Two-Tier Matching Game in Small Cell Networks for Mobile Edge Computing

Yu Du, Jun Li, *Senior Member, IEEE,* Long Shi, *Member, IEEE,* Tingting Liu, *Member, IEEE,* Feng Shu, *Member, IEEE,* and Zhu Han, *Fellow, IEEE,*

Abstract—Mobile edge computing (MEC) enables computing services at the network edge closer to mobile users (MUs) to reduce network transmission latency and energy consumption. Deploying edge computing servers in small base stations (SBSs), operators make profit by offering MUs with computing services, while MUs purchase services to solve their own computation tasks quickly and energy-efficiently. In this context, it is of particular importance to optimize computing resource allocation and computing service pricing in each SBS, subject to its limited computing and communication resources. To address this issue, we formulate an optimization problem of computing resource management and trading in small-cell networks and tackle this problem using a two-tier matching. Specifically, the first tier targets at the association algorithm between MUs and SBSs to achieve maximum social welfare, and the second tier focuses on the collaboration algorithm among SBSs to make efficient usage of limited computing resources. We further show that the two proposed algorithms contribute to stable matchings and achieve weak Pareto optimality. In particular, we verify that the first algorithm arrives at a competitive equilibrium. Simulation results demonstrate that our proposed algorithms can achieve a better network social welfare than baseline algorithms while retaining a close-optimal performance.

Index Terms—Small-cell network, mobile edge computing, matching game, association, collaboration

1 Introduction

It is predicted that data produced by the internet of things (IoT) will occupy 45% of the network data by 2019. Undoubtedly, the ever-growing loT applications will rely on large computation power and stringent response delay in IoT networks [1]. In particular, if a large number of computation-intensive tasks are processed in mobile user (MU) devices, it is bound to accelerate energy consumption and shorten their service life, which becomes the key driving force behind the invent of cloud computing [2], [3]. However, long-distance communications between MU devices and cloud result in unendurable latency in IoT networks, which cannot meet the requirements of delay-sensitive applications [4]. To handle the nail-biting issue, mobile edge computing (MEC) is emerging as an efficient solution to enable timely and efficient local services at the network edges [5], [6].

In recent years, MEC is an increasingly popular computing paradigm due to low latency, high resource utilization efficiency, high transmission, and easy connection with next-generation wireless communication systems. By leveraging these enticing benefits, MEC can help MUs offload computation to MEC servers in small-cell networks [7]. In [8], the authors proposed a new centralized resource management scheme combining the cloud and MEC for resource allocation in the fiber-wireless access network, jointly considering packet delay, response time efficiency, and gain-offload overhead ratio. As follow-up effort, Ref. [9] proposed a centralized energy efficient computation offloading mechanism that optimizes resource allocation to minimize energy consumption in small cell network, meanwhile taking into account the computing power and computing delay requirements of user devices. It is stressed that aforementioned works in [8], [9] concentrated on the centralized algorithms. However, it is widely accepted that the centralized solutions rely on the coordinator to collect the information from the entire network at the cost of high complexity and signalling, which limits MEC efficiency. Compared with centralized algorithms, distributed algorithms decompose an optimization problem into many small parts and process each part locally in a distributed manner. Among distributed solutions, matching theory is an effective method which is also scalable in MEC networks.

In economic literature, matching game theory is a mathematical framework to form reciprocal relationships over time [10]. According to the general classification, matching theory can be divided into three categories, i.e., one-to-one, many-to-one, and many-to-many matching [11]. Matching theory has been widely studied in wireless communications for resource allocation [12], [13]. Under the framework of MEC, matching theory can also be utilized to solve resource allocation problems. For example, in [14], the authors proposed two separate matching games to achieve caching resource allocation in decentralized edge cache network

- *Y.Du, J. Li, and F. Shu are with the School of Electronic and Optical Engineering, Nanjing University of Science Technology Nanjing 210094, China(e-mail:{yudu, jun. li,shufeng}@njust.edu.cn).*
- *Long Shi is with Science and Math Cluster, Singapore University of Technology and Design, Singapore. E-mail:slong1007@gmail.com.*
- *T. Liu is with the School of Electronic and Optical Engineering, Nanjing University of Science and Technology, Nanjing 210094, China, and also with the School of Communication Engineering, Nanjing Institute of Technology, Nanjing 211167, China (e-mail: liutt@njit.edu.cn).*
- *Z. Han is with the Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77004 USA, and also with the Department of Computer Science and Engineering, Kyung Hee University, Seoul 02447, South Korea (e-mail:zhan2@uh.edu).*

with mobile MUs by using distributed algorithm to implement resource allocation. Later on, [15] proposed a joint distributed optimization framework for all edge nodes, data service operators, and data service subscribers in the MEC network, using stackelberg and matching games to solve the computing resource allocation. In [16], matching algorithm was used to maximize the overall uplink throughput while jointly considering the user correlation, subchannel allocation and power allocation in the cognitive femtocell networks. In [17], the authors studied the three-level duplex matching scheme in the MEC-based cloud radio access networks by applying the Gale-Shapley theory and proposed a multi-stage heuristic to minimize the refusal rate for the task offloading requests. Nevertheless, these matching algorithms in [15]–[17] cannot achieve optimal pricing of resources if there exist competition among MUs and competition between MUs and resource owners. Meanwhile, [15]–[17] lack of price adjustment for resource owners to achieve competitive equilibrium.

In practice, MEC is implemented based on a virtualized platform that leverages recent advancements in network functions virtualization, which enables MEC providers to run independent tasks on a single physical server. Add to that, these tasks can access the underlying physical resources while being isolated from each other [18]. For instance, the edge computing server (ECS) independently compute different tasks through different virtual machines (VMs) in various MEC networks [19]. In [20], a novel approach to MEC for the IoT architecture was proposed with the aid of two types of VMs deployed in the edge node and IoT service provider. Considering the MEC provider's profit and MU's cost, how to allocate the limited number of VMs for efficient task computing remains challenging.

In this paper, we study a two-tier matching game in a small cell network with MEC, where each MU first offloads its tasks to the small base stations (SBSs) with the ECSs through association and then the overloaded SBS transfers its tasks to the underloaded SBSs through collaboration. In particular, the virtualization enables each ECS to provide computing services to multiple MUs by creating multiple VMs. To optimize the network social welfare [21], we propose a two-tier matching algorithm that first focuses on the association problem between MUs and SBSs and then addresses the collaboration problem among SBSs sequentially. In the first algorithm, we optimize the VM allocation of SBSs for task offloading from MUs by treating MUs and SBSs as the players in a many-to-one matching game. In the second algorithm, we optimize the computing resources among SBSs, which is formulated as a many-to-many matching game. Our main contributions are summarized as follows:

- We propose a many-to-one matching game integrated with a bidding mechanism to solve the association problem between MUs and SBSs to maximize the social welfare, which guarantees not only the utility of each MU but also the optimal pricing of each SBS (see Section 4.2).
- We propose a many-to-many matching game to solve the SBS collaboration problem, which saves the hardware cost of ECS (see Section 2.2) and minimizes the total transfer delay in collaboration (see Section 4.3).
- We verify the stability and optimality of the proposed matching algorithms. In particular, we validate that the association algorithm arrives at the competitive equilibrium and analyze the impact of price step on the equilibrium (see Section 5.2).
- We illustrate the impacts of different MU demands (concern more about time delay or energy consumption) on social welfare under different uploading powers of MUs through simulations (see Section 6). When the uploading power of MUs ranges from 0W to 1W, the more the MUs concerned about time delay, the larger the social welfare is. Meanwhile, when the uploading power of MU ranges from 1W to 2W, the more the MUs concerned about energy consumption, the larger the social welfare is.

The rest of this paper is organized as follows. Section 2 describes the system model and utility function. The MU and SBS association problem and SBS allocation problem are formulated in Section 3. Section 4 formulates a two-tier matching according to the proposed two matching algorithms. Section 5 investigates the stability and Pareto optimality of the proposed algorithms. In particular, the competitive equilibrium of the association algorithm is investigated. Section 6 presents the simulation results. Finally, Section 7 concludes this paper.

2 System Model

The MEC system consists of M MUs denoted by $\mathcal{U} = \{U_1, U_2, \cdots, U_M\}$ and N SBSs (owned by an operator) denoted by $\mathcal{A} = \{A_1, A_2, \cdots, A_N\}$. As shown in Fig. 1, each SBS in the center of the small cell [1] depolys the same ECS which can open at most G VMs to serve the MUs over different subcarriers, and a single VM handles only one MU's task. The larger the G is, the higher the ECS costs. This is due to the fact that the VM occupies a portion of hardware resources (CPU and memory) [22]. Suppose that each MU is assigned with a computation-intensive and time-sensitive task beyond its limited computing power. As a result, each MU cannot handle its task without the aid of ECS deployed in the SBS. It is assumed that each MU with the same hardware quality has not only sufficient storage capacity to store its task but also sufficient power to uploading its computing task. This paper assumes that each MU in $\mathcal{U}$ offloads its task at the same time. In this work, each SBS has a set of orthogonal subcarriers that can serve at most K MUs simultaneously. Regarding the relation between G and K, we define three types of SBSs as follows:

- $Underloaded$ SBS: The number of associated MUs with the SBS is less than G.
- $Fully\ loaded$ SBS: The number of associated MUs with the SBS is G.
- $Overloaded$ SBS: The number of associated MUs with the SBS is more than G but no more than K.

1. The small cell network is the network composed of multiple small cells. In each single small cell, there exists an SBS located in the center that serves some MUs. In the MEC-enabled small cell network, each MU can offload its task to any SBS in the network, and the tasks can be transferred from one overloaded SBS in one small cell to another underloaded SBS in a different small cell.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2019.2937777, IEEE Transactions on Services Computing

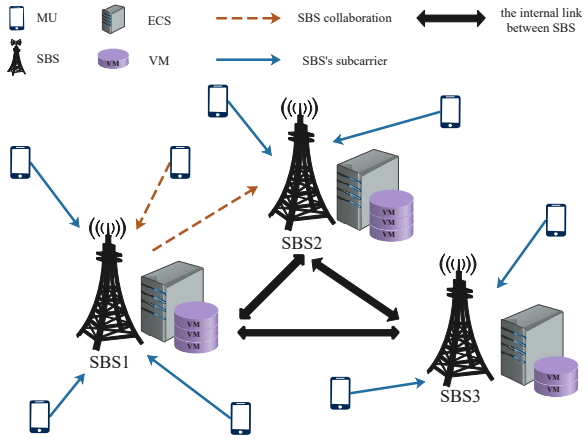


Fig. 1. An MEC network consisting of multiple SBSs each with multiple subcarriers and a single ECS each with multiple VMs, where the overloaded SBS1 with $G = 3$ and $K = 4$ transfers an MU's task to SBS2.

Overall, the MEC focuses on the task offloading from the MUs to SBSs via association and the task computing within the SBSs via collaboration. First, each MU sends a request to all SBSs for the ECS rent. Second, each SBS collects the requests, decides which MUs should be associated with, and leases its ECS to its associated MUs. Third, the overloaded SBS requests the computing resources from the underloaded SBSs[2]. Fourth, each underloaded SBS decides whether to accept the request from the overloaded SBS. Once the request is accepted, the underloaded SBS becomes a helper of the overloaded SBS and the collaboration is established between the two SBSs.

2.1 Association Model

During each period of task offloading, each MU can only be served by a unique VM of a single ECS. Moreover, each SBS has a set of orthogonal subcarriers that can serve at most K MUs simultaneously.

To reduce the cost, each MU wants to connect with the SBS yielding the minimum task offloading consumption. However, limited subcarriers of each SBS does not guarantee that all MUs can connect with the best SBS, especially when the number of MUs is large. This paper considers that each SBS deploys the same ECS. In addition, each ECS can open at most G VMs. Therefore, it is a key issue to find out an cost-efficient solution to the SBS association and computing task allocation.

First, we consider that the $association$ between a pair of MU U_m and SBS A_n is established if U_m is served by A_n. For ease of exposition, we use an $M \times N$ adjacency matrix $\mathbf{X}$ to represent all possible associations between any pair of U_m and A_n, where the $\{m, n\}$-th element of $\mathbf{X}$ is given by

$$x_{m,n} = \begin{cases} 1, & \text{if } U_m \text{ is associated with } A_n, \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

2. We consider that the SBSs distributed in different locations are interconnected with each other through reliable internal links. In this way, they can exchange their offloaded tasks through the shared links.

From (1), U_m uploads its task to A_n if $x_{m,n} = 1$. Then, the uploading rate from U_m to A_n is given by

$$R_{m,n} = W \log_2 \left(1 + \frac{P_m g_{m,n}}{N_0}\right), \quad (2)$$

where W denotes the bandwidth of each uploading link, P_m is the uploading power of U_m, $g_{m,n}$ is the channel gain from U_m to A_n, and N_0 represents the Gaussian noise power. Therefore, the time delay caused by the uploading from U_m to A_n is given by

$$t^{\text{up}}_{m,n} = \frac{Q_{m,n}}{R_{m,n}}, \quad (3)$$

where $Q_{m,n}$ is the data size of the uploaded task from U_m to A_n. From (3), minimizing the uploading time is equivalent to maximizing the transfer rate under fixed data size of each MU task. From (2) and (3), the uploading energy $E^{\text{up}}_{m,n}$ from U_m to A_n is

$$E^{\text{up}}_{m,n} = t^{\text{up}}_{m,n} P_m. \quad (4)$$

The computation in each SBS is characterized by the following key parameters. First, we use f to denote the computing speed of the VM in the ECS of each SBS, namely, the cycles of CPU per second. With reference to [23], the energy consumption of CPU per second is $P = kf^3$, where k is the effective switched capacitance of the chip architecture. Then the computing time of the task offloaded from U_m to A_n is given by

$$t^{\text{com}}_{m,n} = \frac{Q_{m,n} D_m}{f}, \quad (5)$$

where D_m, also called computational complexity, indicates different numbers of CPU cycles consumed by computing different tasks per bit. With reference to [24], [25], the energy consumption in A_n caused by the computation of U_m's task is given by

$$E^{\text{com}}_{m,n} = P t^{\text{com}}_{m,n} = Q_{m,n} D_m k f^2. \quad (6)$$

In practice, a general energy model includes the memory energy consumption caused by the storage of U_m in A_n. As Section 2.3 and 4.2 will elaborate, our proposed association and collaboration algorithms are also applicable in the general energy model, as long as the SBS is aware of the memory energy consumption.

With reference to [26], we consider that the data size of computing result is much smaller than that of offloaded task. Thus, compared with task uploading, we can ignore the delay and energy induced by the data downloading from SBS to MU.

2.2 Collaboration Model

As Section 2.1 points out, each SBS can serve at most K MUs over the fixed K orthogonal subcarriers, and each ECS can open at most G VMs.

2.2.1 ECS Configuration

This part elaborates different ECS configurations in each SBS.

Case 1 : $(K-1)N < M \leq KN$

In this case, there exists at least one SBS associated with K MUs. To guarantee that all MUs can be served simultaneously, we choose $G = K$. Put differently, to meet

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2019.2937777, IEEE Transactions on Services Computing

Case 1, each SBS can serve at most K MUs. Note that the number of MUs served by each SBS may differ. If $M = KN$, each SBS is fully loaded, i.e., $(GN - M) = 0$. Otherwise, the underloaded SBS indeed exists as $GN = KN > M$. It is worth noting that the overloaded SBS does not exist in this case.

Case 2 : $1 \leq M \leq (K-1)N$

The following findings motivate the new setup for this case. First, all SBSs can transfer tasks through the internal links as shown in Fig. 1, and the cost of data transfer among the SBSs is much lower than that of high-performance ECS. Second, we can ignore the transfer delay among the SBSs, since the speed of internal links is much faster than the MUs uploading speed. Driven by these findings, we propose the following strategies for this case:

- Keep the associations between MUs and SBSs unchanged and set $G = \lceil M/N \rceil$, where $\lceil x \rceil$ denotes the ceiling function that maps x to the least integer greater than or equal to x.
- It is possible that the underloaded, fully loaded, and overloaded SBSs coexist. Each overloaded SBS transfers its overloaded tasks to the underloaded SBSs. Notably, each overloaded SBS refunds the associated MUs to compensate for the delay caused by the data transfer [3].

In this case, choosing the ECS with $G = K$ VMs as Case 1 results in higher cost. To illustrate this problem, consider that $N = 4$ SBSs each with $K = 4$ subcarriers and $M = 10$. This example falls into Case 2 as $1 \leq M \leq (K-1)N$. Thus, we set $G = \lceil 10/4 \rceil = 3$. That is, each SBS deploys the ECS which can open 3 VMs at most. However, if we follow Case 1, each SBS will deploy the ECS with higher performance that supports $G = K = 4$ VMs. Apparently, this leads to more idle hardware resources.

We remark that some MUs need to queue up to solve their own tasks if $M > KN$. To study this special case, queuing theory may be cater to the analysis of time delay performance, which is beyond the scope of this paper. Therefore, this paper only considers $M \leq KN$.

2.2.2 SBS Collaboration

In Case 1, the data transfer among SBSs is not required, since the offloaded task from each MU can be accommodated by its associated SBS simultaneously. However, in Case 2, each overloaded SBS needs to transfer its local tasks beyond its computing capability to some underloaded SBSs. We refer to this task transferred from SBS $A_{u_m,h}$ to its peer A_r as the $collaboration$ between $A_{u_m,h}$ and A_r. For ease of exposition, we refer to A_r as a $helper$ of $A_{u_m,h}$. To be specific, $A_{u_m,h}$ denotes the overloaded SBS A_h associated with MU U_m, $h \in \{1, \cdots, H\}$ with H being the number of overloaded SBSs, and A_r is the underloaded SBS, $r \in \{1, \cdots, R\}$ with R being the number of the helpers. We remark that $R < N$, since the number of helpers of $A_{u_m,h}$ is a subset of the SBS set. Furthermore, we use an $M \times R$ adjacency matrix $\mathbf{Y}$ to represent the collaboration between $A_{u_m,h}$ and A_r, where the $\{m, r\}$-th element of $\mathbf{Y}$ is given by

$$y_{r,u_{m,h}} = \begin{cases} 1, & \text{if } A_r \text{ is a helper of } A_{u_{m,h}}, \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

2.2.3 Transfer Delay

Define the time delay caused by the task transfer of U_m in A_h to A_r as

$$t^{\text{tran}}_{u_{m,h},r} = \delta Q_{m,h,r} d_{h,r}, \tag{8}$$

where δ is the reference delay induced by transmission per bit between any two SBSs separated by unit distance, $Q_{m,h,r}$ is the total bits of tasks in $A_{u_m,h}$ transferred to A_r, and $d_{h,r}$ represents the distance between $A_{u_m,h}$ and A_r.

2.3 Utility Functions of MUs and SBSs

Generally, each MU is willing to minimize its cost by taking into account the computation tasks, the computing consumption, and the rent of ECS, while each SBS aims to maximize its profit from the computing services.

First, we define the utility function of MU U_m associated with A_n as

$$W^{\text{MU}}_{m,n} = \varsigma Q_m D_m - \lambda C_{m,n} - \beta_{m,n}, \tag{9}$$

where ς is the reference profit achieved by the computation of every bit with unit complexity, λ is the reference cost caused by unit computing consumption, and $\beta_{m,n}$ is the ECS rental fee of A_n paid by U_m. In (9), the computing consumption $C_{m,n}$ consists of the task offloading delay and energy consumption, given by[4]

$$\begin{aligned} C_{m,n} &= \alpha^{\text{t}}_m (t^{\text{up}}_{m,n} + t^{\text{com}}_{m,n}) + \alpha^{\text{e}}_m E^{\text{up}}_{m,n}, \\ &\alpha^{\text{t}}_m, \alpha^{\text{e}}_m \in [0,1], \alpha^{\text{t}}_m + \alpha^{\text{e}}_m = 1, \end{aligned} \tag{10}$$

where α^{t}_m and α^{e}_m are the weighted parameters of time delay and energy consumption respectively. Notably, the values of α^{t}_m and α^{e}_m depend on different roles of MUs. For example, we set a higher α^{e}_m for the MU with lower battery that has a more stringent power constraint, and a higher α^{t}_m for the MU with more delay-sensitive tasks.

Second, the utility function of the SBS A_n associated with U_m is given by

$$W^{\text{SBS}}_{n,m} = \beta_{m,n} - q - \tau E^{\text{com}}_{m,n}, \tag{11}$$

where q is the hardware cost of a single VM in A_n, τ is the energy price per dBm, and $E^{\text{com}}_{m,n}$ is given in (6). To ensure that each SBS utility is positive, we set $\beta_{m,n} \geq q + \tau E^{\text{com}}_{m,n}$.

3. The transfer delay among SBSs is much smaller than uploading delay and computing delay for the MU, since the speed of internal links among SBSs is much faster than the uploading speed of MUs. Therefore, from the MUs perspective, the MUs can ignore the transfer delay among SBSs, which does not impose any impact on the association. On the other hand, the task offloading caused by data transfer consumes more time than that without data transfer, even though the transfer delay is small. Therefore, due to this transfer delay, the overloaded SBS compensates its associated MUs whose tasks are transferred as shown in (13).

4. The linear utility functions have been widely used in many existing works such as [27] [28] [29].

2.4 Social Welfare

In this network, any underloaded SBS has the idle hardware resources. For example, in Case 1, the operator deploys the ECSs that can support at most $GN = KN$ MUs in total. However, given the M MUs in this case, this setup wastes the hardware resource due to the $KN - M$ idling VMs. Define the network social welfare as the sum of utilities of all entities in the network (i.e., MUs and SBSs in this paper) [21]. In many existing works on resource allocation such as [21] [30], maximizing social welfare is usually a general goal of the algorithms. For example, [21] developed a joint rate adaptation, channel assignment, and routing approach to maximize all utilities of all secondary nodes, i.e., social welfare, in the multihop cognitive radio ad hoc networks. In the following, we characterize the social welfare according to the ECS configurations in Section 2.2.1.

In Case 1, recall that the SBS collaboration is not required. The social welfare function is given by

$$S_1 = \sum_{m=1}^{M}\sum_{n=1}^{N} x_{m,n}(W_{m,n}^{\mathrm{MU}} + W_{n,m}^{\mathrm{SBS}}) - (KN - M)q, \quad (12)$$

where $(KN - M)q$ represents the cost induced by the idle hardware resources in Case 1.

In Case 2, the social welfare function is given by

$$S_2 = \sum_{m=1}^{M}\sum_{n=1}^{N} x_{m,n}(W_{m,n}^{\mathrm{MU}} + W_{n,m}^{\mathrm{SBS}}) - (GN - M)q - \lambda\alpha_m^{\mathrm{t}} \sum_{m=1}^{M}\sum_{r=1}^{R} y_{r,u_{m,h}} t_{u_{m,h},r}^{\mathrm{tran}}, \quad (13)$$

where $(GN - M)q$ is the cost induced by the idle hardware resources in Case 2, and the last term in the right-hand-side of (13) is the total refund to MUs due to transfer delay among SBSs.

3 Problem Formulations

In this section, we first formulate the optimization problems regarding to the association between MU and SBS as well as the collaboration among SBSs, so as to maximize the social welfare. Notably, we bear in mind that each SBS is selfish and greedy to maximize its profit, while each MU also aims to solve its own task at lowest cost.

In the MEC network, we put forth two requirements that all MUs and SBSs have to comply with. First, an MU will not make any other requests to other SBSs if it is associated with an SBS. Second, the SBSs will not accept any other requests if the SBS is fully associated with K MUs (i.e., fully loaded). We stress that the association and collaboration occur in sequence, since each SBS seeks for collaboration with other SBSs after its associations with MUs are completed. Therefore, we formulate two corresponding problems sequentially.

3.1 Optimization Problem of Association

The purpose of the optimal association between MU and SBS is to maximize the social welfare in (12). The optimization problem of association is given by

$$\max_{\mathbf{X}} S_1 \quad (14a)$$

$$s.t. \quad x_{m,n} \in \{0,1\} \quad (14b)$$

$$\sum_{n=1}^{N} x_{m,n} = 1, \quad m = 1,\cdots,M, \quad (14c)$$

$$\sum_{m=1}^{M} x_{m,n} \le K, \quad n = 1,\cdots,N, \quad (14d)$$

where $\mathbf{X}$ is defined in (1). We find that the optimization problem is an NP-hard combinatorial problem. With reference to [30], the optimal solution exists if $x \in \{0,1\}$. We remark that the matrix $\mathbf{X}$ is a feasible solution if it satisfies conditions (14b)-(14d). Moreover, the optimal solution $x_{m,n}^*$ exists if it satisfies $\sum_{m=1}^{M}\sum_{n=1}^{N} x_{m,n}^*(W_{m,n}^{\mathrm{MU}} + W_{n,m}^{\mathrm{SBS}}) > \sum_{m=1}^{M}\sum_{n=1}^{N} x_{m,n}(W_{m,n}^{\mathrm{MU}} + W_{n,m}^{\mathrm{SBS}})$.

3.2 Optimization Problem of Collaboration

First, the optimization problem of collaboration is given by

$$\max_{\mathbf{Y}} S_2 \quad (15a)$$

$$s.t. \ y_{r,u_{m,h}} \in \{0,1\}, \quad (15b)$$

$$-\sum_{m=1}^{M} x_{m,h}\sum_{r=1}^{R} y_{r,u_{m,h}} + \sum_{m=1}^{M} x_{m,h} = G, h = 1,\cdots,H, \quad (15c)$$

$$\sum_{m=1}^{M} y_{r,u_{m,h}} + \sum_{m=1}^{M} x_{m,r} \le G, \quad r = 1,\cdots,R, \quad (15d)$$

where (15b) indicates whether $A_{u_m,h}$ is matched with A_r, $\mathbf{Y}$ is defined in (7), (15c) guarantees that the number of MUs associated with $A_{u_m,h}$ is G, and (15d) guarantees that the total offloaded tasks from the associated MUs and the data transfer from the overloaded SBSs are within the computing capability of the help A_r. Recall that association precedes collaboration. As such, we can optimize S_2 by treating $\sum_{m=1}^{M}\sum_{n=1}^{N} x_{m,n}(W_{m,n}^{\mathrm{MU}} + W_{n,m}^{\mathrm{SBS}})$ and $(GN - M)q$ in (13) as the constants. Therefore, given the optimal solutions in (14), the problem in (16) is reduced to

$$\min_{\mathbf{Y}} \sum_{m=1}^{M}\sum_{r=1}^{R} y_{r,u_{m,h}} t_{u_{m,H},r}^{o}, \quad (16a)$$

$$s.t. \ y_{r,u_{m,h}} \in \{0,1\}, \quad (16b)$$

$$-\sum_{m=1}^{M} x_{m,h}\sum_{r=1}^{R} y_{r,u_{m,h}} + \sum_{m=1}^{M} x_{m,h} = G, h = 1,\cdots,H, \quad (16c)$$

$$\sum_{m=1}^{M} y_{r,u_{m,h}} + \sum_{m=1}^{M} x_{m,r} \le G, \quad r = 1,\cdots,R. \quad (16d)$$

Similarly, the optimization problem in (16) is also an NP-hard combinatorial problem.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2019.2937777, IEEE Transactions on Services Computing

4 Matching Algorithms

In this section, we first brief key preliminaries of matching theory and then develop two matching algorithms to solve the problems in (14) and (16) respectively.

4.1 Preliminaries of Matching

According to the matching theory, MUs and SBSs are regarded as two sides of players and each of them is assumed as a separate and independent set.

In the association problem, the MU is willing to be associated with the best SBS with the minimum computing assumption, while the SBS decides whether to accept or reject the requests from the MUs based on its utility. Given that each SBS is associated with multiple MUs, we treat the association problem between MUs and SBSs as a many-to-one matching.

Given $U_m \in \mathcal{U}$ and $A_n \in \mathcal{A}$, we define ω: $\mathcal{U} \rightarrow \mathcal{A}$ as a many-to-one matching function that maps the MU set $\mathcal{U}$ to the SBS set $\mathcal{A}$. Following the association strategy, the matching between $\mathcal{U}$ and $\mathcal{A}$ follows

$$1)\ \ \omega(U_m) \in \mathcal{A} \ \text{ and } \ |\omega(U_m)| \leq 1, \tag{17a}$$

$$2)\ \ \omega(A_n) \in \mathcal{U} \ \text{and} \ \ |\omega(A_n)| \leq K, \tag{17b}$$

$$3)\ \ \omega(U_m) = A_n \ \Leftrightarrow \ \omega(A_n) = U_m, \tag{17c}$$

where (17a) implies that each MU can be served by a single SBS only, (17b) represents that the maximum number of MUs associated with any SBS is K, and (17c) states that ω builds a bidirectional matching between $\mathcal{A}$ and $\mathcal{U}$.

In the collaboration problem, each overloaded SBS requests all underloaded SBSs to compute its overload tasks, and the underloaded SBSs decides whether to accept the requests. Both actions are determined by the transmit delay between any SBS pair in collaboration. Since each overloaded SBS may connect to multiple helpers, we treat the collaboration problem among SBSs as a many-to-many matching.

Given $A_{u_m,h} \in \mathcal{A}_U^O$ and $A_r \in \mathcal{A}^R$, we define ρ: $\mathcal{A}_U^O \rightarrow \mathcal{A}^R$ as a many-to-many matching function that maps the overloaded SBS set $\mathcal{A}_U^O$ to the helper set $\mathcal{A}^R$. Let $F_h = W_h - G$ be the number of MUs associated with $A_{u_m,h}$ and offloaded to other underloaded SBSs, where W_h is the number of MUs associated with $A_{u_m,h}$. In addition, let $B_r = G - W_r$ be the number of the remaining VMs in A_r after association, where W_r is the number of MUs associated with A_r. Following the association strategy, the matching between $\mathcal{A}_U^O$ and $\mathcal{A}^R$ follows

$$1)\ \ \rho(A_{u_m,h}) \in \mathcal{A}^R \ \text{ and } \ |\rho(A_{u_m,h})| = F_h, \tag{18a}$$

$$2)\ \ \rho(A_r) \in \mathcal{A}_U^O \ \text{ and } \ |\rho(A_r)| \leq B_r, \tag{18b}$$

$$3)\ \ \rho(A_{u_m,h}) = A_r \ \Leftrightarrow \ \rho(A_r) = A_{u_m,h}, \tag{18c}$$

where (18a) means that the overloaded SBS $A_{u_m,h}$ transfers tasks of F_h MUs, (18b) implies that A_r can only accept tasks from B_r MUs, and (18c) indicates that ρ builds a bidirectional matching between $\mathcal{A}_U^O$ and $\mathcal{A}^R$.

4.2 Association Algorithm

Following the matching in (18), we propose an association algorithm to solve the optimal problem in (14), as shown in Algorithm 1. Table 1 presents the commonly used notations in this algorithm.

TABLE 1: Notations in Algorithm 1

Number of subcarriers	K
Profit of U_m associated with A_n in iteration t	$W_{m,n}^{\mathrm{MU,t}}$
Profit of A_n associated with U_m in iteration t	$W_{n,m}^{\mathrm{SBS,t}}$
Price-allocation number in iteration t	$\beta_{m,n}^t$
Number of MU requests received by A_n	sub_n
Set of SBSs each receiving more than K requests	$\mathcal{A}_K$
Price-step number	ε
Received requests of A_n after increasing its price	NUM_n

Algorithm 1 Association Algorithm.

Input: $W_{n,m}^{\mathrm{SBS}}$, $W_{m,n}^{\mathrm{MU}}$, K;
Output: stable matching $\mathbf{X}$, price-allocation number $\beta_{m,n}$;
1: Initialization: t=1, $\beta_{m,n}^t = \beta_{m,n}^{min}$; the set of unmatched MUs $\mathcal{U} = \{U_1, U_2, \cdots, U_M\}$; the set of unmatched SBSs $\mathcal{A} = \{A_1, A_2, \cdots, A_N\}$;
2: **while** $\mathcal{U} \neq \emptyset$ **do**
3: **for** all $A_n \in \mathcal{A}$ **do**
4: A_n broadcasts its price-allocation number $\beta_{m,n}^t$ to all the unmatched MUs;
5: **end for**
6: **for** all unmatched $U_m \in \mathcal{U}$ **do**
7: U_m determines its profit $W_{m,n}^{\mathrm{MU,t}}$ and bids for A_n which provide the biggest profit;
8: Count the number of requests received by A_n as sub_n;
9: Add A_n^K with $\mathrm{sub}_n > K$ into the set $\mathcal{A}_K$;
10: **end for**
11: **while** $\mathcal{A}_K \neq \emptyset$ **do**
12: **for** all $A_n^K \in \mathcal{A}_K$ **do**
13: $t = t + 1$, $\beta_{m,n}^t = \beta_{m,n}^t + \varepsilon$;
14: MUs decide whether to make a further request to A_n^K;
15: Updates NUM_n;
16: **if** $\mathrm{NUM}_n > K$ **then**
17: Go to step 13;
18: **end if**
19: **if** $\mathrm{NUM}_n = K$ **then**
20: A_n^K associated with these K MUs. Remove A_n^K from $\mathcal{A}_K$ and $\mathcal{A}$, and remove these MUs from $\mathcal{U}$;
21: **end if**
22: **if** $\mathrm{NUM}_n < K$ **then**
23: $W_{m,n}^{\mathrm{MU},t} = W_{m,n}^{\mathrm{MU},t-1}$, $W_{n,m}^{\mathrm{SBS},t} = W_{n,m}^{\mathrm{SBS},t-1}$, $\beta_{m,n}^t = \beta_{m,n}^{t-1}$;
24: A_n^K associated with K MUs that provide the first K maximum profits for A_n^K;
25: Remove A_n^K from $\mathcal{A}_K$ and $\mathcal{A}$, and remove these MUs from $\mathcal{U}$;
26: **end if**
27: **end for**
28: **end while**
29: For all U_m associated with A_n, $x_{m,n} = 1$;
30: **end while**
31: **return** Stable matching $\mathbf{X}$, price-allocation number $\beta_{m,n}$.

The algorithm starts with the initialization in lines 1. In line 4, the SBS broadcasts the price-allocation number $\beta_{m,n}$ to all unmatched (or unassociated) MUs, where $\beta_{m,n}$ is prices made by A_n for MU U_m. Each MU then sends the request to the SBS A_n that provides the highest profit and bids for it. Then, the number of requests received by A_n (i.e., sub_n) increases.

If $\mathrm{sub}_n \leq K$, A_n will not increase $\beta_{m,n}$ and keep the unmatched MUs in the following iterations. If $\mathrm{sub}_n \leq K$ until the end of iteration, the SBS will be associated with the MUs that have made the requests. If $\mathrm{sub}_n > K$, the requesting MUs of the SBS will enter the bidding process

TABLE 2: Notations in Algorithm 2

Number of subcarriers	K
Maximum Number of VMs opened by ECS	G
Number of overloaded SBSs	H
Number of fully loaded SBSs	S
Number of underloaded SBSs	R
Number of overloaded MUs associated with A_h	F_h
Number of surplus VMs in A_r	B_r
Number of SBS requests received by A_r	req_r
Preference of A_h to A_r	Γ_r^h
Set of overloaded SBSs	$\mathcal{A}_H^O$

from lines 11 to 28.

Upon receiving the requests yielding $\mathrm{sub}_n > K$ in the t-th iteration, the SBS $\mathcal{A}_n^K \in \mathcal{A}_K$ decides wether to accept these requests. In the $(t+1)$-th iteration, A_n^K increases its price-allocation numbers $\beta_{m,n}^t$ by ε, $m = 1, 2, ...M$, as shown in line 13. As $\beta_{m,n}^t$ increases, there are three possibilities in terms of NUM_n and K. First, $\mathrm{NUM}_n < K$ in line 22. Note that in the t-th iteration, A_n^K has received requests from more than K MUs. The SBS will be associated with K MUs that provide the first K largest profits and keep the remaining MUs in the unmatched list as shown in lines 23-25. Second, $\mathrm{NUM}_n = K$ in line 19. In this case, A_n^K will be associated with all these requesting MUs. Third, $\mathrm{NUM}_n > K$ in line 16. In this case, A_n^K will further increase $\beta_{m,n}^t$ until it is associated with exactly K MUs as shown in line 13.

The last iteration of the algorithm produces an the empty unmatched list. Finally, we obtain the matching matrix $\mathbf{X}$ and $\beta_{m,n}$.

4.3 SBS Collaboration Algorithm

Following the matching in (18), we propose a collaboration strategy in Algorithm 2 to solve the problem in (16). Table 2 presents the commonly used notations in the algorithm.

After the association in Algorithm 1, we can determine not only the numbers of overload, fully loaded, and underloaded SBSs as H, S, and R respectively, but also the numbers of corresponding overloaded MUs and VMs supported by surplus handware resource as F_h and B_r, respectively. To start with the collaboration, each overload SBS sends the requests to the underloaded SBSs according to the $preference$ as follows:

Definition 1. The preference of A_h to A_r is defined as

$$\Gamma_r^h = \frac{1}{t_{u_{m,h},r}^{\mathrm{tran}}}, \tag{19}$$

where $t_{u_{m,h},r}^{\mathrm{tran}}$ is defined in (8).

From (19), lower transfer delay leads to higher preference. We stress that the mutual preference holds. That is, $t_{u_{m,h},r}^{\mathrm{tran}}$ in (19) also characterizes the preference of A_r to A_h. Accordingly, the overloaded SBS A_h sends the requests according to the first F_h largest preferences in $\Gamma_r^h, r = 1, ..., R$, while the underloaded SBS accepts the requests according to the first B_r largest preferences in $\Gamma_r^h, h = 1, ..., H$.

Let us elaborate Algorithm 2. As shown in line 4, if $\sum_{m=1}^{M} x_{m,h} > G$, then $F_h = \sum_{m=1}^{M} x_{m,h} - G$ is the number of MUs in the SBS A_h and offloaded to other underloaded SBSs. As such, A_h makes requests to F_h underloaded SBSs with the F_h largest preferences according Γ^h. Given $\sum_{m=1}^{M} x_{m,r}$ in (14) and $\sum_{m=1}^{M} y_{r,u_{m,h}}$ in (17), $B_r = G - \sum_{m=1}^{M} x_{m,r} - \sum_{m=1}^{M} y_{r,u_{m,h}}$. For any underloaded SBS A_r, there are two possibilities in terms of req_r and B_r. First, if $req_r > B_r$ in line 8, A_r will accept requests from the overloaded SBSs which provide the first B_r largest preferences. Second, if $req_r \leq B_r$ in line 10, the SBS will accept all requests.

The last iteration of the algorithm produces an empty unmatched list. Finally, we obtain the matching matrix $\mathbf{Y}$.

Algorithm 2 Collaboration Algorithm.

Input: Γ_r^h, K, G, $\mathbf{X}$;
Output: stable matching $\mathbf{Y}$;
1: Initialization: the set of overloaded SBSs $\mathcal{A}_H^O = \{A_1, A_2, \cdots, A_H\}$, $y_{r,u_{m,p}} = 0$;
2: **while** $\mathcal{A}_H^O \neq \emptyset$ **do**
3: **for** all $A_h \in \mathcal{A}_H^O$ **do**
4: A_h makes requests to the F_h SBSs in its preference list;
5: Remove the top F SBSs in Γ_r^h;
6: **end for**
7: **for** all A_r **do**
8: **if** $req_r > B_r$ **then**
9: A_r chooses the first B_r tasks transferred by overloaded SBSs according to Γ_r^h and rejects the other requests;
10: **else**
11: A_r accepts the requests from $\mathcal{A}_H^O$;
12: **end if**
13: A_r sets $y_{r,u_{m,h}} = 1$ for the accepted SBSs;
14: **end for**
15: **for** all $A_h \in \mathcal{A}_H^O$ **do**
16: **if** $-\sum_{m=1}^{M} x_{m,h} \sum_{r=1}^{R} y_{r,u_{m,h}} + \sum_{m=1}^{M} x_{m,h} = G$, **then**
17: remove A_h from $\mathcal{A}_H^O$;
18: **end if**
19: **end for**
20: **end while**
21: **return** stable matching $\mathbf{Y}$.

5 Performance Analysis of Proposed Algorithms

In this section, we first prove the stability of proposed algorithms and then verify that the association algorithm can achieve a competitive equilibrium.

5.1 Stability of Association Algorithm

A stable matching ensures that none of players has any motivation to change its matched players. To evaluate the stability, we consider the concept of group stability. The many-to-one matching problem includes multiple stable pairs. The matching game is stable if all the matching pairs are stable. Let $\ell(\mathcal{U}, \mathcal{A})$ denote the set of $final\ matching$ pairs, and (U_m, A_n) denote the subset of $\ell(\mathcal{U}, \mathcal{A})$, where (U_m, A_n) is a matched pair. We introduce the concept of blocking pair and stability as follows. The current matched pair (U_m, A_n) is blocked by the other pair if the following conditions are satisfied [31]:

1) blocking occurs in the pairs of (U_m, A_n) and is not blocked by U_m only or A_n only.

2) both U_m and A_n can achieve a higher utility if each of them matches with the others, compared with their current matching.

where condition 1) states that blocking pair must occur in pairs without the intention of any party, and condition 2) guarantees that the utility of blocking pair must be higher than the current matching.

Definition 2. Given a pair $(U_{m^*}, A_{n^*}) \notin \ell(\mathcal{U}, \mathcal{A})$. In this case, the cost of U_{m^*} associated with A_{n^*} is less than $\omega(U_{m^*})$, while U_{m^*} can provide a higher profit to A_{n^*} than at least one element in $\omega(A_{n^*})$. We define the pair (U_{m^*}, A_{n^*}) as the blocking pair.

Note that the many-to-one matching is stable if and only if there is no blocking pair. Second, by Definition 2 and Algorithm 1, we prove the stability as follows:

Theorem 1. The proposed MU and SBS association in Algorithm 1 is a stable matching.

Proof: This proof largely follows [30]. To prove the stability, we verify that the blocking pair does not exist in the association problem. As lines 7 and 24 of Algorithm 1 show, the MU is associated with the SBS that provides the biggest profits and the SBS is associated with the MUs that provide the first K largest profits. This guarantees that there is no blocking pair. □

To analyze the stability of Algorithm 2, we introduce the concept of pairwise stability [32]. As the SBS collaboration is a many-to-many matching algorithm, we validate that the stability of the many-to-many matching is pairwise stable. Assuming that there is a blocking pair (A_{u_m,h^*}, A_{r^*}) with $A_{u_m,h^*} \notin \rho(A_{r^*})$ and $A_{r^*} \notin \rho(A_{u_m,h^*})$, we have

$$1) \exists A^i_{u_m,h} \in \rho(A_{r^*}) : A_{u_m,h^*} \succ A^i_{u_m,h}, \tag{20a}$$

$$2) \exists A^i_r \in \rho(A_{u_m,h^*}) : A_{r^*} \succ A^i_r, \tag{20b}$$

where ρ is defined in (18) and $\succ$ represents that (A_{u_m,h^*}, A_{r^*}) can provide smaller transfer delay than $(A^i_{u_m,h}, A^i_r)$.

Theorem 2. The final matching result $\mathbf{Y}$ is a pairwise stable.

Proof: To prove the pairwise stability, we verify that the blocking pair does not exist in the collaboration problem. As lines 4 and 9 of Algorithm 2 indicate, the overloaded SBSs make requests to the underloaded SBSs that provide the first F_h smallest transfer delays and the underloaded SBS accepts the requests from the overloaded SBSs that provide the first B_r smallest transfer delays. This guarantees that there is no blocking pair. Thus, the final matching is pairwise stable. □

5.2 Competitive Equilibrium of Association Algorithm

A competitive equilibrium is a traditional concept of economic equilibrium, appropriate for the analysis of commodity markets with flexible prices and many agents [11]. In the association problem, the commodity markets consist of two types agents, i.e., MUs and SBSs. When the commodity market arrives at the competitive equilibrium, there comes to a price at which the number of MUs that will pay is equal to the number of SBSs that will sell. In this sense, the social welfare reaches the maxima among all possibilities of stable matchings due to different price-step numbers.

First, the competitive equilibrium in our problem is defined as

Definition 3. For the matching matrix $\mathbf{X}$ and price-allocation $\beta_{m,n}$, the MU and SBS association problem is in a competitive equilibrium if the following conditions are satisfied:

1) For each SBS $A_n \in \mathcal{A}$, if A_n is associated with an MU $U_m \in \mathcal{U}$, then $\beta_{m,n} \geq C_{m,n}$.
2) For each MU $U_m \in \mathcal{U}$, U_m is associated with the SBS that offers the maximum utility.
3) For each SBS $A_n \in \mathcal{A}$, if A_n is not associated with any MU, then $\beta_{m,n} = C_{m,n}$.

where $\beta_{m,n}$ and $C_{m,n}$ are defined in (9) and (10) respectively.

Second, by Definition 3 and Algorithm 2, we prove achievability of the competitive equilibrium as follows:

Theorem 3. The proposed association in Algorithm 1 arrives at the competitive equilibrium for a sufficiently small ε.

Proof: To prove this theorem, we verify that conditions 1)-3) of Definition 3 hold in Algorithm 1. First, we set $\beta_{m,n} \geq C_{m,n}$ and the price will be higher after each iteration. It proves that condition 1) holds. Second, if the SBS is not associated with any MU, the SBS will not receive any rents and will not cost anything. Therefore, condition 3) is also true. Thrid, let us verify condition 2). Suppose that we set a threshold ε_0 for the price-step number ε in Algorithm 1. Given this ε_0, this algorithm encounters two situations. In the first situation, ε is relatively large such that $\varepsilon > \varepsilon_0$. As such, the SBSs have raised its price by ε in each iteration up to an unaffordable price, such that some MUs give up the bidding and the number of associated MUs with each overloaded SBS may be less than G. Consequently, these SBSs will be associated with the MUs according to the profits, which results in that some MUs are unable to be associated with the SBS that provides the largest benefit. In the second situation, ε is relatively low such that $\varepsilon < \varepsilon_0$. The requesting MUs is reduced one by one to G as the price of SBSs increases. As such, the SBS is associated with the G requesting MUs. This verifies that condition 2) is true. □

Furthermore, we can verify that Algorithm 1 is also a stable matching if it can arrive at the competitive equilibrium. From condition 2) of Definition 3, we conclude that each MU is associated with the SBS that provides the largest utility with the price-step number $\beta_{m,n}$, and each SBS arrives at the price which makes it just receive exactly G requests from the MUs. However, it is possible that a stable matching cannot arrive at the competitive equilibrium.

5.3 Pareto Optimality of Two-Tier Matching Game

First, let us introduce the definition of Pareto improvement as follows:

Definition 4. For a multi-objective function, a Pareto improvement occurs if changing the match can increase the social welfare while meeting participants' wishes [33].

From Definition 4, the final matching is weak Pareto optimal when there is no Pareto improvement.

Second, by definition 4, we prove the weak Pareto optimality of proposed two-tier matching as follows:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2019.2937777, IEEE Transactions on Services Computing

TABLE 3: Parameters in Simulations

$Notations$	$Value$
Transmit power of user equipment P	2.0W
Gaussian noise power N_0	10^{-9}dB
Data size Q_m	[2.0 , 4.0]Gbit
Number of subcarriers K	4
Computational complexity D_m	[2000,2500]cycles/bit
Weighted parameter of time delay α_m^t	0.5
Computing speed of VM f	2.5×10^9
Effective switched capacitance k	10^{-26}
Reference profit of MU's benefit ς	1.2×10^{-6}
Reference cost of system consumption λ	10^5
Hardware cost of a single VM q	500
Reference delay δ	1×10^{-10}

TABLE 4: Social welfare under different numbers of MUs and SBSs

(M, N)	Exhaustive	Random	Association
(4,4)	18250	17997	18250
(8,4)	35354	34588	35354
(12,4)	57904	56488	57904
(16,4)	76412	74998	76410

Theorem 4. The proposed two-tier matching game is weak Pareto optimal.

Proof: Observing two proposed matching algorithm, each participant chooses matching objects according to its preference list. If the next selection has a higher position in his preference list, the participant will change his matching target to the next. The participant cannot receive higher utility by returning to any previous choice. For an MU U_m, there exists an SBS A_n that can provide a higher utility than $A_{n^*} \notin \omega(U_m)$. Therefore, U_m and A_{n^*} are more inclined to establish a matching relationship and form a pair of blocking pairs. As Theorem 1 has verified, the matching algorithm is stable and there are no blocking pairs. Thus, when the first matching comes to an end, there exists no Pareto improvement. Similarly, we can prove that there is no Pareto improvement in Algorithm 2. Therefore, the two-tier matching game is weak Pareto optimal . □

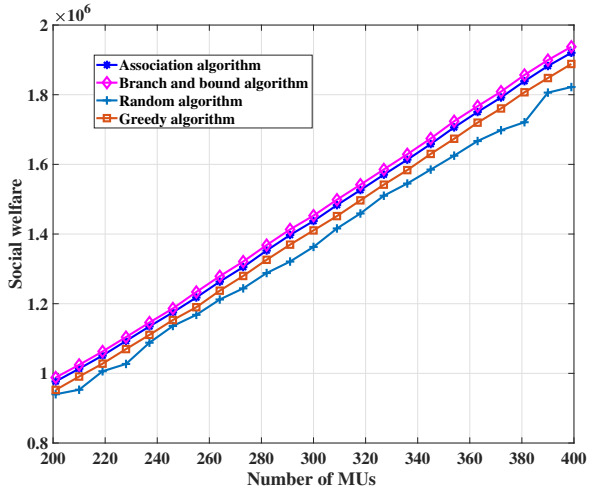


Fig. 2. Social welfare vs. number of MUs

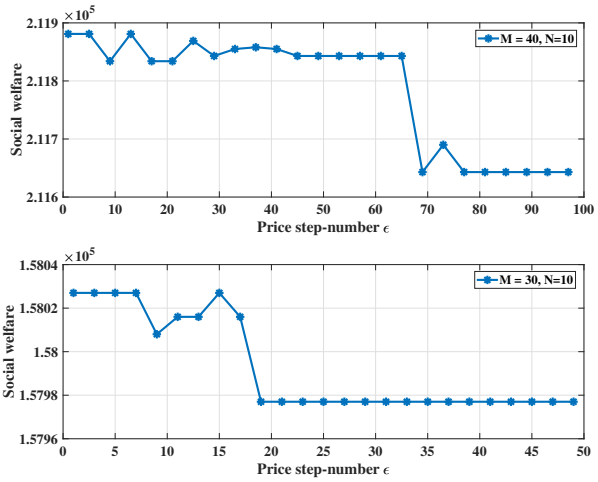


Fig. 3. Social welfare vs. price step-number ε

6 Simulation Results

In this section, we evaluate the performance of the proposed algorithms by numerical results. In the simulations, all MUs and SBSs are randomly located. The transmit power of each MU is 2W. The noise power is set to 10^{-9}dB. The MU tasks are ranges from 2Gb to 4Gb, and the computational complexity of MUs' tasks ranges from 2000cycle/bit to 2500cycle/bit. The number of subcarriers per SBS is 4. The computing power of each VM is 2.5GHz. The simulation parameters are detailed in Table 3.

We compare the proposed algorithms with random allocation, greedy algorithm, and exhaustive searching or branch-and-bound algorithm. In the random allocation, the SBSs and MUs are randomly associated with each other. The performance of random allocation is evaluated by taking the average over 300 times. In the greedy algorithm, we fully load one ECS before moving to the next. When the numbers of SBSs and MUs are small, we employ exhaustive searching (i.e., optimal in general). When the numbers of SBSs and MUs are large, we adopt the branch-and-bound algorithm (i.e., approximately optimal when N and M are large enough) [34].

In Table 4 and Fig. 2, we examine the impact of N, M on the social welfare under the four algorithms. Table 4 presents the social welfare of proposed association, exhaustive searching, and random allocation algorithms when the numbers of MUs and SBSs are relatively small. First, we observe that the social welfare of association is close to that of exhaustive searching and outperforms that of random allocation. Second, we observe that the social welfares of all algorithms increase as the number of MUs increases.

Fig. 2 shows the social welfare of proposed association, branch-and-bound, greedy, and random allocation algorithms when the numbers of MUs and SBSs are relatively large. Consider that the number of SBSs is $N = 100$. First, we observe that the social welfare of association is close to that of the branch-and-bound algorithm and outperforms that of random allocation and greedy algorithm. Second, we observe that the social welfares of all algorithms increase as the number of MUs increases. We remark that both exhaustive searching and branch-and-bound algorithms are imple-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2019.2937777, IEEE Transactions on Services Computing

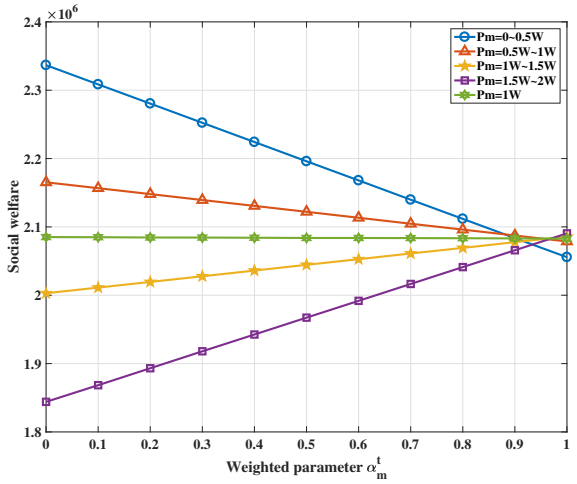


Fig. 4. Social welfare vs. weighted parameter α_m^t

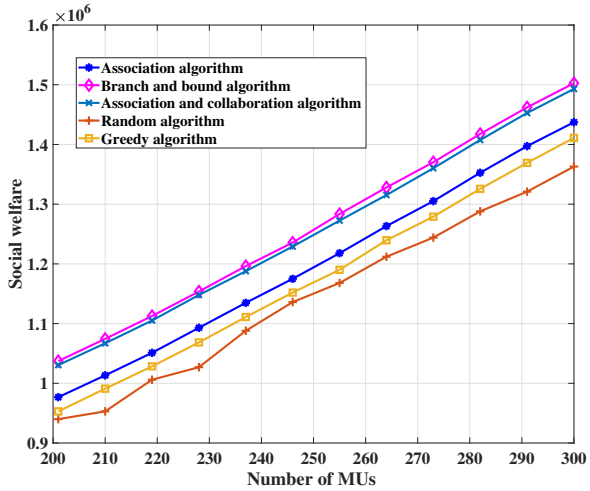


Fig. 6. Social welfare vs. number of MUs

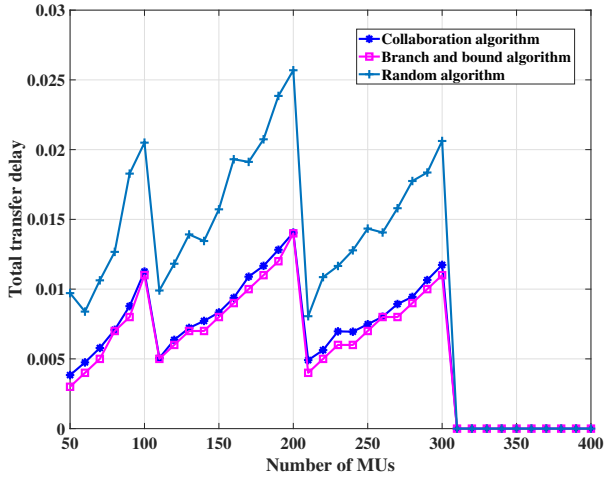


Fig. 5. Total transfer delay vs. number of MUs

mented in a centralized manner regardless of the utilities of each individual MU and SBS. In contrary, the proposed algorithm is a distributed solution with less computational complexity than exhaustive searching whose complexity increases exponentially over the network size.

Fig. 3 depicts the impact of the step number ε on the social welfare. We observe that the matching arrives at the competitive equilibrium and obtains the maximum social welfare when ε is less than a certain threshold. This observation is consistent with $Theorem$ 3. For example, when the number of MUs is 40 and $\varepsilon < 4$, the matching is in the competitive equilibrium state and obtains the maximum social welfare of 2.188×10^5. When the number of MUs is 30 and $\varepsilon < 8$, the competitive equilibrium state corresponds to the maximum social welfare of 1.58027×10^5.

Fig. 4 examines the impact of weighted parameters of time delay α_m^t on the social welfare. Consider that $N = 100$, $M = 400$, and uploading power of MUs ranges from 0W to 2W. First, we observe that the social welfare goes up as α_m^t increases when $1\text{W} < \text{P} \leq 2\text{W}$. This implies that when $1\text{W} < \text{P} \leq 2\text{W}$, the more MUs who are concerned about delay consumption participate, the larger the social welfare is. Second, we observe that the social welfare drops as α_m^t increases when $0\text{W} < \text{P} < 1\text{W}$, This implies that when $0\text{W} < \text{P} < 1\text{W}$, the more MUs who are concerned about energy consumption participate, the larger the social welfare is. Third, we observe that α_m^t has little impact on the social welfare when $P = 1\text{W}$.

Fig. 5 shows the comparison of the transfer delay among the proposed collaboration, random allocation, and branch-and-bound algorithms with respect to different numbers of MUs. First, it is observed that the total transfer delay of each algorithm increases within three regions of $N \in [50, 100], [100, 200], \text{and} [200, 300]$, respectively. When $50 < N < 100$, $G = 1$ according to Case 2 in Section 2.2.1. When $100 < N < 200$, $G = 2$ according to Case 2. When $200 < N < 300$, $G = 3$ according to Case 2. Second, we see that the total transfer delay of each algorithm becomes zero when $N > 300$. This is because the collaboration is not required when $N > 300$ by Case 1 in Section 2.2.1. Third, we observe that the proposed algorithm has similar delay performance to that of the branch-and-bound algorithm, while they both have much smaller total delay than that of the random allocation. For example, when the number of MUs is 200, the proposed algorithm has the delay of 0.014s and the delay of the random allocation is 0.024s, which indicates a 42.1% decrease on the transfer delay.

Fig. 6 compares the social welfare among the proposed two-tier (including the association and collaboration), association, random allocation, greedy, and branch-and-bound algorithms with respect to different numbers of MUs. For $N \in [200, 300]$, the maximum number of VMs supported by every single ECS is 3 according to Case 2 in Section 2.2.1. First, we observe that the two-tier algorithm not only outperforms association, greedy, and random allocation algorithms, but also achieves similar social welfare to the branch-and-bound algorithm. Second, it is observed that the social welfares of all algorithms boosts as M increases.

Differences between Fig. 2 and Fig. 6 are highlighted as follows. First, Fig. 2 focuses on the comparison between the proposed association and other algorithms, while Fig.

6 focuses on the comparison among the proposed two-tier algorithm (including the association and collaboration), association algorithm, and other algorithms. Second, the number of MUs ranges from 200 to 400 in Fig. 2, while the number of MUs is from 200 to 300 in Fig. 6. Notably, the collaboration algorithm is not applicable when the number of MUs is from 300 to 400, as $300 \leq M \leq 400$ belongs to Case 1 and the collaboration is not needed. Third, Fig. 2 and Fig. 6 use different branch-and-bound algorithms for the proposed association algorithm and the two-tier algorithm respectively.

7 Conclusions

In this paper, we have investigated the two-tier matching in small cell networks with MEC. First, we formulated the association problem between MUs and SBSs and the collaboration problem among SBSs as the two-tier matching game. Second, we proposed distributed matching algorithms, i.e., Algorithms 1 and 2, for the association and collaboration problems respectively. Our algorithm not only maximized the social welfare of the network, but also guaranteed the individual utilities of MUs and SBSs. In particular we introduced the bidding mechanism into the association algorithm, and we realized the competition equilibrium between MUs and SBSs and the optimal pricing of computing resources in SBS. Finally, simulation results demonstrated that the proposed algorithms outperformed random allocation and greedy algorithms in terms of social welfare, while retaining a close performance to exhaustive searching and branch-and-bound algorithms.

Several interesting directions follow this work. First, this paper only considered homogenous ECS (i.e., each ECS equipped with the same hardware resources). How to deploy heterogeneous ECSs each with distinct hardware resources remains challenging. Second, our proposed algorithm was applicable in the scenario with $M \leq KN$. It is of interest to apply the queueing theory [35] to address the scenario where the number of offloaded tasks are beyond computation capability of all ECSs, i.e., $M > KN$.

References

[1] S. Abolfazli, Z. Sanaei, E. Ahmed, A. Gani, and R. Buyya, "Cloud-based augmentation for mobile devices: Motivation, taxonomies, and open challenges," *IEEE Commun. Surv. Tut.*, vol. 16, no. 1, pp. 337–368, First quoter, 2014.

[2] S. Azodolmolky, P. Wieder, and R. Yahyapour, "Cloud computing networking: challenges and opportunities for innovations," *IEEE Commun. Mag.*, vol. 51, no. 7, pp. 54–62, Jul. 2013.

[3] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, Jan. 2017.

[4] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surv. Tut.*, vol. 19, no. 4, pp. 2322–2358, 2017.

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

[6] J. Li, C. Shunfeng, F. Shu, J. Wu, and D. N. K. Jayakody, "Contract-based small-cell caching for data disseminations in ultra-dense cellular networks," *IEEE Trans.Wireless Commun.*, pp. 1–1, Jul. 2018.

[7] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.

[8] B. P. Rimal, D. P. Van, and M. Maier, "Mobile-edge computing vs. centralized cloud computing in fiber-wireless access networks," in *INFOCOM WORKSHOPS*, San Francisco, CA, USA, 2016, pp. 991–996.

[9] L. Yang, H. Zhang, M. Li, J. Guo, and H. Ji, "Mobile edge computing empowered energy efficient task offloading in 5g," *IEEE Trans. Veh. Technol.*, vol. 67, no. 7, pp. 6398–6409, Jul. 2018.

[10] A. Roth and M. A. O. Sotomayor, *Two-sided matching :A Study in Game-Theoretic Modeling and Analysis*. Cambridge University, 2017.

[11] S. Bayat, Y. Li, L. Song, and Z. Han, "Matching theory: Applications in wireless communications," *IEEE Signal Processing Mag.*, vol. 33, no. 6, pp. 103–122, Nov. 2016.

[12] Z. Han, Y. Gu, and W. Saad, *Matching Theory for Wireless Networks*. Springer Publishing Company, Incorporated, 2017.

[13] S. W. H. Zhu, N. Dusit and B. Tamer, *Game theory in wireless and communication networks: theory, models, and applications*. Cambridge University Press, 2018.

[14] T. Liu, J. Li, B. Kim, C. Lin, S. Shiraishi, J. Xie, and Z. Han, "Distributed file allocation using matching game in mobile fog-caching service network," in *Proc. IEEE INFOCOM*, Honolulu, HI, USA, 2018, pp. 499–504.

[15] H. Zhang, Y. Xiao, S. Bu, D. Niyato, F. R. Yu, and Z. Han, "Computing resource allocation in three-tier iot fog networks: A joint optimization approach combining stackelberg game and matching," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1204–1215, Oct. 2017.

[16] T. LeAnh, N. H. Tran, W. Saad, L. B. Le, D. Niyato, T. M. Ho, and C. S. Hong, "Matching theory for distributed user association and resource allocation in cognitive femtocell networks," *IEEE Trans. Veh. Technol.*, vol. 66, no. 9, pp. 8413–8428, Sep. 2017.

[17] T. Li, C. S. Magurawalage, K. Wang, K. Xu, K. Yang, and H. Wang, "On efficient offloading control in cloud radio access network with mobile edge computing," in *Proc. ICDCS*, Atlanta, GA, USA, 2017, pp. 2258–2263.

[18] G. Premsankar, M. D. Francesco, and T. Taleb, "Edge computing for the internet of things: A case study," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1275–1284, Apr. 2018.

[19] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surv. Tut.*, vol. 19, no. 4, pp. 2322–2358, 2017.

[20] X. Sun and N. Ansari, "Edgeiot: Mobile edge computing for the internet of things," *IEEE Commun. Mag.*, vol. 54, no. 12, pp. 22–29, Dec. 2016.

[21] F. Tang and J. Li, "Joint rate adaptation, channel assignment and routing to maximize social welfare in multi-hop cognitive radio networks," *IEEE Trans.Wireless Commun.*, vol. 16, no. 4, pp. 2097–2110, Apr. 2017.

[22] F. Zhang, G. Liu, X. Fu, and R. Yahyapour, "A survey on virtual machine migration: Challenges, techniques, and open issues," *IEEE Commun. Surv. Tut.*, vol. 20, no. 2, pp. 1206–1243, Secondquarter 2018.

[23] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4569–4581, Sep. 2013.

[24] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel and Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, Apr. 2015.

[25] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint computation offloading and user association in multi-task mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 12, pp. 12 313–12 325, Dec. 2018.

[26] C. You, K. Huang, H. Chae, and B. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans.Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.

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

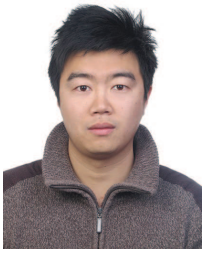
[28] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Trans. Veh. Technol.*, pp. 1–1, Nov. 2018.

[29] M. Chen, B. Liang, and M. Dong, "Multi-user multi-task offloading and resource allocation in mobile cloud systems," *IEEE Trans. Wireless Commun.*, vol. 17, no. 10, pp. 6790–6805, Oct 2018.

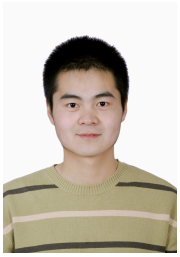
[30] J. Li, M. Liu, J. Lu, F. Shu, Y. Zhang, S. Bayat, and D. N. K. Jayakody, "On social-aware content caching for D2D-enabled cellular networks with matching theory," *IEEE Internet Things J.*, pp. 1–1, 2018.

[31] X. Chen, L. Pu, L. Gao, W. Wu, and D. Wu, "Exploiting massive D2D collaboration for energy-efficient mobile edge computing," *IEEE Wireless. Commun.*, vol. 24, no. 4, pp. 64–71, Aug. 2017.
[32] M. Sotomayor, "Three remarks on the many-to-many stable matching problem," *MATH SOC SCI.*, vol. 38, no. 1, pp. 55–70, Jan. 1999.
[33] C. Xu, C. Gao, Z. Zhou, Z. Chang, and Y. Jia, "Social network-based content delivery in device-to-device underlay cellular networks using matching theory," *IEEE Access*, vol. 5, pp. 924–937, 2017.
[34] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[35] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4005–4018, Jun. 2019.

Yu Du received his B.S. degree from Nanjing University of Science and Technology, P. R. China in 2015. From September 2015 to now, he is a Ph.D. student at the School of Electronic and Optical Engineering, Nanjing University of Science and Technology, Nanjing, China. His research interests include mobile edge computing and matching theory.



Jun Li received his Ph.D. degree in Electronic Engineering from Shanghai Jiao Tong University, Shanghai, P. R. China in 2009. From January 2009 to June 2009, he worked in the Department of Research and Innovation, Alcatel Lucent Shanghai Bell as a Research Scientist. From June 2009 to April 2012, he was a Postdoctoral Fellow at the School of Electrical Engineering and Telecommunications, the University of New South Wales, Australia. From April 2012 to June 2015, he is a Research Fellow at the School of Electrical Engineering, the University of Sydney, Australia. From June 2015 to now, he is a Professor at the School of Electronic and Optical Engineering, Nanjing University of Science and Technology, Nanjing, China. His research interests include network information theory, channel coding theory, wireless network coding, resource allocations in cellular networks.



Long Shi (S'10-M'13) received his Ph.D. degree in electrical engineering from the University of New South Wales, Sydney, Australia, in 2012. He was a visiting student at the Chinese University of Hong Kong and University of Delaware in 2010 and 2011, respectively. From 2013 to 2016, he was a postdoctoral fellow at the Institute of Network Coding, Chinese University of Hong Kong. From 2014 to 2017, he was a lecturer at College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics. Currently, he is a Postdoctoral Research Fellow at the Singapore University of Technology and Design. His current research interests include wireless caching, wireless network coding, and channel coding.

Tingting Liu (M'12) received her B.S. degree in Communication Engineering from Nanjing University of Science and Technology, Nanjing, China, in 2005, and the Ph.D. degree in Information and Communication Engineering from Nanjing University of Science and Technology, Nanjing, China, in 2011. From 2011, she is with the school of Communication Engineering in Nanjing Institute of Technology, China. Currently, she is also a postdoctor in Nanjing University of Science and Technology. Her research interests include game theory, cache-enabled systems, device-to-device networks and cognitive radio networks.

Feng Shu received his Ph.D., M.S., and B.S. degrees from the Southeast University, Nanjing, in 2002, XiDian University, Xian, China, in 1997, and Fuyang teaching College, Fuyang, China, in 1994, respectively. From Sept. 2009 to Sept. 2010, he is a visiting post-doctor at the University of Texas at Dallas. In October 2005, he joined the School of Electronic and Optical Engineering, Nanjing University of Science and Technology, Nanjing, China, where he is currently a Professor and supervisor of Ph.D and graduate students. He is also with Fujian Agriculture and Forestry University and awarded with Mingjian Scholar Chair Professor and Fujian hundred-talent plan in Fujian Province. His research interests include wireless networks, wireless location, and array signal processing. Now, he is an editor for the journal IEEE Access. He has published more than 200 in archival journals with more than 50 papers on IEEE Journals and 80 SCI-indexed papers. He holds six Chinese patents.

Zhu Han (S'01-M'04-SM'09-F'14) received the B.S. degree in electronic engineering from Tsinghua University in 1997, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Maryland at College Park, College Park, MD, USA, in 1999 and 2003, respectively. From 2000 to 2002, he was a Research and Development Engineer of JDSU, Germantown, MD, USA. From 2003 to 2006, he was a Research Associate at the University of Maryland. From 2006 to 2008, he was an Assistant Professor at Boise State University, Boise, ID, USA. He is currently a Professor with the Electrical and Computer Engineering Department and with the Computer Science Department, University of Houston, Houston, TX, USA. His research interests include wireless resource allocation and management, wireless communications and networking, game theory, big data analysis, security, and smart grid. He received an NSF Career Award in 2010, the Fred W. Ellersick Prize of the IEEE Communication Society in 2011, the EURASIP Best Paper Award for the *Journal on Advances in Signal Processing* in 2015, the IEEE Leonard G. Abraham Prize in the field of communications systems (Best Paper Award in the IEEE JSAC) in 2016, and several best paper awards at the IEEE conferences. He is currently an IEEE Communications Society Distinguished Lecturer.