Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH Open Access

A resource competition-based truthful mechanism for IoV edge computing resource allocation with a lowest revenue limit

Jixian Zhang[1], Zhemin Wang[1], Athanasios V. Vasilakos[2] and Weidong Li[3*]

Abstract

Resource allocation in Internet of Vehicles (IoV) edge computing is currently a research hotspot. Existing studies focus on social welfare or revenue maximization. However, there is little research on lowest revenue guarantees, which is a problem of great concern to resource providers. This paper presents the innovative concept of the lowest revenue limit, which enables service providers to preset the revenue *B* and calculate whether the preset revenue can be achieved under the current supply and demand of resources through mechanism design. This approach is very friendly to service providers and can prevent low revenue and waste of resources. Specifically, we improved the ascending price auction mechanism so that it can be used for multi-resource allocation, the unit prices of different resources are calculated according to the intensity of competition among users, and the winning users and the payment are determined by eliminating users with low cost performance. Our mechanism is not sensitive to resource capacity, works well under deployment constraints in edge computing, and satisfies economic characteristics such as individual rationality and truthfulness. Compared with existing algorithms, our approach is shown to enable the service provider to obtain a higher revenue under a lower resource utilization.

Keywords Mechanism design, Internet of vehicles, Edge computing, Resource allocation, Lowest revenue limit

Introduction

The limited computing and storage capabilities of in-vehicle devices make it difficult to meet the large computing demands and low latency of Internet of vehicles (IoV) services. Therefore, introducing edge computing into the IoV is an effective approach to solve the above problems. Edge computing servers (ECSs) have powerful computing and storage capabilities. Typically, ECSs are equipped on the roadside unit, and the vehicles communicate with the ECSs through the vehicle-to-infrastructure (V2I) protocol. The V2I protocol can be used to implement many IoV services, such as the local information distribution [1], in-vehicle information enhancement [2], vehicle online recording and diagnosis [3], auto/assisted driving and emergency failure management [4], and vehicular sensing networks-aided smart cities [5]. In most IoV application scenarios, while the vehicle is driving, various sensors on the vehicle collect a large amount of data at every moment. Some data can be processed directly on the vehicle, such as ultrasonic radar detection data, but there are still some data that are not suitable or cannot be processed directly on the vehicle, such as video and lidar data used to train autonomous driving systems. These data require a large amount of computing resources and take a long time to train. Therefore, some vehicle tasks for IoV services require offloading to nearby edge computing servers.

*Correspondence:
Weidong Li
weidongmath@126.com
[1] School of Information Science and Engineering, Yunnan University, Kunming 650504, Yunnan, People's Republic of China
[2] Center for AI Research (CAIR), University of Agder (UiA), Grimstad, Norway
[3] School of Mathematics and Statistics, Yunnan University, Kunming 650504, Yunnan, People's Republic of China

Motivation

In general, IoV edge computing moves a part of the computing and storage resources from the core network to the edge and employs vehicles to provide them for use. Compared to cloud computing resource allocation, IoV edge computing has two important features. The first is deployment constraints. Because the coverage of roadside units is limited, vehicles can only connect a limited number of ECSs. The other feature is that the service requirements of IoV edge computing differ substantially in time and space. For example, during the morning and evening rush hours or traffic congestion on the main roads of a city, IoV services require more computing and storage resources, while at night or in remote areas with less traffic, less computing and storage resources are required. Therefore, we need a more appropriate way to allocate and use resources on ECSs. Figure 1 shows a typical IoV application scenario with deployment constraints.

An auction is a way to effectively allocate resources through market behavior [6]. In a simple auction environment, users submit their requirements and bids, and the service provider selects the winning users and determines the final price they should pay. But because users are selfish, they may submit untruthful information in the hope of gaining more benefits. Therefore, one of the main goals of mechanism design is truthfulness (also known as incentive compatibility or strategy proofness); that is, users can obtain maximum benefits by submitting truthful information, so that users have no incentive to lie. The design of the auction mechanism mainly includes the allocation algorithm and payment algorithm. However, the problem studied in this paper is much more complex. For example, it includes different types of resources on different edge servers and deployment constraints. This brings great challenges for designing corresponding allocation and payment algorithms.

At present, many studies have used the auction mechanism to solve the resource allocation problem in edge computing [7–9]. Generally speaking, such problems are transformed into social welfare maximization problems with characteristics of multi-server, multi-dimensional resources and deployment constraints. Furthermore, the economic characteristics of truthfulness and individual rationality must also be satisfied. Existing auction mechanism designs include optimal allocation with Vickrey–Clarke–Groves (VCG) payment [10], approximate allocation [11] and monotonic allocation with critical value payment [12]. In these mechanisms, the social welfare and final revenue are not necessarily positively correlated. Even in many social welfare maximization schemes, the final revenue is very low.

Although there are good research results on the mechanism design of revenue maximization, finding the optimal revenue auction mechanism is still a very difficult

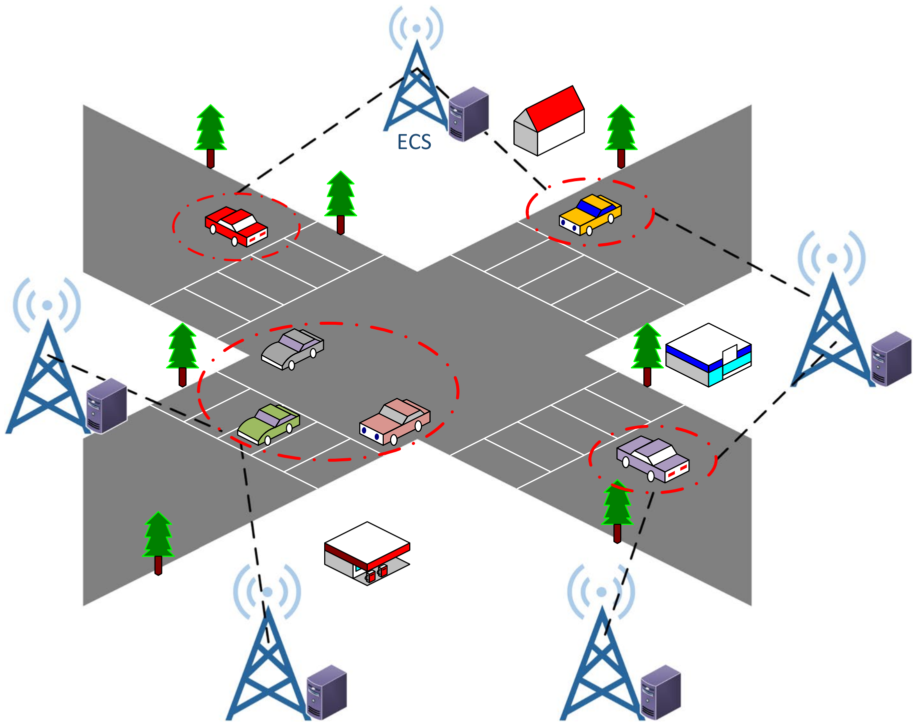

Fig. 1 Typical IOV application scenarios with deployment constraints

problem. In 1981, Myerson [13] solved the single-item optimal revenue auction, but the multi-item and multi-user optimal revenue auction has not been adequately solved in 40 years. Although Duan et al. [14] used machine learning to explore optimal revenue auctions for multiple items, the scale of items and users was small and the practicality was not strong. Therefore, this article considers this matter from another perspective, that is, whether we can ensure that the service provider can obtain at least the revenue B. In this way, the service provider also has a certain initiative in the auction and protects its own interests.

The ascending price auction [15] is a mechanism that can obtain a higher revenue. There are many mechanisms designed based on an ascending price, such as clinching auctions [16]. However, the existing ascending price auction can only be conducted for a single type of item, and it is not suitable for the allocation of multiple resources in edge servers with deployment constraints. Therefore, the main challenge of this paper is to effectively allocate resources in the IoV edge computing environment with unbalanced resource requirements and deployment constraints so that the resource provider can obtain more revenue.

Main contributions

Vehicle driving is a real-time and dynamic process. We can divide the vehicle's driving process into many independent time periods for processing, and each time period is associated with a static auction process (the main research goal of this article). The time period can be divided according to the actual situation, and it may be at the minute level, but at least it should be ensured that within this time period, the user's deployment constraints will not change and the user has enough time to transmit the content that needs to be processed. Notably, the tasks that need to be offloaded may have real-time requirements but cannot be critical; such tasks include congestion prediction services and autonomous driving training, which cannot affect the safe driving of the vehicle.

In the context of auctions within a single time period, we consider a very interesting question: when the users' requirements and the resource capacity of the service provider are known, can a mechanism be designed to ensure that the service provider obtains at least the revenue B? The most significant difference between this problem and those in existing research is that the resource providers have a lowest revenue limit B. This feature is of great significance for actual auctions because compared to the case of social welfare, resource providers are more concerned about how profitable the resources they invest in will ultimately be. For example, when the current road network and time period are determined, the provider has certain expectations for the revenue obtained by the provided edge computing services and whether the bids of users currently using these services are likely to meet this expected revenue. Most of the payment algorithms in the existing mechanism design rely on the critical value theory, and the theory mostly adopts the lowest winning price for achievement, which is one of the reasons that leads to low revenue. Based on the above considerations, we propose a new auction mechanism in the context of resource allocation in IoV edge computing services, which includes the following main features:

1. This article studies a resource competition-based auction mechanism with limited revenue under IoV edge computing services; that is, the resource provider proposes the lowest revenue limit B. The mechanism aims to maximize the total social welfare under this premise. To the best of our knowledge, this is the first article to study mechanism design with revenue limitation under IoV edge computing.
2. The mechanism satisfies the economic features of individual rationality and truthfulness. In addition, the complexity of the mechanism algorithm is polynomial.
3. In addition to meeting the expected revenue B of the resource provider, the algorithm can be used to explore the theoretical maximum revenue (peak B). This can be simply understood as determining the highest revenue when many users buy resources, which is of great significance to resource providers. We discuss this in our experiments.

Although we use IoV edge computing resource allocation as the background for discussion, the mechanism can easily be migrated to other areas for implementation, such as an energy or spectrum auctions.

The remainder of this paper is organized as follows: In Related works section, we discuss the existing studies that inspired our design. In IoV edge computing resource allocation with the lowest revenue limit problem and mechanism design preliminaries section, we describe the resource allocation problem with a lowest revenue limit in IoV edge computing and the mechanism design preliminaries. In IoV edge computing resource allocation mechanism with a lowest revenue limit (IoV-RAM-LRL) section, we propose a truthful ascending-price mechanism to solve the above problem and prove that this mechanism has the economic features of truthfulness and individual rationality. In Numerical results section, we evaluate the mechanisms through extensive experiments. Finally, in Conclusion section, we summarize our results and present possible directions for future research.

Related works

The auction mechanism is an effective resource allocation method. In the resource allocation of edge computing or IoV, multiple servers, multi-dimensional resource types, time-varying, deployment constraints and other characteristics are involved, making the design of the auction mechanism more delicate and attracting considerable attention. Generally, mechanism design includes optimal mechanism design, approximate mechanism design, heuristic mechanism design, double auction, and mechanism design with a budget.

Zaman et al. [17] combined mechanism design theory with cloud computing resource allocation and for the first time proposed a heuristic mechanism based on monotonic allocation with critical value payment to allocate virtual machine resources. Mashayekhy et al. [18] proposed two truthful mechanisms for single-dimensional resource task scheduling: dynamic programming-based allocation and the PTAS resource allocation implemented by maximal-in-range, payment algorithms using VCG. Liu et al. [19] designed an optimal and approximate mechanism for virtual machine allocation under heterogeneous clouds. The optimal mechanism obtains the optimal allocation solution by solving the integer programming (IP) problem and uses VCG to obtain the final payment. Moreover, in the approximate mechanism design, a resource allocation algorithm is designed by combining resource density and a fitness resources strategy, and the final payment price is calculated by dichotomy. Jiao et al. [20] proposed an auction-based market model for efficient computation of the resource allocation in public blockchain networks that used the sub-model optimization method to implement resource allocation and proposed the concept of ex-post estimation to obtain the final payment. Zhang et al. [3] considered a time-varying resource allocation problem in an online environment, applied a waiting period strategy and dominant-resource-based strategy to the resource allocation process, and designed a payment price algorithm based on the dichotomy. Li et al. [21] proposed an online truthful mechanism integrating computation and communication resource allocation and formulated a social-welfare-maximization problem that integrates collaborator selection, communication and computation resource allocation, transmission and computation time scheduling, and pricing policy design. Zhang et al. [22] proposed an online rewards-optimal auction (RoA) to optimize the long-term sum of rewards for processing offloaded tasks, meanwhile adapting to the highly dynamic energy harvesting (EH) process and computation task arrivals. Li et al. [23] formulated an incentive mechanism design problem by jointly optimizing task offloading decisions and allocation of both communications (i.e., power and bandwidth) and computation resources. Zhang et al. [24] addressed the problem of time-varying batch virtual machine (VM) allocation and pricing in the cloud and applied it in the context of online restart mode in [25], proposing a new class of auction for time-varying resource allocation. Bahreini et al. [26] formulated the edge resource allocation problem (ERAP) as a mixed-integer linear program (MILP), proved that the ERAP is NP-hard, and proposed a resource allocation mechanism that is guaranteed to be within a given distance from the optimal solution. He et al. [27] proposed a VCG-based optimal mechanism for computational offloading in a real-time time-varying edge computing environment, and in the approximate mechanism design, a heuristic algorithm based on primitive dual theory was designed to solve the resource allocation and prove the competition ratio. The above mechanism designs are mostly concerned with maximizing social welfare, and as analyzed in the first part of the paper, although social welfare is a very important economic indicator, it is not positively related to service provider revenue; it may even lead to the problem of insufficient revenue, which also needs to be addressed in this paper.

The auction mechanism with maximize revenue has also attracted the attention of many researchers. Deng et al. [28] proposed what economic settings would make the allocation and revenue maximization possible exactly or approximately, especially in cloud computing. Zhu et al. [29] applied deep learning techniques, designing a revenue-optimal auction mechanism for resource allocation in wireless virtualization. Li et al. [30] proposed an auction market in the IaaS cloud, where multiple users with heterogeneous bidding budgets and QoS requirements subscribe cloud resources according to their resource demands. The resource pricing and demand allocation scheme targeting revenue maximization also satisfies essential properties including budget feasibility, incentive compatibility and envy-freeness. Asterios et al. [31] presented a group of efficient allocation and pricing policies that can be used by vendors for their spot price mechanisms. They modeled the procedure of acquiring virtual machines as a truthful knapsack auction and deployed dynamic allocation and pricing rules that achieve near-optimal revenue and social welfare. Although all the above studies take maximum revenue as the research goal, the problem of maximizing the revenue of multiple items and multiple users is still a great challenge, and it is difficult to implement in large-scale environments. Additionally, the greatest difference is that our approach can guarantee a certain revenue *B*.

Some researchers have begun to turn to auction mechanisms with budget constraints. There are two types of auction mechanisms with budget restrictions. One is the user's budget limit, which means that the user cannot pay more than the budget in the auction; the other is mostly used in reverse auctions (such as in mobile crowdsourcing, where the operator pays the user), and the total amount paid by the operator cannot exceed the budget. In terms of the design of reverse auction mechanisms with a budget, Yaron et al. [32] studied a novel class of mechanism design problems in which the outcomes are constrained by the payments. The main result shows that a bounded approximation ratio is achievable for the important class of submodular functions. Nima et al. [33], on the basis of the former, designed a budget-feasible mechanism for large-scale crowdsourcing markets. Zhang et al. [34] began with the assumption of the user coverage probability model and transformed the opportunistic mobile crowdsensing value maximization problem into an ordered submodularity value function model with budget constraints. The notable difference between this article and the above studies is that the budget limit B considered in this article is the payment obtained by the resource provider, not the amount paid out or the user budget limit. In the auction mechanism design with the user's budget limit, the clinching auction is an excellent mechanism. Ausubel proposed the theory of the ascending-bid auction [15] (also called the clinching auction) in 2004. The clinching auction has been favored by many resource providers because of its very good profitability, and it has been used in many scenarios, such as radio spectrum auctions [35, 36] and video advertising auctions [37]. Dobzinski [16] proposed a multi-unit auction mechanism with budget limits based on Ausubel's research. The above mechanisms can only allocate divisible or indivisible homogenous items, which is quite different from the problem studied in this paper.

The mechanism design of this paper benefits in part from the above research. However, it can be seen that most research results take social welfare or revenue maximization as the goal; there are few studies on the guarantee of a minimum revenue for service providers, which is a very practical issue. The research of this article addresses the above shortcomings and also proposes a new idea for the direction of mechanism design.

IoV edge computing resource allocation with the lowest revenue limit problem and mechanism design preliminaries

Assume that the IoV edge computing service provider (referred to as provider) has a total of M ECSs, denoted by a set $\mathcal{M} = \{1, 2, ..., M\}$. Each ECS has R types of resources (such as CPU, memory, or storage) denoted by a set $\mathcal{R} = \{1, 2, ..., R\}$, and the resource capacity of each ECS is determined by the vector $\boldsymbol{c}_j = (c_{j1}, c_{j2}, ..., c_{jR})$, $j \in \mathcal{M}$. Moreover, the provider proposes the lowest revenue limit B, where B is the minimum revenue that the provider expects to obtain in this auction.

Assume there are a total of N vehicle users (referred to as users) to use ECSs resources, defined by the set $\mathcal{N} = \{1, 2, ..., N\}$. Each user $i \in \mathcal{N}$ submits her/his request defined as $\theta_i = (\boldsymbol{s}_i, \boldsymbol{\delta}_i, b_i)$, where $\boldsymbol{s}_i = (s_{i1}, s_{i2}, ..., s_{iR})$ represents the requirement for each type of resource, $\boldsymbol{\delta}_i = (\delta_{i1}, \delta_{i2}, ...\delta_{ij}..., \delta_{iM})$, $i \in \mathcal{N}$ is the deployment constraints vector of user i, δ_{ij} represents the connection status of user i and ECS j, $\delta_{ij} = 1$ represents that the two can be connected, otherwise they cannot be connected, and $\Delta = (\boldsymbol{\delta}_1, \boldsymbol{\delta}_2, ..., \boldsymbol{\delta}_N)$ is defined similarly. In reality, δ_{ij} will be affected by many factors, such as communication power, bandwidth, noise, and obstacles. In our model, δ_{ij} is simplified to a 0-1 constant, which is beneficial for focusing on our problem model. In [8, 38], more in-depth considerations about network connections are given. b_i is the user's bid for her/his resource requirements. The solution of the problem can be represented by a matrix

$$\boldsymbol{X} = \begin{bmatrix} x_{11} & ... & x_{1M} \\ ... & x_{ij} & ... \\ x_{N1} & ... & x_{NM} \end{bmatrix}$$

and a vector $\boldsymbol{p} = (p_1, p_2, ..., p_i, ..., p_N)$, where $x_{ij} = 1$ indicates that the resource requirement of user i is finally allocated by ECS j and p_i indicates the final payment paid by user i (if the user loses in allocation, the payment is 0). Note that any user can be satisfied by at most one ECS. Each group of $\boldsymbol{X}$ corresponds to an allocation solution; therefore, our goal is to maximize the social welfare $V(\boldsymbol{X})$ of the service provider while satisfying the lowest revenue limit and resource constraints.

$$Maximize \;\; V(\boldsymbol{X}) = \sum_{j \in \mathcal{M}} \sum_{i \in \mathcal{N}} b_i \delta_{ij} x_{ij} \tag{1}$$

$$Subject\ to: \sum_{i \in \mathcal{N}} s_{ir} \delta_{ij} x_{ij} \leq c_{jr}, \forall r \in \mathcal{R}, \forall j \in \mathcal{M} \tag{1a}$$

$$\sum_{j \in \mathcal{M}} \delta_{ij} x_{ij} \leq 1, \forall i \in \mathcal{N} \tag{1b}$$

$$\sum_{j \in \mathcal{M}} \delta_{ij} x_{ij} p_i \leq b_i, \forall i \in \mathcal{N} \tag{1c}$$

$$\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{M}}\delta_{ij}x_{ij}p_i \geq B, \forall i \in \mathcal{N}, \forall j \in \mathcal{M} \tag{1d}$$

$$x_{ij} \in \{0,1\}, \forall i \in \mathcal{N} \tag{1e}$$

(1a) indicates that the number of resources allocated on any ECS does not exceed the resource capacity of this ECS, and (1b) indicates that each user is allocated at most once. In our problem, it is reasonable to assume that the user can connect to multiple ECSs, but in the end, she/he can only deploy her/his requirements to one ECS. (1c) indicates that the user's payment is less than or equal to her/his bid, ensuring individual rationality; (1d) indicates that the sum of the users' payments must be greater than the provider's lowest revenue limit B; and (1e) indicates that this is an integer programming problem. The whole model not only includes the constraints of resource allocation (1a) (1b), but also the constraints of payment or revenue (1c) (1d), which is obviously different from the traditional social welfare maximization mechanism design, and is one of the innovations of this paper.

Equation (1) is an ideal problem model. However, in practice, users are selfish and may submit untruthful request for greater benefits; the value of mechanism design lies in addressing this issue. Specifically, to encourage users to participate in the auction process, the mechanism design must satisfy individual rationality; to prevent users from submitting untruthful requests, the mechanism design must satisfy truthfulness. Additionally, to quickly obtain the allocation and payment solution, the mechanism must satisfy computational efficiency.

We use $\theta_i = (\boldsymbol{s}_i, \boldsymbol{\delta}_i, b_i)$ to denote the true request of user i and $\theta'_i = (\boldsymbol{s}_i, \boldsymbol{\delta}_i, b'_i)$ to denote the declared request of user i. Additionally, we assume that the user may lie about her/his bid so that $b'_i > b_i$ or $b'_i < b_i$. We do not discuss the situation of users untruthfully reporting resource requirements $\boldsymbol{s}_i$ and deployment constraints $\boldsymbol{\delta}_i$ because in IoV edge computing services, where data must be offloaded to ECSs or cloud servers for execution, users cannot fake the resource requirements because the data to be processed are generated by sensors and the resources required to process the data are set in advance. Furthermore, the deployment constraints are obtained from the vehicle position, which is provided by GPS and is not easy to fake. We use $\boldsymbol{\theta'} = \{\theta'_1, ..., \theta'_N\}$ and $\boldsymbol{\theta'}_{-i} = \{\theta'_1, ..., \theta'_{i-1}, \theta'_{i+1}, ..., \theta'_N\}$ to denote the declared requests of users submitted to the system and $\boldsymbol{\theta} = \{\boldsymbol{\theta'}_{-i}, \theta_i\}$.

User utility is an important measure for users to determine the value obtained in an auction, and the user always wants to maximize her/his utility in an auction. User utility is typically expressed in the form of a function. In this article, we assume that user i has the following utility function:

$$u_i(\boldsymbol{\theta'}) = \begin{cases} b_i - p'_i, & \text{if user } i \text{ wins in the allocation,} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

p'_i is the final payment price of user i when she/he submits the request $\theta'_i = (\boldsymbol{s}_i, \boldsymbol{\delta}_i, b'_i)$. If the user loses the auction, the utility is 0. Based on the above description, an individually rational and truthful auction mechanism with a revenue limit can be defined.

Definition 1 ***Individual rationality.*** A mechanism that ensures individual rationality should satisfy the condition that when the user submits a truthful request $\theta_i = (\boldsymbol{s}_i, \boldsymbol{\delta}_i, b_i)$, her/his utility will be greater than or equal to zero; i.e., $u_i(\boldsymbol{\theta} \geq 0)$. In other words, as long as the user participates in the auction and reports her/his truthful request, she/he will not incur a loss.

Definition 2 ***Truthfulness.*** A truthful mechanism implies that for every user i, given a truthful declaration request θ_i and declaration requests $\boldsymbol{\theta'}_{-i}$ of the other users, we can obtain $u_i(\boldsymbol{\theta'}_{-i}, \theta_i) \geq u_i(\boldsymbol{\theta'}_{-i}, \theta'_i)$, which is equivalent to $u_i(\boldsymbol{\theta}) \geq u_i(\boldsymbol{\theta'}_i)$. Therefore, submitting a truthful request is the dominant strategy for each user.

Definition 3 ***Revenue limit.*** If an auction mechanism has a revenue limit, the sum of the payments of all users must exceed the lowest revenue limit B proposed by the resource provider; that is, $\sum_{i\in\mathcal{N}} x_i p_i \geq B$.

Definition 4 ***Computational efficiency.*** Because the resource allocation problem is NP-hard, in practice, we need algorithms with polynomial-time complexity to ensure the computational efficiency of the mechanism.

The optimal mechanism design obtains the optimal allocation solution by solving the integer programming problem in Eq. (1), except (1c) and (1d). There are many ways to solve this integer programming problem, such as dynamic programming. Then, VCG is used to determine the payment price. The VCG payment algorithm guarantees the truthfulness of the mechanism under the premise of obtaining the optimal allocation solution [10]. Assuming function OPT(.) is the optimal allocation algorithm, the VCG payment algorithm can be defined as follows:

$$p'_i = \sum_{j\in OPT(\boldsymbol{\theta'}_{-i})} b'_j - \sum_{j\in OPT(\boldsymbol{\theta'}), j\neq i} b'_j \tag{3}$$

$\sum_{j \in OPT(\boldsymbol{\theta'}_{-i})} b'_j$ is the maximum social welfare when user i does not participate, and $\sum_{j \in OPT(\boldsymbol{\theta'}), j \neq i} b'_j$ is the maximum social welfare of all users except user i. p'_i is user i's final payment price. However, constraint (1d) is not considered in the optimal mechanism design because VCG cannot predict the sum of the payments in advance. In the experiment, we use the optimal mechanism as a comparison to our approach. Table 1 lists the notation frequently used in this paper.

IoV edge computing resource allocation mechanism with a lowest revenue limit (IoV-RAM-LRL)

Considering that the lowest revenue limit entails considerable challenges in mechanism design, we must first analyze the reasons for the low revenue of the existing mechanism. Generally, an auction mechanism can be divided into two parts: allocation decisions and payment calculations (also referred to as allocation and payment). Certain features of these two components are the main reasons for low revenue. The first reason is the resource allocation theory used in the auction. Resource allocation can be seen as equivalent to the knapsack problem, in which it is necessary to allocate as many resources as possible and then calculate the payment pricing on this basis. However, in practice, putting all resources on the market may not yield higher revenue because it may lead to oversupply and low transaction prices (sometimes, controlling the amount of resources put on the market may bring higher revenue, such as in the diamond market). The second reason comes from the payment method, specifically from the truthfulness feature of the auction mechanism. In an auction, the seller and bidders (also known as the resource provider and users) are involved in a game, so users may submit untruthful bidding information to obtain greater profits. To ensure that users reveal their truthful request, the auction mechanism must satisfy the truthfulness feature, which means that users obtain the greatest utility when submitting truthful requests. Encouraging users to tell the truth is very important in the mechanism design. A necessary condition for truthfulness is that the user cannot reduce the final payment price by submitting an untruthful resource requirement or bid. Therefore, the user has no incentive to submit untruthful information. To achieve this goal, in the payment stage, existing auction mechanisms adopt the VCG mechanism [10], dichotomy [11] or the last loser bid [39] to determine the final payment price of the user. In general, these auction mechanisms always use the lowest winning price as the user's final payment price, which leads to low final revenue for the resource provider.

In summary, the *principles of allocating as many resources as possible* and using the *lowest winning price* are the main causes of low revenue. Therefore, we must address these two points without destroying the features of truthfulness and individual rationality.

We adopt the idea of an ascending-price auction to design the mechanism. The basic principle is to first calculate the bids of all users. If the total bids are less than B, then even if all users are selected, the lowest revenue limit B cannot be met, and the algorithm exits without a feasible solution.

Otherwise, according to the idea of ascending-price, the global price gp of the system is continuously increased, the users with lower bids are eliminated, and the allocation and payment price calculation is conducted among the remaining active users. Specifically, current active users are traversed in non-increasing order according to their resource requirements, and for each specific user, allocation is attempted in non-decreasing order of the number of active users on the ECSs that she/he can connect to. If user i can be successfully deployed on ECS j, the resources of ECS j are allocated to user i according to the current global price gp, user i is added to the winner set $\mathcal{W}$, the final payment of the user is calculated, and the total payment is updated at the same time. When all users have been traversed and the total payment is greater than B, the auction process is stopped; otherwise, the global price gp is increased and the algorithm enters the next round of execution. If the current

Table 1 Frequently used notation

Notation	Implication
$\mathcal{M} = \{1, 2, ..., M\}$	The set of edge servers
$\mathcal{N} = \{1, 2, ..., N\}$	The set of users
B	Lowest revenue limit
R	Number of resource types
$\boldsymbol{c}_j = (c_{j1}, c_{j2}, ..., c_{jR})$	Resource capacity of the j-th edge server
$\boldsymbol{\theta}_i = (\boldsymbol{s}_i, \boldsymbol{\delta}_i, b_i)$	User i's request
$\boldsymbol{s_i} = (s_{i1}, s_{i2}, ..., s_{iR})$	Resource requirements of user i
$\boldsymbol{\delta}_i = (\delta_{i1}, \delta_{i2}, ..., \delta_{ij}, ..., \delta_{iM})$	Deployment constraints between user i and ECSs
$\Delta = (\delta_1, \delta_2, ..., \delta_N)$	All users' deployment constraints
$\mathcal{A}$	The set of active users
$\mathcal{A}_j$	The set of active users on ECS j
x_{ij}	Decision variables of user i and ECS j
$\mathcal{W}$	The set of winners
λ_r	The price increase parameter of the r-th resource
ε	Fixed step of price increase
$\boldsymbol{gp} = (gp_1, gp_2, ..., gp_R)$	Global unit price of different types of resources
$\boldsymbol{p} = (p_1, p_2, ..., p_i, ..., p_N)$	Payment price of each user
pay	Total revenue

global price *gp* is continuously increased until all users are inactive and the condition that the total payment is greater than B still cannot be satisfied, the algorithm exits and there is no feasible solution. This design method can clearly satisfy all the constraints of formula (1).

In the allocation stage, in the process of increasing the global price *gp*, users with lower bids continue to be eliminated, leaving more valuable users, so the mechanism is not sensitive to resource capacity. Moreover, it is not necessary to allocate all resources, which is an improvement on the principles of allocating as many resources as possible. In the pricing stage, the price paid by each user is calculated according to the global price *gp*, and with the iteration of the algorithm, the global price *gp* increases, as does the final payment of the winning user, which is an improvement on the lowest winning price.

A core problem in the algorithm is how to determine the global price *gp*. In the existing ascending-price auction mechanism [16], since only one-dimensional resources are involved, the global price *gp* can represent the unit price of resources. The scarcer the resource is, the higher the global price *gp*. However, in multi-dimensional resource allocation, different resources have different capacities, resulting in differences in scarcity. How to use the global price *gp* to price resources in different dimensions is a challenge.

Let us consider a simple example, assuming that the CPU, memory, and storage resource capacities of one ECS are (10, 10, 1000), user 1's request is ((5, 5, 10), 200), and user 2's request is ((6, 6, 200), 230), that is, user 1 needs 5 units of CPU, 5 units of memory, 10 units of storage resources, and the bid is 200, similar for user 2. For resource providers, the goal is to determine which user is more cost-effective. When using the optimal mechanism, user 2 will be selected because of the higher bid. When using the resource density defined in [11], which is defined as $d_i = \frac{b_i}{\sqrt{\sum_{r \in \mathcal{R}} \frac{s_{ir}}{c_{jr}}}}$, the resource density of user 1 is $200/\sqrt{1/2 + 1/2 + 1/100} = 199$, and the resource density of user 2 is $230/\sqrt{3/5 + 3/5 + 1/5} = 194$. Thus, the algorithm in [11] is more inclined to select user 1. Different mechanisms select different users. Notably, in this example, the resources of the ECS cannot simultaneously satisfy the requests of the two users. However, the two users form a competitive relationship with respect to only the CPU and memory resources, while the storage resources are sufficient to meet the requirements of the two users, which means that the usage cost of storage resources should be very low, or even zero.

From the users' perspective, it is reasonable that the proportion of the cost of purchasing storage resources in their bid is very low. From the provider's perspective, the capacity of various resources differs. The cost of using abundant resources is not high, while the scarce resources are the main object of competition. This can also be seen in the pricing of virtual machines in the cloud computing market. For example, on the Alibaba Cloud Platform [40], we ordered a virtual machine with 1 core, 2 GB of memory, and 30 GB of storage. The monthly rent cost is 30. If the memory resources are increased from 2 GB to 4 GB, the monthly rent cost is 38, an increase of 8 yuan. However, if only the storage is increased from 30 GB to 60 GB, the monthly rent cost is 32, an increase of only 2. Therefore, for multi-dimensional resources, each type of resource unit price is different, which is reasonable. In the mechanism design of this paper, we use the vector $\boldsymbol{gp} = (gp_1, gp_2, ..., gp_R)$ to represent the unit price of each type of resource. In each iteration of the algorithm, the increase in the unit price of each type of resource is defined by $\lambda_r \varepsilon$ (ε is a small constant), and the design of the price increase parameter λ_r must satisfy the following characteristics: when a certain resource is scarcer, the increase in the unit price of that type of resource is greater; by contrast, when the resource is more abundant, the increase in the unit price of that resource is smaller; when the resource is sufficient to satisfy all users, the increase is fixed at a constant. Therefore, we define the price increase parameter λ_r of resource r as:

$$\lambda_r = e^{\max\{\frac{\sum_{i \in \mathcal{A}} s_{ir} - \sum_{j \in \mathcal{M}} c_{jr}}{\sum_{j \in \mathcal{M}} c_{jr}}, 0\}} \tag{4}$$

where $\sum_{i \in \mathcal{A}} s_{ir}$ represents the requirements of all active users for the rth resources currently, $\sum_{j \in \mathcal{M}} c_{jr}$ represents the amount of the rth resources in all ECSs. When $\sum_{i \in \mathcal{A}} s_{ir} > \sum_{j \in \mathcal{M}} c_{jr}, \lambda_r > 1$ and $\sum_{i \in \mathcal{A}} s_{ir} \leq \sum_{j \in \mathcal{M}} c_{jr}, \lambda_r = 1$. The introduction of the concept of an independent increase in the multi-dimensional resource unit price reflects the scarcity of different resources and is more in line with market rules. We use the following formulas to define how the unit price increases for different types of resources in $\boldsymbol{gp}$.

$$gp_r \leftarrow gp_r + \lambda_r \varepsilon = gp_r + (e^{\max\{\frac{\sum_{i \in \mathcal{A}} s_{ir} - \sum_{j \in \mathcal{M}} c_{jr}}{\sum_{j \in \mathcal{M}} c_{jr}}, 0\}})\varepsilon \tag{5}$$

If the resources provided by ECSs can satisfy all the resource requirements of active users currently but the lowest revenue limit B is still not reached, the unit price of all resources will increase by a fixed step (ε) and feasible solutions will continue to be explored.

Definition 5 ***Active users.*** It refers to the set of users who can still satisfy $b_i \geq \sum_{r \in \mathcal{R}} gp_r s_{ir}$ when the current price is $\boldsymbol{gp} = (gp_1, gp_2, ..., gp_R)$. We use the set $\mathcal{A} \leftarrow \{i \mid b_i \geq \sum_{r\in\mathcal{R}} gp_r s_{ir}, i \in \mathcal{N}\}$ to represent active users, where the set $\mathcal{A}_j \leftarrow \{i \mid \delta_{ij} = 1, i \in \mathcal{A}, j \in \mathcal{M}\}$ is used to denote active users on ECS *j*.

Definition 6 ***The norm of user resource requirements.*** The norm of user *i*'s resource requirements is defined as:

$$|\boldsymbol{s}_i| = \sqrt{(\frac{s_{i1}}{\sum\limits_{j\in\mathcal{M}} c_{j1}})^2 + (\frac{s_{i2}}{\sum\limits_{j\in\mathcal{M}} c_{j2}})^2 + ... + (\frac{s_{iR}}{\sum\limits_{j\in\mathcal{M}} c_{jR}})^2} \tag{6}$$

This value can be used to evaluate the size of the resource requirements of the users. In Algorithm 1, we use this value as the basis for sorting.

In our design, the service provider does not preset the initial price of each resource. We can assume that the initial price of $\boldsymbol{gp}$ is 0 or a very low cost. The algorithm determines the price of each type of resource based on the users' requirements and the resource capacity. Through multiple rounds of iterations, we can find a price $\boldsymbol{gp}$ that is suitable for the current scenario, as well as an allocation and payment solution. This is also the most valuable design of this article.

```
Input: $\boldsymbol{c} = (c_1, c_2, ..., c_M), B, \boldsymbol{\theta} = (\theta_1, \theta_2, ..., \theta_N), \Delta, \mathcal{N}$
Output: $\mathcal{W}, \mathcal{P}, pay$
1: Initialize $\boldsymbol{gp} \leftarrow 0^R, \varepsilon \leftarrow 10^{-6}, \mathcal{A} \leftarrow \mathcal{N}, \mathcal{W} \leftarrow \phi, pay \leftarrow 0, \mathcal{P} \leftarrow \phi, \boldsymbol{c'} \leftarrow \boldsymbol{c}$
2: if ($\sum_{i\in\mathcal{N}} b_i < B$) then
3:     return $notfeasible$
4: end if
5: do
6:     $\lambda_r = e^{\max\{\frac{\sum_{i\in\mathcal{A}} s_{ir} - \sum_{j\in\mathcal{M}} c_{jr}}{\sum_{j\in\mathcal{M}} c_{jr}}, 0\}}$;
7:     $gp_r \leftarrow gp_r + \lambda_r \varepsilon, \forall r \in \mathcal{R}$;
8:     for all each of $i \in \mathcal{A}$ do
9:         if $b_i < \sum_{r\in\mathcal{R}} gp_r s_{ir}$ then
10:            $\mathcal{A} \leftarrow \mathcal{A} \backslash \{i\}$
11:        end if
12:    end for
13:    for each of $j \in \mathcal{M}$ do
14:        $\mathcal{A}_j \leftarrow \{i \mid i \in \mathcal{A}, \delta_{ij} = 1\}$
15:    end for
16:    $\boldsymbol{c'} \leftarrow \boldsymbol{c}$
17:    for each of $i \in \mathcal{A}$ according to the non-increasing order of $|\boldsymbol{s}_i|$ do
18:        for each of $\{j \mid j \in \mathcal{M}, \delta_{ij} = 1\}$, according to the non-decreasing order of current $|\mathcal{A}_j|$ do
19:            if $c'_{jr} - s_{ir} > 0, \forall r \in \mathcal{R}$ then
20:                $\mathcal{W} \leftarrow \mathcal{W} \cup \{x_{ij}\}$, $p_i \leftarrow \sum_{r\in\mathcal{R}} gp_r s_{ir}$, $pay \leftarrow pay + p_i$,
21:                $\mathcal{P} \leftarrow \mathcal{P} \cup \{p_i\}$, $\boldsymbol{c'_j} \leftarrow \boldsymbol{c'_j} - \boldsymbol{s}_i$
22:                for each of $\{ j \mid j \in \mathcal{M}, \delta_{ij} = 1 \}$ do
23:                    $\mathcal{A}_j \leftarrow \mathcal{A}_j \backslash \{i\}$
24:                end for
25:                break;
26:            end if
27:        end for
28:    end for
29:    if $pay \geq B$ then
30:        $feasible \leftarrow 1$
31:    else
32:        $\mathcal{W} \leftarrow \phi, pay \leftarrow 0, \mathcal{P} \leftarrow \phi$
33:    end if
34: while feasible = 0 && ($\mathcal{A} \neq \phi$)
35: if $feasible = 0$ then
36:     return $notfeasible$
37: else
38:     return $\mathcal{W}, \mathcal{P}, pay$
39: end if
```

Algorithm 1 IoV-RAM-LRL

Analysis of IoV-RAM-LRL

At the beginning of the algorithm, we initialize key variables, the most critical of which is to set the unit price of various resources to 0 ($\boldsymbol{gp} \leftarrow 0^R$), and ensures that all users are active($\mathcal{A} \leftarrow \mathcal{N}$). Lines 2-4 of the algorithm calculate whether the bids of all users meet the lowest revenue limit B. If not, the algorithm exits and there is no feasible solution. Lines 6-7 calculate the current unit price of each type of resource based on the active users and resource capacity. Lines 8-15 obtain all current active users and add them to the active users set $\mathcal{A}$ and calculate the active user set $\mathcal{A}_j$ for each ECS j according to the deployment constraints. Lines 17-28 allocate the resources in non-increasing order according to the norm of the current active users' resource requirements. Specifically, for each active user $i \in \mathcal{A}$, it is allocated according to the non-descending order of the $|\mathcal{A}_j|$ of ECSs in set $\{j|j \in \mathcal{M}, \delta_{ij} = 1\}$ to which she/he can be deployed. That is to say, among all the ECSs that user i can be deployed to, she/he is preferentially deployed to the ECS with a smaller number of active users, and this strategy can allow more users to be deployed successfully. Moreover, because user i has been deployed to ECS j, user i must be removed from the active user set of other ECSs (line 22-line 24). When all active users are traversed, calculate whether the payment of the winning users exceeds the lowest revenue limit B. If it does, the algorithm exits and outputs the winning user set $\mathcal{W}$, payment solution $\mathcal{P}$, and the total payment price. Otherwise, the algorithm returns to line 5 to continue increasing the unit price of each type of resource according to formula (5) and then enters the next round of execution. If the unit price increases to the point where there are no active users in the system, then the lowest revenue limit B cannot be reached($\mathcal{A} = \phi$), the algorithm exits, and there is no feasible solution. It is worth noting that it appears that our algorithm (line 17) does not consider the use of bids as a basis for sorting, but in fact, bids still affect the allocation stage. In each round of calculation, users whose bids cannot meet the current unit price will be removed, which means that the remaining active users are more cost-effective. However, we use the norm of user resource requirements to sort among active users, which means that for active users, it is impossible to obtain greater utility by manipulating bids. This ingenious design achieves two goals at the same time: retain cost-effective users while making the final payment of the winning user independent of their bid. The algorithm will run through multiple rounds to verify whether there is a feasible solution (line 29). Because IoV-RAM-LRL is an ascending-price auction, the algorithm will exit when a feasible solution is obtained for the first time. Notably, if algorithm 1 does not output a feasible solution, it means that the current allocation fails and users cannot obtain the corresponding services. This may be caused by two situations. One is that the service provider sets the expected revenue B too high. The other is that the users' bids are very low and cannot reach the preset B value. No matter which situation holds, this is a possibility of market behavior that cannot be avoided in all types of auctions. Therefore, in the next round of auctions, both the service provider and users should fully reconsider this issue. We believe that after a period of running-in, both parties can find a balance.

A simple example can be used to illustrate this process (Fig. 2).

In this example, we consider part of a road, and near the road, there are 2 ECSs (ECS 1 and ECS 2); each of them has two types of resources (CPU and memory), and the resource capacities are (5, 50). There are 3 vehicle users (referred to as users); user 1's resource requirement is (4, 1), and the bid is 5. User 1 can only be deployed in ECS1, and the explanations for other user requests are similar. When using the optimal allocation with VCG payment, the optimal allocation is user 2 deployed on ECS1 and user 3 deployed on ECS2, and the sum of social welfare is

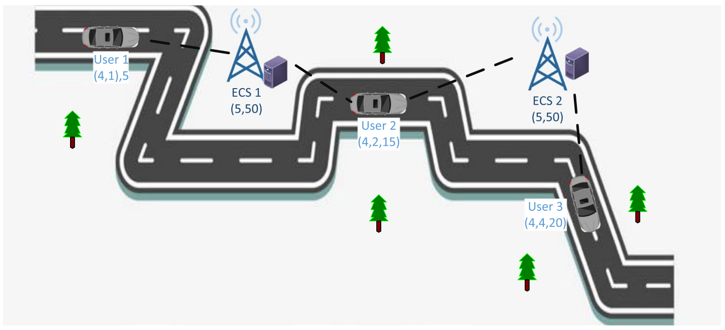

Fig. 2 ECS resource capacity and user requests in the example

Table 2 Results of running the example using our algorithm

Round	$\lambda\varepsilon$	Unit price	Users state Active:1 Inactive:0	Allocation solution	Payment	result
1	(0.61,0.5)	(0.61,0.5)	(1,1,1)	x_{21}, x_{32}	3.44,4.44	$< B$
2	(0.61,0.5)	(1.22,1.0)	(0,1,1)	x_{21}, x_{32}	6.88,8.88	$< B$
3	(0.5,0.5)	(1.72,1.5)	(0,1,1)	x_{21}, x_{32}	9.89,12.89	$< B$
4	(0.5,0.5)	(2.22,2.0)	(0,1,1)	x_{21}, x_{32}	12.89,16.89	$29.78 > B$

35. When using VCG to calculate the payment, the payment of user 2 is $(5+20)-(35-15)=5$, user 3 pays $(5+15)-(35-20)=5$, and the total revenue is 10. When using the algorithm G-PMRM in the literature [11] and executing according to the order of ECS numbers, user 2 and user 3 still win. G-PMRM uses a dichotomy to calculate the payment. First, users 1 and 2 compete on ECS1, and in the end, user 2 wins and pays 5.06. Then, users 2 and 3 compete on ECS2. Because ECS1 is traversed first, user 2 has been allocated successfully on ECS1. Therefore, on the premise that user 2 wins, user 3 can be allocated successfully on ECS2 no matter how much he/she bids; therefore, user 3 can be successfully allocated on ECS2, and the final payment is 0. The total revenue of the auction is 5.06. Although the optimal mechanism or G-PMRM can obtain higher social welfare, the final revenue is very low. When using our approach, where B is set to 25, $\varepsilon = 0.5$. Table 2 shows the results.

In the first round, because the CPU competition is more intense, the CPU unit price increase is 0.61, and the memory unit price increase is 0.5. The final allocation solution is to deploy user 2 to ECS1 and to deploy user 3 to ECS2. The payment price is 3.44 and 4.44, respectively, and the total revenue is $3.44+4.44=7.88$, which is less than B. Therefore, the algorithm enters the second round. After the second round of unit price increase, user 1 becomes inactive because of the low bid; the allocation solution is the same as the previous round. The total revenue of the second round is $6.88+8.88=15.76$, which is still less than B. In the third round, because user 1 is no longer active, the competition for CPU resources is reduced, and the number of resources provided by the system can already satisfy the existing active users (user 2 and user 3). Therefore, the unit price of the two resources is increased by the same step size (ε) of 0.5. The allocation solution is the same as that in round 2, and the total revenue of the third round is $9.89+12.89=22.78$, which is still less than B. In the fourth round, the unit prices of the two resources continue to increase, the allocation solution is the same as that in the previous round, and the total revenue is $12.89+16.89=29.78$, which is greater than B; therefore, the algorithm ends. This example shows that the unit prices of the two resources have different increasing rates due to different capacities. The competition for CPU resources is more intense, and the final unit price is higher, which is consistent with our analysis. Notably, the unit price difference between the two resources in the example is not large. The reason for this result is to make the instance converge as soon as possible. We use a larger ε, and the second reason is because the number of users is small. In the experiments section, we will use extended experiments to illustrate significant differences between resource unit prices. In terms of truthfulness, if user 2 changes her/his bid to 13, user 2 can still win, and the user utility is unchanged; thus, the mechanism is truthful, which we will prove later.

Properties of IoV-RAM-LRL

Notably, due to deployment constraints, some users can be deployed to a small number of ECSs. When resources on these ECSs are exhausted, although these users are still active, they may face situations where they cannot be deployed. This is a common phenomenon in edge computing services. For example, many vehicles suddenly appear around a certain ECS to submit their request, but the ECS cannot meet all the resource requirements. Active users who are not allocated resources at this time are called mechanism victims.

Definition 7 ***Mechanism Victim:*** A user that satisfies $b_i \geq \sum_{r\in\mathcal{R}} gp_r s_{ir}$ in Algorithm 1 but is not allocated resources due to her/his deployment constraints.

The emergence of mechanism victims is not caused by algorithms, but by the deployment constraints of the IoV. Deployment constraints enable users to connect to different ECSs. Because our resource unit price is global, for some ECSs with a large number of connected users, their resources have been allocated completely, but there are still active users. The way to improve this situation is to add ECSs in the user-dense area to resolve the problem of insufficient resources. In traditional cloud computing multi-server allocation, there are no deployment constraints, and each user may be allocated to any server, which is also an important difference between edge computing and cloud computing multi-server resource allocation.

Lemma 1 Mechanism Victim's bid does not affect the current round's allocation result.

Mechanism victims refer to active users who satisfy $b_i \geq \sum_{r\in\mathcal{R}} gp_r s_{ir}$ in the auction but are not allocated resources. According to Algorithm 1, the algorithm allocates resources to active users under the premise of the current unit price $\boldsymbol{gp}$. If the active user i is still not successfully allocated after traversing all the ECSs, then at this time on any ECS, the allocation cannot be successful because the allocation rule is to allocate according to the non-increasing order of the norm of active users' resources requirements, which is not related to the users' bids. Therefore, the mechanism victim still cannot be allocated successfully after changing the bid, which will not affect the result of this round of the allocation solution.

Theorem 1 Individual rationality of IoV-RAM-LRL.

Proof
Winning users in Algorithm 1 must satisfy $p_i = \sum_{r\in\mathcal{R}} gp_r s_{ir} \leq b_i$*; therefore,* $u_i(\boldsymbol{\theta}) = b_i - p_i \geq 0$.

Theorem 2 IoV-RAM-LRL is truthful.

We assume that it is impossible for users to declare untruthful resource requirements and deployment constraints, the reasons have been analyzed in IoV edge computing resource allocation with the lowest revenue limit problem and mechanism design preliminaries section, but users can change their bids.

1. Suppose a user submits an untruthful bid $b'_i < b_i$
   (a) User i wins when the bid is b_i and still wins when the bid is b'_i. Suppose that user i submits a truthful bid b_i and wins when the final unit price is $\boldsymbol{gp} = \{gp_1, gp_2, ..., gp_R\}$. The user still wins when submitting the bid b'_i and when the final unit price is $\boldsymbol{gp'} = \{gp'_1, gp'_2, ..., gp'_R\}$. If $\sum_{r\in\mathcal{R}} gp'_r s_{ir} < \sum_{r\in\mathcal{R}} gp_r s_{ir}$, then when user i's bid is b_i, she/he can also win when the unit price is $\boldsymbol{gp'}$, which is inconsistent with the facts. It can be seen that $\boldsymbol{gp'} = \boldsymbol{gp}$, and the utility of user i is $u(\boldsymbol{\theta'}) = b_i$ - $\sum_{r\in\mathcal{R}} gp'_r s_{ir} = b_i$ - $\sum_{r\in\mathcal{R}} gp_r s_{ir} = u(\boldsymbol{\theta})$, which is unchanged.
   (b) User i wins when the bid is b_i and loses when the bid is b'_i. Because the user decreases the bid, allocation fails. According to Algorithm 1, the user utility is 0, and the utility of user i is $u(\boldsymbol{\theta'}) = 0 \leq b_i - \sum_{r\in\mathcal{R}} gp_r s_{ir} = u(\boldsymbol{\theta})$; thus, the user utility may decrease.
   (c) User i loses when the bid is b_i and loses when the bid is b'_i. According to Algorithm 1, the user loses allocation under the bid b_i; after decreasing the bid, the user still loses. In this case, the user utility is 0, so the utility of user i is $u(\boldsymbol{\theta'}) = 0 = u(\boldsymbol{\theta})$, which is the same as before.
   (d) User i loses when the bid is b_i and wins when the bid is b'_i. According to Algorithm 1, the user loses under bid b_i for two reasons. First, user i is not an active user under the current global unit price $\boldsymbol{gp} = \{gp_1, gp_2, ..., gp_R\}$; that is, $b_i < \sum_{r\in\mathcal{R}} gp_r s_{ir}$. After decreasing the bid, we have $b'_i < b_i < \sum_{r\in\mathcal{R}} gp_r s_{ir}$. Assuming that there is a unit price $\boldsymbol{gp'} = \{gp'_1, gp'_2, ..., gp'_R\}$ that can make the user submit the bid b'_i to be successfully allocated, then $\sum_{r\in\mathcal{R}} gp'_r s_{ir} \leq b'_i < b_i < \sum_{r\in\mathcal{R}} gp_r s_{ir}$ must be satisfied. According to Algorithm 1, when the user submits the bid b_i, the user can also be successfully allocated when the global unit price is $\boldsymbol{gp'} = \{gp'_1, gp'_2, ..., gp'_R\}$, which is inconsistent with the facts. Therefore, this situation does not exist. The second reason is that user i is a mechanism victim in the auction. According to Lemma 1, if user i is a mechanism victim, her/his bid will not affect the allocation result. She/he still cannot be successfully allocated, so this situation cannot exist.

2. Suppose a user submits an untruthful bid $b'_i > b_i$.
   (a) User i wins when the bid is b_i and still wins when the bid is b'_i. The proof is the same as that of 1.(a)
   (b) User i wins when the bid is b_i and loses when the bid is b'_i. According to Algorithm 1, the user wins allocation under bid b_i because $b_i \geq \sum_{r\in\mathcal{R}} gp_r s_{ir}$, and the user is not a mechanism victim. After increasing the bid, $b'_i > b_i \geq \sum_{r\in\mathcal{R}} gp_r s_{ir}$, and the allocation will still be successful; thus, this situation does not exist.

(c) User i loses when the bid is b_i and loses when the bid is b'_i. The proof is the same as that of 1.(c).

(d) User i loses when the bid is b_i and wins when the bid is b'_i. According to Algorithm 1, the user loses allocation under bid b_i; therefore, the utility is 0. There are two reasons for this result. First, user i is not an active user at the final global unit price $\boldsymbol{gp} = \{gp_1, gp_2, ..., gp_R\}$; that is, $b_i < \sum_{r\in\mathcal{R}} gp_r s_{ir}$, and the allocation is successful after increasing the bid to b'_i. Suppose the final unit price under bid b'_i is $\boldsymbol{gp}' = \{gp'_1, gp'_2, ..., gp'_R\}$. If the user is successfully allocated when the bid is b'_i and loses when the bid is b_i, then $b'_i > \sum_{r\in\mathcal{R}} gp'_r s_{ir} > b_i$ must be satisfied. At this time, the utility of user i is $u_i(\boldsymbol{\theta}') = b_i - \sum_{r\in\mathcal{R}} gp'_r s_{ir} s_i \le 0 = u_i(\boldsymbol{\theta})$, and the utility may decrease. Second, user i is a mechanism victim in the auction. According to Lemma 1, if user i is a mechanism victim, her/his bid will not affect the allocation result. She/he still cannot be successfully allocated, so this situation cannot exist; therefore, $u(\boldsymbol{\theta}') = 0 = u(\boldsymbol{\theta})$.

In summary, users submitting an untruthful bid $b'_i \ne b_i$ cannot improve their utility, so the mechanism is truthful.

Theorem 3 The time complexity of IoV-RAM-LRL is polynomial.

The time complexity of Algorithm 1 *is* $O(\frac{B}{\varepsilon}NM(R+M^2))$. *In Algorithm* 1, *lines* 5-34 *are executed at most* $\frac{B}{\varepsilon}$ *times, and each execution has an* $O(NMR)$ *loop. However, after successfully allocating resources to a user, the ECSs must be sorted again because the number of active users has changed. Therefore, the complexity is* $O(\frac{B}{\varepsilon}NM(R+M^2))$.

Theorem 4 The social welfare of IoV-RAM-LRL has an approximate ratio of $\frac{\max(b_i)N}{B}$. *Assuming* $\mathcal{W}$ *is the solution of Algorithm* 1 *and that* $\mathcal{W}^*$ *is the optimal social welfare solution of the problem, it can be seen that* $B \le \sum_{i\in\mathcal{W}} b_i \le \sum_{i\in\mathcal{W}^*} b_i$; *we can obtain* $\frac{\sum_{i\in\mathcal{W}^*} b_i}{\sum_{i\in\mathcal{W}} b_i} \le \frac{\sum_{i\in\mathcal{W}^*} b_i}{B} \le \frac{b_{max}N}{B}$, *where* $b_{max} = \max(b_i), \forall i \in \mathcal{N}$ *and* N *is the number of users. Therefore, the social welfare of IoV-RAM-LRL has an approximate ratio of* $\frac{\max(b_i)N}{B}$.

In theory, we cannot obtain a better approximation than this result. Consider an example in which there are 100 users, the resource requirement of all users is 1, the bid is also 1 (so $b_{max} = 1$), the lowest revenue limit B is 100, and the system resource capacity is 100. In this case, the social welfare of the optimal solution and the social welfare of IoV-RAM-LRL are both 100, and the approximate ratio is $\frac{b_{max}N}{B} = \frac{1\cdot 100}{100} = 1$.

Notably, in existing studies, the social welfare of the allocation solution can easily reach an approximate ratio of 2 or even $1+\varepsilon$ compared with the optimal solution. However, these algorithms are designed without considering the constraints of the lowest revenue limit. Although the algorithm proposed in this paper has only a parameter-related approximation ratio, in most cases, it works very well.

Numerical results

Experimental settings

We adopted the data set of the 2021 Huawei Cloud Software Elite Challenge [41]. The dataset includes various types of server (including CPU, memory, hardware cost, energy consumption) of the Huawei Cloud Platform, as well as user resource request data (CPU, memory, dynamic request sequence, etc.). On this basis, we have made some modifications to facilitate the experiment of the paper. The rules are as follows.

1. From the dataset, we select 100 samples of resource requirements with CPU requirements that do not exceed 25 (an average of 10.3) and memory requirements that do not exceed 35 (an average of 17.85) and add randomly generated storage resource requirements, which range from 40 to 101 (an average of 68.46) as the resource requirements of the users in OPT-VCG, G-PMRM and IoV-RAM-LRL.
2. In terms of bidding, we first calculate the cost of each user's resource requirements according to the resource pricing published in Huawei Cloud [42] and Tencent Cloud [43] and multiply this cost by a random number between 0.2 and 5. (Half of the users satisfy a uniform distribution within 0.2-1, and the other half satisfy a uniform distribution within 1-5), which means that the user's bid is between 0.2 times and 5 times the cost.
3. We determine the resource capacity of the server according to the total requirements of all users (100). We define the server resource capacity parameter C as 1.0, which can just meet the resource requirements of all users. Specifically, in IoV-RAM-LRL, the CPU, memory, and storage requirements are 960, 1680, and 10240, respectively. Similarly, a server resource capacity parameter C of 0.5 yields 480, 840 and 10240, respectively. This is because in reality, the storage resources are relatively sufficient, so the quantity of storage resources does not change.

4. We normalize the resource requirements of the users in IoV-RAM-LRL, and set $\varepsilon = 0.1$ as the step size in increasing the unit price ***gp***.
5. All algorithms use the same data, and to eliminate the influence of the randomness of the data, we test each indicator in the experiment 50 times; the average value is shown in the figure.
6. We use the Python language to implement IoV-RAM-LRL. Specifically, we improve IoV-RAM-LRL to implement IoV-RAM-LRL (Peak B). The principle is to continuously improve B until the algorithm has no solution. At this time, B is Peak B, which is also the highest revenue that can be obtained in the current auction. At the same time, DOCplex is called through Python language to implement OPT-VCG. Another comparison algorithm, G-PMRM [11], is also implemented in Python, but G-PMRM cannot meet the deployment constraints, and we improve it by assigning server numbers.
7. The hardware configuration of the experimental platform is as follows: the processor is an Intel(R) Core(TM) i5-7300HQ CPU with 16 GB memory and a 256 GB SSD.
8. We uploaded the data set and code to https://github.com/WangZHeM/IoV-RAM-LRL/tree/main.

Experimental results

Impact of the lowest revenue limit B

This experiment finds the maximum theoretical revenue of IoV-RAM-LRL by changing the lowest revenue limit B and compares the results with those of the two classical auction mechanisms, OPT-VCG and G-PMRM. In this experiment, the total number of users N is fixed at 100, the number of servers M is fixed at 10, the number of resources R is fixed at 3, the server resource capacity parameter C is fixed at 0.75, and the deployment constraint (average number of users connecting to the server) $\delta = 2$.

Figure 3a shows the difference in revenue of different algorithms, which is the most important indicator in this paper. The revenue is the sum of the payments for all winning users. When B is 30000, the total revenue of IoV-RAM-LRL is lower than that of the other two algorithms because the mechanism is designed in terms of ascending price, and when the predetermined revenue is reached, the algorithm ends. Therefore, when B is low, the corresponding revenue is also low. Moreover, the allocation and payment solution of OPT-VCG and G-PMRM only need to be calculated once, so the allocation and payment solution are fixed. As can be seen from Fig. 3a, OPT-VCG can reach a revenue of about 30,000, while G-PMRM can reach a revenue of about 35,000. If the revenue limit B

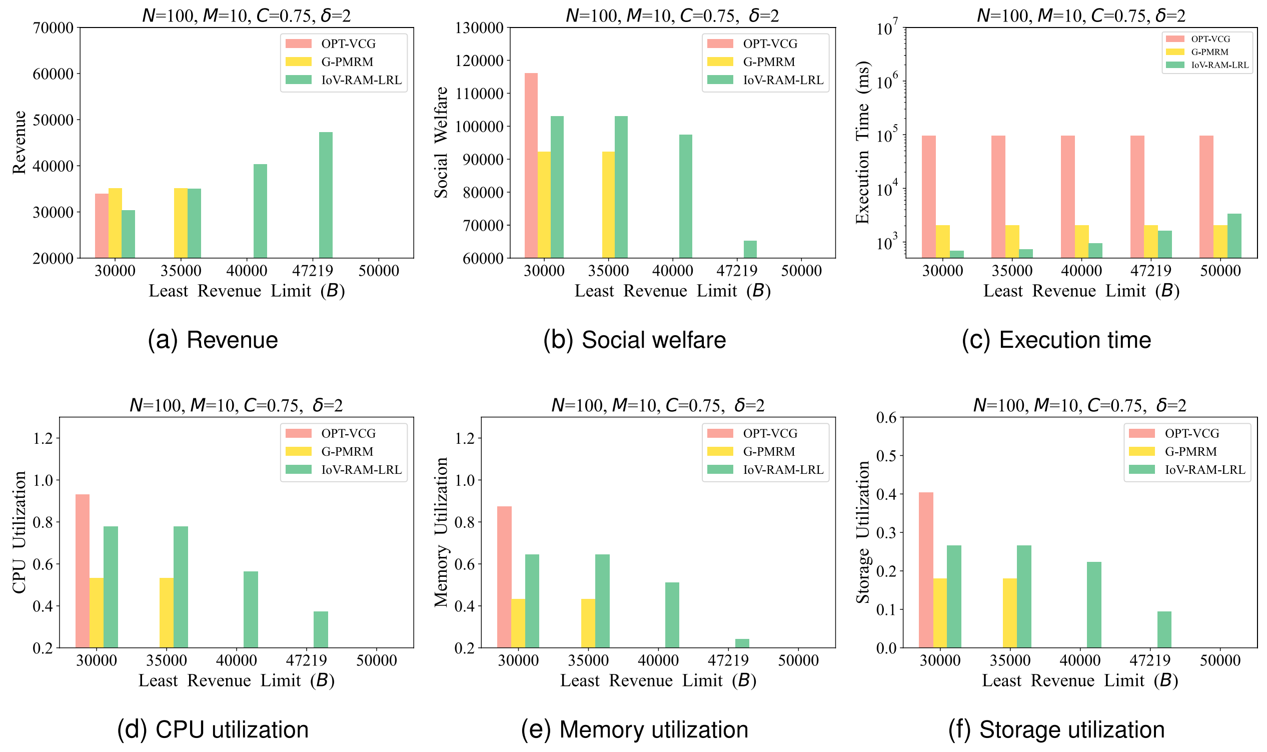

Fig. 3 The impact of the lowest revenue limit *B* (IoV-RAM-LRL)

continues to increase, the revenue generated by the OPT-VCG and G-PMRM mechanisms cannot meet the preset revenue limit, so there is no feasible solution. Therefore, it is not shown in the figure. However, IoV-RAM-LRL continuously increases the unit price of various resources and chooses to eliminate some users with lower bids to calculate the new revenue. When the revenue reduction due to eliminating users is less than the increase in revenue due to the price increase, the revenue will increase until it reaches B or all users have been eliminated and there is no solution. Therefore, the revenue of IoV-RAM-LRL is obtained through continuous improvement and eventually reduces to 0. In this experiment, Peak B is 47219.

Figure 3b shows the difference in social welfare of different algorithms, which is the sum of bids of all winning users. As B increases, the social welfare of IoV-RAM-LRL gradually decreases from constant to 0. The reason is that when B is small, the unit price is low, and the number of users selected is the largest, so the social welfare is the greatest. When B begins to increase, in order to reach the revenue limit B under the condition of deployment constraints, the algorithm eliminates users with lower bids, so the social welfare decreases. However, OPT-VCG satisfies the *principles of allocating as many resources as possible* and will allocate as many resources as possible to users, so more users will be selected, yielding higher social welfare. However, IoV-RAM-LRL achieves higher revenue by improving the *principles of allocating as many resources as possible* and strikes a balance between social welfare and revenue. It is worth noting that the social welfare of IoV-RAM-LRL is higher than that of G-PMRM. The main reason is that G-PMRM does not work well under deployment constraints, which is an another advantage of IoV-RAM-LRL.

From Fig. 3a and b, it can be seen that the IoV-RAM-LRL decreases the optimal social welfare and resource utilization very much when the revenue limit is large. It can be considered that ECSs reserve the resources to pursue a higher revenue. This is an advantage of the mechanism in this paper. The existing mechanism designs tend to clear resources (allocating as many resources as possible), so the revenue is very low when the supply of resources exceeds the demand. However, the IoV-RAM-LRL tends to select more valuable users and let them buy resources, and does not need to allocate resources as much as possible. Specifically, under the premise of the current users requests and ECSs resource capacity, the IoV-RAM-LRL can find the set of users with the highest payment that can be achieved. When the supply of resources exceeds the demand, the algorithm is still applicable.

Figure 3c shows the execution time of the three algorithms. The execution time of IoV-RAM-LRL increases with increasing B because when B is higher, IoV-RAM-LRL must execute more loops (the number of loops is also related to the ε in unit price). Furthermore, OPT-VCG has the longest execution time because OPT-VCG calculates the optimal allocation problem, which is NP-hard, resulting in exponential execution time. Although G-PMRM is designed based on a greedy algorithm, it must be calculated for each server, so the execution time is greater than that of IoV-RAM-LRL in most cases. Moreover, the execution time of G-PMRM and OPT-VCG is not related to the revenue limit B because no matter how B changes, these two algorithms are executed only once.

Figure 3d-f reflect the resource utilization of the three algorithms. OPT-VCG has the highest resource utilization because OPT-VCG satisfies the optimal allocation and aims to allocate resources to users, while IoV-RAM-LRL has the next highest resource utilization, followed finally by G-PMRM. The main reason is similar to that in Fig. 3b. Notably, even in the OPT-VCG algorithm, no resource reaches 100% utilization. The main reason is that some ECS resources cannot be allocated under deployment constraints.

Impact of the resource capacity of the server

This experiment shows the impact of different server resource capacities C on the three algorithms. The number of users N is fixed at 100, the number of servers M is fixed at 10, the number of resources R is fixed at 3, and the deployment constraint is $\delta = 2$.

The algorithm used for comparison in this experiment is IoV-RAM-LRL (Peak B), which can achieve the theoretically highest revenue. Figure 4a shows the revenue of the three algorithms. First, the revenue of IoV-RAM-LRL(Peak B) increases initially and then remains unchanged as the resource capacity increases. This is because when the resource capacity is small (0.25), the resource capacity is insufficient to meet the requirement of cost-effective users. At this time, OPT-VCG has the best performance. However, IoV-RAM-LRL(Peak B) can make cost-effective users win by increasing the unit price, so when the resource capacity increases, the revenue of IoV-RAM-LRL(Peak B) increases rapidly until it stabilizes. On the other hand, because IoV-RAM-LRL(Peak B) always selects the most cost-effective user subset in the system, it is not sensitive to changes in resource capacity. Therefore, as the resource capacity increases, the revenue does not change substantially, which also reveals that most of the revenue comes from the most cost-effective 1/2 of users in the system. However, the revenue

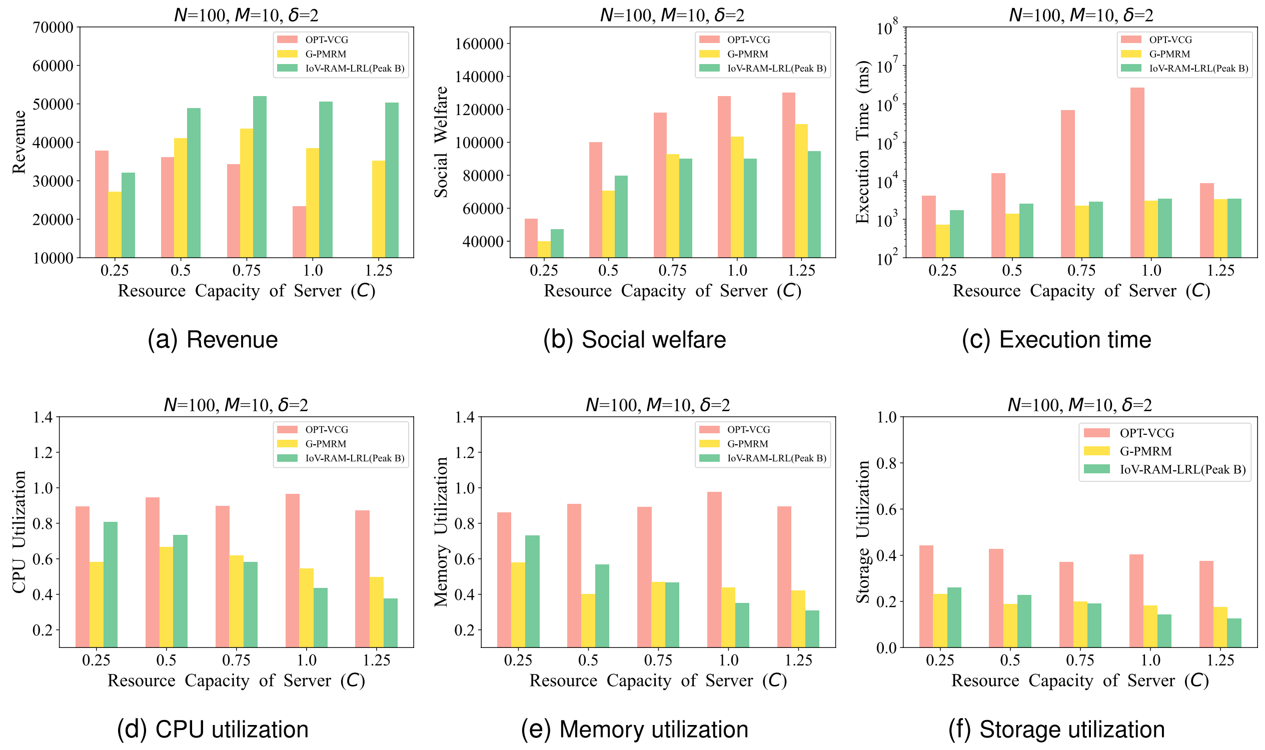

Fig. 4 The impact of the resource capacity of the server

of OPT-VCG and G-PMRM continues to decrease with increasing resource capacity. The main reason is that these two algorithms follow the principle of allocating as many resources as possible, and they use the lowest winning price. Abundance of resources leads to oversupply, resulting in a decrease in revenue.

Figure 4b shows the social welfare of different algorithms. The social welfare of IoV-RAM-LRL(Peak *B*) first increases and then remains unchanged because when the resource capacity is small, the number of winning users is very small. As the resource capacity increases, the number of winning users increases, so social welfare improves. However, when the resource capacity increases to 0.75, the resources are sufficient to satisfy the most cost-effective user subset selected by IoV-RAM-LRL(Peak *B*), so social welfare tends to be stable. The main reason for the increase in social welfare of OPT-VCG and G-PMRM is that the increase in resource capacity increases the number of winning users.

Figure 4c shows the execution times of different algorithms. The execution time of IoV-RAM-LRL(Peak *B*) increases because when the resource capacity is low, the algorithm only needs to eliminate users through capacity constraints. When the resources capacity is large, it is necessary to eliminate users with low cost-effectiveness through continuous iteration, so the execution time is long. OPT-VCG needs to calculate the optimal solution of the allocation, so the execution time is the longest. Furthermore, G-PMRM is executed separately for each server, so when the resource capacity increases, its execution time will also increase.

Figure 4d-f show the resource utilization of the three algorithms. The resource utilization of IoV-RAM-LRL(Peak *B*) continues to decrease. The main reason is that the algorithm tends to become stable after the subset of users with high cost performance is selected; thus, the resource utilization decreases as the resource capacity increases. In addition, OPT-VCG always maintains a high resource utilization rate because OPT-VCG follows *the principle of allocating as many resources as possible.* The reason for the low resource utilization of G-PMRM is that it is greatly affected by deployment constraints. The increase in resource capacity enables the top-ranked server to obtain more users, so the number of users that can be allocated by the bottom-ranked server decreases.

Impact of the deployment constraints

This experiment shows the impact of different deployment constraints (for example, 1-10 indicates that the number of ECSs to which users can connect is uniformly distributed within [1,10]). The number of users *N* is fixed at 100, the number of servers *M* is fixed at 10, the number

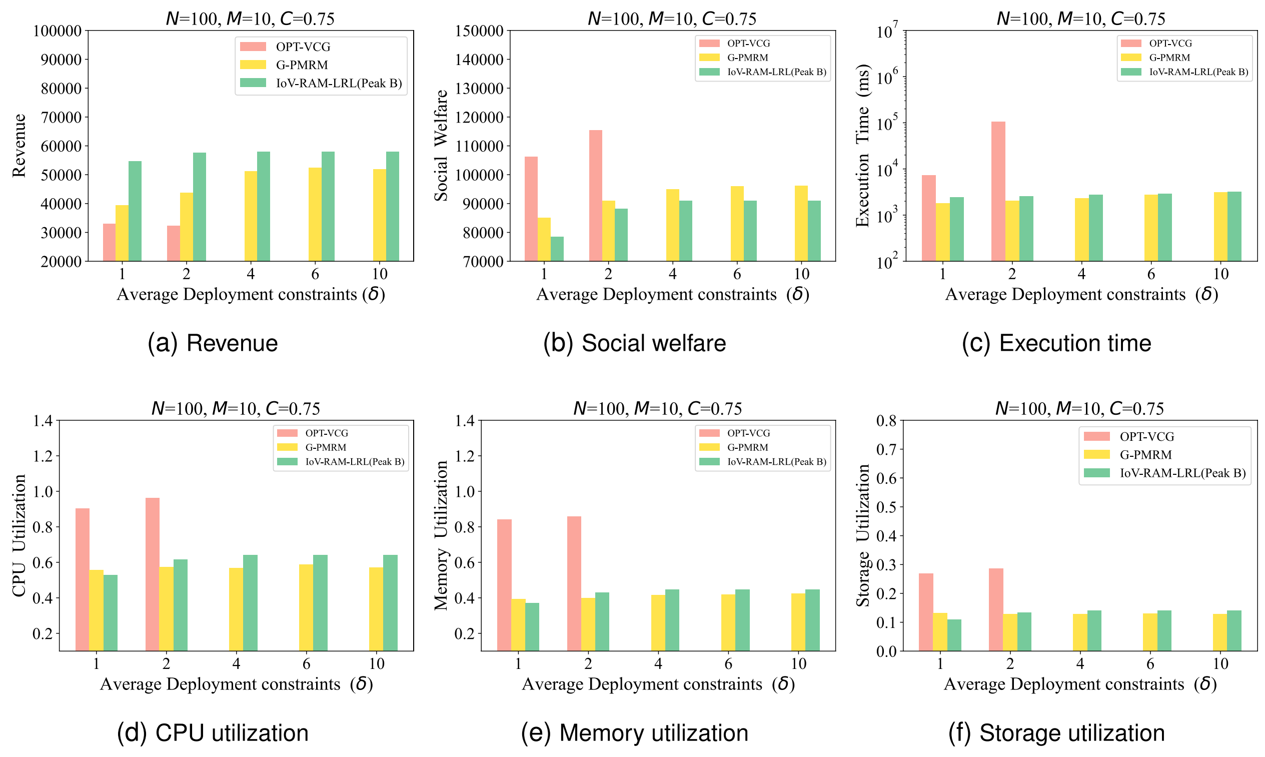

Fig. 5 Impact of the deployment constraints

of resources R is fixed at 3, and the server resource capacity parameter C is fixed at 0.75.

The algorithm used in this experiment is IoV-RAM-LRL(Peak B). Because solving the optimal allocation solution is NP-hard, when δ is greater than 3, OPT-VCG cannot be solved in limited time; thus, OPT-VCG is executed in only the first two experiments. Figure 5a shows the revenue of the three algorithms under different deployment constraints. The IoV-RAM-LRL(Peak B) algorithm has higher revenue under different deployment constraints because when the deployment constraints increase, the number of servers that users can connect to increases and more users can be allocated, so the revenue increases. When the deployment constraints reach a certain number (such as $\delta = [1, 4]$), almost all cost-effective users are successfully allocated, so the revenue remains stable. G-PMRM is based on the monotonic allocation algorithm. When δ increases, the number of users that can be allocated to each server increases, so the revenue also increases. The revenue of the OPT-VCG algorithm is relatively low because it follows *the principle of allocating as many resources as possible* and *the lowest winning price*.

Figure 5b shows the social welfare of the three algorithms. For all algorithms, the number of winning users increases due to the increase in δ, so the social welfare also increases. Among them, the IoV-RAM-LRL(Peak B) algorithm reaches the maximum social welfare when the deployment constraint $\delta = [1, 4]$ and no longer changes with changes in δ. Therefore, the IoV-RAM-LRL(Peak B) algorithm has selected the most valuable users at this time.

Figure 5c shows the execution time of the three algorithms. The execution time of the three algorithms increases with increasing δ because the change in deployment constraints leads to an increase in the number of winning users, which leads to an increase in the computational load of the IoV-RAM-LRL(Peak B) algorithm and the G-PMRM algorithm. Since it is NP-hard to solve the optimal allocation, the OPT-VCG algorithm has the longest execution time and is most affected by deployment constraints.

Figure 5d-f show the resource utilization of the three algorithms. Since OPT-VCG solves the optimal solution of allocation, the resource utilization is high. The IoV-RAM-LRL(Peak B) algorithm and the G-PMRM algorithm increase the number of winning users due to the increase in the deployment constraint δ, so the resource utilization is also improved. However, when the deployment constraint δ increases to a certain value, it is no longer the most important factor affecting allocation (at this time, resource capacity is the most important);

thus, social welfare, revenue, and resource utilization are almost unchanged.

The impact of resource capacity changes on unit prices

This experiment shows the impact of changes in CPU resource capacity on prices. The number of users N is fixed at 100, the number of servers M is fixed at 10, the resource type R is fixed at 3, the memory and storage resource capacity C is fixed at 0.75, and the revenue limit B is set at 20000, while deploying constraints $\delta = 2$ and using the IoV-RAM-LRL(Peak B) algorithm.

It can be seen from Fig. 6 that the unit price of the CPU decreases with increasing CPU resource capacity and finally matches the unit price of the storage resource. This is because when the CPU resource capacity is small, the price increase parameter λ_{cpu} increases, and the unit price of the CPU increases faster. As the resource capacity of the CPU gradually increases, the value of λ_{cpu} decreases, and the increase in the unit price of the CPU will gradually decrease. Notably, the unit price of memory and storage resources is fixed.

The impact of independent unit price changes

This experiment demonstrates the advantages of resource unit prices that vary independently. The unit price increase $\lambda_r \varepsilon$ of each resource in algorithm IoV-RAM-LRL(Peak B)($\lambda\varepsilon$) is calculated independently, while in algorithm IoV-RAM-LRL(Peak B)(ε), the unit price of each resource increases by ε. The number of servers M is fixed at 10, the resource type R is fixed at 3, the resource capacity C is fixed at 0.5, and the deployment constraint $\delta = 2$.

It can be seen from Fig. 7 that as the number of users increases, the final revenue of both algorithms increases, but the IoV-RAM-LRL(Peak B)($\lambda\varepsilon$) algorithm has higher revenue than IoV-RAM-LRL(Peak B)(ε). The main reason is that the former algorithm uses an independent unit price. When a certain resource is small, its unit price will increase faster so that a higher final revenue can be obtained. On the other hand, from a practical perspective, it is more reasonable for different resources to have different unit prices according to the balance of supply and demand.

Truthfulness verification

This experiment verifies the truthfulness of IoV-RAM-LRL from two perspectives. Specifically, 1) the bid of a winning user is changed to observe its utility changes, and 2) the bid of a losing user is changed to observe its utility changes. In this experiment, the number of users N is 100, the number of servers M is 10, the resource type R is 3, the resource capacity C is fixed at 0.75, and the deployment constraint $\delta = 2$.

Figure 8a shows the situation for the winning user 10. Her/his truthful bid is 2542, the resource requirement is (18,30,91), and when she/he wins, the payment is 2021 and the utility is $2542 - 2021 = 521$. By constantly changing her/his bid, it can be found that as long as the bid is higher than 2021, the user can still win, but her/his utility remains at 521, which is unchanged; this is because if the user can win, changing her/his bid will not affect the payment price, and the utility remains the same. When the bid is lower than 2021, allocation fails, so the utility is 0.

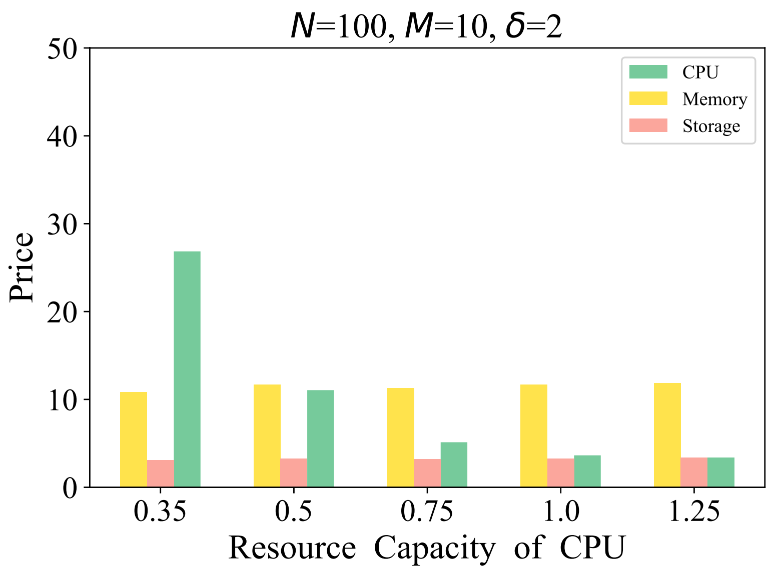

Fig. 6 The impact of CPU capacity changes on prices

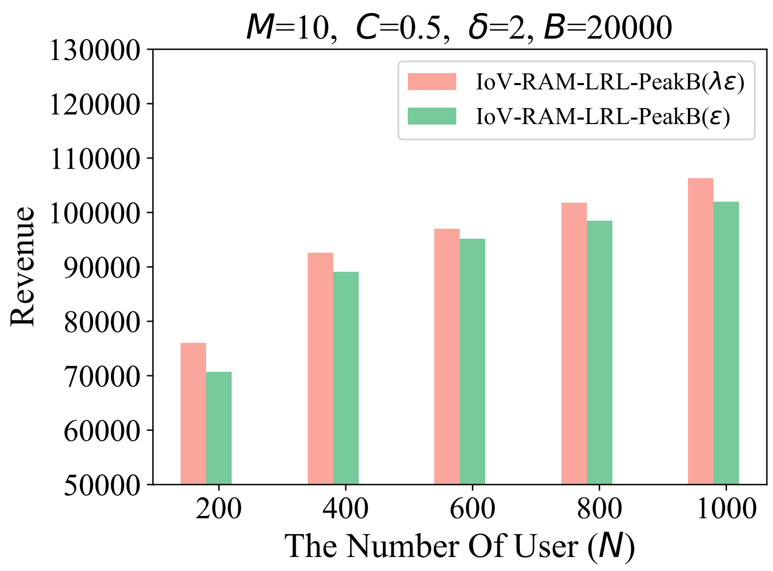

Fig. 7 The impact of independent unit price changes

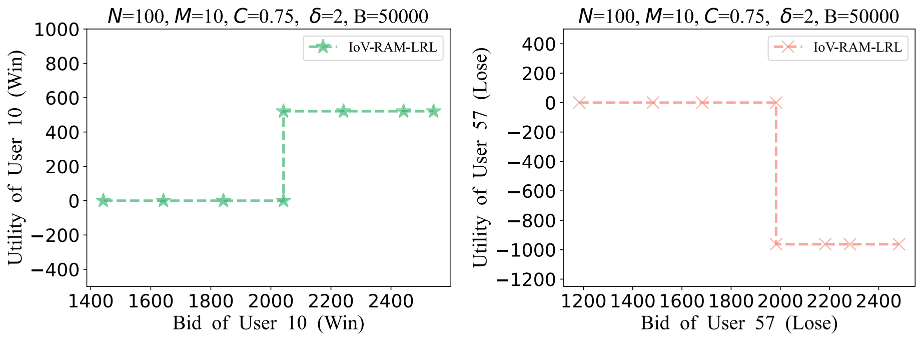

Fig. 8 Truthfulness verification

Figure 8b shows the situation for the losing user 57. Her/his truthful bid is 983, the resource requirement is (16,33,89), the final payment is 0, and the utility is 0. By changing her/his bid, it can be found that when the bid is lower than 1946, the user always fails to allocate, so no fee is paid, and the utility is obviously 0. When the bid is higher than 1946, the user wins the allocation; however, the utility at this time is $983 - 1946 = -963$.

Through the analysis of these two examples, it can be seen that users cannot obtain greater utility by changing the bid, thus verifying the truthfulness of IoV-RAM-LRL.

Conclusion

This paper proposes a truthful mechanism for IoV edge computing resource allocation with a lowest revenue limit. Compared with existing mechanisms, the proposed mechanism for calculating the unit price of resources through the intensity of competition of different resources can obtain greater revenue while ensuring individual rationality and truthfulness. Furthermore, our approach can be used under the edge computing deployment constraints, which improves the practicability of mechanism design in edge computing. However, many problems remain to be studied. For example, when the vehicle moves, the deployment constraints will change, which involves the problem of real-time mechanism design. Another example in the federated learning or metaverse scenarios considers not only the connection between the edge and the device but also involves the cloud-edge collaboration. These novel application scenarios will result in more complex resource allocation problems, which will be our main research work in the future.

Acknowledgements
We sincerely thank the Reviewers and the Editor for their valuable suggestions.

Authors' contributions
J. Zhang and Z. Wang wrote the main manuscript text and A. V. Vasilakos and W. Li proposed the method and revised the manuscript. All authors reviewed the manuscript.

Funding
This work is supported in part by the National Natural Science Foundation of China (Nos. 62062065, 12071417, 61962061), the Education Foundation of Yunnan Province of China (2022J002) and the Program for Excellent Young Talents, Yunnan, China.

Availability of data and materials
Not applicable.

Declarations

Ethics approval and consent to participate
This article does not contain any studies with human participants or animals performed by any of the authors.

Consent for publication
The authors read and approved the final manuscript.

Competing interests
The authors declare no competing interests.

Received: 24 July 2023 Accepted: 9 December 2023
Published online: 08 January 2024

References

1. Luo Q, Li C, Luan TH, Shi W (2020) Edgevcd: Intelligent algorithm-inspired content distribution in vehicular edge computing network. IEEE Internet Things J 7(6):5562–5579. https://doi.org/10.1109/JIOT.2020.2980981
2. Lu S, Yuan X, Shi W (2020) Edge compression: An integrated framework for compressive imaging processing on cavs. In: 2020 IEEE/ACM Symposium on Edge Computing (SEC). pp 125–138. https://doi.org/10.1109/SEC50012.2020.00017
3. Zhang J, Lou W, Sun H, Su Q, Li W (2022) Truthful auction mechanisms for resource allocation in the internet of vehicles with public blockchain networks. Futur Gener Comput Syst 132:11–24. https://doi.org/10.1016/j.future.2022.02.002
4. Hou X, Ren Z, Wang J, Cheng W, Ren Y, Chen KC, Zhang H (2020) Reliable computation offloading for edge-computing-enabled software-defined iov. IEEE Internet Things J 7(8):7097–7111. https://doi.org/10.1109/JIOT.2020.2982292
5. Wang J, Jiang C, Zhang K, Quek TQS, Ren Y, Hanzo L (2018) Vehicular sensing networks in a smart city: Principles, technologies and applications. IEEE Wirel Commun 25(1):122–132. https://doi.org/10.1109/MWC.2017.1600275
6. Reza Dibaj SM, Miri A, Mostafavi S (2020) A cloud priority-based dynamic online double auction mechanism (pb-dodam). J Cloud Comput 9. https://doi.org/10.1186/s13677-020-00213-7
7. Zheng X, Shah SBH, Usman S, Mahfoudh S, Shemim KSF, Kumar Shukla P (2023) Resource allocation and network pricing based on double auction in mobile edge computing. J Cloud Comput 12. https://doi.org/10.1186/s13677-023-00421-x
8. Zhang J, Zong M, Vasilakos AV, Li W (2023) Uav base station network transmission-based reverse auction mechanism for digital twin utility maximization. IEEE Trans Netw Serv Manag 1–1. https://doi.org/10.1109/TNSM.2023.3301522
9. Li Q, Jia X, Huang C (2023) A truthful dynamic combinatorial double auction model for cloud resource allocation. J Cloud Comput 12. https://doi.org/10.1186/s13677-023-00479-7
10. Nisan T, Roughgarden E, Tardos E, Vazirani V (2007) Algorithmic game theory, vol 3. pp 53–78. https://doi.org/10.1017/CBO9780511800481.020
11. Mashayekhy L, Nejad MM, Grosu D (2015) Physical machine resource management in clouds: A mechanism design approach. In: IEEE Transactions on Cloud Computing, vol 3, pp 247–260. https://doi.org/10.1109/TCC.2014.2369419
12. Zhang J, Xie N, Zhang X, Li W (2018) An online auction mechanism for cloud computing resource allocation and pricing based on user evaluation and cost. In: Future Generation Computer Systems, vol 89. pp 286–299. https://doi.org/10.1016/j.future.2018.06.034
13. Myerson RB (1981) Optimal auction design. Math Oper Res 6(1):58–73
14. Duan Z, Tang J, Yin Y, Feng Z, Yan X, Zaheer M, Deng X (2022) A context-integrated transformer-based neural network for auction design. In: Proceedings of the 39th International Conference on Machine Learning, vol 162. PMLR, Baltimore, p 5609–5626
15. Ausubel, Lawrence M (2004) An efficient ascending-bid auction for multiple objects. In: American Economic Review, vol 94. pp 1452–1475. https://doi.org/10.1257/0002828043052330
16. Dobzinski S, Lavi R, Nisan N (2012) Multi-unit auctions with budget limits. In: Games and Economic Behavior, vol 74. pp 486–503. https://doi.org/10.1016/j.geb.2011.08.003
17. Zaman S, Grosu D (2013) A combinatorial auction-based mechanism for dynamic vm provisioning and allocation in clouds. In: IEEE Transactions on Cloud Computing, vol 1. pp 129–141. https://doi.org/10.1109/TCC.2013.9
18. Mashayekhy L, Fisher N, Grosu D (2016) Truthful mechanisms for competitive reward-based scheduling. In: IEEE Transactions on Computers, vol 65. pp. 2299–2312. https://doi.org/10.1109/TC.2015.2479598
19. Liu X, Li W, Zhang X (2018) Strategy-proof mechanism for provisioning and allocation virtual machines in heterogeneous clouds. In: IEEE Transactions on Parallel and Distributed Systems, vol 29. pp 1650–1663. https://doi.org/10.1109/TPDS.2017.2785815
20. Jiao Y, Wang P, Niyato D, Suankaewmanee K (2019) Auction mechanisms in cloud/fog computing resource allocation for public blockchain networks. In: IEEE Transactions on Parallel and Distributed Systems, vol 30. pp 1975–1989. https://doi.org/10.1109/TPDS.2019.2900238
21. Li G, Cai J (2020) An online incentive mechanism for collaborative task offloading in mobile edge computing. In: IEEE Transactions on Wireless Communications, vol. 19. pp 624–636. https://doi.org/10.1109/TWC.2019.2947046
22. Zhang D, Tan L, Ren J, Awad MK, Zhang S, Zhang Y, Wan PJ (2020) Near-optimal and truthful online auction for computation offloading in green edge-computing systems. In: IEEE Transactions on Mobile Computing, vol 19. pp 880–893. https://doi.org/10.1109/TMC.2019.2901474
23. Li G, Cai J, Chen X, Su Z (2022) Nonlinear online incentive mechanism design in edge computing systems with energy budget. In: IEEE Transactions on Mobile Computing, pp 1–1. https://doi.org/10.1109/TMC.2022.3148034
24. Zhang J, Xie N, Yang X, Zhang X, Li W (2021) Strategy-proof mechanism for time-varying batch virtual machine allocation in clouds. In: Cluster Computing, vol 24. pp 3709–3724. https://doi.org/10.1007/s10586-021-03360-x
25. Zhang J, Xie N, Zhang X, Li W (2021) Strategy-proof mechanism for online time-varying resource allocation with restart. In: Journal of Grid Computing, vol 19. pp 25 (20 pp.). https://doi.org/10.1007/s10723-021-09563-1
26. Bahreini T, Badri H, Grosu D (2022) Mechanisms for resource allocation and pricing in mobile edge computing systems. In: IEEE Transactions on Parallel and Distributed Systems, vol 33. pp 667–682. https://doi.org/10.1109/TPDS.2021.3099731
27. He J, Zhang D, Zhou Y, Zhang Y (2020) A truthful online mechanism for collaborative computation offloading in mobile edge computing. In: IEEE Transactions on Industrial Informatics, vol 16, pp 4832–4841. https://doi.org/10.1109/TII.2019.2960127
28. Deng X, Xiao T, Zhu K (2019) Learn to play maximum revenue auction. IEEE Trans Cloud Comput 7(4):1057–1067. https://doi.org/10.1109/TCC.2017.2712142
29. Zhu K, Xu Y, Jun Q, Niyato D (2022) Revenue-optimal auction for resource allocation in wireless virtualization: A deep learning approach. IEEE Trans Mob Comput 21(4):1374–1387. https://doi.org/10.1109/TMC.2020.3021416
30. Li S, Huang J, Cheng B (2021) Resource pricing and demand allocation for revenue maximization in iaas clouds: A market-oriented approach. IEEE Trans Netw Serv Manag 18(3):3460–3475. https://doi.org/10.1109/TNSM.2021.3085519

31. Tsiourvas A, Bitsakos C, Konstantinou I, Fotakis D, Koziris N (2021) A mechanism design and learning approach for revenue maximization on cloud dynamic spot markets. In: 2021 IEEE 14th International Conference on Cloud Computing (CLOUD). pp 427–432. https://doi.org/10.1109/CLOUD53861.2021.00057
32. Singer Y (2010) Budget feasible mechanisms. In: 2010 IEEE 51st Annual Symposium on Foundations of Computer Science. pp 765–774. https://doi.org/10.1109/FOCS.2010.78
33. Anari N, Goel G, Nikzad A (2014) Mechanism design for crowdsourcing: An optimal 1-1/e competitive budget-feasible mechanism for large markets. In: 2014 IEEE 55th Annual Symposium on Foundations of Computer Science. pp 266–275. https://doi.org/10.1109/FOCS.2014.36
34. Zhang J, Zhang Y, Wu H, Li W (2022) An ordered submodularity-based budget-feasible mechanism for opportunistic mobile crowdsensing task allocation and pricing. IEEE Trans Mob Comput 1–18. https://doi.org/10.1109/TMC.2022.3232513
35. Chouayakh A, Amigo I, Bechler A, Maille P, Nuaymi L (2021) Multi-block ascending auctions for effective 5g licensed shared access. In: IEEE Transactions on Mobile Computing, 1–1, https://doi.org/10.1109/TMC.2021.3063990
36. Yi C, Cai J (2018) Ascending-price progressive spectrum auction for cognitive radio networks with power-constrained multiradio secondary users. In: IEEE Transactions on Vehicular Technology, vol 67. pp 781–794. https://doi.org/10.1109/TVT.2017.2744560
37. Yang X, Dong H, Teng X (2017) Ascending-price progressive spectrum auction for cognitive radio networks with power-constrained multiradio secondary users. In: Jisuanji Yanjiu yu Fazhan/Computer Research and Development, vol 54. pp 415–427. https://doi.org/10.7544/issn1000-1239.2017.20160491
38. Luong NC, Van TL, Feng S, Du H, Niyato D, Kim DI (2023) Edge computing for metaverse: Incentive mechanism versus semantic communication. IEEE Trans Mob Comput 1–17. https://doi.org/10.1109/TMC.2023.3317092
39. Nejad MM, Mashayekhy L, Grosu D (2015) Truthful greedy mechanisms for dynamic virtual machine provisioning and allocation in clouds. In: IEEE Transactions on Parallel and Distributed Systems, vol 26. pp 594–603. https://doi.org/10.1109/TPDS.2014.2308224
40. Alibaba cloud vm price (2023) https://www.aliyun.com/price/product#/commodity/vm. Accessed 20 Dec 2023
41. Huawei cloud dataset. (2023). https://github.com/WangZHeM/IoV-RAM-LRL/blob/main/training-1.txt. Accessed 20 Dec 2023
42. Huawei cloud cost. (2023). https://www.huaweicloud.com/product/ecs/recommend.html. Accessed 20 Dec 2023
43. Tencent cloud cost. (2023). https://buy.cloud.tencent.com/price/cvm/. Accessed 20 Dec 2023

Publisher's Note

Jixian Zhang received an M.S. and a Ph.D. degree in computer science from the University of Electronic Science and Technology of China in 2006 and 2010. Currently, he is an associate professor at the School of Computer Science and Engineering at Yunnan University. He has published 30+ articles in peer-reviewed journals and conferences, e.g., TMC, FGCS, JOGC, Cluster Computing, and Computing. His research interests include cloud computing, edge computing and mechanism design.

Zhemin Wang is a master's student from Yunnan University and is expected to obtain a master's degree in 2024. His main study and research directions include resource allocation, intelligence algorithms,edge computing,mechanism design.

Athanasios V. Vasilakos is with CAIR, UiA, and his main research interests include the IoT and mobile nets, artificial intelligence/machine learning, cybersecurity, and big data analytics. He has more than 44,000 citations and an H-index of 113. He has served or is serving as an editor for many technical journals, such as IEEE Transactions on Network and Service Management, IEEE Transactions on Cloud Computing, IEEE Transactions on Information Forensics and Security, IEEE Transactions on Cybernetics, IEEE Transactions on Services Computing, IEEE Transactions on Nanobioscience, IEEE Transactions on Information Technology in Biomedicine, ACM Transactions on Autonomous and Adaptive Systems, and IEEE Journal on Selected Areas in Communications. He was also General Chair of the European Alliances for Innovation.

Weidong Li received a Ph.D. from the Department of Mathematics, Yunnan University, in 2010. He is currently a professor with the School of Mathematics and Statistics at Yunnan University. He has published 90+ articles in peer-reviewed journals and conferences, e.g., TPDS, TMC, ALGO, JOA, TCS, JOCO, FGCS, and JOGC. His main research interests include combinatorial optimization, approximation algorithms, randomized algorithms and cloud computing.