Energy–Latency Tradeoff for Computation Offloading in UAV-Assisted Multiaccess Edge Computing System

Kaiyuan Zhang, Xiaolin Gui, Dewang Ren, and Defu Li

***Abstract*—Unmanned aerial vehicles (UAVs) have been considered as a promising approach for providing additional computation capability and extensive coverage to ground mobile devices (MDs), especially in the scenario where the communication infrastructure is unavailable. This article investigates a UAV-assisted multiaccess edge computing system, where a number of ground MDs are served by a flying UAV and a ground base station, both of them are equipped with computation resources. The system aims to minimize the weighted cost of time latency and energy consumption, subject to the constraints on offloading decisions and resource competition. Since this problem is NP-hard, and the MDs are autonomous, we propose a game theory-based scheme to find the optimal solution and prove the existence of the Nash equilibrium. Meanwhile, we propose another two schemes as the benchmark to evaluate the efficiency and effectiveness of the game-theoretic solution. The simulation results show that the game-theoretic scheme is able to achieve a near-optimal performance, and the convergence time is also stable when the number of MDs increases.**

***Index Terms*—Computation offloading, game theory, multiaccess edge computing (MEC), unmanned aerial vehicle (UAV) assisted.**

I. Introduction

With the rapid development of the mobile Internet, smart mobile devices (MDs), such as the smartphone, smart watch, and augmented reality glass, are becoming prevalent. Meanwhile, lots of new applications, such as augmented reality, virtual reality, and face recognition, have emerged and attracted great attentions [1], [2]. These new applications have several traits in common, i.e., low-latency tolerance, high energy consumption, and computation intensive [3], [4]. Therefore, we call these new applications as computation-intensive applications. However, due to the constraints of MDs' physical size, it is difficult to meet the requirements of these new applications [5]. The contradiction between varied workloads of computation-intensive applications and limited computation resources of MDs brings a challenge for the user experience.

To address this challenge, mobile cloud computing (MCC) has been proposed to enable MDs to offload their computation tasks partially or fully to the cloud [6]–[8]. Since the cloud server is far away from the mobile users, nevertheless, MCC imposes huge additional load both on the radio and backhaul of mobile networks and introduces high latency. To overcome this limitation, recently, multiaccess edge computing (MEC), which can be considered as an extension of MCC, has been proposed as a promising solution [9], [10]. By equipping the base station (BS) with computing resources, the centralized cloud server is gradually replaced by network edge servers. Furthermore, by offloading the computation tasks to the physically proximal edge servers, traffic sent to the remote cloud could be reduced [11], [12]. However, even the existing MEC techniques cannot handle the wireless network scenarios with limited available infrastructures, such as disaster response, emergency relief, or military maneuver environments. In these scenarios, some ground BSs may be damaged by natural disasters or military attacks. Fortunately, unmanned aerial vehicles (UAVs)-assisted MEC has been proposed and envisioned as a potential technique to tackle the challenges [13]–[15].

UAVs can carry out exploration missions to replace human presence in areas that are inaccessible or hazardous. They can also act as aerial BSs to provide communication coverage to areas with limited available infrastructures [16]. In fact, thanks to the maturity of their underlying technology, along with their 3-D aerial mobility, it is a prospective idea to apply UAVs as aerial BSs to improve the efficiency of the current communication system [17]. These new aerial BSs could be defined as UAV-assisted MEC servers. The research community is active in the architectural design of UAV-assisted MEC. Zhao *et al.* [18] considered deploying multiple UAVs for on-demand coverage while maintaining the connectivity among UAVs. Kumar *et al.* [19] studied the relationship between deployment altitude and coverage area, in a delay-aware UAV-assisted communication system. Moradi *et al.* [20] further proposed an alternate edge-evolved packet core (EPC) design, pushing the EPC function to the extreme edge of the core network and achieving a distributed management of UAV mobility. Vamvakas *et al.* [21] introduced a holistic and

Manuscript received March 30, 2020; revised May 11, 2020; accepted May 23, 2020. Date of publication June 1, 2020; date of current version April 7, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 61472316 and Grant 61502380; in part by the National Key Research and Development Project under Grant 2018YFB1800304; in part by the Key Research and Development Project of Shaanxi Province under Grant 2019GY-005, Grant 2017ZDXM-GY-011, and Grant 2020GY-033; in part by the Major Basic Research Project of Shaanxi Province under Grant 2016ZDJC-05; and in part by the Science and Technology Program of Shenzhen under Grant 201771802. *(Corresponding author: Xiaolin Gui.)*

The authors are with the School of Electronic and Information Engineering and the Shaanxi Province Key Laboratory of Computer Network, Xi'an Jiaotong University, Xi'an 710049, China (e-mail: y1073695974@stu.xjtu.edu.cn; xlgui@mail.xjtu.edu.cn; rendewang@stu.xjtu.edu.cn; defuli@stu.xjtu.edu.cn).

Digital Object Identifier 10.1109/JIOT.2020.2999063

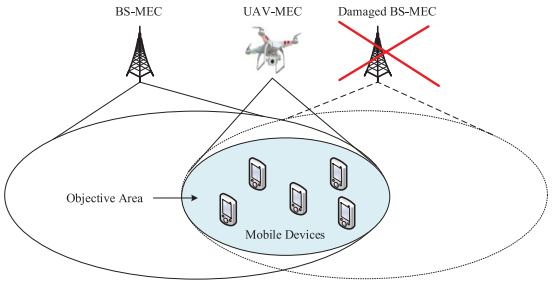


Fig. 1. System model.

realistic framework, which utilized UAVs acting as BSs and complementing the macro BS (MBS) to support the ground users efficient and undisturbed communication. Generally, the idea of UAVs acting as aerial BSs was especially practical for the wireless network scenarios where infrastructures are damaged [22]–[24].

In the UAV-assisted MEC system, the energy consumption and time latency of the offloaded tasks will be a big concern since the UAVs are usually battery limited. In this regard, Cao *et al.* [25] explored the joint UAV trajectory and computation offloading optimization problem for minimizing the time latency of the offloaded tasks. Hu *et al.* [26] studied the resource optimization problem for multiple users served by the UAV-assisted MEC system, in which computation offloading, UAV trajectory, and user scheduling are jointly addressed. Zhang *et al.* [27] investigated joint optimization on computation offloading, resource allocation, and flying trajectory scheduling in the UAV-assisted MEC system while considering the average weighted energy consumption of MDs and the UAV. Zhou *et al.* [28] addressed the computation rate maximization problems in a UAV-assisted MEC system under both partial and binary computation offloading modes. Alsenwi *et al.* [29] focused on minimizing the sum energy consumption of IoT devices and the UAV server in a UAV-assisted mobile-edge computing system, by using the block coordinate descent algorithm. Moreover, all of these above existing works are established in the scenario of one single UAV equipped with an edge server.

Despite the ample research, the problem of computation offloading in UAV-assisted MEC systems still faces many challenges. From the perspective of MDs, computation offloading can help them reduce energy consumption and save battery life. However, due to the limited battery capacity and computation capability of UAVs, if a large number of MDs perform computation offloading, resource competition will lead to a decline in the overall system performance. Thus, the computation offloading decisions on the MDs side are very important. Moreover, to the best of our knowledge, computation offloading in UAV-assisted MEC systems has not been well investigated.

In this article, we consider a UAV-assisted MEC system with a number of ground MDs distributed in the overlapping coverage area of two BSs, as depicted in Fig. 1. The BSs were equipped with edge servers, i.e., the BS-MEC server. The MDs have computation-intensive tasks to complete, which could be executed locally or offloaded to the BS-MEC servers. Suppose one of the two BSs is broken down for some reason, such as a natural disaster or a military attack. Since the broken BS may not be repaired in a short time. Hence, MDs that were originally in the coverage of the damaged BS would move to the objective area to look for a cellular signal. This will increase the load on the undamaged BS. Thus, we introduce a flying UAV, which was equipped with communication and computing capabilities (i.e., the UAV-MEC server), to serve these ground MDs and share the load of the undamaged BS, before the new BS is deployed. Therefore, MDs, in this scenario, have three options for addressing the computation offloading problem.

1) They can execute the computation tasks locally.
2) They can offload their tasks to the powerful BS-MEC server through a cellular network.
3) They can offload their tasks to the flying UAV-MEC server through an orthogonal multiple access manner.

Since the MDs are located at the edge of the coverage area of the BS, the communication quality between MDs and the BS is worse than that between MDs and the UAV. On the other hand, due to the limitation of physical size, the UAV-MEC server has a less computation resource than the BS-MEC server. To achieve efficient computation offloading for this UAV-assisted MEC system, we hence need to carefully tackle this key challenge: how does an MD make its offloading decision among these three options?

Therefore, we solve this kind of computation offloading decision problem by using game theory. This analytical tool has been widely used by the wireless networking community for modeling different types of problems [30]. The objective is to minimize the weighted system total cost, where energy consumption and time latency are jointly addressed. We design a game-theoretic scheme that achieves the best possible trade-off between time latency and energy consumption. Besides, in order to evaluate the efficiency and effectiveness of this game-theoretic scheme, we propose another two different schemes as the benchmark. The main contributions of this article are summarized as follows.

1) We study the problem of the computation offloading decision with one BS-MEC server and one UAV-MEC server in a multiaccess environment, which takes into account both communication and computation aspects of MEC, and formulate it as a constrained optimization problem, with the objective to minimize a weighted cost of energy consumption and time latency for all MDs.
2) We regard each MD as a rational game player and transform the constrained optimization problem into a multiplayer computation offloading game. A game-theoretic computation offloading (GTCO) scheme is proposed for solving the constrained optimization problem. The Nash equilibrium (NE) and convergence of the algorithm are analyzed and proved.
3) We propose another two different schemes as benchmarks to evaluate the efficiency and effectiveness of this game-theoretic scheme. The former is the optimal solution based on the traversal method, and the latter is a near-optimal solution based on sequential tuning. Similarly, the convergence of these two algorithms is also analyzed and proved.

4) Finally, simulation results show that the proposed game-theoretic scheme achieves a near-optimal average system total cost and the lowest time complexity compared with the other schemes.

The remainder of this article is organized as follows. We describe the system model and present the problem formulation in Section II. In Section III, we provide details of the proposed game-theoretic scheme and the other two benchmarks. The simulation results are shown in Section IV and conclusions are discussed in Section V.

II. System Model and Problem Formulation

This section introduces the system model and problem formulation. We consider a set of MDs $N = \{1, 2, \ldots, n\}$ collocated in the objective area (see Fig. 1). Each of them is running computation-intensive and latency-sensitive applications with the aid of a BS-MEC server and a UAV-MEC server. In order to guarantee the quality of experience, each MD is required to execute a highly intensive and delay-sensitive computation task, while preserving its available energy. Different decisions are available for these resource-constrained MDs. They can either: 1) execute their tasks locally; 2) offload them via a cellular connection to a BS-MEC server; or 3) through an orthogonal multiple access connection to a UAV-MEC server. The computation-intensive task, which has to be performed by MD, is characterized by three parameters, the number Y_i of CPU cycles required to execute the calculation, the size D_i of the input data (e.g., in bytes), and the maximum permissible energy consumption $e_i^{\max}$ for accomplishing the task. Besides the input parameters necessary for the computation, the data to be sent can even include the program code if it needs to be executed remotely.

We refer to the vector $\mathbf{x} = \{x_1, x_2, \ldots, x_N\}$ as the offloading decision profile of MDs and we denote the offloading decision for MD i by $x_i \in \{0, 1, 2\}$, where $x_i = 0$ corresponds to local execution (LE), $x_i = 1$ means offloading task to the BS-MEC server, and $x_i = 2$ means offloading task to the UAV-MEC server.

In the following sections, we first introduce the communication model used by the MDs, BS-MEC server, and UAV-MEC server for communication. Then, we give details about the three possible computation models. Finally, we present the cost model and an objective function that needs to be addressed.

A. Communication Model

1) Communication Between MDs and BS-MEC Server: We denote the uplink rate between MD i and the BS-MEC server by R_i^m, which depends on the physical-layer signal characteristics and the corresponding channel gain. By considering the Shannon capacity formulation [31], R_i^m can be defined as

$$R_i^m = B_i^m \log_2\left(1 + \frac{P_i h_i^m}{B_i^m \sigma^2}\right) \tag{1}$$

where B_i^m represents the bandwidth of the link between MD i and the BS-MEC server, P_i is the transmission power of the MD, h_i^m is the channel gain of MD i in the wireless channel, and σ^2 is the background noise. Note that here we focus on the computation offloading problem under the wireless interference model, which adopts some physical-layer channel access schemes (e.g., CDMA) to allow multiple MDs to share the spectrum resources simultaneously and effectively. Consider a bandwidth constraint of the BS-MEC server, we denote it by B^m. Thus, B_i^m is limited by B^m and the number $n(\mathbf{x})$ of MDs that offload tasks to the BS-MEC server

$$\sum_{i=1}^{n(\mathbf{x})} B_i^m \le B^m. \tag{2}$$

2) Communication Between MDs and UAV-MEC Server: The uplink rate between MD i and the UAV-MEC server is denoted by R_i^u, which depends on the physical-layer signal characteristics and the corresponding channel gain. For UAV-assisted networks, as the altitude of the UAV is much higher than that of the ground users, the Line-of-Sight (LoS) channels of the UAV communication links are much more predominant than other channel impairments. The channel gain from the UAV-server to MD i can be described as the free-space path-loss model $h_i^u = \mu_0 (d_i^u)^{-2}$, where μ_0 represents the channel gain at the reference distance $d_0 = 1$ and d_i^u denotes the distance between MD i and the UAV-MEC server. Note that the distance d_i^u is proportional to the flight height of the UAV-MEC server. Thus, R_i^u can be defined as

$$R_i^u = B_i^u \log_2\left(1 + \frac{P_i h_i^u}{B_i^u \sigma^2}\right) \tag{3}$$

where B_i^u represents the bandwidth of the link between MD i and the UAV-MEC server, P_i is the transmission power of the MD, h_i^u is the channel gain of MD i in the wireless channel, and σ^2 is the background noise. Similar to the case of communication between MDs and the BS-MEC server, here, we focus on the computation offloading problem under the wireless interference model, which adopts some physical-layer channel access schemes (e.g., CDMA) to allow multiple users to share the spectrum resources simultaneously and effectively. Consider a bandwidth constraint of UAV-MEC server, we denote it by B^u. Thus, B_i^u is limited by B^u and the number $m(\mathbf{x})$ of MDs that offload tasks to the UAV-MEC server

$$\sum_{i=1}^{m(\mathbf{x})} B_i^u \le B^u. \tag{4}$$

B. Computation Model

Since the offloading decision of MD i's task has three options: 1) local computing; 2) offload to the BS-MEC server; or 3) offload to the UAV-MEC server. We then discuss the computation model for these three options.

1) Local Computing: In the case of local computing, the computation task has to be processed using local computing resources, no actual data ought to be sent via wireless interfaces. As in [32], the local computing time and the corresponding energy consumption are related to the CPU cycles required by the computation task. Hence, we denote f_i^l as the computation capability (i.e., CPU cycles per second) of MD i, and the local computing time needed to perform

its computation task that requires Y_i CPU cycles can be expressed as

$$T_i^l = \frac{Y_i}{f_i^l} \tag{5}$$

and we have E_i^l as the corresponding energy consumption of local computing, which is expressed as

$$E_i^l = z_i Y_i \tag{6}$$

where z_i represents the energy consumption per CPU cycle to complete the computation task.

2) Offload to the BS-MEC Server: In the case of BS-MEC server computing, after the data are transmitted via the cellular network, the processing of the computation task is performed by the BS-MEC server. We denote f^m as the computation capability of the BS-MEC server and we consider that the computation capability f_i^m, assigned to MD i by the BS-MEC server is a nonincreasing function of the number $n(\mathbf{x})$ of MDs that chooses offload to the BS-MEC server, where i belongs to $n(\mathbf{x})$. Given f_i^m, we use a linear model to calculate the execution time of a task $<D_i, Y_i>$ that is offloaded by MD i, $T_i^{m,\text{exe}} = Y_i/f_i^m$. In addition to the execution time, the time delay for transmitting task up to the BS-MEC server will incur an extra overhead. The uplink rate R_i^m together with the data size D_i determines the transmission time $T_i^{m,\text{tran}} = D_i/R_i^m$ of MD i for offloading task to the BS-MEC server. Therefore, the equation for the time function would be written as

$$T_i^m = T_i^{m,\text{exe}} + T_i^{m,\text{tran}} = \frac{Y_i}{f_i^m} + \frac{D_i}{R_i^m}. \tag{7}$$

Compared with MDs, energy resource in the BS-MEC server is abundantly available. Therefore, we only consider the energy consumed for transmitting the computation task from MD i to the BS-MEC server. To model the energy consumption of the MDs, we assume that MD i uses a constant transmit power of P_i for sending the data, thus the energy consumption of MD i for offloading the computation task is

$$E_i^m = \frac{D_i P_i}{R_i^m}. \tag{8}$$

3) Offload to the UAV-MEC Server: In the case of UAV-MEC server computing, MD i will offload its computation task to the nearby UAV-MEC server, and the task will be executed on the flying UAV-MEC server. Similar as above, we denote f^u as the computation capability of the UAV-MEC server and we consider that the computation capability f_i^u, assigned to MD i by the UAV-MEC server is a nonincreasing function of the number $m(\mathbf{x})$ of MDs that chooses offload to the UAV-MEC server, where i belongs to $m(\mathbf{x})$. Given f_i^u, we use a linear model to compute the execution time of a task $<D_i, Y_i>$ that is offloaded by MD i, $T_i^{u,\text{exe}} = Y_i/f_i^u$. In addition to the execution time, the time cost for transmitting task up to the UAV-MEC server will incur an extra overhead. The uplink rate R_i^u together with the data size D_i determines the transmission time $T_i^{u,\text{tran}} = D_i/R_i^u$ of MD i for offloading task to the MEC server. Therefore, the equation for the time function would be written as

$$T_i^u = T_i^{u,\text{exe}} + T_i^{u,\text{tran}} = \frac{Y_i}{f_i^u} + \frac{D_i}{R_i^u}. \tag{9}$$

Compared with the MEC server, energy resource in the UAV-MEC server is limited. Therefore, the energy consumption of task $<D_i, Y_i>$ includes not only the energy consumed for transmitting computation task from MD i to UAV-MEC server but also the energy consumed for UAV hovering and task execution. According to [39], the energy consumption for UAV hovering could be denoted by $E_i^h = P^h/\eta$, where P^h denotes the minimum power for hovering, η denotes power efficiency, during MD i choices to offload computation task to UAV-MEC server. Thus, the equation for the energy consumption would be written as

$$\begin{aligned} E_i^u &= E_i^{u,\text{tran}} + E_i^{u,\text{exe}} + E_i^h \\ &= \frac{D_i P_i}{R_i^u} + \frac{P^u Y_i}{f_i^u} + \frac{P^h}{\eta} \end{aligned} \tag{10}$$

where P^u denotes the energy consumption per second to complete the computation task, that is depending on the chip architecture of the UAV [40].

C. Cost Model

We model the cost of an MD as a linear combination of the energy consumption it takes to finish the computation task and the corresponding time latency. For providing rich flexibility on capturing requirements of MD i, the weighting parameters of time delay and energy consumption for MD i's offloading decision are denoted by $\gamma_i^T, \gamma_i^E \in [0, 1]$, respectively. For example, if MD i is running a latency-sensitive application, then the value of γ_i^T could be set a little higher than γ_i^E. On the contrary, if MD i is at a low-battery state, then the value of γ_i^E could be set a little higher than γ_i^T.

Hence, for the case of local computing, the cost of MD i is determined by the local computing time and the consumed energy per CPU cycle

$$C_i^l = \gamma_i^T T_i^l + \gamma_i^E E_i^l = \left(\frac{\gamma_i^T}{f_i^l} + \gamma_i^E z_i\right) Y_i. \tag{11}$$

For the case of offloading to the BS-MEC server, the cost of MD i is determined by the transmission time, the corresponding energy consumption for transmission, and the execution time in the BS-MEC server

$$\begin{aligned} C_i^m &= \gamma_i^T T_i^m + \gamma_i^E E_i^m \\ &= \left(\gamma_i^T + \gamma_i^E P_i\right) \frac{D_i}{R_i^m} + \gamma_i^T \frac{Y_i}{f_i^m}. \end{aligned} \tag{12}$$

For the case of offloading to the UAV-MEC server, the cost of MD i is determined by the transmission time, the corresponding energy consumption for transmission, the execution time in UAV-MEC server, and the corresponding energy consumption for execution

$$\begin{aligned} C_i^u &= \gamma_i^T T_i^u + \gamma_i^E E_i^u \\ &= \frac{\left(\gamma_i^T + \gamma_i^E P_i\right) D_i}{R_i^u} + \frac{\left(\gamma_i^T + \gamma_i^E P^u\right) Y_i}{f_i^u} + \frac{\gamma_i^E P^h}{\eta}. \end{aligned} \tag{13}$$

Similar to the previous works [33]–[35], we do not model the time needed to transmit the results of the computation back to the MD, as for typical applications, such as face recognition

and speech recognition, the size of the result is much smaller than the size D_i of the input data.

To define the cost of MD i in the offloading decision profile $\mathbf{x}$, we introduce the indicator function $I(x_i, a)$ for MD i as

$$I(x_i, a) = \begin{cases} 1, & \text{if } x_i = a \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

We now express the cost of MD i in the offloading decision profile $\mathbf{x}$ as

$$C_i(\mathbf{x}) = C_i^l I(x_i, 0) + C_i^m I(x_i, 1) + C_i^u I(x_i, 2). \tag{15}$$

Finally, we define the total cost as $C(\mathbf{x}) = \sum_{i=1}^{N} C_i(\mathbf{x})$.

Based on the system model shown above, the computation offloading problem can be formulated as

$$\begin{aligned} \min_{\{\mathbf{x}\}} \ & \sum_{i=1}^{N} C_i(\mathbf{x}) \\ \text{s.t.} \ & C1\text{: } x_i \in \{0, 1, 2\} \quad \forall i \in N \\ & C2\text{: } \sum_{i=1}^{N} \left(E_i^{u,\text{exe}} + E_i^h \right) \le \epsilon \quad \forall i \in N \\ & C3\text{: } f_i^m, f_i^u \ge 0 \quad \forall i \in N \\ & C4\text{: } \sum_{i=1}^{n(\mathbf{x})} B_i^m \le B^m \quad \forall i \in N \\ & C5\text{: } \sum_{i=1}^{m(\mathbf{x})} B_i^u \le B^u \quad \forall i \in N. \end{aligned} \tag{16}$$

In (16), constraint C1 states that each MD can only choose one offloading decision. Constraint C2 guarantees that the energy consumption of the UAV-MEC server must be below a fixed threshold, where ϵ denotes the battery energy budget of the UAV. Constraint C3 means the computation resources allocated to the task of MD i cannot be 0. Constraints C4 and C5 mean the bandwidth resources allocated to MD i is limited. By solving this constrained optimization problem, we can find an optimal computation offloading decision profile $\mathbf{x}$ of all MDs.

Moreover, we can see that if all MDs decide to offload computation tasks to the BS-MEC/UAV-MEC server, competition for resources (i.e., computation and bandwidth resources) will result in significant overhead in terms of time latency and energy consumption. Therefore, the optimal offloading decision profile not only depends on the energy consumption constraints of all MDs but also depends on the competition between MDs for limited wireless bandwidth resources and remote computation resources. It is noticed that problem (16) can be reduced to the classical maximum cardinality bin packing problem, where the computation tasks offloaded by MDs are items to be packed, and the two MEC servers (i.e., BS-MEC and UAV-MEC) are bins. Thus, problem (16) is NP-hard. To obtain the optimal value of $\mathbf{x}$, we will propose three algorithms in the next section.

III. Problem Solution

In this section, we consider the issue of how to find the best offloading decision profile for the MDs. MDs face tradeoff, which means that the offloaded tasks could be executed either on the fixed MEC server or the flying UAV-MEC server. Although the MDs are independent of each other, different offloading decisions are mutually related. This means that one MD's decision will have a direct impact on other MDs' decisions throughout the network. On the one hand, from the perspective of MDs, offloading computation tasks to the remote servers can save their energy cost. On the other hand, if too many MDs choose to offload their computation tasks simultaneously, the competition about computing resources and wireless communication resources will be fierce. Therefore, this may lead to low performance of the offloaded computation tasks. In this case, it would be more efficient to execute the tasks locally. Based on this insight, we first define the concept of beneficial UAV-MEC computing.

Algorithm 1 TMCO

Input: N, D_i, f_i^l, f^m, f^u, z_i, P_i, B^m, B^u, P^u, σ, $\forall i \in N$.

Output: an optimal offloading decision profile $\mathbf{x}^{tm}$, and the minimum system total cost C_{min}.

1: traverse all possible offloading decision profiles $\mathbf{x}$ of MDs;
2: discard those offloading decision profiles that fail to meet the given energy consumption constraints of the computation tasks;
3: calculate the system total cost of remaining offloading decision profiles, and find an overall optimal offloading decision profile $\{\mathbf{x}^{tm}\}$, which has the minimum system total cost C_{min};
4: output $\mathbf{x}^{tm}$ and C_{min}.

Definition 1: Given a computation offloading decision profile $\mathbf{x}$, the offloading decision x_i of MD i that chooses the UAV-MEC computing option is *beneficial* if the UAV-MEC computing option does not incur higher cost than the local computing option or the BS-MEC computing option (i.e., $C_i^u \le C_i^l \& C_i^u \le C_i^m$).

The concept of beneficial UAV-MEC computing plays an important role in the UAV-assisted MEC system. It shows how much load the UAV-MEC server shares from the BS-MEC server. Moreover, it helps to balance the tradeoff between latency and energy consumption for MDs. Next, in order to solve the problem formulated in Section II, we present in more detail our proposed solutions in the following sections.

A. Traversal Method-Based Computation Offloading Solution

To solve problem (16), we first present a traversal method-based computation offloading (TMCO) solution, which is the easiest approach to come up with. We just only need to traverse all possible offloading decision profiles of the MDs, and find the optimal offloading decision profile $\mathbf{x}^{tm}$ which has the minimum system total cost. Detailed procedures of TMCO are briefly summarized in Algorithm 1.

For TMCO in Algorithm 1, we have the following conclusions.

Proposition 1: Algorithm 1 is convergent and it is able to obtain an optimal solution of the problem (16).

Proof: Since TMCO traverses the whole solution space and enumerates all possible computation offloading decision

profiles, it is obvious that TMCO is convergent and it is able to obtain an optimal solution of the problem (16). ■

TMCO in Algorithm 1 is simple and can give an optimal solution of the problem (16), but it has a very high-computation complexity. The upper bound of its computation complexity is given as follows.

Proposition 2: The computation complexity of TMCO is $O(3^n)$, where n is the number of MDs.

Proof: For TMCO, the running time of step 1 is $O(3^n)$ since there are three offloading choices for each MD, i.e., executing the task locally, offloading to the fixed BS-MEC server, and offloading to the flying UAV-MEC server. In steps 2 and 3, in order to discard those offloading decision profiles that fail to meet the given energy consumption constraints and to find an optimal offloading decision profile with the minimum system total cost, we have to traverse all the possible offloading decision profiles, and the running time is also $O(3^n)$. Therefore, the computation complexity of TMCO can be calculated by $O(3^n)$, where n is the number of MDs. ■

The TMCO has to traverse all possible offloading decision profiles and calculate their corresponding system total cost so as to find an optimal offloading profile $\mathbf{x}^{tm}$. Thus, as the number of MDs increases, TMCO is unworkable. Therefore, it is just presented as a benchmark for our following schemes.

B. Greedy-Based Sequential Tuning Solution

As the computation complexity of TMCO is too high, when the number of MDs increases, it is unworkable for TMCO to solve the computation offloading problem (16). Thus, we propose another computation offloading solution, i.e., a greedy-based sequential tuning computation offloading (STCO) scheme, which has lower computation complexity. The details of STCO are given in Algorithm 2.

In particular, Algorithm 2 mainly consists of two stages. In the first stage, the offloading decision profile is initialized via a greedy-based scheme. First, the energy consumption of each task is calculated by LE. For MD i, if $e_i^{\text{loc}} \le e_i^{\max}$, we set the offloading decision of MD i as $x_i = 0$. Then, we pick out the computation tasks that cannot be executed locally, i.e., $e_i^{\text{loc}} > e_i^{\max}$. For those computation tasks, we calculate their time delay separately by offloading to the BS-MEC server or UAV-MEC server, i.e., t_i^{mec} and t_i^{uav}, and choose the offloading decisions with the lower latency. After that, the offloading decision profile is initialized as $\mathbf{x}'$.

In the second stage, to further reduce the system total cost and eventually achieve a local optimal solution for problem (16), we propose a sequential tuning procedure starting from $\mathbf{x}'$. Set $\mathbf{x}'$ as the initial point. For MD $m \in N$, sequentially perform the following two steps.

1) Randomly order the lists of all MDs.
2) Traverse the MD list one by one. For MD $m \in N$, sequentially adjust its offloading decision x_m for the three possible offloading decisions, while the offloading decisions x_{-m} of all other MDs remain unchanged. For each x_m, find the minimum system total cost by solving problem (16). When an MD is found to achieve a lower system total cost by adjusting the offloading decision of its task, update $\mathbf{x}^{st*}$ to the new offloading decision profile that gives the lower cost.

Repeat steps 1) and 2) until the offloading decision profile $\mathbf{x}^{st*}$ converges, i.e., no change for $\mathbf{x}^{st*}$ can be made. Then, output the solution of the STCO algorithm as $\mathbf{x}^{st} = \mathbf{x}^{st*}$.

Algorithm 2 Greedy-Based STCO Algorithm

Input: N, D_i, f_i^l, f^m, f^u, z_i, P_i, B^m, B^u, P^u, σ, $\forall i \in N$.
Output: an optimal offloading decision profile $\mathbf{x}^{st}$, and the minimum system total cost C_{min}.

Stage 1: Initial offloading profile via Greedy
for $i \in N$ **do**
  Calculate e_i^{loc} of each MD separately;
  if $e_i^{loc} \le e_i^{max}$ **then**
    $x_i = 0$;
  else
    Calculate t_i^{mec}, t_i^{uav} without wireless interference;
    if $t_i^{mec} \le t_i^{uav}$ **then**
      $x_j = 1$;
    else
      $x_j = 2$;
    end if
  end if
end for
Stage 2: Sequential Tuning
Set $j = 1$, $\mathbf{x}' = \{x_1, x_2, \cdots, x_N\}$
while $j \le N$ **do**
  Randomly order the list of all MDs and their tasks;
  Set task index $m = 1$;
  Calculate system total cost C' corresponding to $\mathbf{x}'$;
  while $m \le N$ **do**
    While keeping x_{-m} unchanged except x_m, inspect the three possible offloading choices of x_m; find their respective system total costs by solving problem (16); set $C^{st'}$ as the minimum cost among these three choices, and record the corresponding decision profile as $\mathbf{x}^{st'}$;
    if $C^{st'} < C'$ **then**
      Set $\mathbf{x}^{st*} = \mathbf{x}^{st'}$, $C^{st*} = C^{st'}$;
      $m \leftarrow m + 1$;
    else if $m = N$ **then**
      $j \leftarrow j + 1$, $m \leftarrow j$;
    else
      $m \leftarrow m + 1$;
    end if
  end while
end while
output $\mathbf{x}^{st} = \mathbf{x}^{st*}$ and $C_{min} = C^{st*}$.

For STCO in Algorithm 2, we have the following conclusions.

Proposition 3: Algorithm 2 is convergent and it can give a near-optimal solution of the problem (16).

Proof: According to stage 2 in Algorithm 2, STCO adopts a sequential tuning strategy to find the final offloading decision profile, by adopting which the system total cost is minimum. The stage eventually will reach some $\mathbf{x}^{st}$, where the system

total cost cannot be further reduced by adjusting any MD's offloading decision. This is because there is a finite number of possible values for x_i. Since it gives the lowest cost in a finite value of vector $\mathbf{x}$, it is straightforward to show that STCO is convergent and $\mathbf{x}^{st}$ is a near-optimal solution of problem (16). ■

Compared to TMCO, although STCO can only give a near-optimal solution to the problem (16), it has a much lower computation complexity and can be implemented with a large number of MDs in practice. In particular, the following conclusion can be easily proved for STCO.

Proposition 4: The computation complexity of STCO in Algorithm 2 is $O(n^2)$, where n is the number of MDs.

Proof: The running time of STCO mainly consists of two parts. Specifically, the running time of stage 1 is $O(n)$. For steps 1–13, the for loop runs at most n times, and the running time of the stage 1 is $O(n)$. Moreover, in stage 2, for the steps of 15–30, the outer while loop runs at most n times, and the running time of the inner while loop is $O(n)$. Thus, the computation complexity of stage 2 is $O(n \cdot n) = O(n^2)$. Finally, the whole computation complexity of STCO in Algorithm 2 can be calculated by adding them up, i.e., $O(n^2)$. ■

C. Game-Theoretic Computation Offloading Solution

However, STCO is a centralized optimal solution and requires complete information, so a centralized control center is needed to manage computation offloading across the whole network and collect local parameters of MDs. For the scenarios with multiple wireless BS-MEC/UAV-MEC servers, STCO may raise privacy issues and increase the system cost of massive information collection. Thus, a decentralized solution without collecting massive parameters of MDs is required.

The game theory is a powerful tool for analyzing the interactions between multiple independent entities that need to work together to achieve their own goals. In the game theory, players should organize themselves into mutually satisfactory solutions so that no player has the incentive to unilaterally deviate. We can regard each MD as a rational game player who reacts to other players' offloading decisions by carrying out his current optimal offloading decision. Therefore, to design a decentralized solution of the problem (16), we propose a GTCO scheme.

We thus consider that the MDs play a strategic game $\Gamma =< N, \{X_i\}_{i\in N}, \{C_i\}_{i\in N} >$, where N is the number of MDs in the system, $X_i = \{0, 1, 2\}$ is the strategy space, and C_i is the cost function to be minimized by MD i. Details about the possible values of C_i are shown as follows:

$$C_i(x_i, x_{-i}) = \begin{cases} C_i^l, & \text{if } x_i = 0 \\ C_i^m, & \text{if } x_i = 1 \\ C_i^u, & \text{if } x_i = 2. \end{cases} \tag{17}$$

We refer to the game as the computation offloading game, and each MD aims at minimizing its own cost (15), i.e., it aims at finding an offloading decision

$$x_i^* \in \underset{x_i\in\{0,1,2\}}{\arg\min}\ C_i(x_i, x_{-i}) \tag{18}$$

where $x_{-i} = \{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_N\}$ represents the offloading decisions of all MDs except MD i. We are interested in whether an NE of the game Γ exists, i.e., no MD can further decrease its cost by changing its offloading decision.

Definition 2: An NE of the strategic game $\Gamma = <N, \{X_i\}_{i\in N}, \{C_i\}_{i\in N}>$ is an offloading decision profile x^* such that

$$C_i\left(x_i^*, x_{-i}^*\right) \le C_i\left(x_i, x_{-i}^*\right) \quad \forall x_i \in X_i.$$

In order to prove the existence of NE in game Γ, we resort to a powerful tool, the potential game [36]. Since the potential game has at least one NE solution [37], we can prove whether game Γ is a potential game. The definition of a potential game is described in the following.

Definition 3: A game $\Gamma = <N, \{X_i\}_{i\in N}, \{C_i\}_{i\in N}>$ is a potential game if there exists a potential function P such that for all $i \in N$ and all $x_i, x_i' \in X_i$, $C_i(x_i, x_{-i}) - C_i(x_i', x_{-i}) = P(x_i, x_{-i}) - P(s_i', x_{-i})$.

Proof: A game Γ is a potential game if its cost function $\{C_i\}_{i\in N} >$ can be expressed as a potential function $P(x_i, x_{-i})$. This latter is defined as

$$\begin{aligned} P(x_i, x_{-i}) - P\left(x_i', x_{-i}'\right) &= C_i(x_i, x_{-i}) - C_i\left(x_i', x_{-i}'\right) \\ P(x_i, x_{-i}) &= \arg\min\nolimits_{x_i\in X_i} C_i(x_i, x_{-i}) = \psi_i(x_{-i}) \\ P\left(x_i', x_{-i}'\right) &= \psi_i'(x_{-i}) \end{aligned}$$

where $\psi_i(x_{-i})$ is the best possible reply for a player i given the offloading decision profile x_{-i}. It is also defined as a best response potential game [38], which is equal to

$$\psi_i(x_{-i}) = \begin{cases} \arg\min_{x_i\in X_i} C_i^l, & \text{if } x_i = 0 \\ \arg\min_{x_i\in X_i} C_i^m, & \text{if } x_i = 1 \\ \arg\min_{x_i\in X_i} C_i^u, & \text{if } x_i = 2. \end{cases} \tag{19}$$

$\Gamma =< N, \{X_i\}_{i\in N}, \{C_i\}_{i\in N} >$ is a potential game since function (19) satisfies the definition of a potential function and it provides an optimal solution that ensures the best tradeoff between the local cost and system total cost. Therefore, the NE solution exists and is unique. It is equal to $\psi_i(x_{-i})$. ■

The details of GTCO are described in Algorithm 3. During each iteration in GTCO, all MDs first get the current network status and make their current best offloading decisions. Then, MDs contend for the decision update opportunity. If MD j wins the update opportunity, it performs the update step. We say the offloading decision x_j' is an update step for MD j, if $C_j(x_j', x_{-j}) < C_j(x_j, x_{-j})$. Furthermore, we say that a decision x_j^* is the best reply to x_{-j} if it solves (18). Obviously, in NE, all MDs play their best replies to each others' strategies. After a number of iterations, when all the MDs play their best reply, which means that no MD needs to perform the update step, that is the whole system reaches NE. An optimal offloading decision profile $\mathbf{x}^{gt}$ can be obtained.

For GTCO in Algorithm 3, we have the following conclusions.

Proposition 5: Algorithm 3 is convergent and it can give a near-optimal solution to the problem (16).

Proof: In every iteration of Algorithm 3, each MD first makes its own offloading decision according to the current network status. Then, MD j that wins the decision update

Algorithm 3 GTCO Algorithm

Input: N, D_i, f_i^l, f^m, f^u, z_i, P_i, B^m, B^u, P^u, σ, $\forall i \in N$.
Output: an optimal offloading decision profile $\mathbf{x}^{gt}$, and the minimum system total cost C_{min}.

1: Initialize $x_i = 0, \forall i \in N$, and record the corresponding decision profile as $\mathbf{x}^*$;
2: Calculate the initial value of cost function C^*;
3: **repeat**
4: **for** $i \in N$ **do**
5: Get the current network status;
6: Select the new best decision profile $\mathbf{x}'$
7: Calculate the corresponding value of cost function C';
8: **if** $C' \leq C^*$ **then**
9: $C^* = C'$, $\mathbf{x}^* = \mathbf{x}'$, and store the MDs into U;
10: **end if**
11: **end for**
12: **if** $U \neq \emptyset$ **then**
13: each MD in U contends for the decision update opportunity;
14: **if** MD j wins the decision update opportunity **then**
15: Update x_j in $\mathbf{x}^*$;
16: Broadcast the update message to other MDs;
17: **else**
18: Keep x_j unchanged;
19: **end if**
20: **end if**
21: **until** an Equilibrium is achieved;
22: Output $\mathbf{x}^{gt} = \mathbf{x}^*$, $C_{min} = C^*$.

opportunity decides its best reply to other MDs' offloading decision profile x_{-i}. Since the best reply of each MD is only a local optimal decision and the possible values for x_j in Algorithm 3 are finite. Therefore, GTCO in Algorithm 3 is convergent and it can only give a near-optimal solution to the problem (16). ■

Proposition 6: The computation complexity of Algorithm 3 is $O(n)$, where n is the number of MDs.

Proof: Algorithm 3 has two layers of loops in total. The outer loop, which denotes the iteration process of the MDs for reaching to the NE state, converges in finite times. Moreover, the inner for loop of steps 4–11 runs at most n times, and the process from step 12 to 20 takes constant time, i.e., $O(1)$. Finally, the total computation complexity of GTCO can be calculated by $O(n)$, where n is the number of MDs. ■

IV. Simulation Results

In this section, we evaluate the performance of our proposed computation offloading schemes, i.e., 1) TMCO; 2) STCO; and 3) GTCO, through MATLAB simulation under different parameter settings.

A. Simulation Setup

We adopt the MD characteristics from [40], which is based on a Nokia smart device. Each MD has only one independent task. According to [40, Tables 1 and 3], the MD has CPU rate 500×10^6 cycles/s and the LE energy consumption per cycle $[1/(730 \times 10^6)]$ J/cycle. We consider the x264 CBR encode application, which requires 1900 cycle/B, i.e., $Y(i) = 1900D(i)$. The input data sizes of each task are assumed to be uniformly distributed from 10 to 30 MB.

TABLE I
Parameter Settings in the Simulation

notation	description	value
N	number of MDs	60
B^m	wireless channel bandwidth between MD and MEC server	4 MHz
B^u	wireless channel bandwidth between MD and UAV-MEC server	40 MHz
P_i	transmission power of the MD	100 mW
σ^2	background noise	-100 dBm
D_i	input computation data size of task	uniformly distributed from 10 to 30 MB
z_i	consumed energy per CPU cycle	$\frac{1}{730\times10^6}$ J/cycle
e_i^{max}	maximum permissible energy consumption for local execution	50 J
γ_i^E	weight of energy consumption	0.5
γ_i^T	weight of time latency	0.5 J/s

The transmission bandwidth between MD i and BS-MEC/UAV-MEC server is set to 4/40 MHz, respectively. The transmission energy consumptions of the MD is 1.42×10^{-7} J/b as indicated in [40, Table 2]. The transmission power P_i is set to be 100 mW and the background noise $\sigma^2 = -100$ dBm. Here, we assume that the UAV flies at a fixed height $H = 100$ m. The computation capability of the BS-MEC server is 10×10^9 cycles/s and the computation capability of UAV-MEC is 3×10^9 cycles/s. For normalizing the values of energy consumption and time latency, the weights attributed to energy consumption γ_i^E and time latency γ_i^T were set to be $\gamma_i^E = 0.5$ J/s and $\gamma_i^T = 0.5$ J/s, which means that energy consumption and time latency are equally important. Finally, all simulation results are obtained by averaging more than 100 realizations of the input data sizes of each task. Table I lists the key parameters adopted in our simulation.

B. Numerical Results

To demonstrate the efficiency and the effectiveness of our proposed three computation offloading schemes, we carry out the simulations in terms of system total cost and running time. Considering TMCO's high-computation complexity, the number of MDs is only set from 3 to 12. In Figs. 2 and 3, the system total cost and the running time are shown with different number of MDs. It can be observed from Fig. 2 that since TMCO is able to obtain the optimal solution to the problem (16), both STCO and GTCO can achieve near-optimal solutions. In addition, we also find that after many times of simulations, STCO and GTCO have almost the same performance, when the number of MDs is small. Furthermore, from Fig. 3, we can see that the running time of TMCO is exponentially increasing, both STCO and GTCO have higher computation efficiency. The above simulation results verify our previous Propositions 2–6.

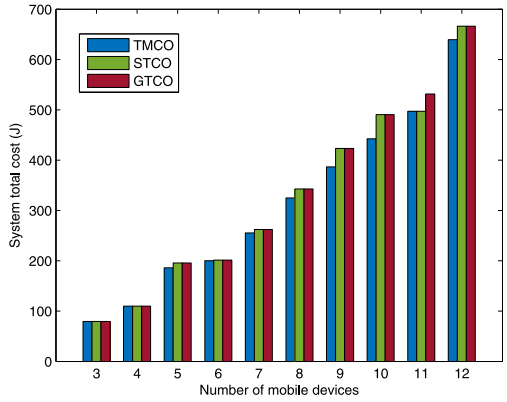


Fig. 2. System total cost with different number of MDs among TMCO, STCO, and GTCO.

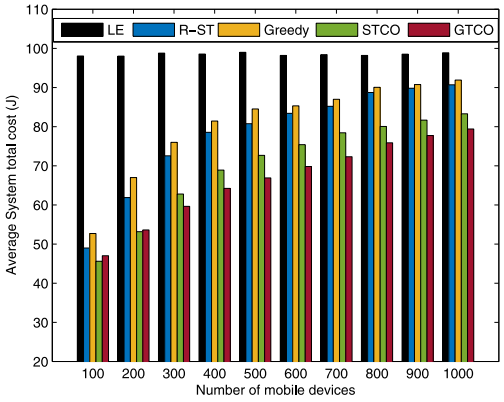


Fig. 4. Average system total cost for variable number of MDs by considering a task generation probability equal to 0.1.

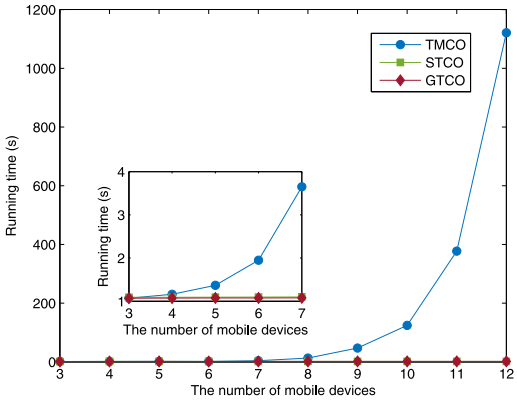


Fig. 3. Running time with different number of MDs among TMCO, STCO, and GTCO.

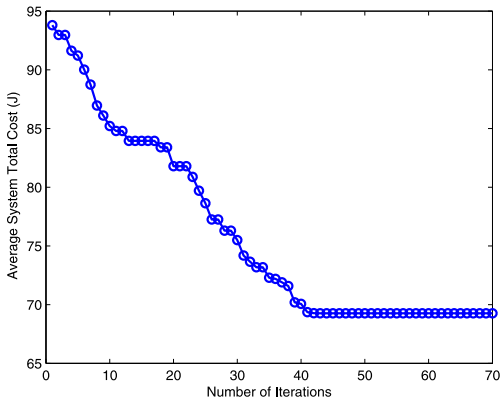


Fig. 5. Dynamics of the average system total cost of GTCO.

Moreover, in order to further verify the performance of our proposed STCO and GTCO scheme, we introduce another three baseline computation offloading policies: 1) LE policy, which executes all computation tasks locally at MDs; 2) a random-based sequential tuning computation offloading (R-ST) policy, where MDs make the offloading decisions randomly and adjust offloading decisions in sequence; 3) a greedy-based computation offloading (Greedy) policy, where each MD only makes the offloading decision with the minimum cost among those three offloading options. We run experiments with different numbers of MDs, respectively.

Fig. 4 shows the average system total cost for an increasing number of MDs by considering a task generation probability equal to 0.1. From Fig. 4, we can easily observe that R-ST, Greedy, STCO, and GTCO can achieve a lower average system total cost than the LE policy, and these results show the necessity of computation offloading. We can also observe that with the number of MDs increases, the benefits brought by computation offloading will decrease. This means that with the number of MDs increases, competition for communication and computation resources becomes more intense. Among the other four computation offloading policies, the greedy-based policy has the worst performance since this policy assumes that each MD only considers its own cost, which leads to poor overall performance. The random-based policy has a little better performance than the greedy-based policy. Furthermore, after many times of simulations, we find that both of our proposed STCO and GTCO schemes can achieve the best performance compared with the other three policies. While for STCO, the complete information is required and hence the server needs to collect the local parameters of the MDs. This would incur a higher system total cost than GTCO, thus, with the number of MDs increases the performance of STCO is worse than GTCO. Furthermore, privacy issues are worthy of attention, when collecting MDs' information. Therefore, considering that there will be tens of thousands of MDs and multiple BS-MEC/UAV-MEC servers in the real world, the STCO scheme is not suitable for providing a solution to the problem (16). Meanwhile, due to the property of NE, GTCO can ensure performance and the self-stability without considering privacy issues.

As stated above, compared with the baseline schemes, the decentralized GTCO scheme is more practical and can be easily extended to the multiserver–multiuser scenarios. We then evaluate the GTCO scheme could converge to an equilibrium. We run experiments with $N = 60$ MDs randomly scattered over the objective area.

Fig. 5 shows the dynamics of the average system total cost with the number of iterations increasing. It can be observed from the figure that GTCO can also keep the average system total cost decreasing and converge to an equilibrium. Fig. 6 shows the dynamics of the achieved number of beneficial

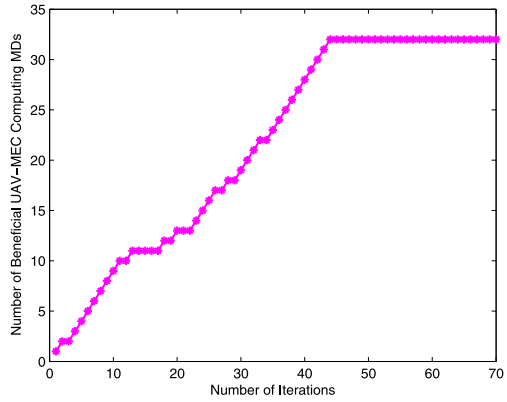


Fig. 6. Dynamics of the number of beneficial UAV-MEC computing users of GTCO.

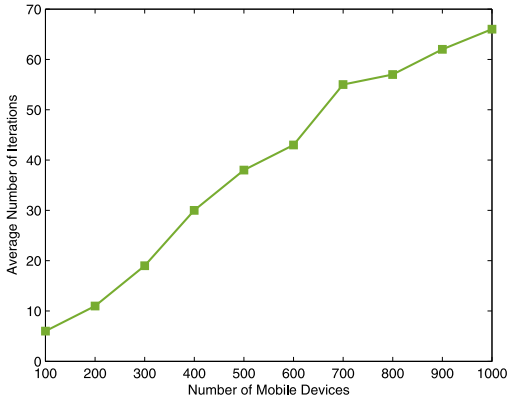


Fig. 7. Average number of iterations for variable number of MDs of GTCO by considering a task generation probability equal to 0.1.

UAV-MEC computing users with the number of iterations increasing. Obviously, as time goes by, the number of beneficial UAV-MEC computing users keeps increasing in the system and converges to an equilibrium finally.

We next evaluate the convergence stability of GTCO, when the number of MDs increases. Fig. 7 shows the relationship between the average number of iterations and the number of MDs, by considering a task generation probability equal to 0.1. From the figure, we can see that as the number of MDs increases, the average number of iterations increases linearly. These results demonstrate that GTCO can converge fast and has strong scalability in practice.

V. Conclusion

In this article, we considered a UAV-assisted MEC system consisting of one BS-MEC server, one UAV-MEC server, and multiple MDs, where each MD has only one independent indivisible task. To minimize a weighted system total cost of energy consumption and time latency, we aimed to find the overall optimal computation offloading decisions of the whole MDs. A game-theoretic scheme (i.e., GTCO) was proposed as our solution, and the existence of NE has been proved either. In order to evaluate the efficiency and effectiveness of GTCO, another two schemes (i.e., TMCO and STCO) were proposed as the benchmarks. By comparison with different parameters, the simulation results showed that the proposed GTCO algorithm and the obtained NE often achieved the minimum average system total cost and confirmed the low-computation complexity.

References

[1] B. P. Rimal, D. P. Van, and M. Maier, "Mobile edge computing empowered fiber-wireless access networks in the 5G era," *IEEE Commun. Mag.*, vol. 55, no. 2, pp. 192–200, Feb. 2017.

[2] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, 2013.

[3] C. C. Coskun, K. Davaslioglu, and E. Ayanoglu, "Three-stage resource allocation algorithm for energy-efficient heterogeneous networks," *IEEE Trans. Veh. Technol.*, vol. 66, no. 8, pp. 6942–6957, Aug. 2017.

[4] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. Heinzelman, "Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Cappadocia, Turkey, 2012, pp. 59–66.

[5] A. U. R. Khan, M. Othman, S. A. Madani, and S. U. Khan, "A survey of mobile cloud computing application models," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 393–413, 1st Quart., 2014.

[6] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: Architecture, applications, and approaches," *Wireless Commun. Mobile Comput.*, vol. 13, no. 18, pp. 1587–1611, 2013.

[7] M. V. Barbera, S. Kosta, A. Mei, and J. Stefa, "To offload or not to offload? the bandwidth and energy costs of mobile cloud computing," in *Proc. IEEE INFOCOM*, Turin, Italy, 2013, pp. 1285–1293.

[8] L. Yang, J. Cao, Y. Yuan, T. Li, A. Han, and A. Chan, "A framework for partitioning and execution of data stream applications in mobile cloud computing," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 40, no. 4, pp. 23–32, 2013.

[9] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1657–1681, 3rd Quart., 2017.

[10] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Netw. Appl.*, vol. 18, no. 1, pp. 129–140, 2013.

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

[12] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 1, no. 2, pp. 89–103, Jun. 2015.

[13] F. Zhou, Y. Wu, H. Sun, and Z. Chu, "UAV-enabled mobile edge computing: Offloading optimization and trajectory design," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Kansas City, MO, USA, 2018, pp. 1–6.

[14] S. W. Loke, "The Internet of flying-things: Opportunities and challenges with airborne fog computing and mobile cloud in the clouds," 2015. [Online]. Available: arXiv:1507.04492.

[15] S. Jeong, O. Simeone, and J. Kang, "Mobile edge computing via a UAV-mounted cloudlet: Optimization of bit allocation and path planning," *IEEE Trans. Veh. Technol.*, vol. 67, no. 3, pp. 2049–2063, Mar. 2018.

[16] L. Ma, M. Li, L. Tong, Y. Wang, and L. Cheng, "Using unmanned aerial vehicle for remote sensing application," in *Proc. 21st Int. Conf. Geoinformat.*, Kaifeng, China, 2013, pp. 1–5.

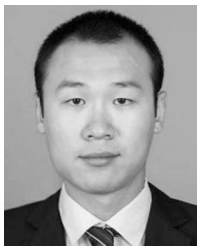
[17] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Mobile unmanned aerial vehicles (UAVs) for energy-efficient Internet of Things communications," *IEEE Trans. Wireless Commun.*, vol. 16, no. 11, pp. 7574–7589, Nov. 2017.

[18] H. Zhao, H. Wang, W. Wu, and J. Wei, "Deployment algorithms for UAV airborne networks toward on-demand coverage," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 9, pp. 2015–2031, Sep. 2018.

[19] S. Kumar, S. Suman, and S. De, "Backhaul and delay-aware placement of UAV-enabled base station," in *Proc. IEEE INFOCOM Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Honolulu, HI, USA, 2018, pp. 634–639.

[20] M. Moradi, K. Sundaresan, E. Chai, S. Rangarajan, and Z. M. Mao, "SkyCore: Moving core to the edge for untethered and reliable UAV-based LTE networks," in *Proc. 24th Annu. Int. Conf. Mobile Comput. Netw.*, 2018, pp. 35–49.

[21] P. Vamvakas, E. E. Tsiropoulou, and S. Papavassiliou, "On the prospect of UAV-assisted communications paradigm in public safety networks," in *Proc. IEEE INFOCOM Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Paris, France, 2019, pp. 762–767.
[22] A. Merwaday and I. Guvenc, "UAV assisted heterogeneous networks for public safety communications," in *Proc. IEEE Wireless Commun. Netw. Conf. Workshops (WCNCW)*, New Orleans, LA, USA, 2015, pp. 329–334.
[23] M. Dong, K. Ota, M. Lin, Z. Tang, S. Du, and H. Zhu, "UAV-assisted data gathering in wireless sensor networks," *J. Supercomput.*, vol. 70, no. 3, pp. 1142–1155, 2014.
[24] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Efficient deployment of multiple unmanned aerial vehicles for optimal wireless coverage," *IEEE Commun. Lett.*, vol. 20, no. 8, pp. 1647–1650, Aug. 2016.
[25] X. Cao, J. Xu, and R. Zhang, "Mobile edge computing for cellular-connected UAV: Computation offloading and trajectory optimization," in *Proc. IEEE 19th Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, Kalamata, Greece, 2018, pp. 1–5.
[26] Q. Hu, Y. Cai, G. Yu, Z. Qin, M. Zhao, and G. Y. Li, "Joint offloading and trajectory design for UAV-enabled mobile edge computing systems," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1879–1892, Apr. 2019.
[27] J. Zhang *et al.*, "Stochastic computation offloading and trajectory scheduling for UAV-assisted mobile edge computing," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3688–3699, Apr. 2019.
[28] F. Zhou, Y. Wu, R. Q. Hu, and Y. Qian, "Computation rate maximization in UAV-enabled wireless-powered mobile-edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 9, pp. 1927–1941, Sep. 2018.
[29] M. Alsenwi, Y. K. Tun, S. R. Pandey, N. N. Ei, and C. S. Hong, "UAV-assisted multi-access edge computing system: An energy-efficient resource management framework," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Barcelona, Spain, 2020, pp. 214–219.
[30] M.-A. Messous, S.-M. Senouci, H. Sedjelmaci, and S. Cherkaoui, "A game theory based efficient computation offloading in an UAV network," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 4964–4974, May 2019.
[31] M. Thomas and J. A. Thomas, "Elements of information theory," *Publ. Amer. Stat. Assoc.*, vol. 103, no. 481, p. 429, 1992.
[32] S. Jošilo, G. Dán, "A game theoretic analysis of selfish mobile computation offloading," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Atlanta, GA, USA, 2017, pp. 1–9.
[33] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, Apr. 2015.
[34] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Trans. Wireless Commun.*, vol. 11, no. 6, pp. 1991–1995, Jun. 2012.
[35] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, Apr. 2010.
[36] Z. M. Fadlullah, C. Wei, Z. Shi, and N. Kato, "GT-QoSec: A game-theoretic joint optimization of QoS and security for differentiated services in next generation heterogeneous networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 2, pp. 1037–1050, Feb. 2017.
[37] 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.
[38] D. Monderer and L. S. Shapley, "Potential games," *Games Econ. Behav.*, vol. 14, no. 1, pp. 124–143, 1996.
[39] K. Kim and C. S. Hong, "Optimal task-UAV-edge matching for computation offloading in UAV assisted mobile edge computing," in *Proc. 20th Asia–Pac. Netw. Oper. Manag. Symp. (APNOMS)*, Matsue, Japan, 2019, pp. 1–4.
[40] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proc. 2nd Usenix Conf. Hot Topics Cloud Comput.*, vol. 10, 2010, p. 4.



Kaiyuan Zhang received the B.S. degree from the School of Computer and Information Engineering, Henan University of Economics and Law, Zhengzhou, China, in 2013. He is currently pursuing the Ph.D. degree with the Shaanxi Province Key Laboratory of Computer Network, Xi'an Jiaotong University, Xi'an, China.

His current research interests are computation offloading, resource allocation, and mobility management in multiaccess edge computing.

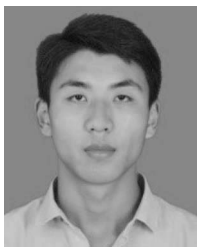
Xiaolin Gui received the Ph.D. degree in computer science from Xi'an Jiaotong University, Xi'an, China, in 2001.

He is currently a Professor and serves as a Deputy Dean of the School of Electronic and Information, Xi'an Jiaotong University since 2013, where he has been also the Director of the Key Laboratory of Computer Network since 2008. His recent research covers high-performance computing, secure computation of open network systems, dynamic trust management theory, and development on community network.

Prof. Gui is a recipient of the New Century Excellent Talents in Universities of China.

Dewang Ren received the B.S. degree from the School of Automation and Electrical Engineering, Lanzhou Jiaotong University, Lanzhou, China, in 2012, and the M.S. degree from the School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing, China, in 2015. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, Xi'an Jiaotong University, Xi'an, China.

His current research interests are mobile cooperative edge caching, mobility management, and resource optimization.



Defu Li received the B.S. degree from the School of Computer Science and Technology, Guizhou University, Guiyang, China, in 2018. He is currently pursuing the M.S. degree with the Shaanxi Province Key Laboratory of Computer Network, Xi'an Jiaotong University, Xi'an, China.

His current research interests include natural language processing, deep learning, and mobile-edge computing.