IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 8, 2022, accepted March 2, 2022, date of publication March 14, 2022, date of current version March 18, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3158319

Joint Social-Aware and Mobility-Aware Computation Offloading in Heterogeneous Mobile Edge Computing

CHENGLIN XU[1], CHENG XU[1], BO LI[2], (Graduate Student Member, IEEE), SIQI LI[1], AND TAO LI[1]

[1]College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China
[2]School of Computer Science and Engineering, Central South University, Changsha 410083, China

Corresponding author: Cheng Xu (chengxu@hnu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61772185, in part by the Hunan Leading Plan for Scientific and Technological Innovation of High-Tech Industries under Grant 2020GK2037, and in part by the Fundamental Research Funds for the Central Universities of Central South University under Grant 2019zzts282.

ABSTRACT With increasing computation-intensive tasks of various applications running on mobile devices, the limitation of computing resources and battery capacity on mobile devices makes it impossible to meet the users' Quality of Service (QoS). Fortunately, with the emergence of mobile edge computing (MEC), mobile devices can offload tasks to edge servers to efficiently solve the above problems. However, meeting the users' QoS requirements with the help of deployed MEC facilities is still challenging since mobile users' service demands vary depending on their dynamic location. In addition, increasing the number of edge servers to meet the requirements of applications would burden the initial investment and maintenance fee accordingly. In this case, using idle resources from nearby mobiles may become an effective solution. Most of the existing works do not consider the mobility of devices and users' willingness to share. Therefore, in this paper, we propose the mobile device selection algorithms (MDSA), in which the social relationship, location correlation, and mobile activity of mobile devices were considered in the selection of target mobile devices, providing device-to-device offloading. In addition, we propose the joint social-aware and mobility-aware computation offloading algorithm (JSMCO) based on the improved Kuhn–Munkres (KM) algorithm to obtain a resource allocation strategy that minimizes the energy consumption while satisfying the minimum latency condition. The proposed algorithms have been verified to reduce the offloading success rate and decrease the users' time and energy consumption in extended real datasets.

INDEX TERMS Computation offloading, social-aware, mobile edge computing, social networks, mobility.

I. INTRODUCTION

Augmented reality, natural language processing, virtual reality [1]–[3], and other similar applications require high computing capability; thus, they have higher performance requirements for mobile devices. However, due to the limitation of the physical size of mobile devices, the resources (computing resources and battery capacity) of mobile devices are usually limited [4]. Cloud computing [5] allows users to migrate computation-intensive tasks to the cloud, thus improving the performance experience and reducing the energy consumption of mobile devices. However, the service quality of cloud computing is significantly affected by the network status. Moreover, as the number of tasks increases, the processing delay also increases. As a result, offloading tasks from mobile devices to the cloud are faced with high latency.

Mobile edge computing (MEC) is an appropriate solution to the problem in cloud computing [6], [7]. An emerging paradigm reduces latency by offloading tasks to edge servers close to users rather than to cloud servers [8]. Because MEC is implemented at the edge of the network, it provides low latency and flexible computing services for device users. However, there are more mobile devices and more offloading task requirements in urban hotspots, and the processing capacity of the edge server cannot be dynamically adjusted

The associate editor coordinating the review of this manuscript and approving it for publication was Pedro R. M. Inacio.

according to the users' needs. If many edge servers are deployed, hardware resources will be wasted. Conversely, if the deployed edge servers are insufficient, they cannot cope with the requirements for task offloading. There are many idle computing resources among mobile devices. Suppose the tasks that the edge server cannot handle can be offloaded to idle mobile devices. In this case, the problem of insufficient processing capacity of the edge server can be solved [9], [10].

However, offloading tasks to nearby mobiles may consume their valuable battery power and computing resources as well as increase security risk, raising the issue of low willingness to share resources among mobile users [11], [12]. Therefore, promoting cooperation and improving user participation have become important issues [13]–[15]. On the other hand, the location of mobile devices varies, and each mobile device has different computing resources to use. Task offloading will fail when the selected device cannot complete the calculation or return the results on time; thus, finding a suitable mobile device to complete the task of computing offloading is also a complex problem. In addition, when dealing with many tasks with different requirements, allocating them efficiently is also a problem that needs to be solved [16], [17] because failure to allocate them rationally can lead to waste of resources and many tasks not being completed on time. Therefore, it has become challenging to utilize the powerful computing capabilities in the cloud, the advantage of ultra-low latency at the edge, and the computing resources provided by idle mobile devices.

Social networks are a continuation of real social relationships, and users with social relationships [18] are more willing to share resources on social networks. Therefore, the social relationships between users could be used to measure their willingness to share computing resources. The mobility of users affects the selection of offloading devices [19]. To select the appropriate device that provides computation offloading, we proposed the mobile device selection algorithms (MDSA) by combining social relationships and mobility (location correlation and mobile activity).

The cloud has powerful computing capability, the edge can provide ultra-low latency services, and the resources of idle mobile devices can solve the problem of insufficient computing resources in hotspots. Therefore, combined use of the above resources can meet the needs of mobile devices, which has practical significance. However, flexibly allocating these resources according to the actual environment becomes a problem that needs to be solved. We propose the joint social- and mobility-aware computation offloading (JSMCO) algorithm to solve this problem.

Our contributions are summarized as follows:

- We considered a multitier system with multiusers, multiple edge servers, and a cloud server. We established a social relationship to describe the relationship between users and measured their willingness to share. In addition, we considered the mobility of users in the system to make the offloading strategy more suitable for real-world scenarios.
- We proposed an MDSA that integrates social relationships, location correlation, and mobile activity attributes to select the appropriate node to provide offloading services according to the actual requirements.
- We employed multiple offloading methods (local offloading, direct cloud offloading, direct edge offloading, device-to-device (D2D) offloading, D2D-assisted cloud offloading, and D2D-assisted edge offloading) to compute offloading comparisons.
- We proposed a multiobjective optimization-based JSMCO algorithm to assign resource allocation under multiple offloading methods and multiple offloading tasks conditions.

II. RELATED WORK

This section provides an overview of works related to edge computation offloading, D2D computation offloading, and joint social-aware computation offloading.

MEC can provide computing and storage services close to the terminal devices at the network's edge. Its advantage is that it reduces device energy consumption and computing time. For these reasons, MEC research has received a considerable attention from academia and industry.

Chen *et al.* [20] studied task offloading in software-defined ultradense networks. They proposed a method to minimize the delay while saving the battery life of user equipment. Guo *et al.* [21] proposed a two-layer game-theoretic greedy offloading scheme to solve the mobile edge computation offloading problem in ultradense Internet of Things networks. They also verified that the approach has certain advantages under the conditions of computation offloading among multiple edge servers. Dinh *et al.* [22] proposed an optimization framework for offloading from a single mobile device to multiple edge devices. This framework minimizes the processing delay of all tasks and the energy consumption of mobile devices by jointly optimizing task allocation decisions and the frequency of the central processing unit (CPU) of the mobile device. You *et al.* [23] studied resource allocation in a multiuser MEC system based on time-division multiple access and orthogonal frequency-division multiple access. Zhao *et al.* [24] studied the problem of computation offloading from multiple mobile devices to one mobile edge server to minimize energy consumption. They proposed a branch delimitation approach based on the reconfiguration linearization technique (Gini coefficient-based greedy heuristic) and a greedy heuristic algorithm based on the Gini coefficient. Ning *et al.* [25] proposed an iterative heuristic MEC resource allocation algorithm that combined cloud computing and MEC. The algorithm can make offloading decisions dynamically to optimize processing latency and offloading efficiency. Ren *et al.* [26] proposed a collaborative cloud edge computing scheme with federated communication and computing resources to improve the efficiency of edge clouds under the condition of limited communication and computation capacity. Xu *et al.* [27] proposed a computation offloading algorithm based on a neural network task

model and presented an adaptive task scheduling algorithm using an improved ant colony algorithm. A cloud edge collaboration framework for distributed neural networks was proposed based on the above algorithms. Zhao *et al.* [28] proposed a cloud edge collaboration approach by jointly optimizing computation offloading decisions and computing resource allocation, which can offload services to vehicles in an in-vehicle network.

Ahani and Yuan [29] proposed the use of BS to assist D2D offloading (BS acts as a relay for task distribution and result collection) and adopted the Lagrangian duality algorithm to balance the increased overhead and reduce the computation offloading time. Lin *et al.* [30] used D2D offloading and local offloading to minimize the total energy consumption. Xing *et al.* [31] studied a multiassisted MEC system that supports D2D offloading. They proposed a heuristic scheme for joint optimization of wireless communication, computing resources, and tasks allocation. Xie *et al.* [32] proposed a computation offloading scheme for precedence-constrained tasks with the aim of minimizing the time consumption and computing cost of computation offloading.

He *et al.* [13] proposed an incentive mechanism based on an online auction where the users' mobile devices dynamically participate in the system, which can obtain better unloading decisions without future information. Pu *et al.* [33] proposed an online incentive-aware task offloading framework, which ensures that the computing resources obtained by mobile devices from other mobile devices do not exceed their contribution. Jin *et al.* [33] proposed a multimarket dynamic double auction mechanism (mobiauc) to promote fair resource exchange. Li *et al.* [34] regarded the collaborative task offloading problem as a social welfare maximization problem and used a prime dual framework to develop an online incentive mechanism for the execution of the tasks. It considers the dynamic participation characteristics of the users' mobile devices. Saha *et al.* [35] designed an incentive mechanism to improve user participation, considering mobile device resource allocation and reputation. Noor *et al.* [36] proposed a cell cloud architecture, which refers to the reputation-based economic incentive model, encouraging mobile users to actively share their mobile device resources. Qiao *et al.* [14] proposed a reputation-based consensus mechanism (Proof of Reputation) so that the proposed computing scheme can be safely and effectively deployed in the device-to-device edge computing networks (D2D-ECN) framework. Chatzopoulos *et al.* [15] put forward a framework of joint incentive mechanism and reputation mechanism so that the devices participating in cooperation can obtain corresponding rewards, whereas the selfish devices are punished accordingly to achieve a better computing unloading effect. Roostaei *et al.* [37] proposed a mobile cloud computing framework supporting device-to-device. At the same time, to encourage mobile devices to contribute to the D2D cloudlet, an incentive mechanism based on credit and reputation was developed. The incentive mechanism uses a second-price reverse auction to measure the value of resources in the D2D cloudlet.

Cao *et al.* [9] leveraged social-aware to select the sharing of idle communication and computing resources among mobile users to reduce energy and communication resource consumptions. In addition, they proposed a joint task-data offloading framework and a matching-based and game theory-based scheme to resolve the association between mobile devices. Chen *et al.* [38] proposed a D2D crowd framework in which network edge devices use network-assisted D2D collaboration to share computing and communication resources so as to reduce energy consumption. To reduce the task computing time and energy consumption, Ciobanu *et al.* [39] proposed transferring data and computing from mobile devices to the cloud, fog nodes, or other mobile devices. Chen *et al.* [40] proposed considering social relationships as an essential factor for collaborative computation offloading and taking the minimum weight of a perfect bipartite graph as the decision-making method of multitask computation offloading. Yu *et al.* [41] proposed a hybrid multicast-based task offloading framework that uses social-aware to establish D2D connections and reduces energy consumption through the task assignment strategy of the framework.

In [20]–[28], only one local computing, edge server computing, cloud computing, or cloud edge collaboration can be selected for offloading computation. Computation offloading using this strategy in urban hotspots will cause a high network load and insufficient server resources. If idle mobile terminal resources can be fully utilized, the efficiency of computation offloading will be improved. D2D communication has become a hot research topic [29]–[32], which does not rely on local base stations but uses physical proximity to reduce end-to-end latency and energy consumption. However, due to the increasing number of mobile devices, the demand for mobile devices for computation offloading is also increasing, which leads to several problems in the use of mobile devices for computation offloading or data transfer. On the one hand, using other mobile devices for computation offloading can pose security risks. Alternatively, mobile devices have a lot of idle computing resources; however, they are unwilling to provide computation offloading to everyone. [13]–[15], [33]–[37], [42] have considered cooperation between devices, incentive mechanism, reputation or a combination of incentive mechanism, and reputation. The above methods can improve the users' effective participation and cooperation, but these methods have higher system overhead and more complex operability. The studies in [9], [38]–[41] considered social-aware for computation offloading, but none considered the mobility of users. Mobile device locations are dynamically determined by user behavior, unlike cloud or edge servers, where the location is relatively constant. Therefore, this paper considers mobility (location correlation and mobile activity) and social-aware while combining cloud servers, edge servers, and mobile devices for offloading computation.

III. SYSTEM MODEL AND PROBLEM FORMULATION

In this section, we first built a task model, task computing model, social relationship model, location correlation model, mobile activity model, and energy consumption minimization problem formulation.

As presented in Fig. 1, we considered a system model with multiuser, multi-edge and cloud servers, where the set of mobile devices can be denoted as $M = \{1, 2 \ldots\ldots, M\}$. Each mobile device location is not fixed, they can move in the area covered by the base stations, and different base stations in different locations can serve them. We assumed that all mobile devices could communicate with wireless base stations and with each other by D2D. When the mobile device handles computation-intensive tasks or latency-constrained tasks, the mobile device can perform computation offloading according to actual conditions. Mobile devices can transmit tasks to the edge or cloud server for computation offloading through the wireless base station. Tasks can be offloaded to other mobile devices when other mobile devices are available. The mobile devices can assist other mobile devices in offloading tasks to the edge or cloud server when far from the base station. In Fig. 2, we also assumed that the relationship between mobile devices is based on the social relationship between users.

The detailed system model is as follows:

A. TASK MODEL

We represent the task as a quadruple $< \lambda_i, \psi_i, \mu_i, lt_i >$; parameter λ_i indicates the size of the input data for the task of device i; parameter ψ_i, the required computing resource (we use the number of CPU clock cycles needed to indicate the required computing resources); μ_i, the size of the output data; and parameter lt_i, the task processing time limit.

The task processing time limit represents the maximum time allowed to complete the task, and the task must be processed within this time to satisfy the requirements of the mobile device. The task processing time is determined by finding the ratio of the required computing resource per task to the CPU performance of the device, which can be defined as follows:

$$t_i = \psi_i / f_i \tag{1}$$

where f_i denotes the computing power of device i using the CPU frequency (number of clock cycles per second) as the computing unit.

If the same devices consume the same amount of energy consumption per unit of time, the energy consumption ρ_i can be calculated based on the task processing time, and it is defined as follows:

$$\rho_i = \rho_i^{ec} \cdot t_i \tag{2}$$

ρ_i^{ec} denotes the energy consumption generated by the computation of device i per unit time, and t_i denotes the time required for the task computation.

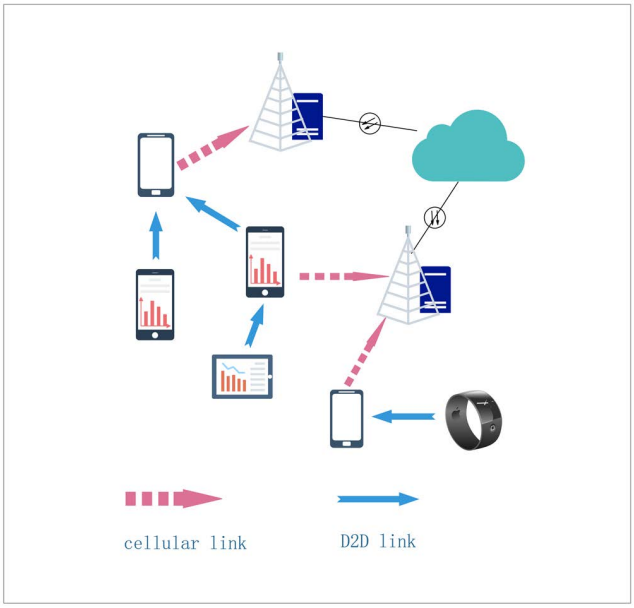


FIGURE 1. Illustration of a system model with multiuser multiedge server and a cloud server.

B. TASK COMPUTING MODEL

As a result of the different computing capabilities of cloud servers, edge servers, and mobile devices, the delay and energy consumption are also different. Comprehensive utilization of these computing resources is necessary to obtain better offloading results. Meanwhile, we will define the energy consumption and the delay associated with each strategy to evaluate offloading strategy.

Since this paper mainly considers the effects of social relationships, location correlation, and mobile activity on the computation offloading results, signal fading will not be considered in the data transmission. Figure 3 presents the computation offloading.

1) LOCAL OFFLOADING (LO)

Energy consumption is determined by the energy consumption of the local device and the computing time of the task when a local device processes the task. The diagram is presented in Fig. 3(a). The local offloading energy consumption is defined as follows:

$$EC_i = \rho_i^{ec} \cdot t_i \tag{3}$$

Because the task was locally processed, its processing time only depends on the performance of the device and the amount of task calculation. Therefore, the task processing time is calculated as follows:

$$t_i = \psi_i / f_i \tag{4}$$

In Fig. 3(a), x_i^i determines whether the task is locally processed:

$$x_i^i = \begin{cases} 1; & \text{if the task } T_i \text{ is processed locally} \\ 0; & \text{otherwise} \end{cases} \tag{5}$$

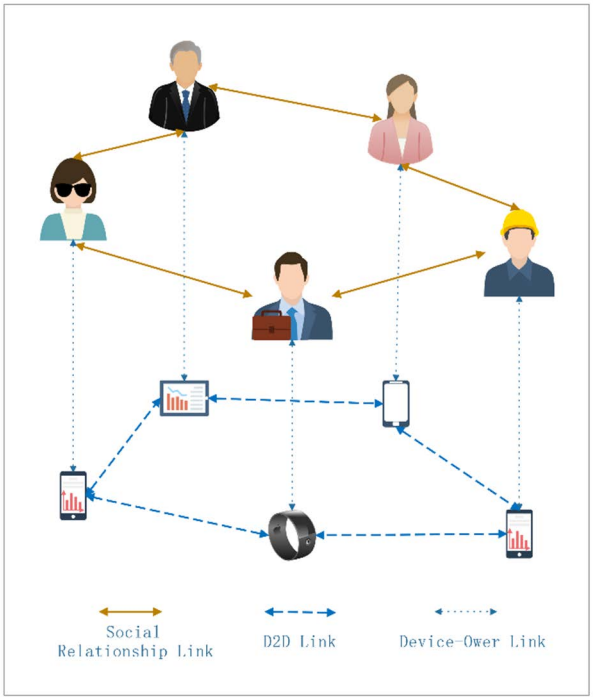


FIGURE 2. Illustration of the connection between users, social relationship, and device relationship.

2) DIRECT CLOUD OFFLOADING (DCO)

Since task offloading aims to increase the processing speed and reduce the energy consumption of the mobile device, the energy consumption of the cloud is not considered. When offloading tasks to the cloud, the energy consumption includes the energy consumption of the mobile device sending data to the cloud server and the mobile device receiving the processing result from the cloud, which is schematically presented in Fig. 3(b). It can be defined as follows:

$$EC_{i,c} = EC^{s}_{i,c} \cdot \lambda_i / R_{i,c} + EC^{r}_{i,c} \cdot \mu_i / R_{c,i} \tag{6}$$

where $EC^{s}_{i,c}$ denotes the power of mobile device i to send data to the cloud, and $EC^{r}_{i,c}$ denotes the power of mobile device i to receive data sent from the cloud.

When offloading tasks to the cloud, the time consumption includes the transmission delay between the mobile device and cloud server and the processing delay of the cloud server. Therefore, the task processing time $t_{i,c}$ is defined as follows:

$$t_{i,c} = \lambda_i / R_{i,c} + \lambda_i / R_{c1,c2} + \psi_i / f^{c}_{i} + \mu_i / R_{c,i} + \mu_i / R_{i,c2} + s_{c1,c2} / R_{c1,c2} \cdot 2 \tag{7}$$

where f^{c}_{i} indicates the computing capacity that the cloud server allocates to node i; $\lambda_i / R_{i,c}$, the time of sending data from the mobile device to the base station; $\mu_i / R_{c,i}$, the time of receiving data from the mobile device; $R_{c1,c2}$, the data transmission rate between the base station and the cloud server; and $s_{c1,c2}$, the distance between the base station and the cloud server.

In Fig. 3(b), x^{c}_{i} determines whether the task is offloaded to the cloud:

$$x^{c}_{i} = \begin{cases} 1; & \text{If the task } \mathrm{T_i} \text{ is offloaded to the cloud server c} \\ 0; & \text{otherwise} \end{cases} \tag{8}$$

3) DIRECT EDGE OFFLOADING (DEO)

The computing power of the edge server is not as strong as the cloud server, but its proximity to the mobile terminal makes the propagation delay lower than the cloud server. When the edge server processes the task, its energy consumption is equal to the sum of the energy consumption of the mobile device sending the data to the edge server and receiving the data result from the edge server. Its schematic is presented in Fig. 3(c), and its energy consumption is defined as follows:

$$EC_{i,e} = EC^{s}_{i,e} \cdot \lambda_i / R_{i,e} + EC^{r}_{i,e} \cdot \mu_i / R_{e,i} \tag{9}$$

where $EC^{s}_{i,e}$ denotes the power sent by a mobile device to the edge server; $R_{i,e}$, the rate sent by a mobile device to the edge server; $EC^{r}_{i,e}$, the power of the mobile device to receive data from the edge server; $R_{e,i}$, the reception rate.

The time consumption includes the time delay in sending and receiving data and the time delay in processing data by the edge server. The time consumption $t_{i,e}$ can be defined as follows:

$$t_{i,e} = \lambda_i / R_{i,e} + \psi_i / f^{e}_{i} + \mu_i / R_{e,i} \tag{10}$$

where f^{e}_{i} denotes the computing capacity of the edge server.

In Fig. 3(c), the x^{e}_{i} determines whether the task is offloaded to the edge:

$$x^{e}_{i} = \begin{cases} 1; & \text{if the task } \mathrm{T}_i \text{ is offloaded to edge device } e \\ 0; & \text{otherwise} \end{cases} \tag{11}$$

4) D2D OFFLOADING (D2DO)

D2DO is a strategy for computation offloading using idle mobile devices. Its data transmission can be done without cellular networks, and it accomplishes computing tasks without cloud or edge servers. The energy consumption of D2DO consists of the energy consumption of the mobile device for sending and receiving data and the energy consumption of the mobile device for completing the computation task. The schematic is presented in Fig. 3(d), and energy consumption $EC_{i,j}$ is defined as follows:

$$EC_{i,j} = EC^{s}_{i,j} \cdot \lambda_i / R_{i,j} + \rho^{ec}_{i} \cdot \psi_j / f_j + EC^{r}_{i,j} \cdot \mu_i / R_{i,j} \tag{12}$$

where $EC^{s}_{i,j}$ denotes the power of mobile device i to send data to j; $EC^{r}_{i,j}$, the power of mobile device i to receive data from j; and $R_{j,i}$, the transmission rate between node i and j. The transmission rate is determined by the type of wireless transmission network (Bluetooth, WLAN) used between node i and j.

The time consumption of D2DO is related to the sending time of mobile device i, the computing time of mobile

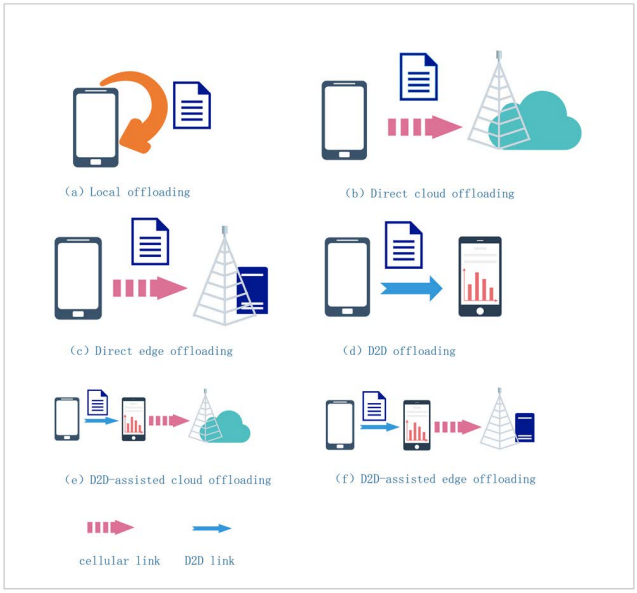


FIGURE 3. Illustration of the different computation offloading methods.

device j, and the receiving time of mobile device i. We define the time consumption $t_{i,j}$ as follows:

$$t_{i,j} = \lambda_i \cdot R_{i,j} + \psi_i \cdot f_j + \mu_i \cdot R_{i,j} \tag{13}$$

In Fig 3(d), the x_i^j determines whether the task is offloaded to other mobile devices:

$$x_i^j = \begin{cases} 1; & \text{if the task } \mathrm{T_i} \text{ is offloaded to} \\ & \text{another mobile device j} \\ 0; & \text{otherwise} \end{cases} \tag{14}$$

5) D2D-ASSISTED CLOUD OFFLOADING (D2DCO)

When mobile devices cannot transmit data directly to the cloud server for computation offloading, other devices can be regarded as a relay to transmit data to the cloud server. Figure 3(e) presents the diagram. The energy consumption of D2DCO consists of device i sending data to j, mobile device i receiving data from j, mobile device j sending data to the cloud server, and mobile device j receiving data from the cloud server. The energy consumption $EC_{i,j,c}$ for D2DCO can be expressed as follows:

$$EC_{i,j,c} = EC^s_{i,j} \cdot \lambda_i / R_{i,j} + EC^r_{i,j} \cdot \mu_i / R_{i,j} + EC^s_{j,c} \cdot \lambda_i / R_{j,c} + EC^r_{j,c} \cdot \mu_i / R_{j,c} \tag{15}$$

where $EC^s_{j,c}$ denotes the power of mobile device j to send data to the cloud, and $EC^r_{j,c}$ denotes the power of mobile device j to receive data from the cloud.

The time consumption of D2DCO includes mobile device i sending data to j, mobile device i receiving data from j, mobile device j sending data to the cloud server, mobile device j receiving data back from the cloud server, and the time consumption required for cloud server calculation. The time consumption of D2DCO $t_{i,j,c}$ can be defined as follows:

$$t_{i,j,c} = \lambda_i \cdot R_{i,j} + \mu_i \cdot R_{i,j} + \lambda_i \cdot R_{j,c} + \mu_i \cdot R_{j,c} + \psi_i / f_j^c + \left(s_{c1,c2} / R_{c1,c2}\right) \cdot 2 \tag{16}$$

In Fig. 3(e), the $x_i^{j,c}$ determines whether the task is offloaded to the cloud with the help of other mobile devices:

$$x_i^{j,c} = \begin{cases} 1; & \textit{if the task } T_i \textit{ is offloaded to the} \\ & \textit{cloud server via mobile device } c \\ 0; & \textit{otherwise} \end{cases} \tag{17}$$

6) D2D-ASSISTED EDGE OFFLOADING (D2DEO)

The signal transmission range of edge servers is limited; therefore, computing resources available to different edge servers are different so that D2DEO can provide a better quality of service (QoS) and a more reasonable allocation of computing resources. Figure 3(f) presents a schematic of D2DEO. The energy consumption is related to mobile device i sending data to j, mobile device i receiving data from j, mobile device j sending data to the edge server, and mobile device j receiving data back from the edge server by analysis. Thus, the energy consumption $EC_{i,j,e}$ for D2DEO can be expressed as follows:

$$EC_{i,j,e} = EC^s_{i,j} \cdot \lambda_i \cdot R_{i,j} + EC^r_{i,j} \cdot \mu_i / R_{i,j} + EC^s_{j,e} \cdot \lambda_i \cdot R_{j,e} + EC^r_{j,e} \cdot \mu_i / R_{j,e} \tag{18}$$

The time consumption of D2DEO includes sending data from mobile device i to j, receiving data from mobile device i to j, sending data from mobile device j to the edge server, receiving data from the edge server to mobile device j, and computing by the edge server. The time consumption $t_{i,j,e}$ can be defined as follows:

$$t_{i,j,e} = \lambda_i \cdot R_{i,j} + \mu_i \cdot R_{i,j} + \lambda_i \cdot R_{j,e} + \mu_i \cdot R_{e,j} + \psi_i / f_j^c \tag{19}$$

In Fig. 3(f), $x_i^{j,e}$ determines whether the task is offloaded to the edge with the help of other mobile devices:

$$x_i^{j,e} = \begin{cases} 1; & \textit{if task } \mathrm{T_i} \textit{ is offloaded to the edge} \\ & \textit{server e via device j} \\ 0; & \text{otherwise} \end{cases} \tag{20}$$

C. SOCIAL RELATIONSHIP MODEL

The concept of community is often used to describe social relationships, where nodes (users) with similarities (interests, behaviors, etc.) are grouped into a community. Nodes within the same community are more closely related than those from different communities. Therefore, to describe the social relationship between nodes, we consider the similarity between nodes and communities as well as the similarity between nodes and nodes to model social relationships.

This paper presents the social similarity [42] between users by the weight matrix calculated from social relationships. This matrix reflects the closeness of social relationships

between mobile devices, and the weight matrix A is expressed as follows:

$$\mathrm{A} = \left[a_{i,j}\right]_{N_{\mathrm{de}} \cdot N_{\mathrm{de}}} \tag{21}$$

where N_{de} denotes the number of mobile device nodes, including edge devices, $a_{i,j} \in [0, 1]$, $a_{i,j} = 0$ indicates that device i and j are not socially related, and $a_{i,j} = 1$ indicates that the social relationship is closely related. Since social relationships are directional, $a_{i,j}$ is different from $a_{j,i}$ (e.g., we recognize celebrities, but they usually do not recognize us). The social similarity attribute determines the value of $a_{i,j}$. In this paper, $a_{i,j}$ is determined by the similarity between node and community and between nodes. It can be defined as follows:

$$a_{i,j} = \eta CS_{i,j} + (1 - \eta)\, NS_{i,j} \tag{22}$$

$CS_{i,j}$ indicates the similarity between node i and community C_j (the community where node j is a member), and $NS_{i,j}$ indicates the similarity between node i and j. η is a variable parameter that adjusts the importance of $CS_{i,j}$ and $NS_{i,j}$.

The similarities between nodes and communities can be used to reflect the relationship between them [43]. The similarity between communities $CS_{i,j}$ is defined as follows:

$$CS_{i,j} = \frac{\sum_{\Psi_k \in \gamma_{i,j}} \Psi_k}{\sum_{\Phi_k \in \Phi_j} \Phi_k} \tag{23}$$

Using (5), we can calculate the similarity between user i and community C_j. Ψ_k denotes the attribute set of nodes i, and Φ_k denotes the attribute set of community C_j. $\gamma_{i,j}$ represents the intersection of the social relationship attribute set of the node i and the community j :

$$\gamma_{i,j} = \Psi_i \cap \Phi_j \tag{24}$$

The similarity between nodes reflects attributes similarity between nodes, which is defined as follows:

$$NS_{i,j} = \frac{\sum_{\Psi_k \in \Omega_{i,j}} \Psi_k}{\sum_{\Psi_k \in j} \Psi_k} \tag{25}$$

where $\Omega_{i,j}$ indicates the intersection of social relationship attributes between user i and user j.

$$\Omega_{i,j} = \Psi_i \cap \Psi_j \tag{26}$$

D. LOCATION CORRELATION MODEL

The exact location of people's movements is uncertain, but the areas frequently visited (e.g., home and office) are relatively fixed. They will stay a long time in these areas. If the mobile devices are selected for computation offloading, they will be more likely to complete it. Therefore, by considering the location correlation when selecting the offloading nodes, better offloading results can be obtained.

In this paper, the position relationship matrix represents the position relationship between nodes. The relationship matrix can reflect the location correlation between the served node and the service node. It reflects whether the service node is in a frequently active area. Thus, the relationship matrix can find more suitable server nodes. The position relationship matrix is defined as follows:

$$\mathrm{P} = \left[b_{i,j}\right]_{N_{\mathrm{d}} N_{\mathrm{d}}} \tag{27}$$

where N_d denotes the number of mobile device nodes. The variable $b_{i,j}$ is defined as follows:

$$b_{i,j} = \begin{cases} 1, & \rho l_i = \rho l_j \; and \; \rho l_i \in Area_j \\ 0 \end{cases} \tag{28}$$

When it is satisfied that node i and j are in the same area, and j is in a frequently active area, $b_{i,j}$ is equal to 1; otherwise, it is equal to 0. ρl_i indicates where the device i is located, and $Area_j$ denotes the area where j is frequently active.

$$Area_i = \{\mathrm{i}_1, \mathrm{i}_2, \mathrm{i}_3 \ldots\ldots \mathrm{i}_n\} \tag{29}$$

The regions are divided by geographical location, and n represents the number of regions.

E. MOBILE ACTIVITY MODEL

Mobile activity indicates the degree of activity, and higher activity indicates that the user's location changes fast, which means that the user stays in a fixed area for less time. This type of user is more suitable for less computationally demanding tasks. We use $Mob_i \in [0, 1]$ to represent mobile activity.

$$Mob_i = \left(\rho l_i, dur, date\right) \tag{30}$$

where ρl_i denotes the location of the user; dur, the duration of the user at the current location; and $date$, the date of the user at the current location. Mob_i is obtained by analyzing the user's movement track data.

F. POWER CONSUMPTION

Under the condition of guaranteed delay, we will find the offloading strategy with the lowest energy consumption. The sum of energy consumption EC is defined as follows:

$$\begin{aligned} EC = & \sum_{l \in M, e \in E} \sum_{k \in M, c \in C} \sum_{c \in C} \sum_{e \in E} \sum_{j \in M} \sum_{\mathrm{i} \in \mathrm{M}} \\ & \times \left(x_i^i EC_i + x_i^j EC_{\mathrm{i,c}} + x_i^e EC_{i,e} + x_i^c EC_{i,j}\right. \\ & \left. + x_i^{k,c} EC_{i,k,c} + x_i^{l,e} EC_{i,l,e}\right) \end{aligned} \tag{31}$$

where EC_i is the energy consumption for local computing; $EC_{i,c}$, the energy consumption of DCO; $EC_{i,e}$, the energy consumption of DEO; $EC_{i,j}$, the energy consumption of D2DO; $EC_{i,j,c}$, the energy consumption of D2DCO; $EC_{i,j,e}$, the energy consumption of D2DEO,; and M, is the set of mobile devices.

Then, the optimization objective is defined as follows:

$$\text{MIN (EC)} \tag{32}$$

$$\sum\nolimits_{i\in M} x_i^e \psi_i \le f_e \tag{32a}$$

$$x_i^i + x_i^j + x_i^e + x_i^c + x_i^{j,c} + x_i^{j,e} = 1 \tag{32b}$$

$$\left(\mathrm{x_i^j} + \mathrm{x_i^{j,c}} + \mathrm{x_i^{j,e}}\right)\left(\rho l_i - \rho l_j\right) = 0 \tag{32c}$$

$$x_i^i t_i + x_i^j t_{i,j} + x_i^e t_{i,e} + x_i^c t_{i,c} + x_i^{j,c} t_{j,c} + x_i^{j,e} t_{j,e} <= lt_i \tag{32d}$$

The constraint in (32a) ensures that the sum of the server resources used cannot exceed the computing resources owned by the edge server. The constraint in (32b) ensures that each task can only be calculated by one offloading mode. The constraint in (32c) guarantees that the offloading task cannot be transmitted to nodes outside of the transmission range. The constraint in (32d) ensures that the task processing time cannot exceed the maximum delay.

IV. JOINT SOCIAL-AWARE AND MOBILITY-AWARE COMPUTATION OFFLOADING (JSMCO)

The JSMCO algorithm mainly addresses the problem of energy consumption minimization under the condition of meeting the task latency. In the algorithm, we considered the social relationship, location correlation, and mobile activity while also making full use of idle resources and realizing the collaborative work of cloud, edge, and mobile devices.

Figure 4 presents the procedure of offloading strategy. The left side of the figure shows the tasks generated by the mobile device node; the right side, the ways that each task can be offloaded; the middle line, the offloading nodes that the task can choose; and the weight of the line, the energy consumption required for offloading. Therefore, we can transform the optimization problem in this paper into the matching problem of solving weighted bipartite graphs. The improved Kuhn–Munkres (KM) algorithm can handle this problem very well, so the objective function (32) can be transformed into the improved KM algorithm to solve the weighted bipartite graph.

A. FUNDAMENTALS OF THE IMPROVED KM ALGORITHM

The KM algorithm [44] is used to solve the weighted bipartite graph matching problem. It transforms the problem of finding the maximum weight match into finding the perfect match of the bipartite graph and finally obtains a maximum weight perfect match.

It can be seen from Fig. 4 that the task generated by the mobile device node corresponds to a part of the bipartite graph. A mobile device node that can complete the task, edge, and cloud corresponds to another part of the bipartite graph. The connection between the task and offloading method nodes represents the offloading method that the task can adopt and the energy consumed by offloading. Therefore, we can convert the minimization optimization problem to the minimum weight matching of the bipartite graph. When using the KM algorithm for bipartite graph matching, the number

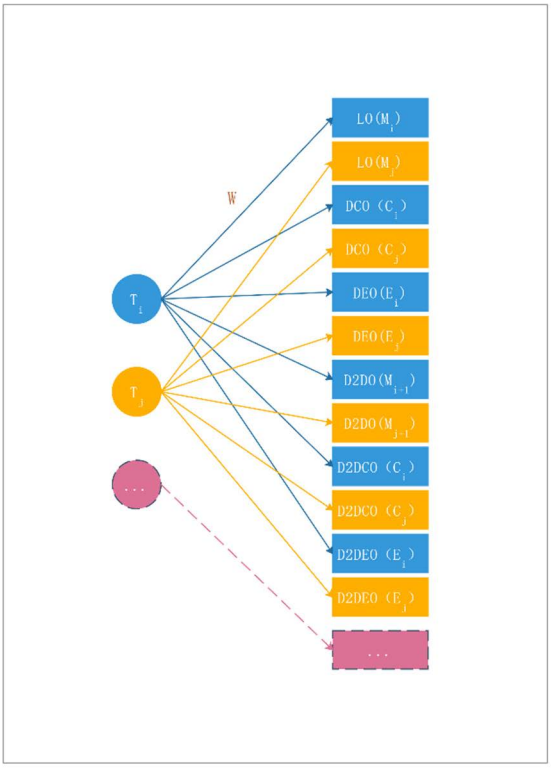


FIGURE 4. Matching model diagram of the weighted computing task and offloading method.

of nodes at both ends of the bipartite graph must be the same, and the maximum weight matching is calculated, which is inconsistent with the conditions in this paper. Therefore, in this paper, the KM algorithm could not be used directly, so the optimized KM algorithm [45] was used instead. First, the virtual task node and the weight between the virtual task node and the offloading method were introduced to satisfy the condition that the left and right nodes of the bipartite graph are equal, and then the matching of the maximum weight of the bipartite graph was transformed into the matching of the minimum weight.

B. JSMCO ALGORITHM

The JSMCO algorithm is used to minimize energy consumption to meet the minimum-delay demand. It needs to use the MDSA to obtain the optimal auxiliary unloading mobile device nodes and the optimized KM algorithm to obtain the optimal offloading strategy. The pseudocode of the algorithm is as follows:

Algorithm 1 is used to find the optimal computation offloading strategy to obtain the least-energy consumption while satisfying the time delay. The algorithm can be divided into nine steps: (1) obtain the list of task nodes and the available resources (cloud, edge, and mobile devices); (2) obtain a set of nodes around the task node that can assist in offloading according to the current time and the task node; (3) obtain all the offloading methods available for the task nodes; (4) get the optimal offloading node using algorithm 2; (5) calculate

Algorithm 1 Joint Social-Aware and Mobility-Aware Computation Offloading Algorithm

1. Initialization
get TaskList () Get a list of task nodes
get Resource () Getting cloud, edge, and mobile device available resources
end initialization
For all TaskList **do**
2. procedure getUnloadedNode()
Get the nodes around a task node
end procedure
3. procedure disIsSuitable ()
Get all available offloading methods for the current task
end procedure
procedure OptAssNode()
4. Obtain the best assisted offloading node for the current task *via* algorithm 2
end procedure
5. Procedure energyCo ()
Calculate the energy consumption of all offloading methods
end procedure
Procedure timeCon ()
Calculate the time consumption of all offloading methods
end procedure
End For
6. procedure getAllTaskUnloadReList()
Obtain offloading methods and results that meet the requirements
end procedure
7. procedure dataConversion ()
Data Conversion
end procedure
8. procedure UnloadingOptimization ()
Obtain weight-minimizing offloading methods
end procedure
9. procedure Unloading ()
Computation offloading according to the offloading scheme and calculation of the offloading time and energy consumption
end procedure

the energy and time consumption of the offloading strategy that meets the offloading time requirements; (6) obtain the initial offloading results for all tasks; (7) transform the offloading result for calculation; (8) obtain the offloading strategy that minimizes the energy consumption by the minimum weight matching of the improved KM algorithm; and (9) calculate the offloading according to the offloading strategy in the previous step.

C. MOBILE DEVICE SELECTION ALGORITHMS (MDSA)

We select the optimal offloading mobile device nodes based on three factors, namely, social relationship, location correlation, and mobile activity. Considering the combination of these factors, it is possible to select a more appropriate mobile device that provides computation offloading. This paper assigns weight to social relationships, location correlation, and mobile activity. The core formula $om_{\mathrm{I,j}}$ of the MDSA is expressed as follows:

$$om_{\mathrm{i,j}} = \left(\partial a_{i,j} + \beta b_{i,j} + \delta Mob_j\right) \tag{33}$$

$\partial, \beta, and\ \delta$ are variable parameters, which are adjusted according to the actual demand.

Algorithm 2 Mobile Device Selection Algorithms (MDSA)

1.initialization:
Initializing the values of the ∂, β, δ parameters of the mobile device selection algorithms.
end initialization
2.procedure ParserData ()
Parsing datasets to obtain social data, trajectory data, interaction data.
end procedure
Set computation offloading start time StartTime
Set computation offloading end time endTime
Set Number of cycles division
For all division **do**
3. initialization:
Set the node range for allocation and size for the task.
Initialize the properties of all mobile devices.
end initialization
4. **procedure** CreatTask ()
Task creation based on task and mobile device settings
end procedure
5.For all Obtain the set of task nodes **do**
Set The time point is the time point of the task node
Set The current node is the task node involved in the calculation
proceduregetUnloadedNode ()
Get the set of nodes that meet the task nodeNodesLocation
end procedure
6. for all The set of all encounter nodes
if chooseNode<>null
procedure community ()
Calculate the similarity of nodes
end procedure
procedure location ()
Calculating the nodes location correlation
end procedure
procedure nodeMobility ()
Calculating the mobile activity of nodes
end procedure
$\mathbb{N}_{\mathrm{i,j}} = (\partial \mathrm{similarity} + \beta \mathrm{location} + \delta \mathrm{mobility})$
end if
7. for all Node Matching Sets $\mathbb{N}_{\mathrm{I,j}}$
Get the result of the pairing with the largest value in the set
end for
End For
End for

The MDSA is used to obtain the optimal mobile device for providing offloading services. The algorithm can be divided into seven steps: (1) initialize the parameters according to the optimized parameter values obtained from the experimental results; (2) parse the data from the dataset; (3) select the offloading task node and the initialization task; (4) create tasks based on initialization parameters; (5) obtain the list of contact nodes according to the offloading time and the offloading task node; (6) calculate the social relationship, location correlation, and mobile activity and then used $om_{\mathrm{i,j}}$ to obtain a matching set; and (7) obtain the optimal node that provides offloading service.

V. NUMERICAL RESULTS

An extended real dataset was used to evaluate the performance of the MDSA and JSMCO algorithms proposed in this paper. The performance of the algorithms cannot be

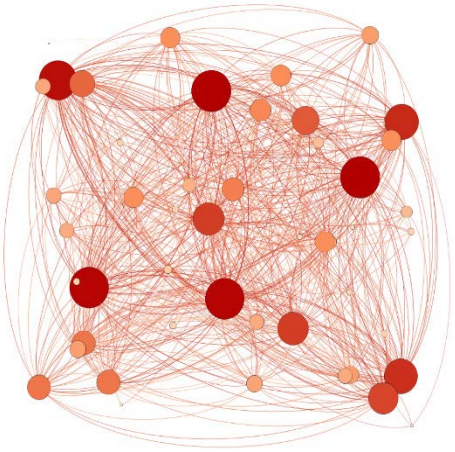


FIGURE 5. Dataset interaction network diagram.

thoroughly evaluated since relatively few mobile devices were available in the real dataset to assist with offloading. Therefore, the dataset was expanded following the distribution law of the original data. The dataset used was based on the upb-hyccups2012 [39], [46] dataset from a social tracking experiment conducted at Politehnica University, which recorded the interactions between the experiment participants and showed that their social relationships are based on their Facebook profiles. The dataset social relationship network is presented in Fig. 5.

The dots in Fig. 5 represent each node in the dataset, and the different sizes indicate the degree of the node [47]. Larger nodes indicate a greater degree, which means stronger social relationships. The lines between nodes indicate the social relationships between nodes, and the nodes with different colors belong to different communities. We compared the performance of the proposed MDSA and JSMCO algorithms using the following baseline algorithms.

The social relationship algorithm [40] considers the social relationship between mobile devices. They present a social-aware bipartite matching-based cooperative task offloading algorithm by incorporating social tie structure into the device computation and network resource-sharing processes. However, the algorithm does not consider the mobility characteristics of the devices.

The randomized algorithm does not consider user social relationships. When computation offloading is required, the algorithm randomly determines the computation offloading strategy, and the result is uncertain. We used the average value of 100 experimental results as the final result since the randomized algorithm is very unstable. The experiments included only a single-hop D2D communication, and multi-hop was not considered.

To evaluate the success rate of computation offloading, computation offloading latency, and computation offloading energy consumption, we conducted experiments to determine the task density and size factors.

A. IMPACT OF TASK DENSITY

By adjusting task density ρ (the ratio of tasks to helper device), we observed the effect of D2D offloading on the offloading success rate, energy consumption, and time delay. The task density ρ_i can be expressed as follows:

$$\rho_i = TN_i / \mathrm{Sum}_i \tag{34}$$

TN_i denotes the number of task nodes in region i, and Sum_i denotes the number of nodes in region i that can assist offloading. When ρ is lower, the task can choose more assistance nodes. Conversely, the task can choose fewer assistance nodes.

The D2D offloading success rate ω can be expressed as follows:

$$\omega = sum_{\mathrm{suc}} / \mathrm{sum}_{\mathrm{all}} \tag{35}$$

$\mathrm{sum}_{\mathrm{suc}}$ denotes the number of successful offloading tasks; $\mathrm{sum}_{\mathrm{all}}$, the total number of offloading tasks; and ω, the probability that the task assigned to the first offloading node can complete the offloading task.

In this experiment, the randomized algorithm randomly selects the assistance nodes based only on the nodes found. The algorithm based on the social relationship selects nodes according to the closeness of their relationship, and it gives preference to nodes with closer relationships for computation offloading. The proposed MDSA considers social relationships, location correlation, and mobile activity. In the experiment of task density factor, we set $\partial = 3$, $\beta = 1$, $\delta = 1$, $\eta = 0.3$.

In Fig. 6(a), all algorithms maintained a 100% offloading success rate as the task density increases when the task is a low-computing-requirement task. Because all the assistance nodes around the task node can complete the offloading task, selecting any assistance nodes will yield the same result.

Figure 6(b) and (c) demonstrates that the offloading success rate decreases as the task density increases. The offloading success rate of the randomized algorithm starts to decrease at $\rho = 0.2$; the offloading success rate of the social network-based algorithm starts to decrease at $\rho = 0.5$ and $\rho = 0.3$, respectively; and the offloading success rate of the MDSA starts to decrease at $\rho = 0.8$ and $\rho = 0.6$, respectively. Meanwhile, Fig. 6(b) demonstrates that the offloading success rate of the social network-based algorithm at $\rho = 0.7$ and $\rho = 0.8$ is higher than that of $\rho = 0.6$. This is because the social network-based algorithm does not consider the mobility of mobile devices.

The experimental results in Fig. 7(a) indicate that the energy consumption of the three algorithms is the same for the low-computing-requirement task. In Fig. 7(b), the energy consumption of the JSMCO and social network algorithms are similar when the task density is in the interval of 0.1–0.5, and the energy consumption gap is more apparent when the task density is in the interval of 0.6–0.7 and 0.8–1. It can also be seen in Fig. 7(c) that the JSMCO algorithm consumes less energy compared with the other two algorithms.

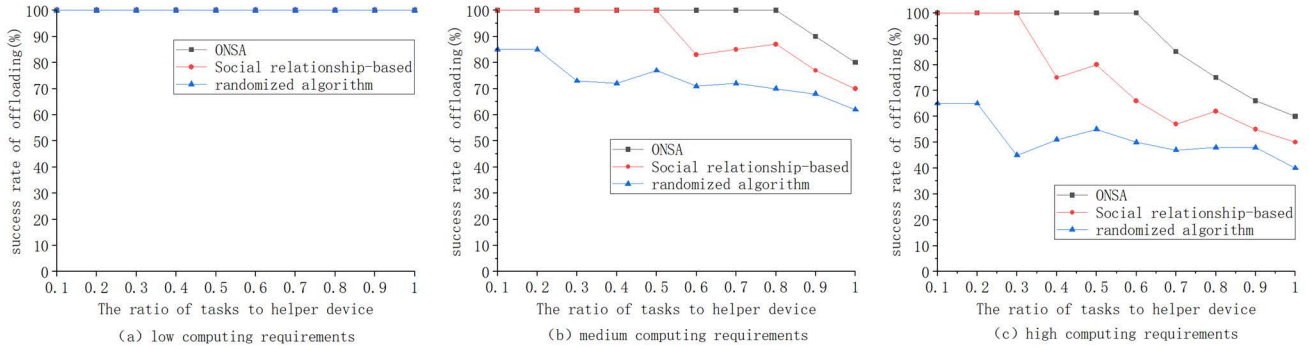


FIGURE 6. The success rate of offloading vs. the ratio of tasks to helper device under different computing requirements.

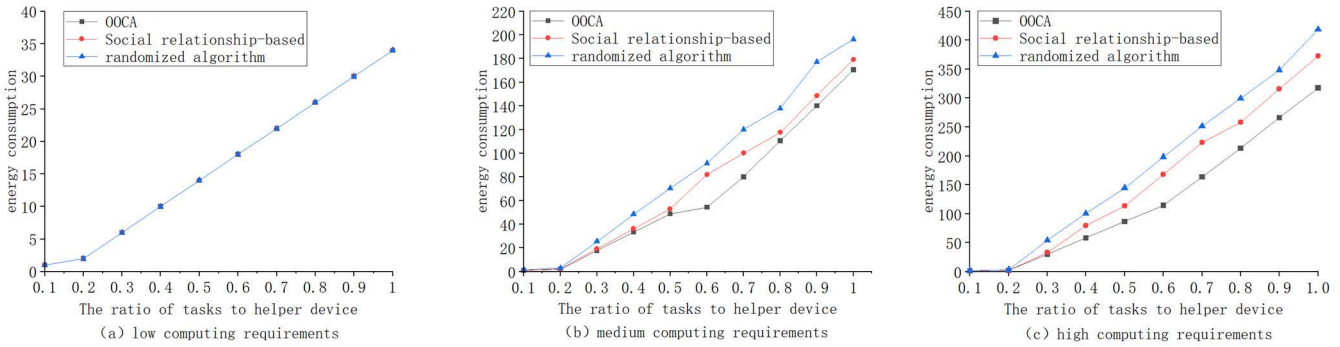


FIGURE 7. Energy consumption.

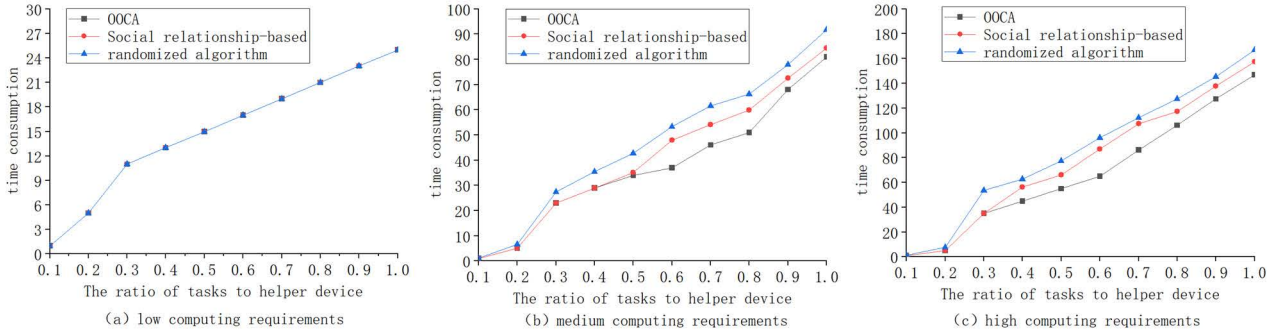


FIGURE 8. Time consumption.

The experimental results in Fig. 8(b) and (c) indicate that the JSMCO algorithm has a shorter time overhead than the other two algorithms when solving medium- and high-computing-requirement tasks. Moreover, when the task density is in the middle part, the JSMCO algorithm offers more apparent advantages than the other two algorithms.

When the task density is in the range of 0.1–0.2, the computation offloading of the task is mainly performed by the edge servers, so their energy consumption gap is relatively small. When the task density is in the range of 0.2–0.8, the mobile devices are more involved in computation offloading, so the energy consumption advantage is more obvious. When the task density is in the range of 0.8–1, the selection of mobile devices available is relatively small, so the energy consumption gap tends to be stable. This also shows the superiority of the JSMCO algorithm.

B. IMPACT OF TASK SIZE

Different users have different offloading requirements. Therefore, to better accommodate the needs of different users, the task offloading strategy should be able to handle the offloading requirements of different task sizes. In the experiment of task size factor, we set $\partial = 3$, $\beta = 1$, $\delta = 1$, $\eta = 0.3$.

The results in Fig. 9(a) indicate that the MDSA and social network algorithms have the same offloading success rates

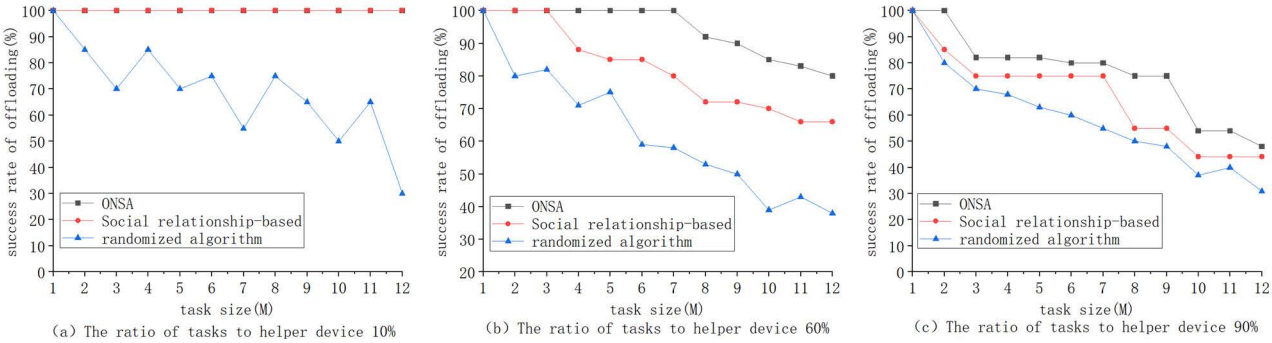


FIGURE 9. The success rate of offloading vs. the ratio of tasks to helper device under different computing requirements.

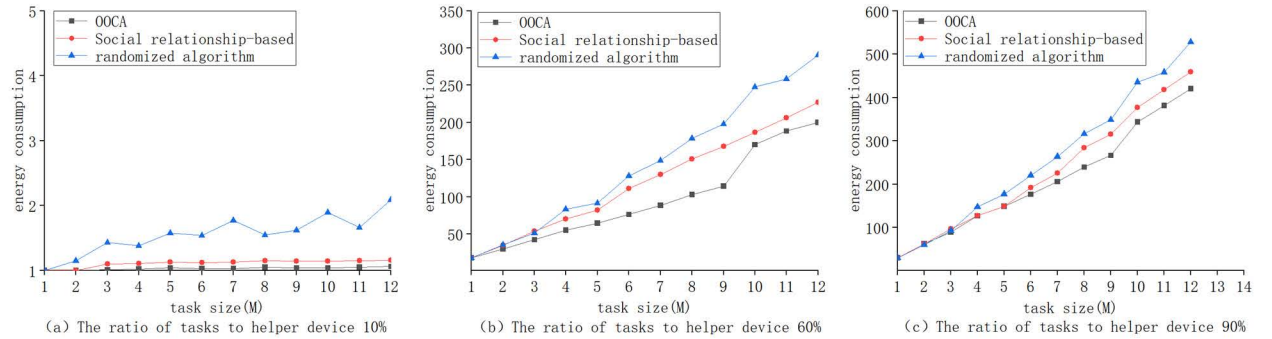


FIGURE 10. Energy consumption.

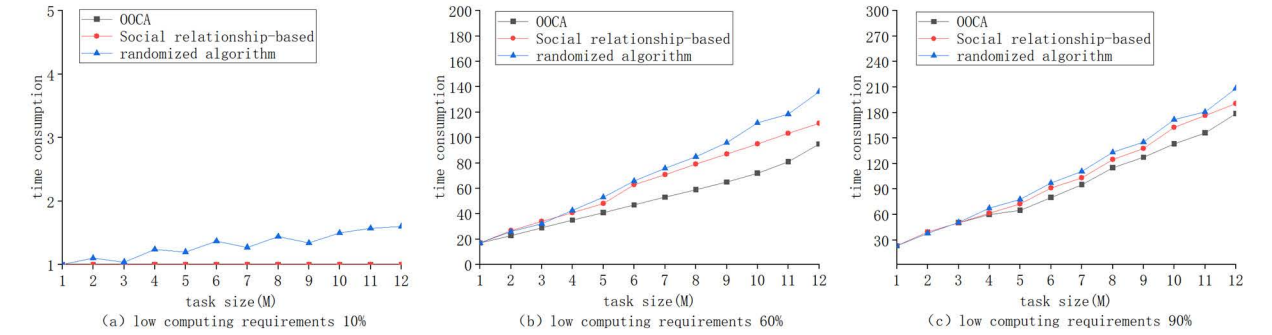


FIGURE 11. Time consumption.

when the task density is 10%. However, the randomized algorithm is relatively low, and its results are unstable. Figure 9(b) demonstrates that the MDSA is consistent with the social network algorithm when the task density is 60% and the task size is relatively low (1–2 M). When the task size exceeds 3 M, the offloading success rate of the social network algorithm starts to decrease. The offloading success rate of the MDSA starts to decrease only when the task size exceeds 7 M. The results in Fig. 9(c) indicate that the offloading success rate of all three algorithms significantly decreases when the task density is 90%.

The MDSA performs better than the other two algorithms with varying task densities and task sizes according to the above experimental results. The reason is that the randomized algorithm uses a random way to select the offloading nodes, and the social network-based algorithm does not consider the mobility characteristics of mobile devices.

Figures 10 and 11 present that the JSMCO algorithm obtains optimal energy and time consumption results in the case of increasing task size. Figure 10(a) shows that both the JSMCO and social relationship-based algorithms have lower energy consumption in the low task density. The energy consumption of the randomized algorithm is higher relative to the other algorithms and shows irregular variations. Figure 9(b) shows a significant increase in the energy consumption of the three algorithms for moderate task densities. As the task size increases, the JSMCO algorithm has less energy consumption compared with the other two algorithms. The gap between the

energy consumption of the JSMCO and social network algorithm decreases when the task size is larger than 10 M; however, the algorithm in this paper is still optimal. Figure 10(c) shows that the energy consumption gap between algorithms becomes less at high task densities, but the JSMCO is still optimal. The results of time consumption (a), (b), and (c) in Fig. 11 indicate that the variation pattern is similar to those of energy consumption (a), (b), and (c) in Fig. 10.

The above results suggest that the proposed algorithm still outperforms the other two algorithms in terms of time and energy consumption.

VI. CONCLUSION AND FUTURE WORK

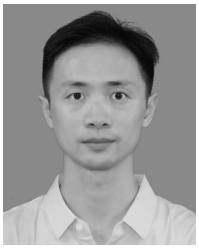
This paper has developed a computation offloading policy for a multiuser multiserver MEC system, where the social relationship among users, user mobility, and dynamic computing resource was considered. To make full use of idle resources of mobile devices and increase the users' willingness to share, we propose the MDSA, which considers social relationships, location relevance, and mobile activity. The MDSA can find suitable mobile devices to provide computational offloading, thus improving the offloading success rate. At the same time, we proposed a matching theory-based algorithm to fully utilize computing resources and reduce energy consumption while satisfying task latency requirements. Finally, the numerical results verified that these methods have better performance than the baseline methods in terms of offloading success rate and time and energy consumption.

In this paper, we have considered the single-hop case for computation offloading. To better utilize the resources in the whole system to provide a better QoS, we will consider the multihop-assisted offloading problem in the future.

REFERENCES

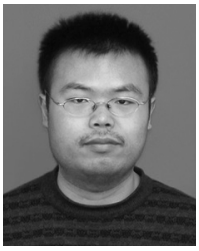
[1] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog: A survey and analysis of security threats and challenges," *Future Gener. Comput. Syst.*, vol. 78, pp. 680–698, Jan. 2018.
[2] S. Vimal, M. Khari, N. Dey, R. G. Crespo, and Y. H. Robinson, "Enhanced resource allocation in mobile edge computing using reinforcement learning based MOACO algorithm for IIOT," *Comput. Commun.*, vol. 151, pp. 355–364, Feb. 2020.
[3] J. Chen and X. Ran, "Deep learning with edge computing: A review," *Proc. IEEE*, vol. 107, no. 8, pp. 1655–1674, Aug. 2019.
[4] X. Xu, Q. Liu, Y. Luo, K. Peng, X. Zhang, S. Meng, and L. Qi, "A computation offloading method over big data for IoT-enabled cloud-edge computing," *Future Gener. Comput. Syst.*, vol. 95, pp. 522–533, Jun. 2019.
[5] C. Stergiou, K. E. Psannis, B.-G. Kim, and B. Gupta, "Secure integration of IoT and cloud computing," *Future Gener. Comput. Syst.*, vol. 78, pp. 964–975, Jan. 2018.
[6] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on Internet of Things: Architecture, enabling technologies, security and privacy, and applications," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1125–1142, Oct. 2017.
[7] A. Ndikumana, N. H. Tran, T. M. Ho, Z. Han, W. Saad, D. Niyato, and C. S. Hong, "Joint communication, computation, caching, and control in big data multi-access edge computing," *IEEE Trans. Mobile Comput.*, vol. 19, no. 6, pp. 1359–1374, Jun. 2020.
[8] Y. Deng, Z. Chen, X. Chen, and Y. Fang, "Throughput maximization for multi-edge multi-user edge computing systems," *IEEE Internet Things J.*, vol. 9, no. 1, pp. 68–79, May 2021.
[9] Y. Cao, C. Long, T. Jiang, and S. Mao, "Share communication and computation resources on mobile devices: A social awareness perspective," *IEEE Wireless Commun.*, vol. 23, no. 4, pp. 52–59, Aug. 2016.
[10] Y. Huang, Y. Liu, and F. Chen, "NOMA-aided mobile edge computing via user cooperation," *IEEE Trans. Commun.*, vol. 68, no. 4, pp. 2221–2235, Apr. 2020.
[11] H. Baek, H. Ko, and S. Pack, "Privacy-preserving and trustworthy device-to-device (D2D) offloading scheme," *IEEE Access*, vol. 8, pp. 191551–191560, 2020.
[12] J. Xu, L. Chen, K. Liu, and C. Shen, "Designing security-aware incentives for computation offloading via device-to-device communication," *IEEE Trans. Wireless Commun.*, vol. 17, no. 9, pp. 6053–6066, Sep. 2018.
[13] J. He, D. Zhang, Y. Zhou, and Y. Zhang, "A truthful online mechanism for collaborative computation offloading in mobile edge computing," *IEEE Trans. Ind. Informat.*, vol. 16, no. 7, pp. 4832–4841, Jul. 2020.
[14] G. Qiao, S. Leng, H. Chai, A. Asadi, and Y. Zhang, "Blockchain empowered resource trading in mobile edge computing and networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–6.
[15] D. Chatzopoulos, M. Ahmadi, S. Kosta, and P. Hui, "FlopCoin: A cryptocurrency for computation offloading," *IEEE Trans. Mobile Comput.*, vol. 17, no. 5, pp. 1062–1075, May 2018.
[16] X. Diao, J. Zheng, Y. Wu, and Y. Cai, "Joint computing resource, power, and channel allocations for D2D-assisted and NOMA-based mobile edge computing," *IEEE Access*, vol. 7, pp. 9243–9257, 2019.
[17] C. Yi, S. Huang, and J. Cai, "Joint resource allocation for device-to-device communication assisted fog computing," *IEEE Trans. Mobile Comput.*, vol. 20, no. 3, pp. 1076–1091, Mar. 2021.
[18] D. Wu, B. Liu, Q. Yang, and R. Wang, "Social-aware cooperative caching mechanism in mobile social networks," *J. Netw. Comput. Appl.*, vol. 149, Jan. 2020, Art. no. 102457.
[19] G. Cao, S. Wang, M. Hwang, A. Padmanabhan, Z. Zhang, and K. Soltani, "A scalable framework for spatiotemporal analysis of location-based social media data," *Comput., Environ. Urban Syst.*, vol. 51, pp. 70–82, May 2015.
[20] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 587–597, Mar. 2018.
[21] H. Guo, J. Liu, J. Zhang, W. Sun, and N. Kato, "Mobile-edge computation offloading for ultradense IoT networks," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4977–4988, Dec. 2018.
[22] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Trans. Commun.*, vol. 65, no. 8, pp. 3571–3584, Aug. 2017.
[23] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.
[24] P. Zhao, H. Tian, C. Qin, and G. Nie, "Energy-saving offloading by jointly allocating radio and computational resources for mobile edge computing," *IEEE Access*, vol. 5, pp. 11255–11268, 2017.
[25] Z. Ning, P. Dong, X. Kong, and F. Xia, "A cooperative partial computation offloading scheme for mobile edge computing enabled Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4804–4814, Jun. 2019.
[26] J. Ren, G. Yu, Y. He, and G. Y. Li, "Collaborative cloud and edge computing for latency minimization," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 5031–5044, May 2019.
[27] S. Xu, Z. Zhang, M. Kadoch, and M. Cheriet, "A collaborative cloud-edge computing framework in distributed neural network," *EURASIP J. Wireless Commun. Netw.*, vol. 2020, no. 1, pp. 1–17, Oct. 2020.
[28] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7944–7956, Aug. 2019.
[29] G. Ahani and D. Yuan, "BS-assisted task offloading for D2D networks with presence of user mobility," in *Proc. IEEE 89th Veh. Technol. Conf. (VTC-Spring)*, Apr. 2019, pp. 1–5.
[30] Q. Lin, F. Wang, and J. Xu, "Optimal task offloading scheduling for energy efficient D2D cooperative computing," *IEEE Commun. Lett.*, vol. 23, no. 10, pp. 1816–1820, Oct. 2019.
[31] H. Xing, L. Liu, J. Xu, and A. Nallanathan, "Joint task assignment and resource allocation for D2D-enabled mobile-edge computing," *IEEE Trans. Commun.*, vol. 67, no. 6, pp. 4193–4207, Jun. 2019.
[32] J. Xie, Y. Jia, Z. Chen, Z. Nan, and L. Liang, "D2D computation offloading optimization for precedence-constrained tasks in information-centric IoT," *IEEE Access*, vol. 7, pp. 94888–94898, 2019.
[33] A.-L. Jin, W. Song, P. Wang, D. Niyato, and P. Ju, "Auction mechanisms toward efficient resource sharing for cloudlets in mobile cloud computing," *IEEE Trans. Services Comput.*, vol. 9, no. 6, pp. 895–909, Nov. 2016.

[34] G. Li and J. Cai, "An online incentive mechanism for collaborative task offloading in mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 624–636, Jan. 2020.
[35] S. Saha, M. A. Habib, T. Adhikary, M. A. Razzaque, M. M. Rahman, M. Altaf, and M. M. Hassan, "Quality-of-experience-aware incentive mechanism for workers in mobile device cloud," *IEEE Access*, vol. 9, pp. 95162–95179, 2021.
[36] S. A. Noor, R. Hasan, and M. M. Haque, "CellCloud: A novel cost effective formation of mobile cloud based on bidding incentives," in *Proc. IEEE 7th Int. Conf. Cloud Comput. (CLOUD)*, Jun. 2014, pp. 200–207.
[37] R. Roostaei, M. Sheikhi, and Z. Movahedi, "Computation offloading in D2D-enabled MCC for precedence-constrained components," *Ad Hoc Netw.*, vol. 124, Jan. 2022, Art. no. 102700.
[38] 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.
[39] R.-I. Ciobanu, C. Dobre, M. Balanescu, and G. Suciu, "Data and task offloading in collaborative mobile fog-based networks," *IEEE Access*, vol. 7, pp. 104405–104422, 2019.
[40] X. Chen, Z. Zhou, W. Wu, D. Wu, and J. Zhang, "Socially-motivated cooperative mobile edge computing," *IEEE Netw.*, vol. 32, no. 6, pp. 177–183, Nov. 2018.
[41] S. Yu, B. Dab, Z. Movahedi, R. Langar, and L. Wang, "A socially-aware hybrid computation offloading framework for multi-access edge computing," *IEEE Trans. Mobile Comput.*, vol. 19, no. 6, pp. 1247–1259, Jun. 2020.
[42] Y. Meng, C. Jiang, H. H. Chen, and Y. Ren, "Cooperative device-to-device communications: Social networking perspectives," *IEEE Netw.*, vol. 31, no. 3, pp. 38–44, Mar. 2017.
[43] K. Gu, D. Liu, and K. Wang, "Social community detection scheme based on social-aware in mobile social networks," *IEEE Access*, vol. 7, pp. 173407–173418, 2019.
[44] R. J. Munkres, "Algorithms for the assignment and transportation problems," *J. Soc. Ind. Appl. Math.*, vol. 10, no. 1, pp. 196–210, Mar. 1957.
[45] L. Rui, Y. Yang, Z. Gao, and X. Qiu, "Computation offloading in a mobile edge communication network: A joint transmission delay and energy consumption dynamic awareness mechanism," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10546–10559, Dec. 2019.
[46] R. I. Ciobanu, R. C. Marin, and C. Dobre, "MobEmu: A framework to support decentralized ad-hoc networking," in *Studies in Big Data, Modeling and Simulation in HPC and Cloud Systems*, J. Kołodziej, F. Pop, and C. Dobre, Eds. New York, NY, USA: Springer, 2018, pp. 87–119.
[47] G. Rossetti and R. Cazabet, "Community discovery in dynamic networks: A survey," *ACM Comput. Surv.*, vol. 51, no. 2, pp. 1–37, Jun. 2018.



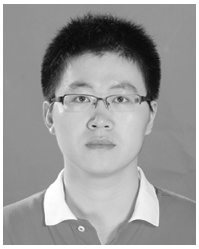
CHENGLIN XU received the M.S. degree in software engineering from Central South University, Changsha, China, in 2018. He is currently pursuing the Ph.D. degree in computer science and technology with Hunan University. His research interests include social networks, edge computing, and data mining.

CHENG XU received the Ph.D. degree in computer science and engineering from the Wuhan University of Technology, Wuhan, China, in 2006. He is currently a Professor of computer science and electronic engineering with Hunan University, Changsha, China, where he is also a Ph.D. Supervisor with the College of Information Science and Engineering. He has presented 28 articles and has hosted several national and provincial nature fund projects. His current research interests include cyber-physical systems, embedded systems, digital video processing, computer vision, machine learning, and edge computing. He is a member of the China Computer Federation.



BO LI (Graduate Student Member, IEEE) was born in 1988. He is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, Central South University. His research interests include social networks, edge computing, Internet of Vehicles, and software defined networks.

SIQI LI was born in 1990. He received the B.S. degree in physics and the M.S. degree in control engineering from Shandong University, in 2012 and 2015, respectively. He is currently pursuing the Ph.D. degree in computer science and technology with Hunan University. His research interests include artificial intelligence, reinforcement learning, and robotics.



TAO LI received the M.S. degree in communication engineering from Hunan University, where he is currently pursuing the Ph.D. degree with the College of Computer Science and Electronic Engineering. His current research interests include parallel I/O systems, file and storage systems, high-performance computing, and distributed computing.