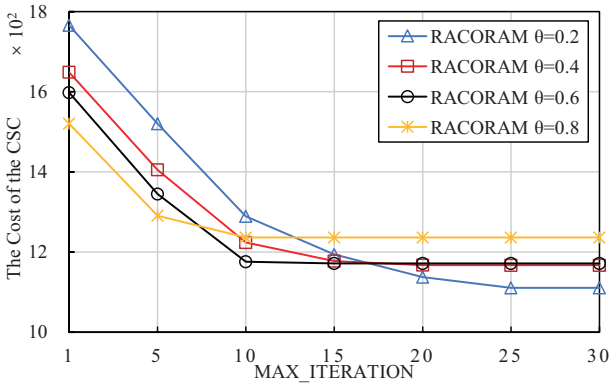

Fig. 2. The cost of the CSC under RACORAM with different greedy values.

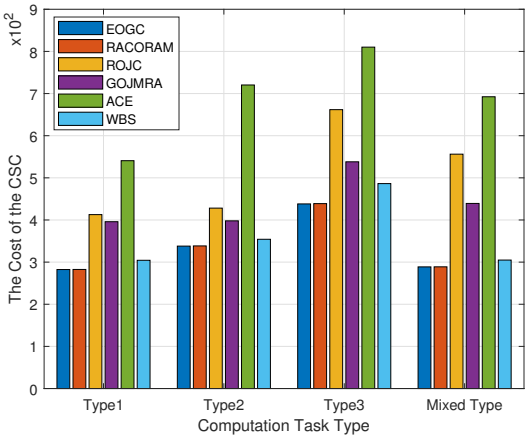

Fig. 3. Comparison of the CSC's cost under different computation task types.

MDs as $N = 25$, the number of SBSs as $S = 8$, and each SBS has $M = 4$ sub-bands.

Fig. 2 shows the performance comparison of four types of RACORAM in terms of the CSC's cost with the increase of the value of **Max-Iteration**. It can be found that with the increase of the value of **Max-Iteration**, the quality of the final solution found by RACORAM is better, so the CSC's cost will decrease continuously. However, as the solution found by RACORAM tends to the optimal solution, the CSC's cost will gradually stabilize. Furthermore, as the greedy value θ increases, the minimum value of **Max-Iteration** required to stabilize the cost of the CSC decreases. This is because the increase of θ will reduce the size of RCL, so RACORAM only needs a smaller value of **Max-Iteration** to find the corresponding suboptimal solution. On the other hand, as θ decreases, the CSC's cost becomes smaller when it stabilizes. This is because reducing θ will increase the size of RCL, so the higher probability of a better solution of RACORAM is found.

Considering the CSC's cost and the running time of RACORAM, we set greedy value $\theta = 0.6$ and **Max-Iteration** $= 10$ in the following simulations.

7.4 Performance Comparison

In this part, the performance of RACORAM is compared with the optimal solution obtained by EOJC and also with the other

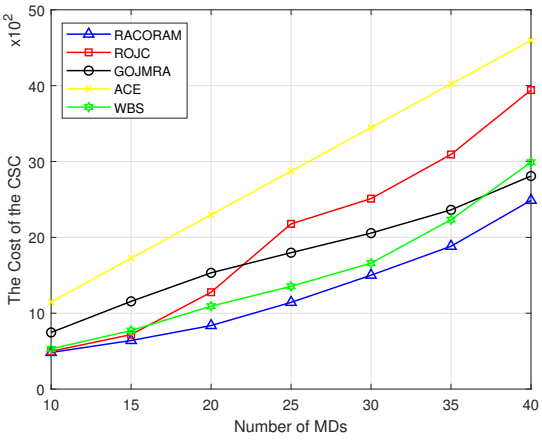

Fig. 4. Comparison of the CSC's cost under different number of MDs.

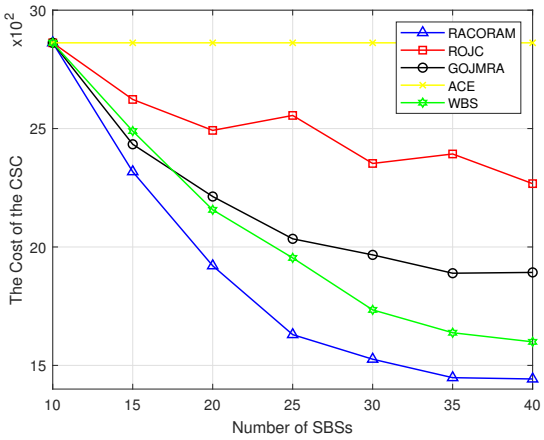

Fig. 5. Comparison of the CSC's cost under different number of SBSs.

baseline methods in terms of the CSC's cost. Then, we compare the performance of RACORAM with ROJC, GOJMRA, ACE, and WBS in terms of the CSC's cost under different scenarios, while the EOJC method is omitted due to the enormous computational complexity.

7.4.1 Suboptimality

Firstly, we verify the suboptimal nature of our proposed RACORAM. Although EOJC can obtain the optimal solution, its enormous computational complexity makes the simulation only available in the sparse network. Therefore, we set the number of MDs as $N = 6$, the number of SBSs as $S = 4$, and each SBS has $M = 2$ sub-bands.

Fig. 3 shows the performance comparison of RACORAM with other baseline methods in terms of the CSC's cost under different computation task types. Since some tasks may be more sensitive to delay, or have larger data size, or have larger required CPU cycles, we use the task type to represent different types of tasks generated in the system. Through this process, we can evaluate whether our proposed solution is suitable for general situations. For the abscissa, Type 1 (or Type 2, Type 3) means that all computation tasks generated by MDs in the system are Type 1 (or Type 2, Type 3), but Mixed Type means that the computation tasks generated by each MD will randomly be one of the three types. It can be found that the performance of RACORAM is very close to the optimal solution EOJC, and its performance is significantly better than other baseline methods. In addition, the average running time of EOJC is more than 100 times that of RACORAM in such a small network. This is because the greedy randomized adaptive search procedures used by RACORAM have excellent convergence stability and global exploration capabilities, and can obtain a suboptimal solution that is comparable to the exhaustive method in a short time. We also observe that with the increase of the maximum tolerable delay required for different types of computation tasks, the performance superiority of RACORAM compared to other baseline methods is more obvious. The main reason is that RACORAM considers the improvement of MDs' QoS in the optimization problem.

7.4.2 The CSC's Cost Under Different Number of MDs

This part compares RACORAM with ROJC, GOJMRA, ACE, and WBS in terms of the CSC's cost under different number of MDs.

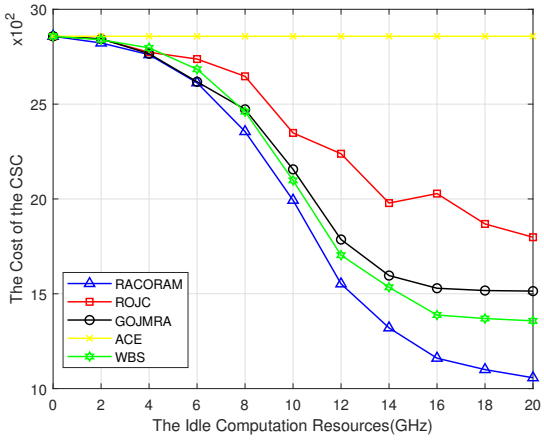

Fig. 6. Comparison of the CSC's cost under different idle computation resources of SBSs.

We set the number of SBSs as $S = 12$, and each SBS has $M = 4$ sub-bands. Meanwhile, the number of MDs is in the range of $[10, 40]$.

Fig. 4 shows the performance comparison of RACORAM, ROJC, GOJMRA, ACE, and WBS in terms of the CSC's cost under different number of MDs. It can be found that with the increase of the number of MDs, more computation tasks will be offloaded by MDs, so the cost of the CSC will increase continuously. The CSC's cost in RACORAM increases slowly as the number of MDs increases, compared with other baselines, which illustrates that RACORAM performs best. When the number of MDs increases to a large value, the CSC's cost of our proposed RACORAM is close to GOJMRA and WBS. This is because when the SBS serves too many MDs, the idle computation resources of the SBS cannot meet or only meet the maximum delay of MDs. For WBS, since it cannot avoid the local optimum problem in the selection of the offloading strategy and does not consider the improvement of MDs' QoS, it performs worse than our proposed RACORAM. For GOJMRA, since the CSC selects the SBS with the largest utility for each MD in turn, without considering allocating additional computation resources to MDs, the performance of GOJMRA is slightly worse than WBS. For ROJC, SBSs have sufficient

idle resources to provide high-quality services to nearby MDs when the number of MDs is small, so the current performance of ROJC is close to RACORAM and better than WBS and GOJMRA. However, the randomly generated offloading strategies in ROJC cannot meet the resource constraints of SBSs when the number of MDs is larger, which causes most MDs to offload their computation tasks to the CSC. In this case, ROJC's performance is the worst compared with RACORAM, WBS, and GOJMRA.

7.4.3 The CSC's Cost Under Different Number of SBSs

This part compares RACORAM with ROJC, GOJMRA, WBS, and ACE in terms of the CSC's cost under different number of SBSs. We set the number of MDs as $N = 25$, and each SBS has $M = 4$ sub-bands. Meanwhile, the number of SBSs is in the range of $[0, 12]$.

Fig. 5 shows the performance comparison of RACORAM, ROJC, GOJMRA, WBS, and ACE in terms of the CSC's cost under different number of SBSs. With the increase of the number of SBSs, more SBSs can assist the CSC to offload computation tasks, so the cost of the CSC will decrease continuously. Our proposed RACORAM still performs best. This is because RACORAM uses a GWBSM algorithm to obtain the near-optimal offloading strategy in each SBS's coverage area, and considers the improvement of MDs' QoS in the optimization problem, so the CSC has a greater possibility to select more valuable SBSs to participate in computation offloading. In contrast, WBS only considers assigning computation tasks to the SBS that can bring the maximum profit, while ignoring the potential impact of channel conditions and unassigned computation tasks. Therefore, WBS can only find the local optimal solution. For GOJMRA, it assigns computation tasks to the SBS that can bring the maximum profit in turn, while ignoring the impact of the resource allocation. Therefore, the performance of GOJMRA is worse than WBS. Furthermore, it's obvious that ROJC performs worst. The main reason is that ROJC randomly assigns the computation tasks of each MD to nearby SBSs, resulting in a large number of SBSs unable to provide services under delay constraints.

7.4.4 The CSC's Cost Under Different Idle Computation Resources of SBSs

This part compares RACORAM with ROJC, GOJMRA, WBS, and ACE in terms of the CSC's cost under different idle computation resources of SBSs. We set the number of MDs as $N = 25$, the number of SBSs as $S = 12$, and each SBS has $M = 4$ sub-bands. Meanwhile, the idle computation resources of each SBS is in the range of $[0, 20]$ GHz.

Fig. 6 shows the performance comparison of RACORAM, ROJC, GOJMRA, WBS, and ACE in terms of the CSC's cost under different idle computation resources of SBSs. With the increase of the idle computation resources of SBSs, more MDs can be served by SBSs, so the CSC's cost will decrease continuously. Furthermore, the cost of the CSC in RACORAM decreases significantly as the idle computation resources of SBSs increase, compared with that of WBS, GOJMRA, and ROJC, which demonstrates that RACORAM performs best. When the idle computation resources of SBSs increase to a large value, i.e, 18 GHz, the CSC's cost in RACORAM, WBS, and ROJC still slowly decrease while GOJMRA is almost unchanged. This is because when the idle computation resources of SBSs are sufficient, each SBS can meet the computation resource requirements of all MDs in its coverage area under the maximum tolerable delay. We know that the computation resources allocated to MDs in GOJMRA only meet their maximum tolerable delay requirements, so if the idle computation resources of SBSs are sufficient to complete the computation task within the maximum tolerable delay, then GOJMRA will almost unchange. In contrast, RACORAM and ROJC use the CGDAM algorithm to allocate additional idle computation resources when the idle computation resources of SBSs are sufficient, thereby reducing the expected cost of the CSC. Obviously, ROJC still performs worst.

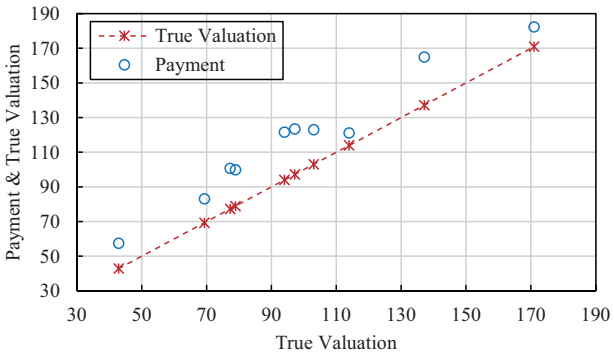

Fig. 7. Payment & True Valuation of Winning SBSs.

7.5 Evaluation of Individual Rationality and Truthfulness

In this part, we verify the individual rationality and truthfulness properties of the proposed payment determination in RACORAM. Firstly, we verify the individual rationality by comparing the payment of each winning SBS and its corresponding true valuation of cost in computation offloading. Then, we verify the truthfulness by randomly selecting a winning SBS to observe how its utility (according to Eq. (43)) evolves with the variant of bidding price.

Fig. 7 shows the individual rationality of the proposed payment determination in RACORAM. It can be found that the payment of each winning SBS is higher than its true valuation of the cost in computation offloading, which means that each winning SBS can get a positive utility when its bidding price is truthful. Therefore, we verify that the proposed payment determination can guarantee the individual rationality of the winning SBSs in RACORAM.

Fig. 8 shows the truthfulness of the proposed payment determination in RACORAM. We randomly select a winning SBS according to RACORAM, and its true valuation of cost in computation offloading is estimated to be 81.27 or 0.12 per unit of computation resource. Then, it can be found that when the bidding price of the SBS (per unit of computation resource) is lower than its true cost, it will not be selected as the winning bidder to ensure the property of individual rationality. However, when the bidding price of the SBS is much higher than its true cost, it will not be selected as the winning bidder either. This SBS can be selected as the winning bidder only when its bidding price is equal to or close to its true cost (0.12), i.e., within the range of $[0.12, 0.15]$. And even if SBS's bidding price increases, the payment provided by the CSC is always equal to $107.8 > 81.27$, which proves that the SBS cannot get higher payment from untruthful bidding price. Therefore, we verify that the proposed payment determination can guarantee the truthfulness of the winning SBSs in RACORAM.

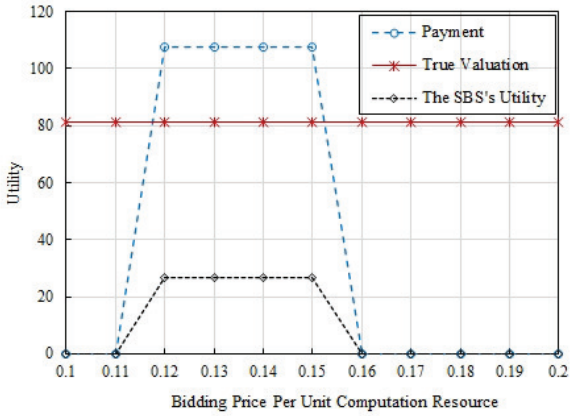

Fig. 8. Utility & Bidding Price of Wining SBSs.

TABLE 3. Running Time(ms)

Network Type $N/(S*M)$	Sparse $6/(4*2)$	Denser $25/(8*4)$	Ultra-Denser $40/(12*4)$
EOJC	2.28×10^6	-	-
RACORAM	1.73×10^4	9.89×10^5	6.94×10^6
ROJC	2.41×10^2	7.19×10^2	4.26×10^3
GOJMRA	1.92×10^2	8.61×10^2	9.53×10^3
WBS	1.47×10^4	7.80×10^5	1.06×10^6

7.6 Computational Efficiency

This part investigates the computational efficiency of RACORAM, and records the running time of RACORAM and other baseline methods in different network scenarios, as shown in Table 3. For each network scenario, we perform 50 iterations of each method and take the average of the running time to measure the computational efficiency.

From the table, N represents the number of MDs, and S and M represent the number of SBSs and each SBS's sub-bands respectively. We observe that the running time of RACORAM gradually rises as the network scenario grows from sparse to ultra-denser. It is worth noting that the relationship between the growth rate of running time and the network size is not exponential. This is consistent with the claim that RACORAM follows a polynomial computation time with respect to the scale of the network, which is shown in Remark 4. On the one hand, RACORAM can find sub-optimal solutions in a shorter time than EOJC. On the other hand, compared with GOJMRA, ROJC, and WBS, RACORAM further reduces the cost of the CSC with an acceptable increase in the computational overhead. Furthermore, RACORAM can greatly reduce running time through parallel computing.

To summarize, the greedy value θ and **Max-Iteration** have a significant impact on the performance of RACORAM, and RACORAM is very close to the optimal method EOJC while it significantly outperforms the other baseline methods under different scenarios. In addition, RACORAM is superior to EOJC in terms of the computational efficiency. Furthermore, we prove that the proposed payment determination can guarantee the individual rationality and truthfulness properties.

8 Conclusion

In this paper, we have proposed a novel Reverse Auction-based Computation Offloading and Resource Allocation Mechanism, named RACORAM for the mobile Cloud-Edge computing. The optimization problem is formulated as a Mixed Integer Nonlinear Programming (MINLP) problem, aiming to minimize the cost of the CSC. We decomposed the original problem into an equivalent master problem and subproblem, and proposed low-complexity algorithms to solve the related optimization problems. Specifically, a Constrained Gradient Descent Allocation Method (CGDAM) is first proposed to determine the computation resource allocation strategy, and then a Greedy Randomized Adaptive Search Procedure based Winning Bid Scheduling Method (GWBSM) is proposed to determine the computation offloading strategy. Meanwhile, a VCG scheme-based payment determination for the winning SBSs is also presented. Simulation results illustrate that RACORAM is very close to the optimal method EOJC with significantly reduced computational complexity, and greatly outperforms the other baseline methods in terms of the CSC's cost under different scenarios.

References

[1] Y. Hui, Z. Su, T. H. Luan, C. Li, G. Mao and W. Wu, "A Game Theoretic Scheme for Collaborative Vehicular Task Offloading in 5G HetNets," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 16044-16056, 2020.

[2] S. Hu, and G. Li, "Dynamic Request Scheduling Optimization in Mobile Edge Computing for IoT Applications," *IEEE Internet Things J.*, vol. 7, no. 2, pp. 1426-1437, Feb. 2020.

[3] K. Jiang, C. Sun, H. Zhou, X. Li, M. Dong, and V. Leung. "Intelligence-Empowered Mobile Edge Computing: Framework, Issues, Implementation, and Outlook," *IEEE Netw.*, vol. 35, no. 5, pp. 74-82, 2021.

[4] S. Yu, X. Chen, Z. Zhou, X. Gong and D. Wu, "When Deep Reinforcement Learning Meets Federated Learning: Intelligent Multitimescale Resource Management for Multiaccess Edge Computing in 5G Ultradense Network," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2238-2251, 2021.

[5] P. Lai, et al., "Cost-Effective User Allocation in 5G NOMA-based Mobile Edge Computing Systems," *IEEE Trans. Mobile Comput.*, vol.PP, no. 99, pp. 1-1, 2021.

[6] H. Zhou, T. Wu, X. Chen, S. He, and J. Wu, "RAIM: A Reverse Auction-based Incentive Mechanism for Mobile Data Offloading through Opportunistic Mobile Networks," *IEEE Trans. Netw. Sci. Eng.*, vol. PP, no. 99, pp. 1-1, 2022.

[7] X. Chen, W. Li, S. Lu, Z. Zhou, and X. Fu, "Efficient resource allocation for on-demand mobile-edge cloud computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 9, pp. 8769-8780, Sep. 2018.

[8] H. Zhou, K. Jiang, X. Liu, X. Li, and V. C. M. Leung, "Deep Reinforcement Learning for Energy Efficient Computation Offloading in Mobile Edge Computing," *IEEE Internet Things J.*, vol. 9, no. 2, pp. 1517-1530, 2022.

[9] H. Zhou, Z. Wang, N. Cheng, D. Zeng, and P. Fan, "Stackelberg Game-based Computation Offloading Method in Cloud-Edge Computing Networks," *IEEE Internet Things J.*, vol. PP, no. 99, pp. 1-1, 2022.

[10] Z. Xu, W. Liang, M. Jia, M. Huang, and G. Mao, "Task Offloading with Network Function Requirements in a Mobile Edge-Cloud Network," *IEEE Trans Mobile Comput.*, vol. 18, no. 11, pp. 2672-2685, Nov. 2019.

[11] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update 2017–2022, Global Mobile Data Traffic Forecast, Cisco Syst., San Jose, CA, USA, 2019.

[12] B. Bangerter, S. Talwar, R. Arefi and K. Stewart, "Networks and devices for the 5G era," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 90-96, Feb. 2014.

[13] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450-465, Feb. 2018.

[14] F. Rebecchi, M. Dias de Amorim, V. Conan, A. Passarella, R. Bruno, and M. Conti, "Data offloading techniques in cellular networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 2, pp. 580-603, 2015.

[15] H. Zhou, T. Wu, H. Zhang, and J. Wu, "Incentive-driven Deep Reinforcement Learning for Content Caching and D2D Offloading," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 8, pp. 2445-2460, 2021.

[16] H. Zhou, Z. Zhang, D. Li, and Z. Su, "Joint Optimization of Computing Offloading and Service Caching in Edge Computing-based Smart Grid," *IEEE Transactions on Cloud Computing*, vol. PP, no. 99, pp. 1-1, 2022.

[17] N. Cheng, W. Xu, W. Shi, Y. Zhou, N. Lu, H. Zhou, and X. Shen, "Air-ground integrated mobile edge networks: Architecture, challenges, and opportunities," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 26-32, Aug. 2018.

[18] J. Ren, H. Guo, C. Xu, and Y. Zhang, "Serving at the edge: A scalable IoT architecture based on transparent computing," *IEEE Netw.*, vol. 31, no. 5, pp. 96-105, Aug. 2017.

[19] C. You, K. Huang, and H. Chae, "Energy efficient mobile cloud computing powered by wireless energy transfer," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 5, pp. 1757-1771, May 2016.

[20] X. Lin, Y. Wang, Q. Xie, and M. Pedram, "Task scheduling with dynamic voltage and frequency scaling for energy minimization in the mobile cloud computing environment," *IEEE Trans. Services Comput.*, vol. 8, no. 2, pp. 175-186, 2015.

[21] Y. Zhang, X. Lan, Y. Li, L. Cai, and J. Pan, "Efficient Computation Resource Management in Mobile Edge-Cloud Computing," *IEEE Internt Things J.*, vol. 6, no. 2, pp. 3455-3466, April 2019.

[22] J. Xie, D. Guo, C. Qian, and H. Chen, "A Fast Hybrid Data Sharing Framework for Hierarchical Mobile Edge Computing," *in Proc. of IEEE INFOCOM*, pp. 2609-2618, 2020.

[23] J. Liu, S. Guo, K. Liu, and L. Feng, "Resource Provision and Allocation Based on Microeconomic Theory in Mobile Edge Computing," *IEEE Trans. Services Comput.*, vol. PP, no. 99, pp. 1-1, 2020.

[24] X. Peng, K. Ota, M. Dong, and H. Zhou, "Online Resource Auction for Edge-assistant Vehicular Network with Non-price Attributes," *IEEE Trans. Veh. Technol.*, vol. 70, no. 7, pp. 7127-7137, 2021.

[25] N. C. Luong, P. Wang, D. Niyato, Y. Wen and Z. Han, "Resource Management in Cloud Networking Using Economic Analysis and Pricing Models: A Survey," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 2, pp. 954-1001, 2017.

[26] H. Zhou, X. Chen, S. He, J. Chen, and J. Wu, "DRAIM: A Novel Delay-constraint and Reverse Auction-based Incentive Mechanism for WiFi Offloading," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 4, pp. 711-722, 2020.

[27] P. Porambage, J. Okwuibe, M. Liyanage, M. Ylianttila and T. Taleb, "Survey on Multi-Access Edge Computing for Internet of Things Realization," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 2961-2991, 2018.

[28] M. Chen, B. Liang, and M. Dong, "Joint offloading and resource allocation for computation and communication in mobile cloud with computing access point," *in Proc. of IEEE INFOCOM*, pp. 1863-1871, Apr. 2017.

[29] C. Kai, H. Zhou, Y. Yi, and W. Huang, "Collaborative Cloud-Edge-End Task Offloading in Mobile-Edge Computing Networks With Limited Communication Capability," *IEEE Trans. Cognit. Commun. Netw.*, vol. 7, no. 2, pp. 624-634, June 2021.

[30] T. X. Tran, and D. Pompili, "Joint Task Offloading and Resource Allocation for Multi-Server Mobile-Edge Computing Networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 856-868, Jan. 2019.

[31] L. Dong, S. Han, Y. Gao, and Z. Tan, "A Game-Theoretical Approach for Resource Allocation in Mobile Edge Computing," *in Proc. IEEE ICCT*, pp. 436-440, 2020.

[32] M. Huang, W. Liang, X. Shen, Y. Ma, and H. Kan, "Reliability-Aware Virtualized Network Function Services Provisioning in Mobile Edge Computing," *IEEE Trans. Mobile Comput.*, vol. 19, no. 11, pp. 2699-2713, 1 Nov. 2020.

[33] X. Yang, X. Yu, H. Huang, and H. Zhu, "Energy Efficiency Based Joint Computation Offloading and Resource Allocation in Multi-Access MEC Systems," *IEEE Access*, vol. 7, pp. 117054 - 117062, Aug. 2019.

[34] X. Chen, Y. Cai, L. Li, M. Zhao, B. Champagne, and L. Hanzo, "Energy-Efficient Resource Allocation for Latency-Sensitive Mobile Edge Computing," *IEEE Trans. Veh. Technol.*,vol. 69, no. 2, pp. 2246-2262, Feb. 2020.

[35] S. H. Park, S. Jeong, J. Na, O. Simeone, and S. Shamai, "Collaborative Cloud and Edge Mobile Computing in C-RAN Systems With Minimal End-to-End Latency," *IEEE Trans. Signal Inf. Process. Netw.,* vol. 7, pp. 259-274, 2021.

[36] W. Hou, H. Wen, N. Zhang, J. Wu, W. Lei, and R. Zhao, "Incentive-Driven Task Allocation for Collaborative Edge Computing in Industrial Internet of Things," *IEEE Internet Things J.,* vol. PP, no. 99, pp. 1-1, 2021.

[37] L. Li, T. Q. S. Quek, J. Ren, H. H. Yang, Z. Chen, and Y. Zhang, "An Incentive-Aware Job Offloading Control Framework for Multi-Access Edge Computing," *IEEE Trans. Mobile Comput.*, vol. 20, no. 1, pp. 63-75, 1 Jan. 2021.

[38] Y. Liu, C. Xu, Y. Zhan, Z. Liu, J. Guan, and H. Zhang, "Incentive mechanism for computation offloading using edge computing: A Stackelberg game approach," *Comput. Netw.*, vol. 129, pp. 399-409, Mar. 2017.

[39] R. Chattopadhyay, and C. Tham, "Fully and Partially Distributed Incentive Mechanism for a Mobile Edge Computing Network," *IEEE Trans. Mobile Comput.*, vol. PP, no. 99, pp. 1-1, 2020.

[40] Q. Wang, S. Guo, J. Liu, C. Pan, and L. Yang, "Profit Maximization Incentive Mechanism for Resource Providers in Mobile Edge Computing," *IEEE Trans. Services Comput.*, vol. PP, no. 99, pp. 1-1, 2020.

[41] G. Li, and J. Cai, "An Online Incentive Mechanism for Collaborative Task Offloading in Mobile Edge Computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 624-636, Jan. 2020.

[42] L. Ma, X. Wang, X. Wang, L. Wang, Y. Shi and M. Huang, "TCDA: Truthful Combinatorial Double Auctions for Mobile Edge Computing in Industrial Internet of Things," *IEEE Trans. Mobile Comput.,* vol. PP, no. 99, pp. 1-1, 2021.

[43] W. Sun, J. Liu, Y. Yue, and H. Zhang, "Double Auction-Based Resource Allocation for Mobile Edge Computing in Industrial Internet of Things, " *IEEE Trans. Ind. Inf.,* vol. 14, no. 10, pp. 4692 - 4701, 2018.

[44] G. Gao, M. Xiao, J. Wu, H. Huang, S. Wang, and G. Chen, "Auction based VM allocation for deadline-sensitive tasks in distributed edge cloud," *IEEE Trans. Services Comput.*, vol. PP, no. 99, pp. 1-1, 2020.

[45] W. Sun, J. Liu, Y. Yue, and P. Wang, "Joint Resource Allocation and Incentive Design for Blockchain-Based Mobile Edge Computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 9, pp. 6050-6064, Sept. 2020.

[46] T. H. Thi Le, N. H. Tran, Y. K. Tun, O. Tran Thi Kim, K. Kim and C. S. Hong, "Sharing Incentive Mechanism, Task Assignment and Resource Allocation for Task Offloading in Vehicular Mobile Edge Computing," *in Proc. of IEEE/IFIP NOMS*, pp. 1-8, 2020.

[47] Y. Yue, W. Sun, and J. Liu, "Multi-Task Cross-Server Double Auction for Resource Allocation in Mobile Edge Computing," *in Proc. of IEEE ICC*, pp. 1-6, May 2019.

[48] J. He, D. Zhang, Y. Zhou, and Y. Zhang, "A Truthful Online Mechanism for Collaborative Computation Offloading in Mobile Edge Computing," *IEEE Trans. Ind. Inf.*, vol. 16, no. 7, pp. 4832-4841, July 2020.

[49] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[50] W. Zhan, C. Luo, G. Min, C. Wang, Q. Zhu and H. Duan, "Mobility-Aware Multi-User Offloading Optimization for Mobile Edge Computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 3, pp. 3341-3356, March 2020.

[51] V. V. Vazirani. "Approximation algorithms," *Springer Sci. Business Media*, 2013.

[52] W. Liu, S. Chen, Y. Hou, and Z. Yang, "Trilevel Mixed Integer Optimization for Day-Ahead Spinning Reserve Management of Electric Vehicle Aggregator With Uncertainty," *IEEE Trans. Smart Grid*, vol. 13, no. 1, pp. 613-625, Jan. 2022.

[53] S. Burer, and A. N. Letchford, "Non-convex mixed-integer nonlinearprogramming: A survey," *Surv. Operations Res. Manage. Sci.,* vol. 17, no. 2, pp. 97–106, 2012.

[54] X. Chu, D. Lopez-Perez, Y. Yang, and F. Gunnarsson, *Heterogeneous Cellular Networks: Theory, Simulation and Deployment*. Cambridge, U.K.: Cambridge Univ. Press, 2013.

Huan Zhou (M'14) received his Ph. D. degree from the Department of Control Science and Engineering at Zhejiang University. He was a visiting scholar at the Temple University from Nov. 2012 to May, 2013, and a CSC supported postdoc fellow at the University of British Columbia from Nov. 2016 to Nov. 2017. Currently, he is a full professor in the College of Computer and Information Technology, China Three Gorges University. He was a Lead Guest Editor of Pervasive and Mobile Computing, TPC Chair of EAI BDTA 2020, and Local Arrangement Chair of I-SPAN 2018. He has published more than 60 research papers in some international journals and conferences, including IEEE JSAC, TPDS, TWC, and so on. His research interests include Mobile Social Networks, VANETs, Mobile Edge Computing, and Mobile Data Offloading. He receives the Best Paper Award of I-SPAN 2014 and I-SPAN 2018, and is currently serving as an associate editor for IEEE ACCESS and EURASIP Journal on Wireless Communications and Networking.

Tong Wu received his B.S. Degree from Jincheng College of Sichuan University. Currently, he is a graduate student at the College of Computer Information and Technology, China Three Gorges University. His research interests include mobile edge caching and opportunistic mobile networks.

Xin Chen received his B.S. Degree from China Three Gorges University. Currently, he is a graduate student at the College of Computer Information and Technology, China Three Gorges University. His main research interests are Mobile Data Offloading, Mobile Edge Computing and VANETs.

Shibo He (M'13-SM'19) received the Ph.D. degree in control science and engineering from Zhejiang University, Hangzhou, China, in 2012. He is currently a Professor with Zhejiang University. He was an Associate Research Scientist from March 2014 to May 2014, and a Postdoctoral Scholar from May 2012 to February 2014, with Arizona State University, Tempe, AZ, USA. From November 2010 to November 2011, he was a Visiting Scholar with the University of Waterloo, Waterloo, ON, Canada. His research interests include wireless sensor networks, crowdsensing, big data analysis, etc. Dr. He serves on the editorial board for the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, Springer Peer-to-Peer Networking and Application and KSII Transactions on Internet and Information Systems, and is a Guest Editor for Elsevier Computer Communications and Hindawi nternational Journal of Distributed Sensor Networks. He was a Symposium Co-Chair for the IEEE ICC 2017, a Finance and Registration chair for ACM MobiHoc 2015, a TPC Co-Chair for the IEEE ScalCom 2014, a TPC Vice Co-Chair for ANT 2013–2014, a Track Co-Chair for the Pervasive Algorithms, Protocols, and Networks of EUSPN 2013, a Web Co-Chair for the IEEE MASS 2013, and a Publicity Co-Chair of IEEE WiSARN 2010, and FCN 2014.

Deke Guo received the B.S. degree in industry engineering from the Beijing University of Aeronautics and Astronautics, Beijing, China, in 2001, and the Ph.D. degree in management science and engineering from the National University of Defense Technology, Changsha, China, in 2008. He is currently a Professor with the College of System Engineering, National University of Defense Technology. His research interests include distributed systems, software-defined networking, data center networking, wireless and mobile systems, and interconnection networks. He is a senior member of the IEEE and a member of the ACM, and has been awarded as the CCF-IEEE CS Yong Computer Scientist 2020.

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.