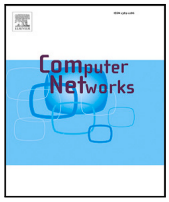
Computer Networks 246 (2024) 110428

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



On incentivizing resource allocation and task offloading for cooperative edge computing

Weibo Chu [a,*], Xinming Jia [a], Zhiwen Yu [b,a], John C.S. Lui [c], Yi Lin [a]

[a] Northwestern Polytechnical University, Xi'an, China
[b] Harbin Engineering University, Harbin, China
[c] The Chinese University of Hong Kong, Hong Kong

ARTICLE INFO

Keywords:
Cooperative edge computing
Resource allocation
Incentive mechanism
Double auction
Decentralized mechanism

ABSTRACT

Cooperative edge computing serves as an effective solution to provide reliable and elastic edge computing services through pooling geographically proximate edge resources and efficiently allocating them to users. The incentive mechanism is critical for realizing cooperative edge computing. In this paper, we propose a virtual machine (VM) or container based resource allocation scheme and two market-based mechanisms to stimulate collaboration among edge servers (ESs). Our first mechanism is a novel two-stage double auction based approach where edge resources are allocated to services/VMs at the first stage and then distributed among ESs at the second stage via a two-sided multi-unit auction. To make it more practical, we use markups to set the ask and bid prices for the participants in the market. Moreover, with the concept of virtual sellers and virtual buyers, we cast the multi-unit auction as single-unit one so that the auction becomes tractable. We further propose a price-based decentralized mechanism in case there is no central authority. With an appropriately constructed objective function, we formulate the resource allocation task as a network utility maximization (NUM) problem and leverage the dual-based decomposition theory for a distributed solution. We theoretically prove that both mechanisms guarantee properties such as budget balance and individual rationality for all ESs. Numerical studies with real-world dataset are presented to demonstrate the superior performance of our mechanisms over baselines.

1. Introduction

1.1. Background and motivation

Multi-access Edge Computing (MEC) [1], formally known as Mobile Edge Computing [2,3], is an emerging computing paradigm that enables a variety of new applications such as augmented reality, interactive gaming and autonomous driving. Being resource-hungry and delay-sensitive, these applications generally cannot be well supported by today's cloud-centric paradigm due to the excessively long delay, unstable network connection, security/privacy concerns, etc. MEC, on the other hand, brings computation, storage and networking resources at the edge of the network, so that services can be accessed locally and with low latency.

MEC has attracted significant attention since the concept was proposed, and numerous effort has been dedicated to the development of its theory and application. However, despite the fast growth, MEC is still in its infancy stage. The current practice is that edge resources are often sporadically deployed in distributed regions, configured in an *ad hoc* manner and a set of applications privately make use of them [4]. For example, civic authorities use edge devices to process data from sensors on public infrastructure, industrial executives rely on edge to monitor and analyze energy use in factories, families deploy edge computing services within the home to ensure real-time response and enhanced privacy for smart home applications, to name a few.

On one hand, the ad hoc, private and sporadic deployment of edge resources are far from realizing the vision of MEC — providing the public with reliable and elastic edge services, just like cloud computing. On the other hand, given these distributed (and isolated) edge resources, if we can group and efficiently make use of them, i.e., through intelligently allocating resources to services and scheduling users' tasks, the performance of the system and users' QoE then can be significantly improved in terms of reliability, scalability, cost, etc. This is essentially the aim of cooperative edge computing [5–8], which calls for federating geographically distributed edge resources and providing edge

* Corresponding author.
E-mail addresses: wbchu@nwpu.edu.cn (W. Chu), jxm12f@mail.nwpu.edu.cn (X. Jia), zhiwenyu@nwpu.edu.cn (Z. Yu), cslui@cse.cuhk.edu.hk (J.C.S. Lui), ly_cs@nwpu.edu.cn (Y. Lin).

https://doi.org/10.1016/j.comnet.2024.110428
Received 27 December 2023; Received in revised form 11 March 2024; Accepted 15 April 2024
Available online 20 April 2024

services in a systematical way, i.e., through handling variable requested workloads in a cooperative manner. Moreover, from the economic perspective, cooperative edge computing is also a more cost-effective solution as compared to building the network from scratch. Today, the academia and industry communities are working together on building an open edge computing environment/platform and providing services on a global scale, i.e., the Open Edge Computing [9], Openfog [10], etc.

There are several challenges in realizing cooperative edge computing, not only from technical, but also from social, legal and geopolitical considerations [4]. Among them, we consider two fundamental problems: *(1) How should we incentivize edge computing service providers (SPs) so that they are willing to share their resources with other SPs?* Indeed, edge resources are owned by self-interested providers who provision and operate their edge servers (ESs) in a way such that their revenue/profit is maximized. Without proper incentive mechanisms, SPs will be unwilling to contribute their resources to other SPs and have their workload processed. This is also true for the services/applications as users may be reluctant to offload their tasks to other ESs due to the reduced quality of experience; (2) *Given that SPs have incentive to collaborate with others, how should each SP allocate its edge resources among others (including itself), and which edge server should be selected for remote computation if a task is to be offloaded to the network?* Note that these problems of incentive mechanism design, resource allocation and task offloading are tightly coupled and each one is nontrivial to answer, given that SPs may have different preferences towards others, ESs are heterogeneous in resource capacity, cost, and the workload from SPs are with diverse characteristics, etc.

1.2. Contribution

In this paper, we seek to answer the aforementioned questions and our goal is to stimulate collaboration among SPs while at the same time, to maintain a high resource utilization efficiency and revenue for each SP. We propose a market-based resource allocation mechanism, where SPs with diverse characteristics act as buyers and sellers, and ES resources act as divisible goods in the market. In particular, we propose a two-stage double-auction based mechanism, where a third-party platform is introduced into the system for allocating resources and determining the prices at which resources are traded. Our proposed mechanism ensures certain desired properties such as individual rationality, budget balance, computation efficiency, etc. Moreover, we also propose a price-based allocation mechanism where edge resource allocation is performed by ESs in a fully decentralized manner, i.e., when there is no third-party platform as a central coordinator for organizing the market.

More specifically, we make the following contributions:

1. We consider a heterogeneous *multi-server* and *multi-service/application* system, where edge servers may have different amount of resources and services/applications at each server can vary in workload, cost, QoS requirements, etc. In line with the current technology, we propose a VM or container based resource allocation scheme, i.e., both the allocation of server resources and task offloading among edge servers are performed in terms of individual VMs/containers. This makes our model more realistic and also different from existing work that typically adopts a task based resource allocation and workload distribution scheme [11,12].

2. To incentivize collaboration among edge servers, we propose a novel two-stage double auction based resource allocation mechanism. The first stage of the mechanism aims at distributing resources of ESs to services, i.e., determining the number of VMs to be hosted for each service at each server, and also figuring out the demand and supply of the edge resources in the market. Resource allocation among ESs is performed at the second stage via a two-sided multi-commodity multi-unit auction, where we use *markups* [13] to capture the dishonest behaviors of both the buyers and sellers in the market. Moreover, to efficiently solve the auction, we adopt the concept of *virtual buyers* and *virtual sellers* so that the concerned auction is cast as single-unit auctions that can be well addressed by existing algorithms. We prove that the proposed two-stage mechanism achieves the desired economic properties such as individual rationality, budget balance, etc.

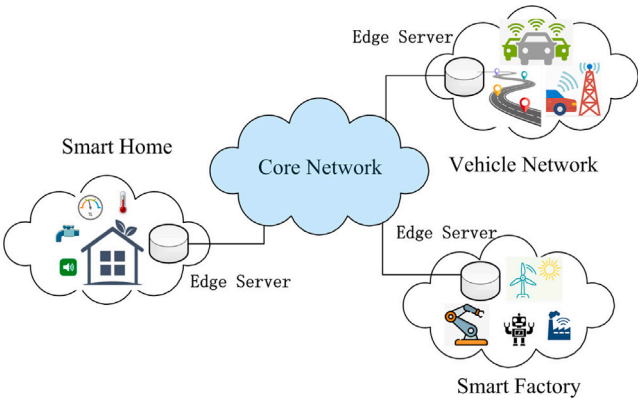


Fig. 1. A heterogeneous edge computing network.

3. In case there is no central coordinator as that required by the double auction based approach, we propose a price-based decentralized resource allocation mechanism. The proposed mechanism models the task with a network utility maximization (NUM) formulation, and leverages the dual-based decomposition theory for a distributed solution. Note that since the utility function of ESs are linear, the canonical NUM framework cannot be readily applied. To this end, we add to the utility function of each service a concave term that is appropriately constructed so that: (1) the problem under study becomes convex; and (2) the gap between the distributed solution and the optimal solution to the original problem can be well bounded. In addition, we give pricing strategies for ESs and prove their individual rationality under the proposed mechanism.

4. We evaluate both mechanisms through numerical studies with real-world dataset from today's cloud computing market. Results indicate that our mechanisms outperform the baselines, i.e., both mechanisms are able to provide an optimized social welfare, guarantee individual rationality for all ESs, and improve their revenue through resource allocation.

The remainder of this paper is organized as follows. In Section 2 we introduce system model and problem formulation. Section 3 describes our two-stage double auction based resource allocation mechanism. Section 4 elaborates on the price-based decentralized mechanism. Section 5 presents numerical studies and evaluation results. We give related work in Section 6 and conclude the paper in Section 7.

2. System model and problem formulation

2.1. System model

We consider a simple yet generic edge computing network as shown in Fig. 1, which consists of multiple Edge Servers (ES) and User Applications (Services). Each ES is responsible for serving its local users, and is connected via external links to other ESs. Note that to cover as many edge computing scenarios as possible, these ESs could be micro data centers, edge clouds, computing servers with high capacities, or even gateways at home and offices that endowed with computing/storage/networking resources. Furthermore, without loss of generality, we assume that these ESs are owned and operated by different edge computing service providers (SPs), and in practice, they can be geographically distributed and sporadically deployed.

One of the main challenges faced by the edge computing SPs in their current practice is how to provision ESs so as to maximize their

profit/revenue while at the same time, to maintain a high resource utilization efficiency. Given the high dynamics and unpredictability of user-generated workload, and the fact that edge resources are often configured in an *ad hoc* manner, it is almost impossible to meet both goals simultaneously. For example, putting too much resources into ESs will certainly bring high payoffs by customers through satisfying all their requests, however, the resource utilization efficiency in this case would be low, which results in a waste of investment. On the other hand, a lack of capacities of ESs ensures a high resource utilization, while at the cost of a decrease in revenue since only part of customers' workload can be processed at the network edge.

To address the above issue, one promising solution is to federate these geographically distributed ESs, i.e., to form a logically centralized resource pool. In this setting, a computing task from user can be either performed at its local ES as it was supposed, or be offloaded to a nearby ES given that the required service happens to be cached there and no QoS violation is incurred. This brings significant benefits such as improved system throughput, enhanced service reliability, and elasticity.

2.2. Problem formulation

For simplicity of notations, we denote the set of ESs and services/applications in the network as $\mathcal{M}$ and $\mathcal{N}$, respectively, and the number of ESs and services are M and N (see Table 1). Since we assume different ESs are owned by different SPs, we use ESs and SPs interchangeably throughout this paper. Moreover, we sometimes also use VMs and services interchangeably, for the fact that services are hosted by virtual machines or containers according to the current technology. Each VM is equipped with certain amount of resources and each one is able to process a specific amount of requests per time interval. We regard this workload processed by a VM or container per time interval as *basic workload*, and use it to characterize the workload of each service at each ES. In other words, the workload of each service is measured as the number of VMs required per time interval. In this work, we consider four types of edge resources for allocation: (*CPU, memory, bandwidth, disk space*), which is in accordance with the current cloud computing market for configuring and renting VMs [14].

Let $R_i = (R_{i1}, R_{i2}, R_{i3}, R_{i4})$ be the amount of the four types of resources at ES i, and $r_j = (r_{j1}, r_{j2}, r_{j3}, r_{j4})$ be the resources required by a VM of service j. Also denote k_{ij} be the number of VMs of service j that are to be hosted at ES i. Obviously, to have a valid resource allocation, we have the following constraint:

$$\sum_{j\in\mathcal{N}} k_{ij}\cdot r_{js} \le R_{is}, \quad \forall i, s \in \{1,2,3,4\}. \tag{1}$$

Let λ_{ij} be the workload of service j at ES i, and x_{ijk} be the number of VMs of service j at ES i that are allocated to ES k. Since we assume local workload has priorities and each SP will provide services to other SPs only when they have extra resources, we have:

$$0 \le \sum_{k\in\mathcal{N}(i)} x_{ijk} \le [k_{ij} - \lambda_{ij}]^+, \quad \forall i, j \tag{2}$$

where $[a]^+ = \max\{a, 0\}$, and $\mathcal{N}(i)$ is the set of ESs that can offload its workload to ES i without QoS violations, i.e., the offloading delay is no larger than a threshold pre-set by the service/application. Note that both k_{ij}'s and x_{ijk}'s are unknown variables and needs to be figured out by our proposed resource allocation mechanism. In addition, denote by l_{ij} be the number of VMs of service j at ES i for processing its local workload, we have:

$$l_{ij} = \min\{\lambda_{ij}, k_{ij}\}. \tag{3}$$

Also denote by c_{ij} be the cost of hosting a VM j at ES i. The cost can be due to hardware (CPU, memory, storage, etc.), software (database, load balancer, Firewall/IDS/IPS, etc.), management, maintenance, operation expenditure (i.e., energy consumption), etc. The cost of serving local workload at ES i, denoted as U_i^1, thus can be characterized as follows:

$$\mathrm{U}_i^1 = \sum_{j\in\mathcal{N}} l_{ij}\cdot c_{ij} \tag{4}$$

Likewise, the cost of serving workload from other SPs (providing services to other ESs) at ES i, denoted as U_i^2, can be given as:

$$\mathrm{U}_i^2 = \sum_{j\in\mathcal{N}}\sum_{k\in\mathcal{N}(i)} x_{ijk}\cdot c_{ij} \tag{5}$$

In an edge-computing market, end-users pay their local SPs for enjoying edge computing services. Normally, users tend to give a higher payment if SPs can provide them with a better quality of service (QoS). Let v_{ijk} be the payoff received by ES i, from its users, for having the (basic) workload of service j processed at ES k. In general, we have $v_{iji} \ge v_{ijk}, \forall k \ne i$, i.e, local computation brings the highest payoff due to its superior QoS.[1] The payoffs received by ES i from its users thus can be characterized as:

$$\mathrm{U}_i^3 = \sum_{j\in\mathcal{N}} (l_{ij}\cdot v_{iji} + \sum_{k\in\mathcal{I}(i)} x_{kji}\cdot v_{ijk}) \tag{6}$$

where $\mathcal{I}(i)$ is the set of ESs to which i can offload its tasks without QoS violations. Note that it is possible we have $\mathcal{N}(i) \ne \mathcal{I}(i)$.

From Eq. (6), we can see that the payoffs of each ES from serving its end-users comprises of two parts. The first one is from local computation, i.e., the workload of its users is processed at the ES, and the other one is from offloading users' tasks to other ESs and having them processed there. This, to some extent, gives rise to the concept of local edge computing and remote edge computing, as illustrated in Fig. 2. Note that in this work, it is our assumption that users will offload tasks only to their local ESs, and these local ESs will then determine whether to perform tasks locally or direct them to nearby ESs for remote computation. The process of offloading tasks to other ESs and having them processed there, however, is transparent to end-users albeit they may be aware of the decreased QoS, i.e., large delay.

Since ESs are self-interested, proper incentive mechanism is needed to stimulate collaboration. Let o_{ijk} be the reward provided by ES i to ES k if ES k allocates a VM of service j to ES i, and has i's workload processed. The reward received by ES i from other ESs thus can be calculated as:

$$\mathrm{U}_i^4 = \sum_{j\in\mathcal{N}}\sum_{k\in\mathcal{N}(i)} x_{ijk}\cdot o_{kji} \tag{7}$$

On the other hand, the reward that ES i provides/pays to other ESs is:

$$\mathrm{U}_i^5 = \sum_{j\in\mathcal{N}}\sum_{k\in\mathcal{I}(i)} x_{kji}\cdot o_{ijk} \tag{8}$$

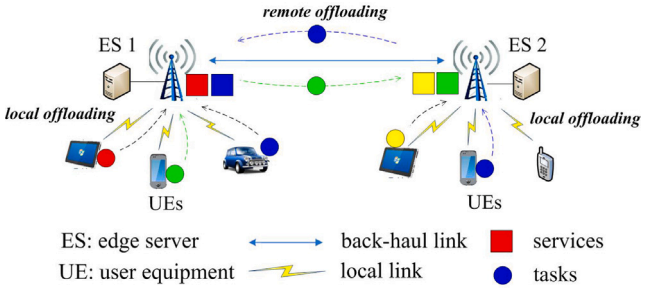


Fig. 2. The concept of local edge computing and remote edge computing. At ES 1, the workload marked as red and blue is locally processed, whereas the workload marked green is offloaded to ES2 for remote computation; Likewise, at ES 2 the workload marked yellow is locally processed whereas the workload marked blue is offloaded to ES 1. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

[1] If $v_{iji} = v_{ijk}$, it means that users' payment is oblivious to QoS.

Summarizing all the above, we have the revenue of ES i, denoted as U_i, as follows:

$$U_i = U_i^3 + U_i^4 - U_i^1 - U_i^2 - U_i^5 \tag{9}$$

Moreover, as we assume ESs are rational and each one tries to maximize its profit, we can formulate the following problem for each ES i:

$$\underset{\{k_{ij},x_{ijk},o_{ijk}\}}{\text{Maximize:}}\ U_i \tag{10a}$$

$$\text{s.t.}\quad 0 \le \sum_{k\in\mathcal{N}(i)} x_{ijk} \le [k_{ij} - \lambda_{ij}]^+, \quad \forall j \tag{10b}$$

$$0 \le \sum_{k\in\mathcal{I}(i)} x_{kji} \le [\lambda_{ij} - k_{ij}]^+, \quad \forall j \tag{10c}$$

$$\sum_{j\in\mathcal{N}} k_{ij}\cdot r_{js} \le R_{is}, \quad \forall s \in \{1,2,3,4\} \tag{10d}$$

$$k_{ij} \in \mathbb{N}, \quad x_{ij} \in \mathbb{N}, \quad o_{ijk} \in \mathbb{R}^+, \quad \forall j,k. \tag{10e}$$

Note that constraint (10c) gives the upper bound of the workload of each service that can be offloaded to the network (other ESs).

Our goal is to maximize the overall payoffs/revenue of the network, denoted by U, through optimization of incentive design o_{ijk}'s, resource allocation k_{ij}'s and task offloading x_{ijk}'s. The problem can be mathematically formulated as follows:

$$\underset{\{k_{ij},x_{ijk},o_{ijk}\}}{\text{Maximize:}}\ U = \sum_{i\in\mathcal{M}} U_i \tag{11a}$$

$$\text{s.t.}\quad 0 \le \sum_{k\in\mathcal{N}(i)} x_{ijk} \le [k_{ij} - \lambda_{ij}]^+, \quad \forall i,j \tag{11b}$$

$$0 \le \sum_{k\in\mathcal{I}(i)} x_{kji} \le [\lambda_{ij} - k_{ij}]^+, \quad \forall i,j \tag{11c}$$

$$\sum_{j\in\mathcal{N}} k_{ij}\cdot r_{js} \le R_{is}, \quad \forall i,s \in \{1,2,3,4\} \tag{11d}$$

$$k_{ij} \in \mathbb{N}, \quad x_{ij} \in \mathbb{N}, \quad o_{ijk} \in \mathbb{R}^+, \quad \forall i,j,k. \tag{11e}$$

Note that since $\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{N}}\sum_{k\in\mathcal{N}(i)} x_{ijk}\cdot o_{kji} = \sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{N}}\sum_{k\in\mathcal{I}(i)} x_{kji}\cdot o_{ijk}$, i.e., $\sum_{i\in\mathcal{M}} U_i^4 = \sum_{i\in\mathcal{M}} U_i^5$, we have $U = \sum_{i\in\mathcal{M}} U_i = \sum_{i\in\mathcal{M}}(U_i^3 - U_i^1 - U_i^2)$. In other words, the overall payoffs of the network does not depend on o_{ijk}'s. However, since ESs are selfish, we need o_{ijk}'s to stimulate their collaboration. Meanwhile, as we adopt a market-based mechanism, we also want our solution to have the following properties:

(1) *Individual Rationality* (IR): A seller is paid more than its ask (cost), and a buyer pays less than its bid (true valuation). In other words, both the sellers and buyers can gain profit from resource allocation/trading.
(2) *Budget Balance* (BB): The total payment made by the participants is at least zero and there is no deficit, i.e., the market organizer does not have to subsidize the market [15].
(3) *Computation Efficiency*: Given that problem (11) is a Mixed Integer Linear Program (MILP) with $2MN + 4M$ constraints, which is NP-hard in nature, we need algorithms to solve it efficiently, i.e., in polynomial time.

3. A two-stage double-auction based mechanism

In this section, we elaborate on our two-stage double-auction based mechanism to solve problem (11) with the desired properties. The main idea is as follows: we decompose problem (11) into two sub-problems, where the first one is to determine resource allocation to services at each ES, i.e., to decide the number of VMs hosted for each service (we call it resource configuration), and the second one is to distribute these VMs among ESs and also to determine at what prices these resources are traded. Correspondingly, our proposed mechanism consists of two stages as depicted in Fig. 3, where each stage is dedicated to solve one sub-problem.

Table 1
Main notations.

Symbol	Definition
$\mathcal{M}$	Set of ESs in system
$\mathcal{N}$	Set of services in system
R_{is}	The amount of resource of type s at ES i
r_{js}	The amount of resource of type s required by a VM j
λ_{ij}	Local workload of service j at ES i
k_{ij}	Number of VMs of service j hosted at ES i
l_{ij}	Number of VMs of service j at ES i for processing local workload
c_{ij}	Cost of hosting a VM of service j at ES i
v_{ijk}	Payoff received by ES i from having its workload of service j processed at ES k
o_{ijk}	Reward provided by ES i to ES k from having i's workload of service j processed at ES k
x_{ijk}	Number of VMs of service j at ES i allocated to ES k
U_i	Revenue of ES i
L_{ij}	Supply of VMs of service j at ES i
D_{ij}	Demand of VMs of service j at ES i
Bid_{ijk}	Bid submitted by ES i for a VM of service j at ES k
Ask_{jk}	Ask submitted by ES k for providing a VM of service j
$\mathcal{B}_j$	Set of buyers for VMs of service j
$\mathcal{S}_j$	Set of sellers for providing VMs of service j
$\mathcal{VB}_i$	Set of virtual buyers of ES i
$\mathcal{VS}_k$	Set of virtual sellers of ES k
Prc_{ijk}	The price that service ij pays to ES k for using its resources
Prc_{ik}	The price that ES i pays to ES k for using its edge resources

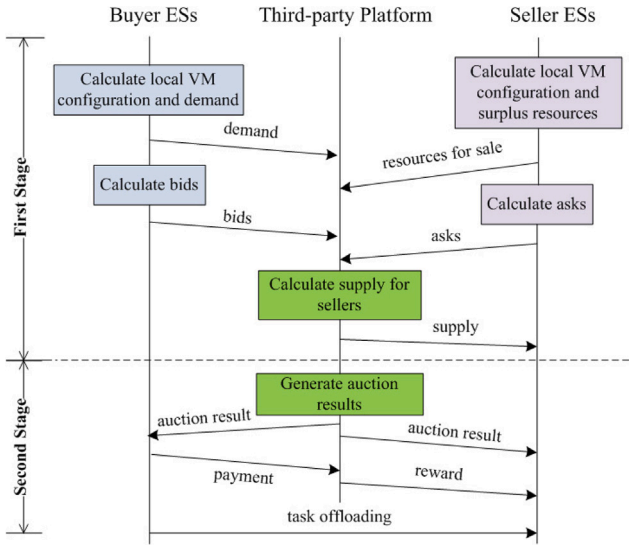


Fig. 3. Two-stage double auction for resource allocation among edge computing servers.

3.1. First stage for resource configuration

At this stage, each ES preferentially dedicates its resources to local workload, given the fact that: (1) serving local workload brings higher payoff than offloading it to other ESs, i.e., due to reduced QoS; and (2) providing computing services to other ESs is also sub-optimal since it is reasonable to believe that local workload has priority over non-local workload, even if other ESs can provide a high reward. Let l_{ij}^{Local} be the number of VMs of service j that are to be hosted at ES i for processing its local workload, we can formulate the following Knapsack problem for each ES i, where the objective is again to maximize i's

payoff through proper resource allocation:

$$\underset{\{l_{ij}^{\text{Local}}\}}{\text{Maximize:}} \sum_{j\in\mathcal{N}} l_{ij}^{\text{Local}}(v_{iji} - c_{ij}) \tag{12a}$$

$$\text{s.t.} \sum_{j\in\mathcal{N}} l_{ij}^{\text{Local}} r_{js} \le R_{is}, \quad \forall s \in \{1,2,3,4\} \tag{12b}$$

$$l_{ij}^{\text{Local}} \le \lambda_{ij}, \quad \forall j \tag{12c}$$

$$l_{ij}^{\text{Local}} \in \mathbb{N}, \quad \forall j. \tag{12d}$$

Note that the above problem can be locally addressed by ES i, since all the information required, i.e., $(v_{iji}, c_{ij}, R_{is}, \lambda_{ij})$, are locally available.

Knapsack problems have been well studied and many efficient heuristic/approximation algorithms [16] have been proposed. Once problem (12) is solved, each ES is aware of whether it is in short of resources or can provide resources to other ESs. Let $\{L_{ij}^{\text{Local}}\}$ be the solution to problem (12), and D_{ij} be the number of VMs of service j that ES i demands in order to satisfy all its local workload, we have:

$$D_{ij} = [\lambda_{ij} - L_{ij}^{\text{Local}}]^+, \quad \forall i,j \tag{13}$$

The remaining resources of type s at each ES i, denoted as R_{is}^{Left}, is:

$$R_{is}^{\text{Left}} = [R_{is} - \sum_{j\in\mathcal{N}} L_{ij}^{\text{Local}} r_{js}]^+, \quad \forall i,j \tag{14}$$

Based on D_{ij}'s and R_{is}^{Left}'s, we can partition the set of ESs into two sub-sets $\mathcal{B}$ and $\mathcal{S}$, where $\mathcal{B}$ is the set of ESs that demand resources, i.e, $\mathcal{B} = \{i \in \mathcal{M} | \exists j \in \mathcal{N} \text{ if } D_{ij} > 0\}$, and $\mathcal{S} = \mathcal{M} \setminus \mathcal{B}$. Note that $\mathcal{B}$ and $\mathcal{S}$ are also the set of buyers and sellers in the market.

At this point, although we know the set of buyers and sellers, and how much resources is available at each seller for potential trading, we still need to determine how these remaining resources are allocated to services, as the commodity in the market are individual VMs. This problem can be addressed by the third-party platform based on the bids and asks submitted from ESs. Hereafter we first give the bidding and asking strategies of the buyers and sellers, and then describe the detailed allocation scheme.

In a real market, it is common that traders submit bids lower than its true valuation to the commodity or asks higher than its cost to gain more profit. The markup [13] concept, which is from the field of economics, can be used to capture this dishonest behaviors. In this work, we adopt nonlinear markups for both the buyers and sellers. More specifically, for each buyer $i \in \mathcal{B}$, the true valuation to a VM of service j that provided by seller $k \in \mathcal{S}$ is v_{ijk}. We define the buyer markup MB_{ijk} for that VM as:

$$\text{MB}_{ijk} = \frac{v_{ijk}}{\max\limits_{s\in\mathcal{T}(i)} \{v_{isk}\}} \in (0,1] \tag{15}$$

where $\mathcal{T}(i)$ denotes the set of services/VMs that i demands, i.e., $\mathcal{T}(i) = \{j \in \mathcal{N} | D_{ij} > 0\}$. Obviously, this markup reflects how thirsty the buyer i is for a VM of that service from ES k. The bid submitted by buyer i is then defined as:

$$\text{Bid}_{ijk} = v_{ijk} \times \frac{9}{10 - \text{MB}_{ijk}} \tag{16}$$

From (15) and (16), we can see that the higher true valuation to the VM, the higher bids that each buyer submits.

On the other hand, for each seller $k \in \mathcal{S}$, the cost of providing a VM of service j is c_{kj}. We define the seller markup MS_{kj} as:

$$\text{MS}_{kj} = \frac{c_{kj}}{\max\limits_{s\in\mathcal{N}} \{c_{ks}\}} \in (0,1] \tag{17}$$

which is the ratio of the cost to the largest cost of providing a VM. The asks submitted by seller k is then defined as:

$$\text{Ask}_{kj} = c_{kj} \times \frac{10}{10 - \text{MS}_{kj}} \tag{18}$$

Obviously, we can see that the higher cost of a VM/service, the higher markups and also the higher asks. This reflects the fact that each seller wants to gain more profit from a commodity with high cost, which is common in a real market.

With the bids and asks from ESs, the third-party platform determines how the remaining resources at each seller are allocated to VMs of services. This task, also refereed to as *resource configuration/provision* at sellers, is achieved by solving the following optimization problem for each $i \in \mathcal{S}$:

$$\underset{\{l_{ijk}^{\text{Offer}}\}}{\text{Maximize:}} \sum_{k\in\mathcal{B}} \sum_{j\in\mathcal{N}} l_{ijk}^{\text{Offer}}(\text{Bid}_{kji} - \text{Ask}_{ij}) \tag{19a}$$

$$\text{s.t.} \sum_{k\in\mathcal{B}} \sum_{j\in\mathcal{N}} l_{ijk}^{\text{Offer}} r_{js} \le R_{is}^{\text{Left}}, \quad \forall s \in \{1,2,3,4\} \tag{19b}$$

$$l_{ijk}^{\text{Offer}} \le D_{kj}, \quad \forall k,j \tag{19c}$$

$$l_{ijk}^{\text{Offer}} \in \mathbb{N}, \quad \forall j. \tag{19d}$$

where l_{ijk}^{Offer} is the number of VMs of service j that i offers to the buyer $k \in \mathcal{B}$, and the objective is to maximize i's revenue. Note that problem (19) is also a Knapsack problem, where (19b) gives the upper bounds on the amount of resources for allocation, and (19c) states that the supply should not exceed demand.

Let $\{L_{ijk}^{\text{Offer}}\}$ be the solution to (19). We would like to emphasize that although this solution gives a system-wide resource allocation, it is not optimal, i.e., from the social welfare point of view, and may even violate certain desired economic properties. In fact, this solution is only used to determine the amount of VMs for trading in the market. For example, the number of VMs of service j provided by seller $i \in \mathcal{S}$, denoted as L_{ij}, can be given as:

$$L_{ij} = \sum_{k\in\mathcal{B}} L_{ijk}^{\text{Offer}} \tag{20}$$

3.2. Second stage for resource allocation

This stage is dedicated to allocating VMs among ESs and also determining the prices at which VMs are traded, with a double-auction based mechanism. Note that since each seller has multiple VMs for sale, and each bidder may require multiple ones as well, the mechanism we consider obviously belongs to the type of multi-unit auctions. Moreover, as VMs can be of different services, the auction is also for the market with heterogeneous goods. This is almost the most complex situation for designing auction mechanisms.

Our proposed mechanism is based on the following key observation: *both the seller's and buyer's valuations for the VMs are additively separable*, i.e., for each buyer the value of obtaining several VMs is the sum of these VMs' valuation. This is also true for the sellers as for each seller the cost of providing several VMs is the sum of these VMs' cost. Based on this observation, we can decompose the auction into multiple ones, where each one is for the assignment of VMs of one service. Moreover, for each of these auctions, we can further convert it into the standard single-unit double auction, through leaveraging the concept of *virtual sellers* and *virtual buyers*. Once the auction is finished, the assignment of VMs between sellers and buyers can be recovered by grouping.

We now elaborate on the details of the mechanism. Note that it suffices to focus on the auction for assigning VMs of one service. Now assume VMs of service j is to be allocated, and let $\mathcal{S}_j$ and $\mathcal{B}_j$ be the set of sellers and buyers, respectively, i.e., $\mathcal{B}_j = \{i \in \mathcal{M} | D_{ij} > 0\}$, and $\mathcal{S}_j = \{i \in \mathcal{M} | L_{ij} > 0\}$. For each buyer $i \in \mathcal{B}_j$ its demand for the VMs is D_{ij}, we can virtualize it into D_{ij} virtual buyers with each one requiring one VM. Likewise, for each seller $k \in \mathcal{S}_j$, its supply of the VMs is L_{kj}, we can also virtualize it into L_{kj} virtual sellers with each one offering one VM. The concept of virtual sellers and virtual buyers is depicted in Fig. 4.

Let $\mathcal{VB}_i$ be the set of virtual buyers of $i \in \mathcal{B}_j$, and $\mathcal{VS}_k$ be the set of virtual sellers of $k \in \mathcal{S}_j$. For each $l \in \mathcal{VB}_i$, its valuation to a VM of service j provided by each $t \in \mathcal{VS}_k$, denoted as v_{ljt}, is the same as i's

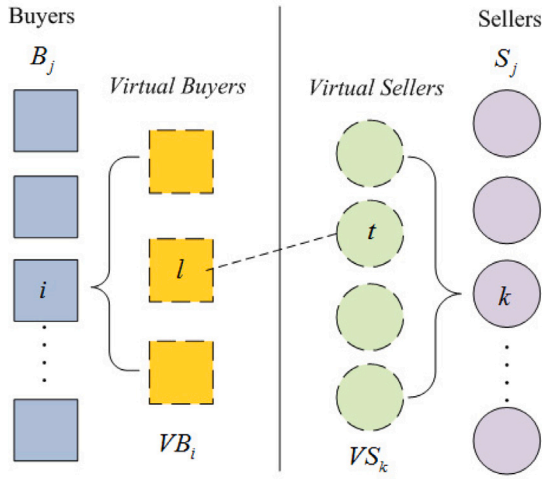


Fig. 4. The concept of virtual buyers and virtual sellers in the trade of VMs of service j: here $i \in B_j$ and $D_{ij} = 3$, $k \in S_j$ and $L_{kj} = 4$, l is a virtual buyer of ES i, and t is a virtual seller of ES k.

valuation to the VM if it is provided by k, i.e., $v_{ljt} = v_{ijk}$. As a result, the markup and bid will also be the same, which is as follows:

$$\mathrm{MB}_{ljt} = \mathrm{MB}_{ijk} \tag{21}$$

and

$$\mathrm{Bid}_{ljt} = \mathrm{Bid}_{ijk} \tag{22}$$

Likewise, the cost of providing a VM of service j by each $t \in \mathcal{VS}_k$, denoted by c_{tj}, is the same as the cost that the VM is provided by k, i.e., $c_{tj} = c_{kj}$. It follows that the markup and ask will also be the same, i.e., we have:

$$\mathrm{MS}_{tj} = \mathrm{MS}_{kj} \tag{23}$$

and

$$\mathrm{Ask}_{tj} = \mathrm{Ask}_{kj} \tag{24}$$

Let Pay_l^t be the price that $l \in \mathcal{VB}_i$ pays to the auctioneer if it wins out and obtains a VM of service j from $t \in \mathcal{VS}_k$. The payoff of l thus can be expressed as:

$$\mathrm{PO}_l = v_{ljt} - \mathrm{Pay}_l^t \tag{25}$$

On the other hand, the payoff of the seller t, if winning out, is:

$$\mathrm{PO}_t = \mathrm{Rec}_t^l - c_{tj} \tag{26}$$

where Rec_t^l is the price that t receives.

As we assume that the third-party platform seeks no profit in the auction, it can set Pay_l^t and Rec_t^l as follows [17,18]:

$$\mathrm{Pay}_l^t = \mathrm{Rec}_t^l = \frac{\mathrm{Ask}_{tj} + \mathrm{Bid}_{ljt}}{2} \tag{27}$$

Note that if the third-party platform does seek profit, which is typical in real world, we can set Pay_l^t and Rec_t^l as:

$$\mathrm{Pay}_l^t = \frac{\mathrm{Ask}_{tj} + \mathrm{Bid}_{ljt}}{2} + \frac{\alpha}{2} \times (\mathrm{Bid}_{ljt} - \mathrm{Ask}_{tj}) \tag{28}$$

$$\mathrm{Rec}_t^l = \frac{\mathrm{Ask}_{tj} + \mathrm{Bid}_{ljt}}{2} - \frac{\alpha}{2} \times (\mathrm{Bid}_{ljt} - \mathrm{Ask}_{tj}) \tag{29}$$

where $\alpha \in [0, 1]$ is a parameter for regulating the profit. In this case, the gain for the third-party platform is $\alpha \times (\mathrm{Bid}_{ljt} - \mathrm{Ask}_{tj})$.

Let x_l^t be a binary variable indicating whether l and t are successfully matched in the auction. The social welfare SW from this trade can be expressed as:

$$\begin{aligned} \mathrm{SW} &= \sum_{i \in B_j} \sum_{k \in S_j} \sum_{l \in \mathcal{VB}_i} \sum_{t \in \mathcal{VS}_k} (\mathrm{PO}_t + \mathrm{PO}_l) \cdot x_l^t \\ &= \sum_{i \in B_j} \sum_{k \in S_j} \sum_{l \in \mathcal{VB}_i} \sum_{t \in \mathcal{VS}_k} (v_{ljt} - c_{tj}) \cdot x_l^t \end{aligned} \tag{30}$$

In this work, we aim to maximize social welfare and thus can formulate the following optimization problem for the third-party platform:

$$\text{Maximize: } \mathrm{SW} \tag{31a}$$

$$\text{s.t.} \quad \sum_{t \in \mathcal{VS}_k} x_l^t \le 1, \quad \forall i, k, l \tag{31b}$$

$$\sum_{l \in \mathcal{VB}_i} x_l^t \le 1, \quad \forall i, k, t \tag{31c}$$

$$(\mathrm{Bid}_{ljt} - \mathrm{Ask}_{tj}) \cdot x_l^t \ge 0, \quad \forall l, t \tag{31d}$$

$$x_l^t \in \{0, 1\}, \quad \forall l, t \tag{31e}$$

Note that constraint (31b) states that a virtual buyer can obtain a VM from at most one virtual seller, and (31c) tells that a virtual seller can provide a VM to at most one virtual buyer. The two constraints together implies a one-to-one mapping between virtual sellers and virtual buyers on the VM allocation. Constraint (31d) ensures that the transaction between a seller and a buyer will only be concluded if the bid is higher than the ask, which essentially guarantees the economic property of individual rationality.

Problem (31) is an ILP problem with many linear constraints. To solve it efficiently, we convert it as the maximum weighted matching problem (MWM) over a bipartite graph $G(\pi, \zeta, \xi, \psi)$, where π and ζ denotes the set of virtual buyers and virtual sellers, respectively, and ξ is the set of edges between them, i.e., $\xi = \{(l, t) | (\mathrm{Bid}_{ljt} - \mathrm{Ask}_{tj}) \ge 0\}$. ψ denotes the weight on edges, i.e., $\psi(l, t) = v_{ljt} - c_{tj}, \forall (l, t) \in \xi$. Note that this weighting process is done over all the edges and hence the complexity for the construction of the bipartite graph is $O(n^2)$, where n is the number of nodes. Moreover, MWM is a well-known assignment problem and can be efficiently solved by existing algorithms [19]. In this work, we adopt the classical Hungarian algorithm whose complexity is $O(n^3)$ [20]. The overall complexity to solve problem (31) is therefore $O(n^3)$.

Once the auction is finished, we can recover the VM assignment among ESs by grouping. Let $\{x_l^{t,\#}\}$ be the solution to (31), the number of VMs of service j that ES k allocates to ES i, denoted as x_{ijk}, can be calculated as follows:

$$x_{ijk} = \sum_{l \in \mathcal{VB}_i} \sum_{t \in \mathcal{VS}_k} x_l^{t,\#} \tag{32}$$

Correspondingly, the price that ES i pays to k for these VMs, denoted as Prc_{ijk}, is:

$$\mathrm{Prc}_{ijk} = \frac{1}{2}(\mathrm{Ask}_{kj} + \mathrm{Bid}_{ijk}) \cdot x_{ijk}. \tag{33}$$

3.3. Mechanism analysis

Here we show that our mechanism satisfies the desired properties.

Theorem 3.1. *The proposed mechanism is computationally tractable.*

Proof. Let $S = max\{L_{ij}, D_{ij}\}_{(i,j) \in \mathcal{M} \times \mathcal{N}}$. From Section 3.1, we know that the complexity of the first stage is on solving two Knapsack problems (12) and (19) for all ESs, which are $O(MN)$ and $O(M^2N)$, respectively. On the other hand, the complexity of the second stage is on constructing a bipartite graph and executing the Hungarian algorithm for each service. The complexity of this stage is at most $O(S^3M^3N)$. It follows that the complexity of our mechanism is at most $O(S^3M^3N)$. □

Theorem 3.2. *The proposed mechanism guarantees individual rationality for all ESs.*

Proof. From the bidding strategies as described in Eqs. (15) and (16), we know that the bid submitted by a buyer is no larger than its true valuation to a VM, i.e., we have $v_{ijk} \geq \text{Bid}_{ijk}, \forall i, j, k$. Likewise, according to the asking strategies as described in Eqs. (17) and (18), the ask submitted by a seller is no less than its true cost of providing the VM, i.e., $\text{Ask}_{kj} \geq c_{kj}, \forall k, j$. Now assuming that a VM of service j from ES k is successfully assigned to ES i, from (31d) we know that $\text{Bid}_{ijk} \geq \text{Ask}_{kj}$. These inequalities together with Eq. (27) (or Eq. (28) and (29)) implies that $\text{Pay}_i^k < \text{Bid}_{ijk} \leq v_{ijk}$ and $\text{Rec}_k^i > \text{Ask}_{kj} \geq c_{kj}$, which is exactly the individual rationality for the sellers and buyers by definition. □

Theorem 3.3. *The proposed mechanism is budget balanced.*

Proof. The property of budget balance is obvious since from Eq. (27) (or Eqs. (28) and (29)), we can see that the payment made by each virtual buyer equals to the price received by the corresponding virtual seller plus the gain of the third-party platform, i.e., there is no deficit in the market. □

Remarks. In this work, as our goal is to maximize social welfare, we do not guarantee truthfulness as most auction mechanisms pursue. In fact, researchers have shown that there is a conflict between the two goals of truthfulness and system efficiency maximization [21], i.e., although truthfulness makes the auction mechanism design simpler, it may degrade system performance considerably at the same time. In our mechanism, we use markups to capture the dishonest behaviors of both the buyers and sellers, which makes it more practical. Meanwhile, from the economic point of view, our mechanism also leads to a faster growth of the economy as the higher social welfare, the more efficient of the market as more ESs will participate [22].

4. A price-based decentralized allocation mechanism

In this section, we present a price-based decentralized algorithm to perform resource allocation for the edge computing network, as the 2-stage algorithm proposed in the above section requires a third-party platform as a central coordinator. The third-party platform is assumed to be trusted and fair, which limits the applicability of the proposed auction mechanism. On the other hand, decentralized mechanisms are able to achieve superior performance through nodes collaboration and local computation, and therefore they are more preferred.

4.1. Decentralized algorithm

Our decentralized mechanism is based on the lagrangian dual-based decomposition theory. The core idea is to decompose the resource allocation problem into multiple sub-problems, each one for an individual ES or service that can be locally solved. Since there is no central authority, for an optimal allocation the two parties interact with each other through exchanging some information that can be locally derived by each party, but which is also needed by the other party for solving its sub-problem. This process of local computation and information exchange iterates until convergence.

More specifically, our decentralized mechanism also comprises of two stages, where the first one is exactly the same as that of the double-auction based mechanism, i.e., each ES preferentially dedicates its resources to local workload by solving problem (12). The remaining resources are allocated by the price-based decentralized algorithm. Let $\boldsymbol{x_{ij}} = (x_{1ji}, x_{2ji}, \dots, x_{Mji})$ denote the resource allocation of service ij (service j at ES i for short), and $\boldsymbol{x} = (\boldsymbol{x_{11}}, \boldsymbol{x_{12}}, \dots, \boldsymbol{x_{MN}})$. Also denote by $\mathcal{X}_{ij} = \{\boldsymbol{x}_{ij} | \mathbf{1}^T \cdot \boldsymbol{x_{ij}} \leq D_{ij}, \boldsymbol{x_{ij}} \geq \mathbf{0}\}$, and $\mathcal{X} = \prod_{ij} \mathcal{X}_{ij}$. We assume that the amount of the remaining resources for allocation at each ES is much larger than that required by a VM, i.e., $R_{is}^{\text{Left}} \gg r_{js}, \forall i, j, s$, so that x_{kji}'s can be regarded as real numbers as the rounding error will be negligible. The resource allocation task at the second stage then can be reformulated as the following optimization problem:

$$\underset{\boldsymbol{x}}{\text{Maximize:}} \sum_i \sum_j \sum_{k \in \mathcal{I}(i)} x_{kji}(v_{ijk} - c_{kj}) \tag{34a}$$

$$\text{s.t.} \quad \boldsymbol{x} \in \mathcal{X} \tag{34b}$$

$$\sum_i \sum_j x_{kji} \cdot r_{js} \leq R_{ks}^{\text{Left}}, \quad \forall k \in \mathcal{I}(i), s \tag{34c}$$

Note that problem (34) is a linear program and hence the canonical dual-based decomposition method for the NUM framework cannot be applied, i.e., the utility function is not strictly concave [23].

To model the resource allocation task with the standard NUM framework and leverage the existing technique, our solution is to add a concave term to the objective function of problem (34), so that: (1) the problem under study becomes convex; and (2) the gap between the solution of the new problem and that of the original one can be bounded. Here we choose the term as $\sum_i \sum_j \sum_{k \in \mathcal{I}(i)} (\log(x_{kji} + 1) - x_{kji})/M^2$, which satisfy the above two requirements (the proof is given in Section 4.2). Now let us consider the following problem:

$$\underset{\boldsymbol{x}}{\text{Max:}} \sum_{i,j} \sum_{k \in \mathcal{I}(i)} \left(x_{kji}(v_{ijk} - c_{kj}) + \frac{\log(x_{kji} + 1) - x_{kji}}{M^2} \right) \tag{35a}$$

$$\text{s.t.} \quad \text{(34b), (34c)} \tag{35b}$$

Let $\alpha_{ks} \geq 0$ and $\boldsymbol{\alpha} = (\alpha_{11}, \alpha_{12}, \dots, \alpha_{M4})$, where α_{ks} is the Lagrangian multiplier for the constraint (34c) and can be physically interpreted as the price of resource s at ES k. Define Lagrangian:

$$\begin{aligned} L(\boldsymbol{x}, \boldsymbol{\alpha}) &= \sum_{i,j,k \in \mathcal{I}(i)} \left(x_{kji}(v_{ijk} - c_{kj}) + \frac{\log(x_{kji} + 1) - x_{kji}}{M^2} \right) \\ &\quad + \sum_{k,s} \alpha_{ks} (R_{ks}^{\text{Left}} - \sum_{i,j} x_{kji} \cdot r_{js}) \\ &= \sum_{i,j,k} \left(x_{kji}(v_{ijk} - c_{kj} - \frac{1}{M^2} - \sum_s \alpha_{ks} r_{js}) + \frac{\log(x_{kji} + 1)}{M^2} \right) \\ &\quad + \sum_{k,s} \alpha_{ks} R_{ks}^{\text{Left}} \end{aligned} \tag{36}$$

The key to the decomposition of problem (35) is to examine its dual. With Lagrangian, the dual objective can be written as:

$$g(\boldsymbol{\alpha}) = \underset{\boldsymbol{x} \in \mathcal{X}}{\text{Maximize:}} \; L(\boldsymbol{x}, \boldsymbol{\alpha}) \tag{37}$$

and the dual problem is to find:

$$\underset{\boldsymbol{\alpha} \geq \mathbf{0}}{\text{Minimize:}} \; g(\boldsymbol{\alpha}) \tag{38}$$

Note that problem (38) is convex even when problem (35) is not.

Let $\boldsymbol{U}_{ij} = \sum_k (x_{kji}(v_{ijk} - c_{kj} - \frac{1}{M^2} - \sum_s \alpha_{ks} r_{js}) + \frac{\log(x_{kji}+1)}{M^2})$. As the first term in the RHS of the last equation in (36) is separable, we have: $\underset{\boldsymbol{x} \in \mathcal{X}}{\text{Max}} \sum_{i,j} \boldsymbol{U}_{ij} = \sum_{i,j} \underset{\boldsymbol{x}_{ij} \in \mathcal{X}_{ij}}{\text{Max}} \boldsymbol{U}_{ij}$. According to [24], we can now readily formulate an optimization problem for each service ij as follows:

Service ij's problem:

$$\underset{\boldsymbol{x}_{ij}}{\text{Maximize:}} \; \boldsymbol{U}_{ij} \tag{39a}$$

$$\text{s.t.} \quad \boldsymbol{x}_{ij} \in \mathcal{X}_{ij} \tag{39b}$$

Theorem 4.1. *The edge resource allocation problem (39) has a unique optimal solution.*

Proof. Given $\boldsymbol{\alpha}$, we can see that $\boldsymbol{U}_{ij}$ is a concave function in $\boldsymbol{x}_{ij}$. Now by its definition, $\mathcal{X}_{ij}$ is an affine set. It follows that problem (39) is a convex optimization program and hence a unique maximizer, called optimal solution, exists. □

Algorithm 1 Decentralized algorithm for edge resource allocation.

```
# For each service ij:
Start with an initial resource demand x_ij^0.
for t = 0, 1, 2, ... do
    Collect from ESs the price of resources α^t.
    Obtain x_ij^{t+1} through solving problem (39).
    Broadcast x_ij^{t+1} to the corresponding ESs.
# For each ES k:
Start with an initial price α_k^0.
for t = 0, 1, 2, ... do
    Collect demand from ESs/services x^t.
    Update α_k^{t+1} according to Eq. (40).
    Broadcast α_k^{t+1} to the corresponding ESs/services.
```

Once each ES k receives resource requirements from services, its price can be updated through a projected sub-gradient approach, as follows:

$$\alpha_{ks}^{t+1} = [\alpha_{ks}^{t} + \gamma(\sum_{i,j} x_{kji}^{t} \cdot r_{js} - R_{ks}^{\text{Left}})]^{+} \tag{40}$$

where $\gamma > 0$ is a step size and t denotes time. Eq. (40) implies that when the services require too much resources, i.e., $\sum_{i,j} x_{kji}^{t} \cdot r_{js} > R_{ks}^{\text{Left}}$, each ES will increase its price of resources so that the demand from services will fall (recall that $\boldsymbol{U}_{ij}$ is a decreasing function in α_{ks}). On the other hand, when there are extra resources, i.e., $\sum_{i,j} x_{kji}^{t} \cdot r_{js} < R_{ks}^{\text{Left}}$, the price of resources at each ES will be decreased so that these resources can be consumed by services.

In short, in the mechanism each ES locally calculates the price of its resources based on the amount of resources required by each service, and then broadcasts the price to the services. On the other hand, each service locally calculates the amount of resources it demands, based on the prices published by ESs. Each service then broadcasts its requirement of resources back to the ESs. This process iterates until convergence, as depicted in Alg. 1.

Let $\{x_{kji}^{\#}\}$ and $\{\alpha_{ks}^{\#}\}$ be the optimal solution to problem (38) and (39) respectively, i.e., when Alg. 1 converges. The price that service ij pays to ES k for using its resources, denoted as $\text{Prc}_{ij\to k}$, can be given as:

$$\text{Prc}_{ij\to k} = x_{kji}^{\#} \cdot (c_{kj} + \sum_{s} \alpha_{ks}^{\#} r_{js}) \tag{41}$$

Correspondingly, the price that ES i pays to ES k, denoted as $\text{Prc}_{i\to k}$, is:

$$\text{Prc}_{i\to k} = \sum_{j} \text{Prc}_{ij\to k} = \sum_{j} x_{kji}^{\#} \cdot (c_{kj} + \sum_{s} \alpha_{ks}^{\#} r_{js}) \tag{42}$$

Remark. (1) From Eq. (42) we can see that the price paid by each ES consists of two parts. The first one is for the cost of the VMs provided by other ESs, and the second one is for the resources consumed by these VMs. This is unlike the traditional price-based dual decomposition approach where the price paid by each party is exactly for the resources consumed, i.e., the second part; (2) To solve problem (39), it is required that each service ij is aware of the cost of providing VMs at each ES, i.e., $\{c_{kj}\}$. We argue that this information is publicly available, i.e., in the cloud-computing market the price of renting a VM at platforms such as Amazon Web Services (AWS) or Microsoft Azure is posted on their websites. Alternatively, one can also obtain this information through querying ESs during resource allocation; (3) Although problem (39) is for each service, in practice it is solved by the corresponding ES as we are allocating resources and stimulating collaboration among ESs.

4.2. Mechanism analysis

Theorem 4.2. *$\{x_{kji}^{\#}\}$ is the optimal solution to problem (35).*

Proof. By definition, $\mathcal{X}$ is a product of the affine sets $\{\mathcal{X}_{ij}\}$ and hence it is convex. Since constraint (34c) is linear, the feasible set of problem (35) is convex. Therefore, problem (35) is a convex optimization problem and hence the duality gap between the primal solution and the dual solution is zero, i.e., $\{x_{kji}^{\#}\}$ solves problem (35). □

Theorem 4.3. *The proposed mechanism guarantees individual rationality for all ESs and services.*

Proof. When $x > 0$, we have $\log(x+1) - x < 0$, and when $x = 0$, $\log(x+1) - x = 0$. Now from problem (39) we can see that for each service ij with $x_{kji}^{\#} > 0$, we have $x_{kji}^{\#}(v_{ijk} - c_{kj}) > 0$, since $(\log(x_{kji}^{\#}+1) - x_{kji}^{\#})/M^2 - \sum_s x_{kji}^{\#}\alpha_{ks}r_{js} < 0$. Otherwise, if $x_{kji}^{\#}(v_{ijk} - c_{kj}) < 0$, then we can set $x_{kji}^{\#} = 0$ to improve the utility, which tells that $x_{kji}^{\#} > 0$ is not the optimal solution. This means service ij will request resources from ES k only when $v_{ijk} > c_{kj}$, i.e., when the payoff from task offloading is greater than the cost for the resource, which implies individual rationality for the service. Likewise, for each ES k with $x_{kji}^{\#} > 0$, according to Eq. (42) we have $\text{Prc}_{i\to k} > \sum_j x_{kji}^{\#} \cdot c_{kj}$, i.e., ES k is paid more than the cost of the resources it provides, which, by definition, implies individual rationality for k. □

Theorem 4.4. *Let f and $\mathcal{F}$ be the objective function of problem (34) and problem (35), respectively. Also denote by $\boldsymbol{x}^*$ be the optimal solution to (34), and $D_{max} = \max\{D_{ij}\}$. Then we have $f(\boldsymbol{x}^*) - f(\boldsymbol{x}^{\#}) \le N D_{max}$.*

Proof. It is clear that for all $x_{kji} \in [0, D_{ij}]$, we have $0 \ge \sum_{ijk}(\log(x_{kji}+1) - x_{kji})/M^2 \ge \sum_{ijk}(\log(D_{ij}+1) - D_{ij})/M^2 \ge -\frac{1}{M^2}\sum_{ijk} D_{max} \ge -N D_{max}$, which implies that $f(\boldsymbol{x}^*) - \mathcal{F}(\boldsymbol{x}^*) \le N D_{max}$. Now since $\mathcal{F}(\boldsymbol{x}^{\#}) \ge \mathcal{F}(\boldsymbol{x}^*)$, $f(\boldsymbol{x}^{\#}) \ge \mathcal{F}(\boldsymbol{x}^{\#})$, it follows that $f(\boldsymbol{x}^*) - f(\boldsymbol{x}^{\#}) \le N D_{max}$. □

5. Performance evaluation

In this section, we perform numerical studies to evaluate the performance of our proposed mechanisms. To make it more practical, we use real-world dataset from today's cloud-computing market. In particular, the resource configuration of VMs and their prices are drawn from Alibaba Cloud [14].

5.1. Evaluation setup

We consider an edge-computing network with 6 services, i.e., $N = 6$. The resource configuration of each VM of these services are characterized as follows: *General Purpose Instance* (2 GHz, 8 GB, 40 GB, 10 Gbps), *Computation Type* (2 GHz, 4 GB, 20 GB, 10 Gbps), *Memory Instance* (2 GHz, 16 GB, 40 GB, 10 Gbps), *Computation-Optimized* (2 GHz, 2 GB, 20 GB, 1 Gbps), *Big Data Instance* (24 GHz, 88 GB, 11100 GB, 12 Gbps) and *Local SSD Instance* (2 GHz, 16 GB, 447 GB, 15 Gbps). Note that these VMs are of different types and have diverse resource requirements. The price of renting each of these VMs on Alibaba Cloud is $\boldsymbol{c} = (c_j) = (251.66, 195.7, 334.19, 120.15, 3300.0, 459.5)$. We use these prices to set the cost of hosting a VM at each ES and the payoff from having the corresponding workload processed at each ES. More specifically, the cost of hosting a VM of service j at each ES k, c_{kj}, is uniformly and randomly (u.a.r) drawn from $[c_j - \delta_j, c_j + \delta_j]$, where $\boldsymbol{\delta} = (\delta_j) = (40, 30, 50, 20, 200, 70)$. Likewise, the payoff from having workload of each service locally processed, v_{iji}, is u.a.r drawn from $[2c_j, 3c_j]$. The payoff from offloading workload to other ESs, v_{ijk}, is set according to the following rule: at probability 0.8 it is u.a.r drawn from $[2c_j, v_{iji}]$, and at probability 0.2 drawn from $[0.9c_{ij}, 1.2c_{ij}]$. This models the scenario that it is not always profitable for each ES

providing services to other ESs. Moreover, we configure the resources at each ES as follows: the CPU frequency (GHz) at each ES is set as $128 + 2^{\text{randint}(1,\ 6)}$, the memory capacity (GB) as $256 + 2^{\text{randint}(1,\ 8)}$, the storage capacity (GB) as $20000 + 200 \times \text{randint}(1, 100)$, and the network bandwidth (Gbps) is u.a.r drawn from [201, 300]. The workload of each service j at each ES i, λ_{ij}, is u.a.r drawn from [0, 20] at probability 1/2, and u.a.r drawn from [0, 2] at probability 1/2. This results in a skewed workload distribution.

5.2. Benchmarks

We adopt the following five benchmarks for a comprehensive performance comparison:

(1) **Random Allocation**: This is the mechanism that randomly allocates the spare resources of each ES to services. The allocation of VMs among ESs is performed according to the second stage of the proposed auction-based mechanism.

(2) **Without Incentive**: This mechanism dedicates resources of each ES solely to its local workload. In other words, there is no collaboration among ESs even when some ESs have spare resources.

(3) **Optimal SocialWelfare**: This mechanism allocates resources of ESs in a way such that the social welfare is maximized, i.e., through solving problem (11).

(4) **TASC** [25]: This is also an auction-based mechanism that originally proposed for cooperative communications. It was proven to have individual rationality, budget balance and truthfulness properties, but may sacrifice system efficiency.

(5) **MDA** [26]: Proposed for E-Markets, this canonical multi-unit double auction mechanism has properties of individual rationality, budget balance and truthfulness, while at the same time guarantees high efficiency.

Note that for benchmarks (1), (2) and (5), resource allocations are determined optimally by running our proposed algorithm, i.e., only when the first stage of the proposed mechanism completes do we know the remaining resources for allocation at each ES. For benchmark (3), we adopt the PuLP solver to address problem (11), although it takes long time to get the solution. Unless otherwise specified, for our auction-based mechanism we set the parameter for regulating the profit of the third-party platform as $\alpha = 0.1$.

5.3. Numerical results

5.3.1. Performance of two-stage incentive mechanism

We first investigate the performance of our two-stage double auction based resource allocation mechanism. Fig. 5(a) gives the social welfare of different algorithms when the number of ESs is 5, i.e., $M = 5$. From the figure, we can see that our proposed mechanism is able to achieve as much as 96% the optimal social welfare, which is much better than Random Allocation (74%) and Without Incentive (65%). Moreover, it also outperforms TASC by more than 32% where the third-party platform seeks no profit in the auction, and MDA by 5.5%. Furthermore, it is surprising to observe that in this case study the performance of Random Allocation is even better than Without Incentive and TASC, which strongly suggests collaboration among ESs for improved social welfare.

To see how each ES performs under the six mechanisms, we trace their revenue through resource allocation and the result is illustrated in Fig. 5(b). Obviously we can see that although Optimal SocialWelfare attains the maximum system performance, it does lead to negative revenue for some ESs, i.e., ES 1 and ES 3. This tells us the fact that in order to obtain an optimal overall system performance, Optimal SocialWelfare may sacrifice individual interests. On the other hand, our proposed mechanism guarantees individual rationality for all ESs, i.e., each ES gains profit from collaboration. Moreover, as compared with the four counterparts, our mechanism is able to achieve a higher revenue for each ES.

Fig. 6 depicts the resource utilization efficiency of different mechanisms. Here, we show the usage of all the four types of resources at each ES. We can see that: (1) for some type of resources, Without Incentive leads to considerable waste at ESs, i.e., the CPU usage at ES 1 and ES 3, the memory usage at ES 1 and ES 3; (2) Out of our expectation, Optimal SocialWelfare does not provide the highest usage at every ES, i.e., the CPU usage at ES 2 is lower than that of the other mechanisms; and (3) In average, our proposed two-stage mechanism provides satisfactory resource utilization efficiency at all ESs.

We also investigate the performance of our mechanism when the third-party platform seeks more profit in the auction, i.e., $\alpha = 0.3$, and the result is depicted in Fig. 7. Again we find that our mechanism outperforms baselines except Optimal SocialWelfare, and it is able to achieve a high revenue for all ESs.

Figs. 8 and 9 give the performance of the concerned mechanisms when there are more ESs in the network, i.e., $M = 10$. We observe similar trends, i.e., as compared with other baselines, our proposed mechanism provides an optimized social welfare, and it ensures individual rationality for all ESs while at the same time maintains a high resource utilization efficiency.

5.3.2. Performance of price-based decentralized mechanism

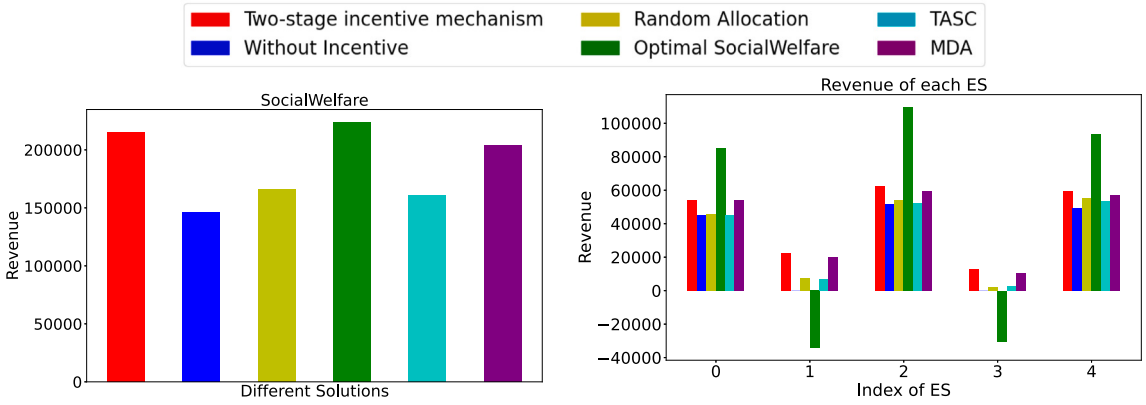
To investigate the performance of our price-based decentralized algorithm, we compare it with the double-auction based mechanism, and the results are shown in Figs. 10 to 13. From these figures, we can see that our price-based allocation mechanism achieves comparable social welfare with the double-auction based mechanism, and it provides individual rationality for all ESs. Meanwhile, it is interesting to observe that the price-based mechanism even provides more revenue for some ESs, i.e., ES 2 as shown in Fig. 10(b) and ES 7 in Fig. 12(b). Moreover, from Figs. 11 and 13, it can be seen that for some type of resources, our price-based allocation mechanism consumes less resources, which from the viewpoint of economy, implies that it is more efficient than the double-auction based mechanism. In fact, according to Theorem 4.4, we can see that the cost of decentralization, as compared with the centralized solution, can be made arbitrarily small as long as we adopt a larger constant factor in the added concave term. For example, we can choose the term as $\sum_i \sum_j \sum_{k \in \mathcal{I}(i)} \frac{\log(x_{kji}+1) - x_{kji}}{KM^2}$ and let $K \to +\infty$.

Figs. 14 and 15 show the convergence behavior of the price-based decentralized mechanism, where Figs. 14(a) and 15(a) give the social welfare of the system as time elapses (the solution may violate the resource constraints). We can see that the maximum social welfare is obtained at the very beginning, i.e., when the price vector $\boldsymbol{\alpha}$ is initialized to be $\mathbf{0}$ and all the services require resources as much as possible. According to the price update rule as given in Eq. (40), each ES then increases its price and the resource demand then decreases. This results in a decrease of social welfare. The system eventually fluctuates within a given interval, as shown in Fig. 14(a). Note that this behavior is in accordance with the sub-gradient based approach where only the optimal social welfare converges if we keep track of the best solution found, as shown in Figs. 14(b) and 15(b) where the optimal valid social welfare of the system is depicted (with no resource constraints violated). Obviously, we can see that the optimal social welfare is quickly found and the system converges then.

6. Discussions and related work

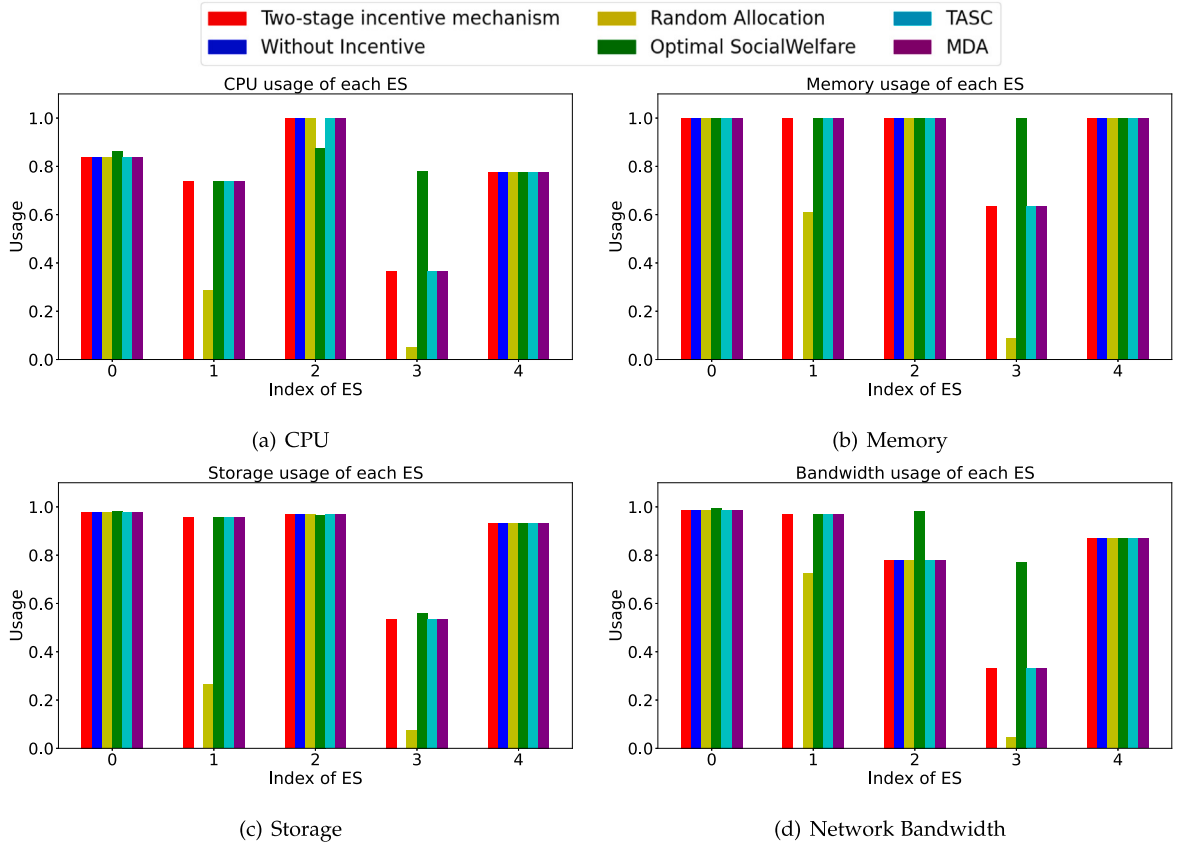
In this section, we explore the implications of our mechanisms and present some future research directions. We end with a brief discussion of the related work.

(1) Performance assurance for offloaded tasks. When offloading tasks to other edge servers, it is crucial to guarantee that the performance requirements of the offloaded tasks can be satisfied. Recall that in our model we use two sets $\mathcal{N}(i)$ and $\mathcal{I}(i)$ to represent the QoS requirements of services, which respectively denote the set of ESs that can offload tasks to ES i and the set of ESs to which i can offload its



(a) Social Welfare (b) Revenue

Fig. 5. Social welfare and revenue of different mechanisms: $M = 5$ and $\alpha = 0.1$.

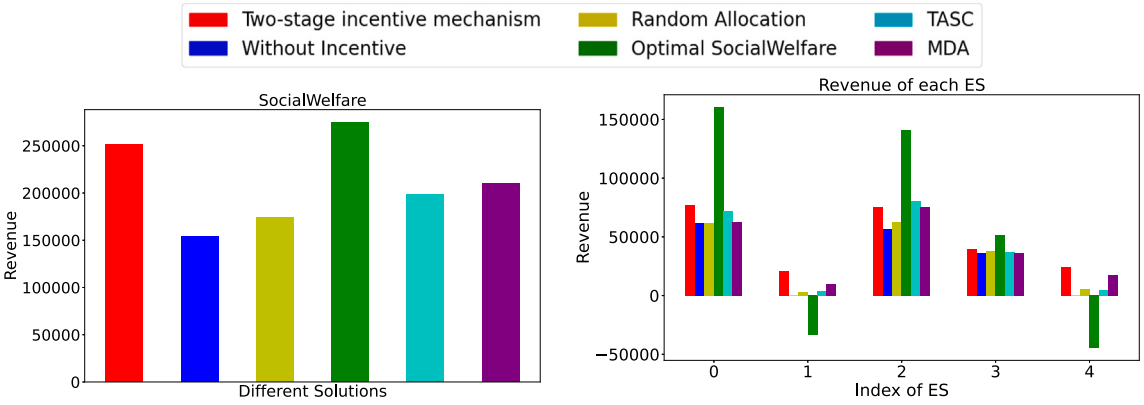


(a) CPU (b) Memory

(c) Storage (d) Network Bandwidth

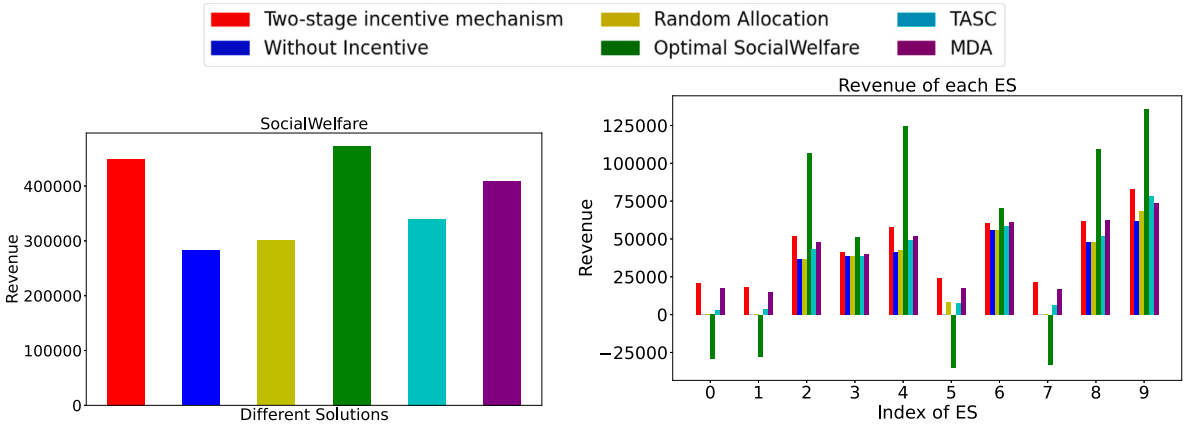
Fig. 6. Resource efficiency of different mechanisms: $M = 5$ and $\alpha = 0.1$.

tasks, both without QoS constraints. The merit of this abstraction is that it allows us to incorporate various QoS requirements. For example, if the transmission delay between ES i and ES k is too large or the network bandwidth between them is too small, then in either case ES k will not be a member of $\mathcal{N}(i)$, i.e., $k \notin \mathcal{N}(i)$. Another advantage of this abstraction is that it can be easily extended to handle the fine-grained QoS requirements of individual applications. For example, for delay-sensitive applications, since the transmission delay between end-users and their local ES is much smaller than that between different ESs, the transmission delay of a task, if offloaded to other ES, is mainly due to the delay between ESs. This implies that when offloading tasks to other ESs, we can determine whether the delay requirement can be satisfied based on the delay between ESs. More specifically, if different applications have diverse delay requirements, then we can define the sets $\mathcal{N}(ij)$ and $\mathcal{I}(ij)$ for each service j, which respectively denote the set of ESs that can offload tasks of j to ES i and the set of ESs to which i can offload tasks of j, without violating j's delay requirement. In this way, we can guarantee that the delay requirement of individual applications can be satisfied when their tasks are offloaded to other edge servers. Of course, in order to achieve this, we need systems that can be leveraged to monitor and provision resources of the underlying MEC network [27]. Some virtualization and service-oriented solutions such as network function virtualization (NFV) [28] by ETSI ISG, IETF service function chaining [29] (SFC), 3GPP/5G-pp 5G Network Slicing [30], can be adopted to provide such functions and support.



(a) Social Welfare (b) Revenue

Fig. 7. Social welfare and revenue of different mechanisms: $M = 5$ and $\alpha = 0.3$.



(a) Social Welfare (b) Revenue

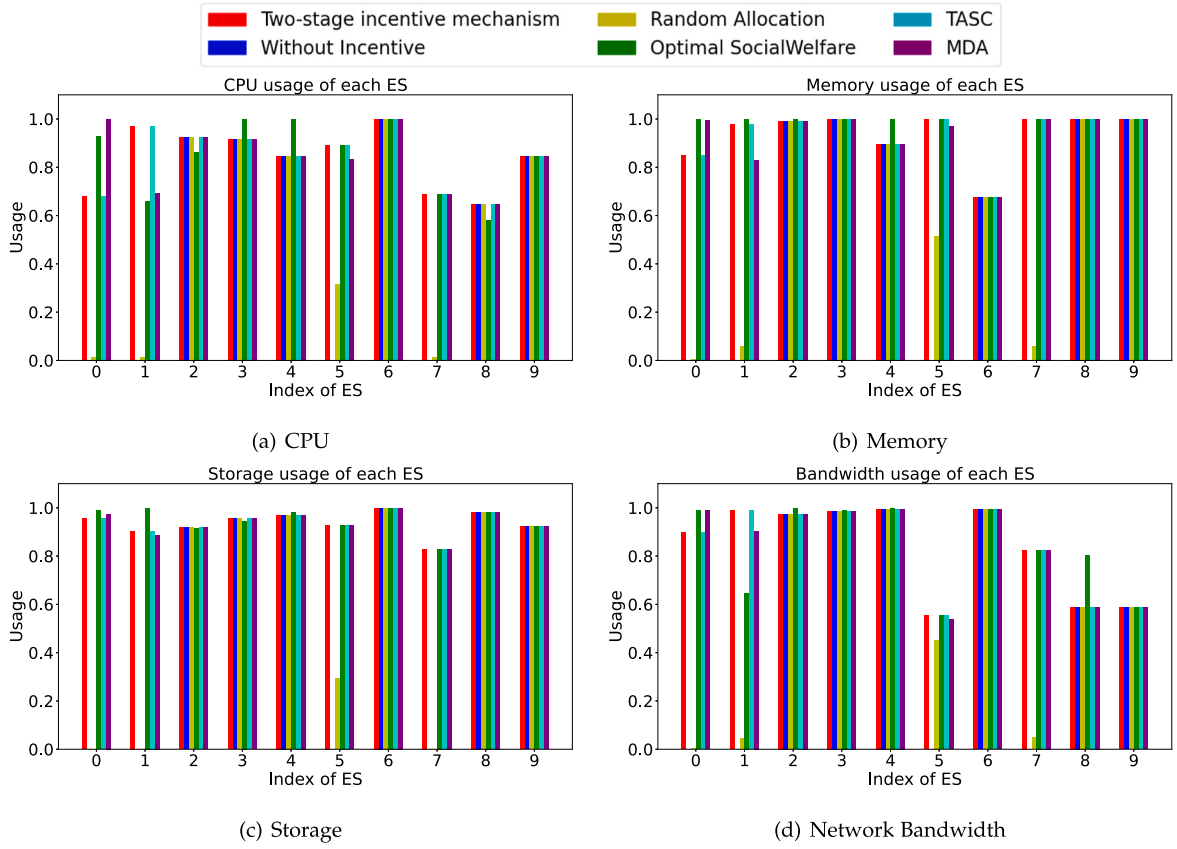
Fig. 8. Social welfare and revenue of different mechanisms: $M = 10$ and $\alpha = 0.1$.

(2) Multiple ESs for each SP. We assume in problem formulation that each SP owns a single ES, which may not always hold in practice. It is thus interesting to see whether our mechanisms still work when each SP owns multiple ESs. We argue that this issue can be addressed by extending our model, i.e., we can regard each ES as a virtual SP, and each SP then comprises of multiple virtual SPs. The revenue of each SP is then the sum of the revenue of the corresponding virtual SPs. The proposed mechanisms, i.e., double-auction based and price-based resource allocation, remain effective.

Related Work The problem of incentive mechanism design [31–33] for multi-access edge computing is essentially to devise a mechanism for system so that profit-driven entities (i.e., mobile devices, edge servers) are willing to share their resources to improve profit and resource efficiency. The profit is usually modeled as utilities in the literature. Accordingly, there are three type of utilities: utility of users, utility of servers, and social welfare.

Utility of users is often defined as the valuation obtained by offloading tasks minus the cost of offloading. The valuation can be simply the economic value from users' perspective [34–36], or modeled as a function of QoE improvement [37,38] such as delay reduction and energy savings. The offloading decision is the key design parameter if we want to optimize this utility. On the other hand, utility of servers is often characterized as the payment received from all users for task offloading minus the cost for processing their tasks. In this context, designing an appropriate pricing scheme is the focus of the research [39]. Typical pricing strategies include uniform pricing [37], differentiated pricing [40] and bid-based pricing [36,38]. The last type of utility is social welfare, which is defined as the sum of utilities of all the participants in system [41]. Since the payment appears in both the utility of users and that of servers, it equals to the valuation from users offloading tasks minus the cost of resources consumed for processing tasks. To maximize this utility, both the offloading/matching policy between users and servers and the pricing strategy should be considered.

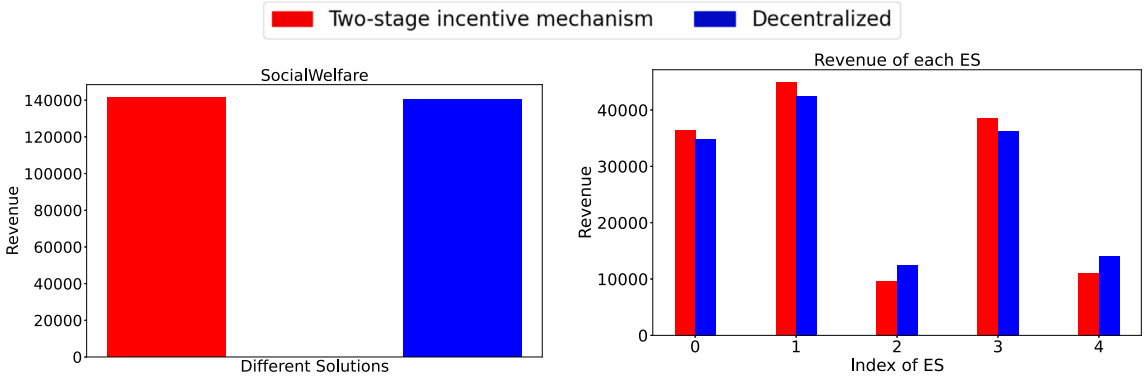
Existing incentive mechanisms for multi-access edge computing can be generally categorized into game-theory-based-mechanisms [37,40,42,43] and auction-based-mechanisms [34,36,38,41]. The game-theory-based-mechanisms can be further divided into the categories of non-cooperative-game-based and cooperative-game-based. For non-cooperative-game-based mechanisms, [37] models the process of users competing edge resources as a non-cooperative game, and tries to maximize the utility of users. [40] establishes a stackelberg game for the system where the resources of an edge server (leader) is shared among multiple mobile devices (followers). In this case, the problem is mainly about how to set the price of resource at the server side, and whether or not to offload tasks to the server at the user side. Unlike non-cooperative-game-based mechanisms, the cooperative-game-based mechanisms [43] usually adopt a coalitional model among users, which can play the role as collaborators for task relaying, execution, etc. Here, the key challenge is how to choose the best collaborator and also determine the task offloading mode.



(a) CPU (b) Memory

(c) Storage (d) Network Bandwidth

Fig. 9. Resource efficiency of different mechanisms: $M = 10$ and $\alpha = 0.1$.



(a) Social Welfare (b) Revenue

Fig. 10. Social welfare and revenue of the two mechanisms: $M = 5$ and $\alpha = 0.1$.

The auction-based-mechanisms establish an auction model between edge servers providing resources and user devices with tasks for potential offloading. Different from game-theory-based-mechanisms, a third-party platform is often needed as an auctioneer for setting the prices of resources and also determining the matching between users and servers. Most often, the goal is to maximize social welfare. For example, [44] studies the problem of allocating VMs of Edge Cloud Nodes (ECNs) to nearby mobile users, and proposes a truthful Auction-based VM Allocation (AVA) mechanism to solve it. [45] considers the scenario that a Cloud Service Centre (CSC) recruits small edge servers to process the offloaded tasks and proposes a reverse-auction based mechanism to solve the task offloading and resource allocation problem. To stimulate edge servers to provide resources to MDs, [46] proposes a truthful and budget-balanced auction mechanism and takes into account locality constraints of MEC system. [47] proposes a breakeven-based double auction (BDA) and a dynamic pricing based double auction (DPDA) mechanism to determine the matched pairs between MDs and ESs, which were proven to meet the desired economic properties. He et al. in [41] develop an auction based online mechanism that stimulates mobile devices offering computing resources to other MDs, and show that it achieves near optimal long-term social welfare as compared to the offline optimum. Among these mechanisms, [34,41] adopt the regular forward auction framework where the bid is offered by the buyers, [38] adopts the reverse auction framework where the bid is offered by the sellers, and [35,36,47] adopt double auction framework where the buyers offer bids and the sellers offer asks.

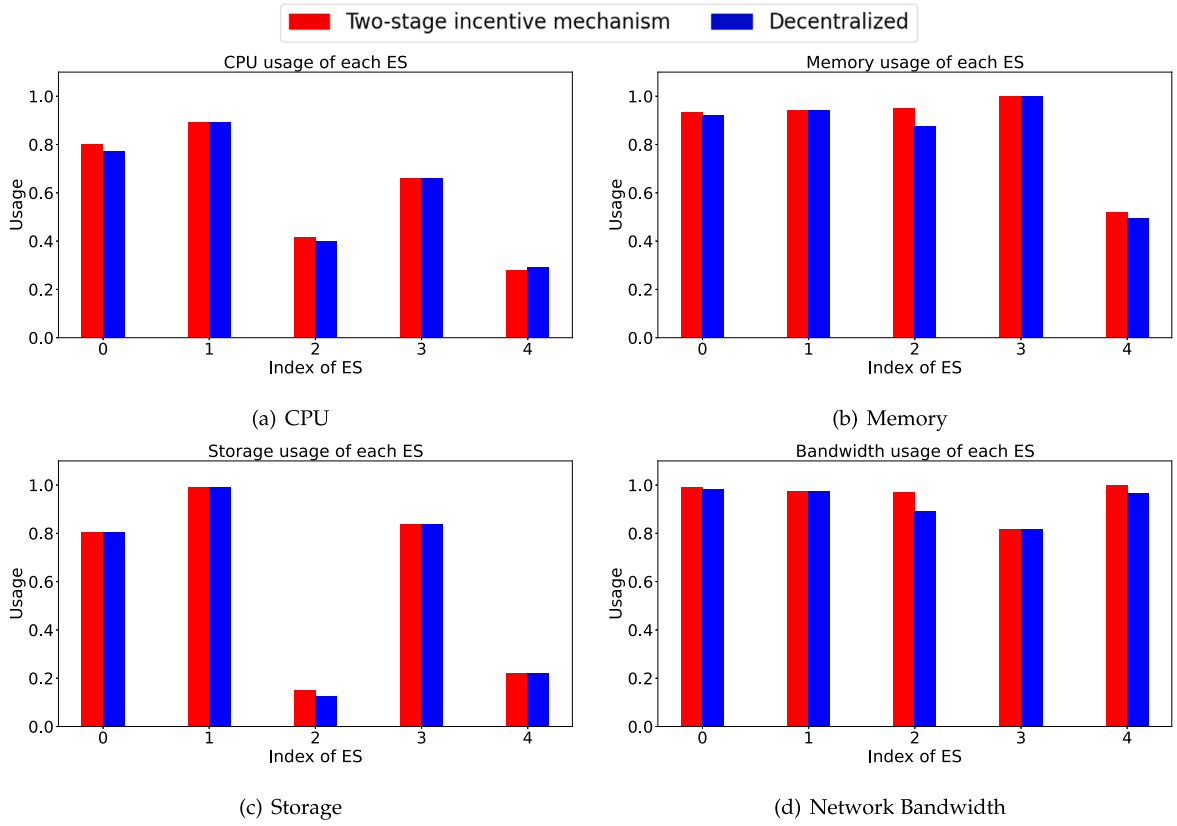


Fig. 11. Resource efficiency of the two mechanisms: $M = 5$ and $\alpha = 0.1$.

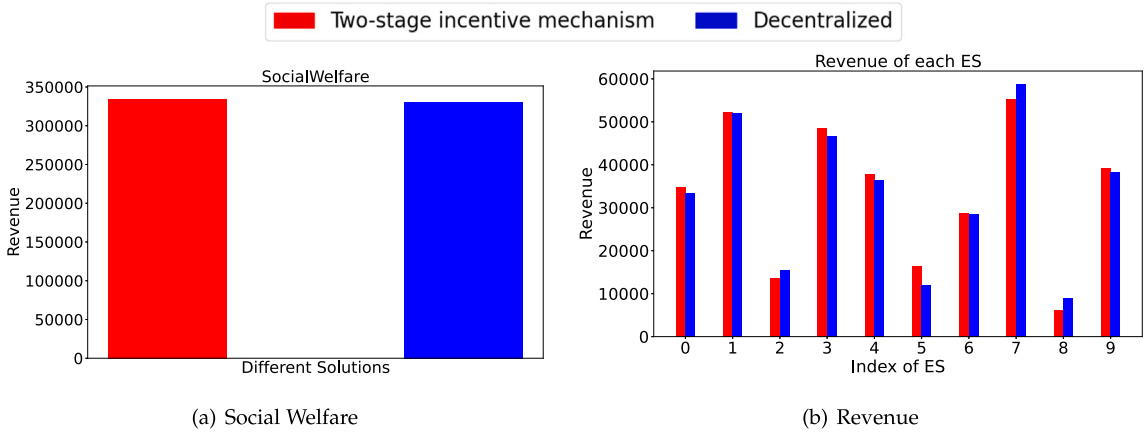
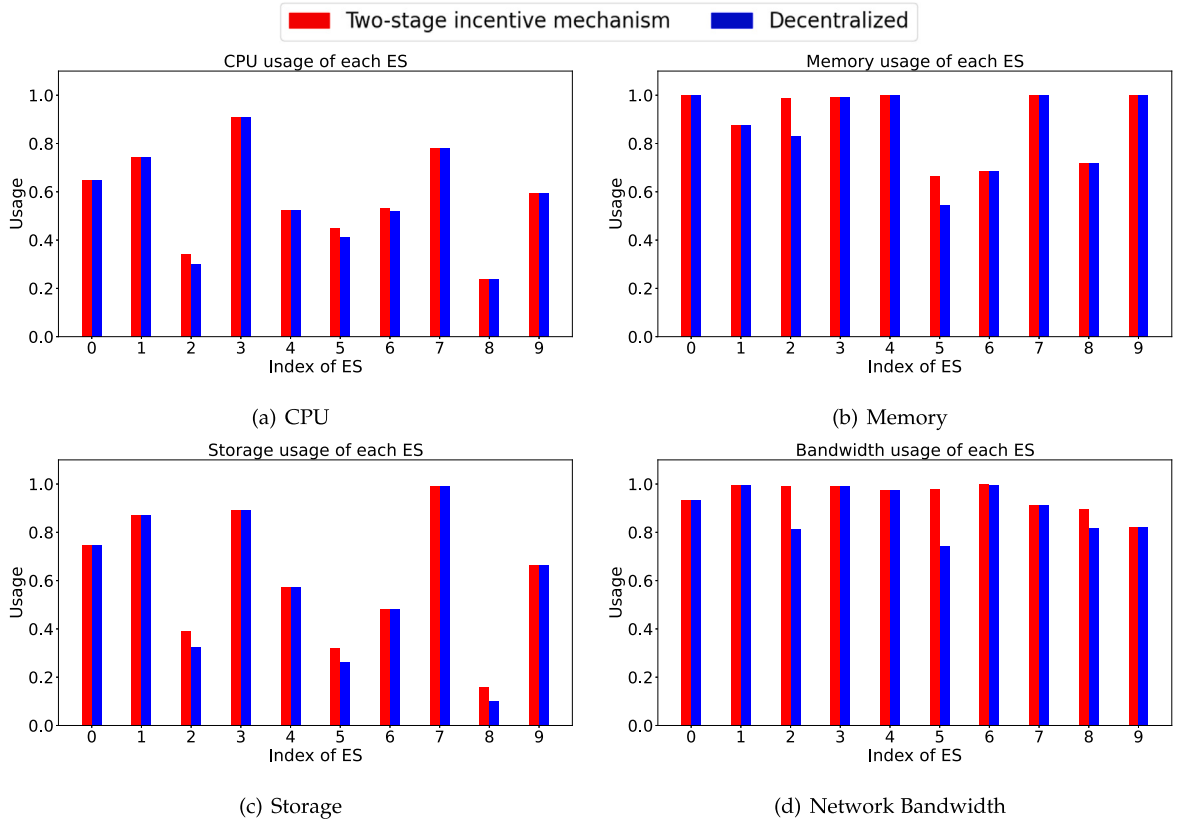


Fig. 12. Social welfare and revenue of the two mechanisms: $M = 10$ and $\alpha = 0.1$.

Existing incentive mechanisms for MEC can also be categorized into offline mechanisms and online mechanisms. The offline mechanisms [48] assume all the information about tasks and resources at edge servers are available before the decision is made. An optimized solution thus can be derived using such global knowledge. On the other hand, in online mechanisms [41,49], tasks from users arrive dynamically and there is no prior knowledge about the resources at edge servers. The system has to make decisions in an online fashion. In general, although offline mechanisms are able to provide much better performance than online mechanisms, the latter is more flexible and suitable in dynamic environment such as a real edge computing network, where the characteristics of tasks and resources can change over time and are difficult to predict.

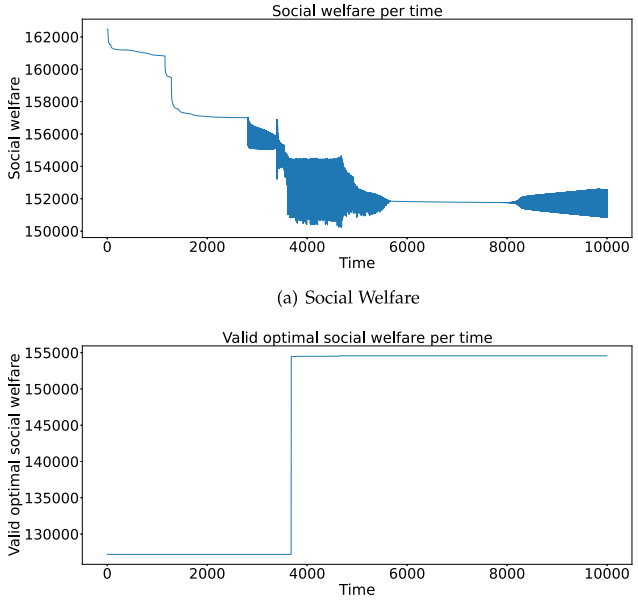
Our work differs from existing ones in that: (1) we propose to incentive resource allocation and task offloading between edge servers, whereas most existing work focuses on allocating edge resources to MDs (or end-users), or offloading tasks between the edge and cloud. In other words, there is no cooperation between ESs. In addition, resource allocation in our mechanism is carried out in terms of VMs/containers, whereas most existing work typically adopt a task-based resource allocation scheme; (2) We provide a two-stage resource allocation mechanism, where at the first stage each ES preferentially allocates its resources to local workload. This prevent each ES from allocating its resources fully to other ESs, i.e., when other ESs offer a high reward, at the cost of sacrificing the revenue of its local users, which is obviously not reasonable. Meanwhile, it also significantly reduces the computational complexity of the mechanism. On the contrary, most



(a) CPU (b) Memory

(c) Storage (d) Network Bandwidth

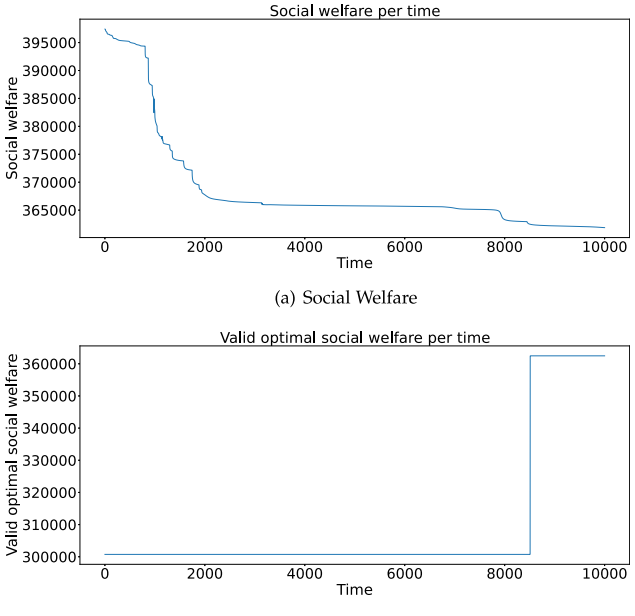
Fig. 13. Resource efficiency of the two mechanisms: $M = 10$ and $\alpha = 0.1$.



(a) Social Welfare

(b) Valid Optimal Social Welfare

Fig. 14. Convergence of the price-based mechanism: $M = 5$ and $\alpha = 0.1$.



(a) Social Welfare

(b) Valid Optimal Social Welfare

Fig. 15. Convergence of the price-based mechanism: $M = 10$ and $\alpha = 0.1$.

existing work does not provide such treatment; and (3) We consider a general multi-server MEC system, where multiple heterogeneous services/applications compete for the multi-server edge resources. Meanwhile, to make our mechanism more practical and computationally cost efficient, we model resource allocation with a multi-commodity multi-unit double auction, and adopt the concept of markups and virtual sellers/buyers. To the best of our knowledge, these ideas have not been adopted in existing work on cooperative edge computing.

7. Conclusion

We study the problem of stimulating resource sharing among different edge servers in a generic edge computing network, and adopt a VM/container based resource allocation scheme. A two-stage market based resource allocation mechanism is proposed, where edge resources at each ES are first allocated to services according to its local workload, and then distributed among other ESs through a multi-unit double auction. We further develop a price-based decentralized mechanism in which resource allocation is performed fully by ESs through their local computation and limited information exchange. We give pricing strategies for the two mechanisms and prove their fundamental properties. The efficacy of our mechanisms are validated via numerical studies.

CRediT authorship contribution statement

Weibo Chu: Writing – original draft, Methodology, Funding acquisition, Formal analysis, Conceptualization. **Xinming Jia:** Validation, Software, Methodology. **Zhiwen Yu:** Supervision, Resources, Project administration, Funding acquisition. **John C.S. Lui:** Writing – review & editing, Supervision, Resources, Methodology. **Yi Lin:** Writing – review & editing, Visualization, Validation, Investigation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Grant No. 62172333) and the Natural Science Basic Research Plan in Shaanxi Province of China (Grant No. 2021JM-073). The work of John C.S. Lui was supported in part by the RGC SRFS2122-4S02.

References

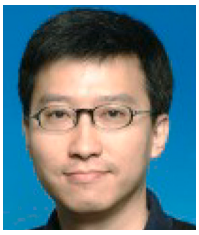
[1] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, D. Sabella, On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration, IEEE Commun. Surv. Tutor. 19 (3) (2017) 1657–1681.
[2] Y. Mao, C. You, J. Zhang, K. Huang, K.B. Letaief, A survey on mobile edge computing: The communication perspective, IEEE Commun. Surv. & Tutor. 19 (4) (2017) 2322–2358.
[3] M. Patel, B. Naughton, C. Chan, N. Sprecher, S. Abeta, A. Neal, et al., Mobile-edge computing introductory technical white paper, 29, 2014, pp. 854–864, White paper, mobile-edge computing (MEC) industry initiative.
[4] A.C. Baktir, C. Sonmez, C. Ersoy, A. Ozgovde, B. Varghese, Addressing the challenges in federating edge resources, in: Fog and Edge Computing: Principles and Paradigms, John Wiley & Sons, Inc., Hoboken, NJ, USA, 2019, pp. 25–49.
[5] C. Gong, F. Lin, X. Gong, Y. Lu, Intelligent cooperative edge computing in internet of things, IEEE Internet Things J. 7 (10) (2020) 9372–9382.
[6] U. Saleem, Y. Liu, S. Jangsher, Y. Li, T. Jiang, Mobility-aware joint task scheduling and resource allocation for cooperative mobile edge computing, IEEE Trans. Wireless Commun. 20 (1) (2020) 360–374.
[7] S. Zhong, S. Guo, H. Yu, Q. Wang, Cooperative service caching and computation offloading in multi-access edge computing, Comput. Netw. 189 (2021) 107916.
[8] Z. Jiang, N. Ling, X. Huang, S. Shi, C. Wu, X. Zhao, Z. Yan, G. Xing, CoEdge: A cooperative edge system for distributed real-time deep learning tasks, in: Proceedings of the 22nd International Conference on Information Processing in Sensor Networks, 2023, pp. 53–66.
[9] Open Edge Computing, http://openedgecomputing.org.
[10] Openfog, https://opcfoundation.org/markets-collaboration/openfog/.
[11] X. Chen, L. Jiao, W. Li, X. Fu, Efficient multi-user computation offloading for mobile-edge cloud computing, IEEE/ACM Trans. Netw. 24 (5) (2015) 2795–2808.
[12] Q. Peng, C. Wu, Y. Xia, Y. Ma, X. Wang, N. Jiang, Dosra: A decentralized approach to online edge task scheduling and resource allocation, IEEE Internet Things J. 9 (6) (2021) 4677–4692.
[13] Markup, https://en.wikipedia.org/wiki/Markup_(business).
[14] Alibaba Cloud: Cloud Computing Services, http://www.alibabacloud.com.
[15] Budget-balanced mechanism, https://en.wikipedia.org/wiki/Budget-balanced_mechanism.
[16] Knapsack problem, https://en.wikipedia.org/wiki/Knapsack_problem.
[17] R.B. Myerson, M.A. Satterthwaite, Efficient mechanisms for bilateral trading, J. Econ. Theory 29 (2) (1983) 265–281.
[18] W. Yong, Y. Li, L. Chao, C. Wang, X. Yang, Double-auction-based optimal user assignment for multisource–multirelay cellular networks, IEEE Trans. Veh. Technol. 64 (6) (2015) 2627–2636.
[19] R. Duan, S. Pettie, Linear-time approximation for maximum weight matching, J. ACM 61 (1) (2014) 1–23.
[20] D.B. West, et al., Introduction to Graph Theory, vol. 2, Prentice hall Upper Saddle River, 2001.
[21] E. Anshelevich, S. Das, Y. Naamad, Anarchy, stability, and utopia: creating better matchings, Auton. Agents Multi-Agent Syst. 26 (1) (2013) 120–140.
[22] W.A. Kamakura, B.T. Ratchford, J. Agrawal, Measuring market efficiency and welfare loss, J. Consum. Res. 15 (3) (1988) 289–302.
[23] J.-W. Lee, R.R. Mazumdar, N.B. Shroff, Non-convex optimization and rate control for multi-class services in the internet, IEEE/ACM Trans. Netw. 13 (4) (2005) 827–840.
[24] D.P. Palomar, M. Chiang, A tutorial on decomposition methods for network utility maximization, IEEE J. Sel. Areas Commun. 24 (8) (2006) 1439–1451.
[25] D. Yang, X. Fang, G. Xue, Truthful auction for cooperative communications, in: Proceedings of the Twelfth ACM International Symposium on Mobile Ad Hoc Networking and Computing, 2011, pp. 1–10.
[26] P. Huang, A. Scheller-Wolf, K. Sycara, Design of a multi–unit double auction e–market, Comput. Intell. 18 (4) (2002) 596–617.
[27] Q. Duan, S. Wang, N. Ansari, Convergence of networking and cloud/edge computing: Status, challenges, and opportunities, IEEE Netw. 34 (6) (2020) 148–155.
[28] Virtualization, Network Functions, Network function virtualization, 2020, NFV_White_Paper3. pdf.
[29] J. Halpern, C. Pignataro, Service Function Chaining (SFC) Architecture, Technical Report, 2015.
[30] S. Zhang, An overview of network slicing for 5G, IEEE Wirel. Commun. 26 (3) (2019) 111–117.
[31] X. Huang, B. Zhang, C. Li, Incentive mechanisms for mobile edge computing: Present and future directions, IEEE Netw. 36 (6) (2022) 199–205.
[32] R. Chattopadhyay, C.-K. Tham, Fully and partially distributed incentive mechanism for a mobile edge computing network, IEEE Trans. Mob. Comput. 21 (1) (2020) 139–153.
[33] X. Pu, T. Lei, W. Wen, W. Feng, Z. Wang, Q. Chen, S. Jin, Incentive mechanism and resource allocation for collaborative task offloading in energy-efficient mobile edge computing, IEEE Trans. Veh. Technol. (2023).
[34] G. Li, J. Cai, An online incentive mechanism for collaborative task offloading in mobile edge computing, IEEE Trans. Wireless Commun. 19 (1) (2019) 624–636.
[35] W. Sun, J. Liu, Y. Yue, P. Wang, Joint resource allocation and incentive design for blockchain-based mobile edge computing, IEEE Trans. Wirel. Commun. 19 (9) (2020) 6050–6064.
[36] W. Lu, S. Zhang, J. Xu, D. Yang, L. Xu, Truthful multi-resource transaction mechanism for P2P task offloading based on edge computing, IEEE Trans. Veh. Technol. 70 (6) (2021) 6122–6135.
[37] L. Li, T.Q. Quek, J. Ren, H.H. Yang, Z. Chen, Y. Zhang, An incentive-aware job offloading control framework for multi-access edge computing, IEEE Trans. Mob. Comput. 20 (1) (2019) 63–75.
[38] Q. Wang, S. Guo, J. Liu, C. Pan, L. Yang, Profit maximization incentive mechanism for resource providers in mobile edge computing, IEEE Trans. Serv. Comput. 15 (1) (2019) 138–149.
[39] B. Liang, R. Fan, H. Hu, Y. Zhang, N. Zhang, A. Anpalagan, Nonlinear pricing based distributed offloading in multi-user mobile edge computing, IEEE Trans. Veh. Technol. 70 (1) (2020) 1077–1082.
[40] Z. Chang, W. Guo, X. Guo, Z. Zhou, T. Ristaniemi, Incentive mechanism for edge-computing-based blockchain, IEEE Trans. Ind. Inform. 16 (11) (2020) 7105–7114.
[41] J. He, D. Zhang, Y. Zhou, Y. Zhang, A truthful online mechanism for collaborative computation offloading in mobile edge computing, IEEE Trans. Ind. Inform. 16 (7) (2019) 4832–4841.

[42] Y. Liu, C. Xu, Y. Zhan, Z. Liu, J. Guan, H. Zhang, Incentive mechanism for computation offloading using edge computing: A stackelberg game approach, Comput. Netw. 129 (2017) 399–409.
[43] S. Luo, X. Chen, Z. Zhou, X. Chen, W. Wu, Incentive-aware micro computing cluster formation for cooperative fog computing, IEEE Trans. Wireless Commun. 19 (4) (2020) 2643–2657.
[44] G. Gao, M. Xiao, J. Wu, H. Huang, S. Wang, G. Chen, Auction-based VM allocation for deadline-sensitive tasks in distributed edge cloud, IEEE Trans. Serv. Comput. (2019).
[45] H. Zhou, T. Wu, X. Chen, S. He, D. Guo, J. Wu, Reverse auction-based computation offloading and resource allocation in mobile cloud-edge computing, IEEE Trans. Mob. Comput. (2022).
[46] L. Ma, X. Wang, X. Wang, L. Wang, Y. Shi, M. Huang, TCDA: Truthful combinatorial double auctions for mobile edge computing in industrial internet of things, IEEE Trans. Mob. Comput. 21 (11) (2021) 4125–4138.
[47] W. Sun, J. Liu, Y. Yue, H. Zhang, Double auction-based resource allocation for mobile edge computing in industrial internet of things, IEEE Trans. Ind. Inform. 14 (10) (2018) 4692–4701.
[48] D.T. Nguyen, L.B. Le, V.K. Bhargava, A market-based framework for multi-resource allocation in fog computing, IEEE/ACM Trans. Netw. 27 (3) (2019) 1151–1164.
[49] G. Li, J. Cai, X. Chen, Z. Su, Nonlinear online incentive mechanism design in edge computing systems with energy budget, IEEE Trans. Mob. Comput. (2022).

Weibo Chu received the B.S. degree in software engineering in 2005 and the Ph.D. degree in control science and engineering in 2013, both from Xi'an Jiaotong University, Xi'an, China. He has participated in various research and development projects on network testing, performance evaluation and troubleshooting, and gained extensive experiences in the development of networked systems for research and engineering purposes. From 2011–2012 he worked as a visiting researcher at Microsoft Research Asia, Beijing. From 2013 he was with the School of Computer Science and Technology, Northwestern Polytechnical University. His research interests include internet measurement and modeling, traffic analysis and performance evaluation.

Xinming Jia received the B.E. degree from Northwestern Polytechnical University, Xi'an, China, in 2021. He is currently working towards his M.S. degree at the School of Computer Science and Technology, Northwestern Polytechnical University, Xi'an, China. His research interests include resource management and incentive mechanisms design for edge computing systems.

Zhiwen Yu received the Ph.D. degree in computer science from Northwestern Polytechnical University, Xi'an, China, in 2005. He is currently a Professor in the School of Computer Science, Northwestern Polytechnical University, Xi'an, China, and a vice-president of Harbin Engineering University, Harbin, China. He was an Alexander Von Humboldt Fellow with Mannheim University, Germany, and a Research Fellow with Kyoto University, Kyoto, Japan. His research interests include ubiquitous computing and mobile computing.



John C.S. Lui received the Ph.D. degree in computer science from UCLA. He is currently a professor in the Department of Computer Science and Engineering at The Chinese University of Hong Kong. His current research interests include communication networks, network/system security (e.g., cloud security, mobile security, etc.), network economics, network sciences (e.g., online social networks, information spreading, etc.), cloud computing, large-scale distributed systems and performance evaluation theory. He serves in the editorial board of IEEE/ACM Transactions on Networking, IEEE Transactions on Computers, IEEE Transactions on Parallel and Distributed Systems, Journal of Performance Evaluation and International Journal of Network Security. He was the chairman of the CSE Department from 2005 to 2011. He received various departmental teaching awards and the CUHK ViceChancellor's Exemplary Teaching Award. He is also a corecipient of the IFIP WG 7.3 Performance 2005 and IEEE/IFIP NOMS 2006 Best Student Paper Awards. He is an elected member of the IFIP WG 7.3, fellow of the ACM, fellow of the IEEE, and croucher senior research fellow.

Yi Lin received the Ph.D. degree in computer science from Northwestern Polytechnical University, Xi'an, China, in 2005. He is currently an Associate Professor of the School of Computer Science, Northwestern Polytechnical University, Xi'an, China. His research interests include data storage and software engineering.