Mobility-Aware Resource Allocation Based on Matching Theory in MEC

Bin Niu[1(✉)], Wei Liu[1], Yinghong Ma[1], and Yue Han[2]

[1] State Key Labs of ISN, Xidian University, Xi'an 710071, China
niubin@stu.xidian.edu.cn, liuweixd@mail.xidian.edu.cn, yhma@xidian.edu.cn
[2] College of Information and Communication,
National University of Defense Technology, Xi'an 710106, China

Abstract. Mobile Edge Computing (MEC) is a technology that provides communication, computing, and storage resources at the edge of a mobile network to improve the Quality of service (Qos) for mobile users. However, the conflict between the mobility of the user and the limited coverage of the edge server may interrupt the ongoing service and cause a decrease in the quality of the service. In this context, we jointly formulate service migration and resource allocation in MEC by considering user mobility, service migration, communication and computing resources in the edge server to minimize the total service delay. Then we propose a matching algorithm that takes into account the selection preferences of users and Edge servers, and effectively solves the integer nonlinear programming problem we formulated. Finally, the simulation results prove the effectiveness of the proposed algorithm.

Keywords: MEC · Mobility · Resource allocation · Matching theory

1 Introduction

With the rapid development of 5G and the Internet of Things (IoT), many new applications have emerged in mobile terminals, which put forward high requirements for delay and bandwidth. However, the limited computing resources on mobile devices are not enough to support these applications. Although traditional cloud computing provides a lot of computing resources, due to the centralized architecture of cloud computing, a lot of time is wasted in the process of data transmission. Mobile Edge Computing (MEC) has become an effective method to provide users with communication, computing and storage resources closer to users [1]. Therefore, users can offload tasks to Edge servers to improve the Qos of user.

The financial support of the National Natural Science Foundation of China (61871452), and the Fundamental Research Funds for the Central Universities under Grant JB210106.

D. Jiang and H. Song (Eds.): SIMUtools 2021, LNICST 424, pp. 75–88, 2022.
https://doi.org/10.1007/978-3-030-97124-3_7

Resource allocation and task offloading in MEC network has been widely considered. In [2], jointly optimizing local computing frequency, task splitting and transmission power, while ensuring strict delay requirements and remaining energy constraints, an algorithm for dynamic resource allocation based on user tasks is proposed. The work in [3] takes into account the service delay of users, channel quality, Edge server resources and service provider revenue, and proposes a SPA matching algorithm, which effectively reduces the service delay of users and increases the revenue of service provider.

However, none of the above works consider the mobility of users, which is fundamental for mobile users in MEC. Due to the conflicts between the mobility of users and limited coverage of Edge servers, providing continuous service to users becomes challenging. Considering the mobility of users, the work in [4,5] formulated the mobility of users as a 1-D and 2-D Markov decision process respectively and proposed an algorithm for service migration. Different from the previous study, the authors of [6–10] exploited user mobility can be predicted accurately based on the movement trajectory of user [11]. In [6], considering the user mobility, distributed resources, task properties and energy constraint of the user equipment, jointly formulate task assignment and power allocation to minimize the service delay of user.

Although the mobility of users is considered in the above work, the proposed resource allocation algorithms are all based on providing services for a single user. In fact, in the real mobile edge computing scenario, the edge server provides services for multiple users at the same time, and there is task conflict between users, which is not considered in the above work. The work in [7] jointly considered the limited battery of mobile devices and the service delay requirement of task, two algorithms are proposed for single user and multi-user to minimize the service delay. In [8], a joint service migration and mobility optimization method for vehicle edge computing is developed. In addition, a multi-agent deep reinforcement learning algorithm is proposed. The proposed method can effectively reduce system cost and vehicle service delay. The work in [9] focus on the provisioning of virtualized network function services for mobile users in MEC that takes into account user mobility and service delay requirements. The above work is based on the resource allocation on the user side without considering the perspective of the edge server. The work in [10] proposed a matching algorithm in Vehicular MEC to minimize the service delay and power consumption. However, the work in [10] did not consider the migration of some user tasks, only considered the case where calculations are performed on the same server and the calculation results are returned through the connection between Edge servers.

Although the prior works have studied mobility-aware computation offloading and resource allocation, it is still a challenging problem. Our contributions can be summarized as follows.

1) We study the service migration and resource allocation in mobile edge computing. Considering both the mobility of users and the limited resources of Edge Servers, we formulate the resource allocation problem as a integer nonlinear problem to minimize the service delay.

2) We propose an effective matching algorithm, taking into account the selection preferences of users and Edge servers, and effectively solve the proposed inte-

ger nonlinear problem. Compared with other algorithms, the proposed matching algorithm achieves lower service delay.

The rest of the paper is organized as follows. Section 2 presents the system model and discusses the mobility of user, communication model, computation model and migration model in detail. Section 3 discusses the service delay minimization problem formulation, while Sect. 4 describes our proposed Matching algorithm to obtain effective resource allocation. Section 5 presents the results and performance analysis. Section 6 finally concludes this work and highlights future direction.

2 System Model

2.1 System Configuration

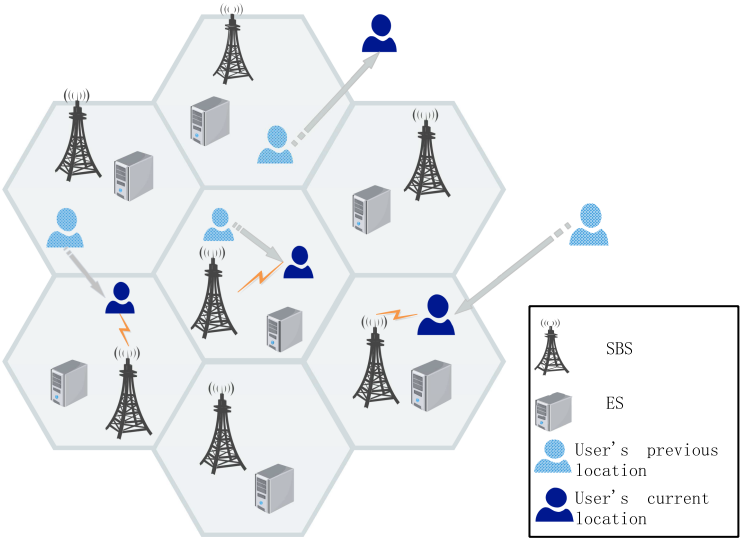

Fig. 1. Mobile edge computing network

As shown in Fig. 1, we consider a mobile edge computing system that includes K Small base stations ($SBS = \{SBS_1, SBS_2, \ldots, SBS_K\}$) with co-located Edge servers ($ES = \{ES_1, ES_2, \ldots, ES_K\}$) and N users ($U = \{u_1, u_2, \ldots, u_N\}$). In the paper, we consider the mobility of users and the delay-sensitive requirement of task. The task of user u_i can be described by a tuple $\{D_i, DC_i, T_i^{req}, Mob_i(t)\}$ [9], where D_i is the data size (bits) of u_i, DC_i is the required computing resources (CPU cycles) of u_i, T_i^{req} is the delay requirement of u_i and $Mob_i(t)$ is the mobility profile of u_i that will be introduced later [9]. Due to the limited computing resources of user, the user has to offload its task to the Edge server.

The k-th Edge server ES_k can be described by a tuple $\{C_k, \gamma_k\}$, where C_k is the computing capacity (CPU cycles per second) of Edge Server ES_k, and γ_k is the maximum numbers of users that the k-th Edge server ES_k can simultaneously serve [12]. Furthermore, B_k is the bandwidth that Small base station SBS_k provides to a single user [3]. It should be noted that the coverage of Small base station is relatively small, so users will not stay in a certain area for a long time because of the mobility [13]. It's noted that Small base stations are connected by backhaul links [8].

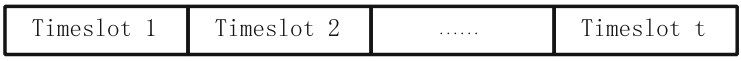

Fig. 2. Time-slotted system

As users move frequently in the system, we simply consider a time-slotted system [5] as shown in Fig. 2. And we denote the length of a single timeslot as t_{slot}. We simply assumed that the position of the user remains unchanged for a single timeslot. At the next timeslot, users in the system may move to a new position. Let $P_{i,k}(t)$ denotes the probability that u_i moves to the coverage of ES_k at timeslot t. Thus, the mobility profile of u_i is $Mob_i(t) = \{P_{i,k}(t)|ES_k \in ES\}$, which can be estimated based on the historical movement traces of the user by using trajectory data mining [11]. This assumption has been adopted in [9,14] too. Without loss of generality, We simply assume that the mobility of different users is independent.

Due to the contradiction between the mobility of user and limited coverage of single Small base station, the ongoing service of user may be interrupted [15]. Therefore, the service migration is an effective measure to ensure the continuity of the service of user. There are two main types of services of user, when a user moves from one Small base station to another, (a) the user can still choose the previous Edge server for task execution, and transmit the computing result to the user, (b) or the user can migrate the service to the closest Edge server for task execution.

In summary, the service delay of user consists of communication delay, computation delay and migration delay.

2.2 Communication Model

When user in the system has a computation task, it offloads task to ES through the closest SBS. As we mentioned earlier, we consider the delay-sensitive requirement of the task, and the transmission delay is an important part of the service delay of the user. Moreover, we assume the radio resources used between users are orthogonal to avoid the interference [3]. Thus, the transmission rate $R_{i,k}$ between user u_i and Small base station SBS_k is given by

$$R_{i,k} = B_k log_2(1 + \mathrm{SNR}_{i,k}) \tag{1}$$

where B_k is the bandwidth provided to u_i, and $\mathrm{SNR}_{i,k}$ is the received signal-to-noise ratio given by

$$\mathrm{SNR}_{i,k} = \frac{P_i h d_{i,k}^{-\theta}}{\delta^2} \tag{2}$$

where P_i is the transmit power of u_i, h is the channel power gain, and $d_{i,k}$ is the distance from u_i to SBS_k, θ is the path loss, and δ^2 is the Noise power. Thus, the transmission delay $T_{i,k}^{comm}$ from the u_i to SBS_k is [3]:

$$T_{i,k}^{comm} = \frac{D_i}{R_{i,k}} \tag{3}$$

2.3 Computation Model

Due to the delay-sensitive requirement of user, computation delay by the Edge server significantly affects the Qos of the user. For Edge server, it allocates its computing resources reasonably to serve multiple users simultaneously. We assume that users share the computing resources of the Edge server equally. Therefore, we define a matching matrix $M(t) \in \mathbb{R}^{N \times K}$ to represent the relationships between users and Edge servers in timeslot t. The matching matrix $M(t)$ of timeslot t is a correspondence between users and Edge servers.

$$M_{i,k}(t) = \begin{cases} 1, & \text{if } u_i \text{ chooses } ES_k \text{ to execute in timeslot t} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

We also assume that the task of user u_i can only be computed in single Edge server in any timeslot [8]. As we mentioned earlier, the users of any Edge server ES_k can served simultaneously is γ_k. Therefore, the two constraints are as follows.

$$\sum_{ES_k \in ES} M_{i,k}(t) \leq 1, \forall u_i \in U \tag{5}$$

$$\sum_{u_i \in U} M_{i,k}(t) \leq \gamma_k, \forall ES_k \in ES \tag{6}$$

Since the task of user u_i may have been partially calculated in the previous timeslot, let denote the requirement of computation resources by u_i in timeslot t as $DC_i(t)$, which is given by:

$$DC_i(t) = \mu(t) DC_i \tag{7}$$

where the $\mu(t)(\mu(t) \in (0, 1))$ is the percentage of task that has not been calculated at the beginning of timeslot t.

As we mentioned earlier, the users share the computation resources of Edge server equally. Due to the limited computing capacity of Edge server, not all

tasks can be completed in the current timeslot and the unfinished tasks need to be calculated in the next timeslot. Thus, we denote the computation delay of u_i who use the ES_k to execute the task in timeslot t as $T_{i,k}^{comp}(t)$, which is given by:

$$T_{i,k}^{comp}(t) = \begin{cases} \frac{DC_i(t)}{C_k / \sum_{u_i \in U} M_{i,k}(t)} & u_i \text{ can be completed in timeslot t} \\ t_{slot} & \text{otherwise} \end{cases} \tag{8}$$

2.4 Migration Model

As we mentioned earlier, providing continuous service to users becomes challenging due to mobility of users and limited coverage of Edge servers. In order to ensure that users continue to receive services, service migration is an important measure. Since the user move in the coverage of multiple Small base stations, the user should make a decision to choose whether to migrate. Therefore, there are two main types of service of user, when a user moves from one Small base station to another, (a) the user can still choose the previous Edge server for task execution, and transmit the computing result to the user, (b) or the user can migrate the service to another Edge server for task execution [15]. Furthermore, it is also essential to decide where to migrate. As we mentioned earlier, the migration strategy of migrating computing tasks to the closest server for task processing is considered to be an effective migration strategy [4]. When the migration is completed, the Edge server restores the previous computing service of user.

As we noted earlier, Small base stations are connected by backhaul links. Thus, the migration delay is composed of transmission, propagation, processing and queuing [8]. The transmission delay between Small base station SBS_p and Small base station SBS_k in timeslot t is denoted as: $\eta(t)D_i/\omega_{p,k}$, where $\eta(t)$ ($\eta(t) \in (0,1)$) is the percentage of task which has not been computed in timeslot t, and $\omega_{p,k}$ is the transmit rate (bps) through the Small base station SBS_p and Small base station SBS_k. Due to the data of computing result is relative small, we ignore the transmission time of computing result. In addition, the propagation, processing and queuing delays are determined by the hop count between the previous Edge server ES_p and the current Edge server ES_k [8]. Therefore, we denote the migration delay of u_i between Edge server ES_p and Edge server ES_k in timeslot t as $T_{i,p,k}^{mig}(t)$, which is described as:

$$T_{i,p,k}^{mig}(t) = \begin{cases} 0 & ES_k = ES_p \\ \eta(t)D_i/\omega_{p,k} + \lambda h(ES_p, ES_k) & ES_k \neq ES_p \end{cases} \tag{9}$$

where $M_{i,p}(t-1) = 1$, $M_{i,k}(t) = 1$, the λ is a positive coefficient, and $h(ES_p, ES_k)$ represents the hop count between the ES_p and ES_k. And we also denote the binary $\alpha_i^{p,k}(t)$ to represent the migration between Edge server ES_p and Edge server ES_k of user u_i in timeslot t.

$$\alpha_i^{p,k}(t) = \begin{cases} 1 & \text{migrate } u_i \text{ service from } ES_p \text{ to } ES_k \text{ in timeslot t} \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

We assume that the task of user u_i can only be computed in single Edge server in any timeslot. Thus, the migration of any user task can not occur on different Edge servers. The constraint is as follows.

$$\sum_{ES_p \in ES} \sum_{ES_k \in ES} \alpha_i^{p,k}(t) \leq 1, \forall u_i \in U \tag{11}$$

3 Problem Formulation

In the previous section, we discuss the communication delay, computation delay and migration delay of user, all of which are very essential for the total service delay of user. When user u_i has a new task, it offloads the task to the closest Edge server ES_{close}, and the transmission delay is $T_{i,close}^{comm}$. Since the user u_i may experience multiple timeslots in the process of receiving service, we denote that the start timeslot for user u_i to receive service from Edge server as ts_i, and the last time for user u_i to receive service from Edge server as tl_i. Thus, the total computation delay of user u_i can be denoted as T_i^{comp}, which can be described as:

$$T_i^{comp} = \sum_{t=ts_i}^{tl_i} \sum_{ES_k \in ES} M_{i,k}(t) T_{i,k}^{comp}(t) \tag{12}$$

Furthermore, user u_i may move in the coverage of multiple base stations during the service period. Therefore, the service of user may be migrated many times, and the total migration delay of user u_i can be denoted as T_i^{mig}, which can be described as:

$$T_i^{mig} = \sum_{t=ts_i}^{tl_i} \sum_{ES_p \in ES} \sum_{ES_k \in ES} \alpha_i^{p,k}(t) T_{i,p,k}^{mig}(t) \tag{13}$$

To sum up, the total service delay of user u_i is composed of the transmission delay, total computation delay and total migration delay. As a result, the service delay of u_i is denoted as T_i, which can be described as follows:

$$T_i = T_{i,close}^{comm} + T_i^{comp} + T_i^{mig} \tag{14}$$

It is obvious that the service delay of user u_i must satisfy request latency T_i^{req} of user u_i, which is shown as follows.

$$T_i \leq T_i^{req}, \forall u_i \in U \tag{15}$$

Thus, we are ready to formulate the optimization problem, which is shown below.

$$\begin{aligned} \min_{M(t)} \quad & \frac{\sum_{u_i \in U} T_i}{N} \\ \text{s.t} \quad & \text{Service delay constraint: (15)} \\ & \text{Edge server constraint: (6)} \\ & \text{Computation constraints: (4), (5)} \\ & \text{Migration constraints: (10), (11)} \end{aligned} \tag{16}$$

where (16) is the optimal target of the system, representing the average service delay of all users. The formulated problem falls in the form of a Integer nonlinear program, which is an NP-hard problem. Therefore, we adopt the matching theory to find the suboptimal solution.

4 Matching Based Computation Resource Allocation and Task Migration Scheme

As we mentioned earlier, the service of user may last for multiple timeslots. Therefore, we consider the resource allocation scheme in each timeslot, that is, the user chooses the best resource in each timeslot. Thus the final resource allocation scheme for users is also a suboptimal solution.

This resource allocation problem can be seen as a matching problem between users and Edge servers [16]. In our model, the Edge servers provide computing resources for users to select, and the matching is between the set of users ($U = \{u_1, u_2, \ldots, u_N\}$) and the set of Edge servers ($ES = \{ES_1, ES_2, \ldots, ES_K\}$). Therefore, we design a matching algorithm to get a competitive resource allocation scheme.

Since the matching is between the users and Edge servers, each user gives a preference list over the Edge servers that it can find acceptable, and each Edge server gives a preference list over the users that it can provide service before matching process. Since our goal is to minimize the service delay of user, the preference list of each user is determined by the service delay of the Edge servers available, while the preference list for each Edge server is determined by the computation delay of user. Let PL^{User} denotes preference list of users, and PL^{ES} denotes the preference list of Edge servers.

As we mentioned earlier, a single Edge server can provide computation service for multiple users simultaneously, and the users who choose the same Edge server share the computation resources equally. Although any Edge server ES_k has quota γ_k to limit the maximum number of users, the number of users that a Edge server serves in timeslot t is unknown for both users and Edge servers before matching. Therefore, we assume that user who choose any Edge server ES_k shares $1/\gamma_k$ of Edge server ES_k computation resources when build the

preference list of users and Edge servers [3]. Denote the maximal computation delay of user u_i who use Edge server ES_k in timeslot t as $T_{i,k}^{maxcomp}(t)$.

$$T_{i,k}^{maxcomp}(t) = \frac{DC_i(t)}{C_k/\gamma_k} \tag{17}$$

Furthermore, when the user selects which Edge server to compute task, the user consider the service delay of user. Denote the preference of user u_i over Edge server ES_k in timeslot t as $PL_i^{User}(k,t)$, which is based on the transmission delay, maximal computation delay and migration delay.

$$PL_i^{User}(k,t) = T_{i,k}^{comm} + T_{i,k}^{maxcomp}(t) + T_{i,p,k}^{mig}(t); \tag{18}$$

where p satisfied the $M_{i,p}(t-1) = 1$.

When Edge server selects which the users to serve, the Edge server only consider the maximal computation delay of user. Denote the preference of Edge server ES_k over user u_i in the current timeslot t as $PL_k^{ES}(i,t)$, which is based on the maximal computation delay.

$$PL_k^{ES}(i,t) = T_{i,k}^{maxcomp}(t) \tag{19}$$

Then, we design the matching algorithm to decide whether to migrate and where to compute the task [17].

Step 1: Establishment of user preference list

As we mentioned earlier, there are at most two Edge servers for any user u_i to select: one is the Edge server ES_{pre} which the user selected in the previous timeslot $t-1$, and the other is the Edge server ES_{close} in timeslot t which is the closest to the user. If the previous Edge server ES_{pre} is different from the closest Edge server ES_{close}, the user can select one of them. If the user select the previous Edge server, it means the Edge server continues to serve user. If the user select the closest Edge server, it means the service needs to be migrated. Furthermore, if the previous Edge server is also the closest Edge server in current timeslot t, the user has only one Edge server to select.

When the previous Edge server is different from the closest Edge server in timeslot t, the user has two Edge servers to select: the closest Edge server and the previous Edge server. And the preference list of user is determined by the service delay which is calculated by (18). The smaller the service latency, the higher the priority. For example, if the $PL_i^{User}(close,t) \leq PL_i^{User}(pre,t)$, the preference list of user u_i in timeslot t is $PL_i^{User}(t) = [ES_{close}, ES_{pre}]$. When the user has only one Edge server to select. Thus, the preference list of user in timeslot t is $PL_i^{User}(t) = [ES_{close}]$.

Step 2: Establishment of Edge server preference list

For any Edge server ES_k, the user it can provide service consists of two kinds of users: one is the user who selected ES_k in the previous timeslot $t-1$, the other is the closest Edge server of user who did not select ES_k in the previous timeslot $t-1$. Define the number of the above two kinds of users as N_1 and N_2 respectively. Therefore, we assume the number of users which can potentially be

Algorithm 1: Matching Algorithm

Input: $ES, U, PL^{User}, PL^{ES}$
Output: Matching matrix $M(t)$

```
while some user u_i has task to process and u_i has not matched with Edge server do
    for u_i ∈ U do
        u_i proposes the first Edge server ES_first in current preference list PL_i^user(t) of u_i;
        remove the first Edge server ES_first from current preference list PL_i^user(t);
        M(t) ← M(t) ∪ (u_i, ES_first);
    end
    for ES_k ∈ ES do
        while ES_k is over-subscribed do
            Find the worst users u_wst from the users who have selected ES_k according to the current preference list PL_k^ES(t) of ES_k;
            M(t) ← M(t)/(u_wst, ES_wst);
        end
    end
end
Return matching results M(t).
```

served by the Edge server ES_k is $N_1 + N_2$. It is noted that $N_1 + N_2$ may be more than γ_k and the user who selected the Edge server in previous timeslot has higher priority than other users. Therefore, the preference list of Edge server ES_k consists of two parts, the two kinds of users are sorted by the maximal computation delay respectively, which is described by (19).

For example, we simply assume that the maximal computation delay of users which selected the Edge server ES_k in previous timeslot $t-1$ is $PL_k^{ES}(1,t) \leq PL_k^{ES}(2,t) \leq \cdots \leq PL_k^{ES}(N_1,t)$ and the maximal computation delay of other users which Edge server ES_k can potentially serve is $PL_k^{ES}(N_1+1,t) \leq PL_k^{ES}(N_1+2,t) \leq \cdots \leq PL_k^{ES}(N_1+N_2,t)$. Therefore, the preference list of Edge server ES_k in timeslot t is $PL_k^{ES}(t) = [user_1, user_2, \ldots, user_{N_1}, user_{N_1+1}, \ldots, user_{N_1+N_2}]$.

Step 3: Match selection of user

For any user u_i, if the Edge server ES_k is the first Edge server in its preference list, the matching matrix is $M(t)_{i,k} = 1$. Then, the user removes the first Edge Server from its preference list. For example, if the preference list of user u_i before Step 3 is $PL_i^{User}(t) = [ES_{pre}, ES_{close}]$, then the preference list after the matching selection of user is $PL_i^{User}(t) = [ES_{close}]$, and the matching matrix is $M(t)_{i,pre} = 1$.

Step 4: Match selection of Edge server

According to the current matching matrix $M(t)$, for any Edge server ES_k in the system, it judges whether it has reached the limit on the number of service users γ_k. If it has not reached the limit, go straight to the next step. Otherwise,

if the number of user who selects the ES_k is $\gamma_k + k_1$, the Edge server ES_k will select the k_1 worst users among the current users based on the preference list and set the element corresponding to the matching matrix to 0. For example, if user u_i is one of the k_1 worst users, the matching matrix is $M(t)_{i,k} = 0$.

Step 5: Estimate of Matching result

If there still some users who does not match with the Edge servers, the process will go back to Step 3. Otherwise, if all the users match with the Edge servers, the process of matching in current timeslot t is finished. Furthermore, the details of matching algorithm is shown as Algorithm 1.

According to the matching result we get, we recalculate the service delay of users in the current timeslot, which is given by (14). At the next timeslot, due to the mobility of the user, the user may move to a location which is in the coverage of another Small base station. If the task of user has not completed, the process of this timeslot will be repeated. Meanwhile, the preference list will be recalculated, and the matching algorithm will be used until all users in the network have finished the services.

5 Performance Evaluation

In this section, we evaluate the performance of the proposed algorithm through experimental simulations.

We consider a mobile edge computing system that includes 20 Small base stations with co-located Edge servers and the number of users is from 50 to 140. The computing capacity of Edge Server varies from 1 GHz to 10 GHz, and the maximum numbers of users that Edge server can simultaneously serve is 7. Furthermore, the data size of user task varies from 10 MB to 20 MB, and the required CPU cycle of user varies from 5 GHz to 10 GHz. The other parameters used in the simulation are shown in Table 1.

Table 1. Parameters description

Parameter	Value
The delay requirement of user, T_i^{req}	5–6 s
The length of timeslot, t_{slot}	2 s
The bandwidth, B_k	200 kHz
The transmit rate through Base station $\omega_{p,k}$	100 Mbps
hop	0.05 s
The transmit power of user, P	200 mW
Noise spectral density	−174 dBm/Hz
Pass loss exponent, α	3

We evaluate the performance of Algorithm 1 against the Algorithm MobMig which is proposed by [12] and the Algorithm AMA which means always migrating

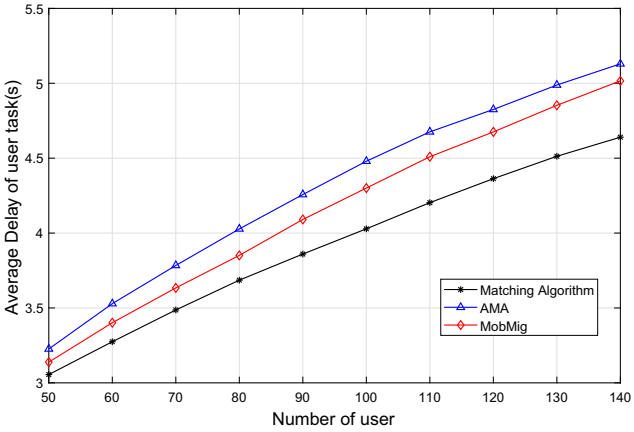

Fig. 3. The average service delay of user task

the service with the movement of user. In Fig. 3, the service delay of user versus the number of user is plotted for the above three algorithms. As seen from Fig. 3, the service delay of user increases with the increasing of number of user, as the computation resources provided to single user decreases with the increasing of the number of user task, resulting the increasing of user service delay. As our proposed matching algorithm makes the task offload decision and allocates the computing resources of the edge servers reasonably, the performance of Matching Algorithm we proposed is obviously better than the other two algorithms.

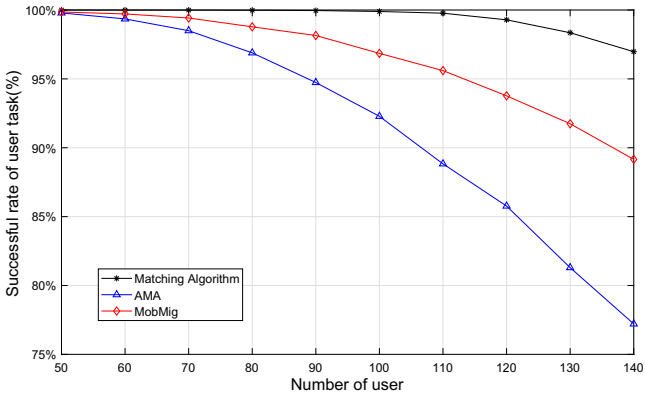

Fig. 4. The successful rate of user task

In Fig. 4, the successful rate of user task versus the number of user is plotted for the above three algorithms. As seen from Fig. 4, the successful rate of user task decreases with the increasing of number of user. When user service delay

exceeds the delay requirements of tasks, the task fails. As the number of server service users increases, service delay for users increase, resulting the decrease of successful rate of user task. Furthermore, Fig. 4 shows that the Matching Algorithm also has the best performance among the three algorithms.

6 Conclusion

In this paper, we study the service migration and resource allocation in mobile edge computing. We use the timeslot system to represent the continuous movement process of the user discretely. We formulate the resource allocation problem as a integer nonlinear problem to minimize the service delay by taking account of the mobility of user. Then we propose the Matching Algorithm taking into account of the preferences of users and Edge servers to solve the formulated problem. Finally, the simulation results show the effectiveness of the proposed algorithm.

References

1. Mach, P., Becvar, Z.: Mobile edge computing: a survey on architecture and computation offloading. IEEE Commun. Surv. Tutor. **19**(3), 1628–1656 (2017)
2. Qin, M., et al.: Service-oriented energy-latency tradeoff for IoT task partial offloading in MEC-enhanced multi-RAT networks. IEEE Internet Things J. **8**(3), 1896–1907 (2021)
3. Gu, Y., Chang, Z., Pan, M., Song, L., Han, Z.: Joint radio and computational resource allocation in IoT fog computing. IEEE Trans. Veh. Technol. **67**(8), 7475–7484 (2018)
4. Wang, S., Urgaonkar, R., He, T., Zafer, M., Chan, K., Leung, K.K.: Mobility-induced service migration in mobile micro-clouds. In: 2014 IEEE Military Communications Conference, pp. 835–840 (2014)
5. Wang, S., Urgaonkar, R., Zafer, M., He, T., Chan, K., Leung, K.K.: Dynamic service migration in mobile edge computing based on Markov decision process. IEEE/ACM Trans. Netw. **27**(3), 1272–1288 (2019)
6. Saleem, U., Liu, Y., Jangsher, S., Li, Y., Jiang, T.: Mobility-aware joint task scheduling and resource allocation for cooperative mobile edge computing. IEEE Trans. Wirel. Commun. **20**(1), 360–374 (2021)
7. Zhu, T., Shi, T., Li, J., Cai, Z., Zhou, X.: Task scheduling in deadline-aware mobile edge computing systems. IEEE Internet Things J. **6**(3), 4854–4866 (2019)
8. Yuan, Q., Li, J., Zhou, H., Lin, T., Luo, G., Shen, X.: A joint service migration and mobility optimization approach for vehicular edge computing. IEEE Trans. Veh. Technol. **69**(8), 9041–9052 (2020)
9. Ma, Y., Liang, W., Li, J., Jia, X., Guo, S.: Mobility-aware and delay-sensitive service provisioning in mobile edge-cloud networks. IEEE Trans. Mob. Comput. 1 (2020)
10. Gu, B., Zhou, Z.: Task offloading in vehicular mobile edge computing: a matching-theoretic framework. IEEE Veh. Technol. Mag. **14**(3), 100–106 (2019)
11. Feng, Z., Zhu, Y.: A survey on trajectory data mining: techniques and applications. IEEE Access **4**, 2056–2067 (2016)

12. Peng, Q., et al.: Mobility-aware and migration-enabled online edge user allocation in mobile edge computing. In: 2019 IEEE International Conference on Web Services (ICWS), pp. 91–98 (2019)
13. Liu, Z., Wang, X., Wang, D., Lan, Y., Hou, J.: Mobility-aware task offloading and migration schemes in SCNs with mobile edge computing. In: 2019 IEEE Wireless Communications and Networking Conference (WCNC), pp. 1–6 (2019)
14. Taleb, T., Ksentini, A., Frangoudis, P.A.: Follow-me cloud: when cloud services follow mobile users. IEEE Trans. Cloud Comput. **7**(2), 369–382 (2019)
15. Wang, S., Xu, J., Zhang, N., Liu, Y.: A survey on service migration in mobile edge computing. IEEE Access **6**, 23511–23528 (2018)
16. Zhang, Y., Qin, X., Song, X.: Mobility-aware cooperative task offloading and resource allocation in vehicular edge computing. In: 2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW), pp. 1–6 (2020)
17. El-Atta, A.H.A., Moussa, M.I.: Student project allocation with preference lists over (student, project) pairs. In: 2009 Second International Conference on Computer and Electrical Engineering, vol. 1, pp. 375–379 (2009)