Location-Aware Edge Service Migration for Mobile User Reallocation in Crowded Scenes

Xuan Xiao[1], Yin Li[2], Yunni Xia[1](✉), Yong Ma[3], Chunxu Jiang[4], and Xingli Zhong[5]

[1] College of Computer Science, Chongqing University, Chongqing 400044, China
xiayunni@hotmail.com

[2] Institute of Software Application Technology, Guangzhou and Chinese Academy of Sciences, Guangzhou 511000, China

[3] School of Computer and Information Engineering, Jiangxi Normal University, Nanchang 330022, China

[4] Chongqing Key Laboratory of Smart Electronics Reliability Technology, Chongqing 401331, China

[5] CISDI R&D Co. Ltd., Chongqing 401122, China

Abstract. The mobile edge computing (MEC) paradgim is evolving as an increasingly popular means for developing and deploying smart-city-oriented applications. MEC servers can receive a great deal of requests from equipments of highly mobile users, especially in crowded scenes, e.g., city's central business district (CBD) and school areas. It thus remains a great challenge for appropriate scheduling and managing strategies to avoid hotspots, guarantee load-fairness among MEC servers, and maintain high resource utilization at the same time. To address this challenge, we propose a coalitional-game-based and location-aware approach to MEC Service migration for mobile user reallocation in crowded scenes. Our proposed method includes multiple steps: 1) dividing MEC servers into multiple coalitions according to their inter-euclidean distance by using a modified k-means clustering method; 2) discovering hotspots in every coalition area and scheduling services based on their corresponding cooperations; 3) migrating services to appropriate edge servers to achieve load-fairness among coalition members by using a migration budget mechanism; 4) transferring workloads to nearby coalitions by backbone network in case of workloads beyond the limit. Experimental results based on a real-world mobile trajectory dataset for crowded scenes, and an urban-edge-server-position dataset demonstrate that our method outperforms existing approaches in terms of load-fairness, migration times, and energy consumption of migrations.

Keywords: Service migration · Load fairness · Coalitional game · Crowded scenes · Hotspot discovery · Mobile trajectory · User reallocation · Backbone network

H. Gao et al. (Eds.): CollaborateCom 2020, LNICST 349, pp. 441–457, 2021.
https://doi.org/10.1007/978-3-030-67537-0_27

1 Introduction

Mobile edge computing (MEC) refers to the concept of processing data at mobile edge network. The edge is similar to a distributed cloud with proximity close to end users. It is usually built upon small-scale data centers close to the data sources to guarantee low latency, high reliability, and scalability [8,11]. As shown in Fig. 1, services in edge nodes are situated close to users and an edge server (ES) can only contact the users that fall into its coverage area. Due to user mobility and the constraint of the user-server proximity, in practice, server-side services need to be migrated from their original nodes to new ones to maintain the user-server proximity. In reality, population in motion in crowded areas, e.g., supermarkets, can lead to unbalanced distribution of highly mobile users [20], and thus edge servers can receive quite different amounts of requests from users even when they are located at nearby areas [21]. For example, a subway station usually attracts a lot of users while a park or a bookstore near the subway usually attracts much fewer. Consequently, edge servers deployed in the station can show higher load than those in the park or bookstore. A smart strategy that is capable of handling user mobility and migrating tasks from highly-loaded servers to ones with low load is thus in high need [2]. An ES should deal with load peaks and very spiky patterns s. Moreover, MEC servers are generally equipped with lightweight computing components and limited storage. This exacerbates the challenge to avoid hotspot effects on ESs, and load-fairness among MEC servers, as well as appropriate utilization rates of edge servers [1,2]. Thus, user reallocation from highly-loaded ES to lowly-loaded ES should be performed by service migration [3,5]. To overcome these limitations, we propose a novel coalitional game-theoretic approach to location-aware MEC service migration (CGL-SM) for crowded scenes. It includes a coalition formation strategy and mechanisms for load-balancing. The coalitions are formed by using a modified k-means algorithm, where its payoff is proportional to load-fairness of the ESs in the corresponding coalition. Load-balancing is maintained through service migrations among ESs in the coalition and avoiding hotspots by

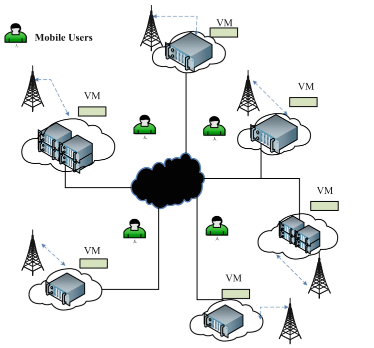

Fig. 1. Mobile users upon MEC System

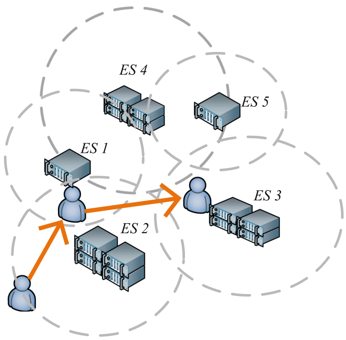

Fig. 2. Mobile User's Service Migration

using an ES workloads detection mechanism. To validate our proposed method, we carry out a case study based on a well-known edge users allocation dataset (EUA dataset) [25] and a crowded-scene mobile user trajectory dataset [29]. We show that our method beats its competitors in terms of: 1) workload-fairness of the involved ESs in highly-crowded and highly-overloaded situations; 2) number of service migrations performed; 3) energy consumption of migrations.

2 Related Work

Most existing works in this direction focus on the edge user allocation problem in the MEC environment [7,15,19,27], or location prediction of mobile users for MEC systems [17,21,22]. Some work [3] has researched crowded scenes from the perspective of computer graphics. However, how to migrate services upon MEC infrastructures in crowded scenes is less considered and studied. Robicquet *et al.* [28,29] provided a mobile trajectory dataset in crowded scenes by drone in the Stanford campus, as shown in Fig. 4 and 5. Some work has researched hotspot discovery issues. For example, Anchuri *et al.* [23] have researched hotspot discovery problem in service-oriented architecture, however, it is not in a specific scenario. Huang *et al.* [14] have researched the roadside hotspot in edge computing based on internet of vehicle from the perspective of protecting security from attacks, and their method is based on a Stackelberg game approach. This is the most similar research that is also focusing hotspot issues in edge, however, their proposed method is mainly for the purpose of avoiding internet attacks and thus service migration is not considered.

In a crowded scene, a single MEC server is inadequate and a group of MEC servers is usually required for cooperative tasks [12]. Generally, user requests and mobile resource allocation technology could be applied in this scenario [23,30]. A number of works considered focus on D2D communications to appropriately allocate user requests with multiple MEC servers. For example, He *et al.* [13] considered using D2D communication for task offloading and resource management in a multi-user and distributed mobile edge cloud resource environment. However, in a highly mobile environment, D2D is not a mainstream solution due to the fact it tends to lose connectivity stability when load is high [4].

Recently, service migration is regarded as a highly effective means for load balancing, and task offloading with mobile users. As can be seen in the example illustrated in Fig. 2, a mobile user moves freely and it is assumed generally that their moving area is a circle (note that this assumption is widely used in related works [19,27]). By radio access network, edge servers could cooperate to execute tasks collectively. For example, Pang *et al.* [7] developed a loosely coupled fog radio access network model leveraging low-end infrastructures such as small cells' power to achieve ultralow latency by exploiting the joint-edge-computing and near-range communications techniques. Some incooperative-game-based model is usually used for user reallocation or resource reallocation. He *et al.* [19] proposed a game-theoretic approach that formulates the edge users allocation problem as a potential game. Decreasing System-cost is taken as the metric for service migrations among edge servers, and the less system-cost is better. Wang *et al.* [15] formulated the service migration problem as a Markov decision process. Their formulation captured general cost models and provided a mathematical framework to design optimal service migration policies. Locations of mobile users in service migration are also researched frequently. Wu *et al.* [20] promoted a user-centric location prediction approach by leveraging users' social information. They considered that each user is with private location information that is not shared to others, and proposed a factor-graph-learning model that takes into account not only user's social and network information, but also inter-user correlation information. Tan *et al.* [21] proposed a location-aware load prediction which deals with user mobility by correlating load fluctuations of edge datacenters in physical proximity. Yin *et al.* [22] presented a decision-support framework for provisioning edge servers for online services providers.

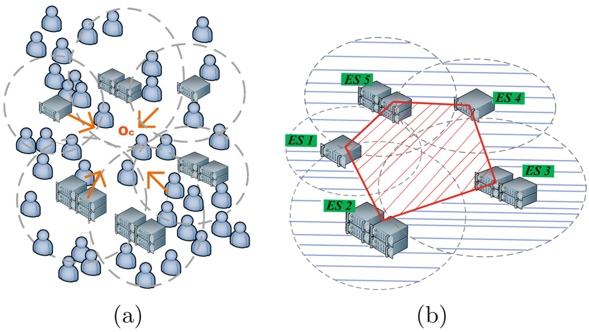

Fig. 3. Crowded Scene and Hotspot in an ES Coalition Area (Color figure online)

3 System Model

3.1 Crowd Model Scenario

If a large number of users gather toward the same destination, such as a subway station, a school gate or a plaza, then a crowded area O_c emerges. Figure 3(a) shows an illustrative example of the gathering pattern, where highly mobile users' paths are bidirectional and users move periodically towards and away from O_c. Figure 3(b) shows the ES coalition graph with such patterns, where crowded areas are marked with blue lines and hotspot areas are marked with red lines. As various existing works [20,28] did, we consider that the coverage area of the MEC ESs is larger than the hotspot area. Figure 3(c) demonstrates that ES2 and ES4 are marked red and they are affected by crowds of users. Therefore, they should transfer some workloads to other ESs in the same coalition. If all ESs in the same coalition have no remaining capacity, then the workload is transferred to a nearby coalition by backbone network [24].

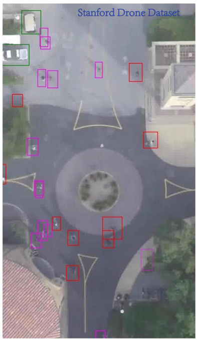

Fig. 4. Deathcircle Scene in Stanford

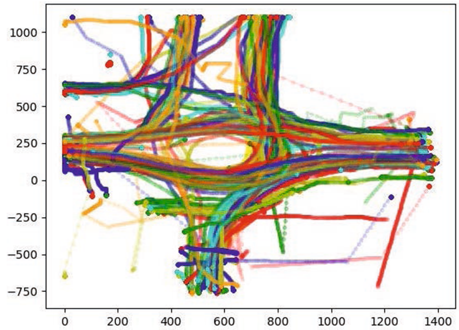

Fig. 5. Trajectories in Deathcircle

3.2 Service Migration Model

Originally, an application user u_i can be allocated to an ES e_j only if it is covered by e_j, i.e., $COV(e_j)$. It is usually covered by many ESs, however, it will choose the nearest one originally as shown in (1):

$$\begin{aligned} &u_1 \in COV(e_{j1}),\ COV(e_{j2}),\ COV(e_{j3}) \\ &u_i \longrightarrow e_{j1},\ s.t.\ d_i^{j1} < d_i^{j2} < d_i^{j3} \end{aligned} \tag{1}$$

where d_i^{j1} is the distance between e_{j1} and u_i. s_i could be migrated to the ES in another coalition when u_i is at the boundary of any two coalitions. $traj$ is the set of trajectories of all users, which is also the input of *Algorithm 2* below. Each user's movement is decided by its corresponding mobile trajectory as given in (2). $traj_i$ denotes the moving trajectory of u_i's location: y_i^t, and it can be denoted by:

$$\begin{aligned} &traj = \{traj_i \mid u_i \in U\} \\ &traj_i = \{\ y_i^t = <LO_i^t, LA_i^t> \mid t \in T\} \end{aligned} \tag{2}$$

where LO_i^t and LA_i^t denote the longitude and latitude values at time point t, U is the set of users, T denotes all time slices, N_s the number of all services. According to [20], we use the following model to estimate service migration overhead. We consider that every service have a budget B_m. A unit migration budget, B_i, is consumed, whenever s_i is migrated. Migration stops when the corresponding budget is used out. The remaining budget of all services: B_r, is thus:

$$\begin{aligned} &\mathrm{B}_m^i = B_m - \mathrm{n}^i * B_i \quad s.t.\ \ B_i \propto w_i,\ B_m^i > 0 \\ &B_r = N_s * B_m - \sum_{i=1}^{N_s} * n^i *\ B_i \end{aligned} \tag{3}$$

where B_m^i is the remained budget of s_i, w_i denotes the workload of s_i, B_i is a unit budget cost related to w_i, n^i is the total migration times of s_i. Thus, the service with a larger workload should be moved for a fewer times. In our problem, we denote the MEC service latency at time point t as $l(p_i^t, y_i^t)$, which is relevant to the physical proximity between u_i and the matched ES. Then latency of s_i in all time slices T, i.e., L_i follows:

$$L_i = \sum_{t=1}^{T} l(p^t, y_i^t),\ \ s.t.\ B_m^i > 0 \tag{4}$$

where u_i's location at t is y_i^t and location of service s_i in ES at t is p^t. With the constraint of B_m^i, s_i can not be migrated within the coalition before it is migrated to another coalition. We use M_i to denote the maximum delay of s_i when it's in the coverage area of coalition c. It follows:

$$L_i < M_i \tag{5}$$

If u_i steps into another coalition's area, s_i gets a new budget. If the user is still in the coalition area, but out of the coverage area of the original ES, the service is migrated to another ES as well. The overall energy consumption for service migration, i.e., E_S can be obtained as:

$$E_S = \sum_{i=1}^{N_s} n^i * E_m * w_i \tag{6}$$

where E_m is the migration energy for a unit workload.

3.3 Capacity and Workload Model

Each edge server e_j has a capacity of F^j, and F_i^j denotes the service workload of u_i placed on server e_j. It should satisfy that The aggregate workload of each resource type incurred by all allocated users must not exceed the capacity of their assigned server in (6). The total workloads generated by all users allocated to an edge server must not exceed its remaining capacity as shown in (6). Assume that if services are placed on e_j, they should satisfy the function $PL(e_j)$ in (6), which indicates the set of services placed on e_j.

$$\begin{aligned} &\mathrm{Q(e}_j) = \{u_i \mid \mathrm{u}_i \in e_j\} \\ &\mathrm{PL(e}_j) = \{\mathrm{s}_i \mid s_i \in e_j\} \\ &F^j > \sum_{u_i \in Q(e_j)} F_i^j \end{aligned} \tag{7}$$

F^j is the capacity value of an edge server e_j, $Q(e_j)$ is a function to indicate the set of users whose services are placed on e_j. The total workloads of users in $Q(e_j)$ mustn't exceed F^j. Assume that W_j is the workload of e_j, and it equals all the services' workloads on e_j. R_j is the resource utlization rate of e_j, and it can be computed as:

$$R_j = \frac{W_j}{F^j} \quad s.t.\ W_j \neq 0, R_j < z_j \tag{8}$$

where z_j is the maximum utilization rate of e_j, and R_j should not exceed z_j.

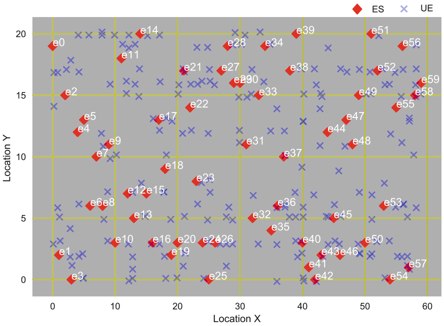

Fig. 6. Map of ESs and Users (Color figure online)

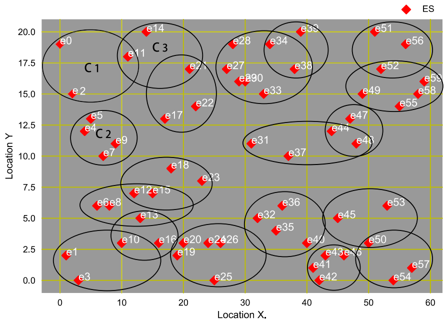

Fig. 7. Coalitions by Modified K-means

4 Coalitional Game for Load-Balancing

4.1 Framework Overview

The overall framework includes 3 major steps:1) using k-means to divide the coalitions as in pseudocode lines 1–11 in *Algorithm 1*; 2) discovering the hotspot happened in the area covered by ESs in the coalition, and then reallocate the workloads inside the coalition members; 3) transferring workloads to members of nearby coalitions from the overloaded ESs by backbone network [24] in pseudocode lines 21–23 in *Algorithm 2*. As shown in Fig. 7, c_1 could transfer workloads to nearby coalitions, such as: c_2, c_3. Coalitions, which are next to the overloaded ES, i.e., overloaded ES2 and ES4 in Fig. 3(c), should be choosen in prior.

4.2 Location-Aware Coalition Formation

Algorithm 1: Coalition Formation Based on k-means

Input: P, H, k, J
Output: C
1 Build a coordinate system for the ESs in map, and confirm their x, y coordinates
2 Compute the ES distances set E according to (10)
3 **for** *every h in H* **do**
4 | Form a coalition according to k-means
5 **end**
6 Finally, it groups e_1,e_2...e_N as k coalitions
7 **if** *e_{j1} is close to another coalition* **then**
8 | Add e_{j1} to that coalition, renew C
9 **end**
10 **if** *for any coalition c, its n_c is larger than J* **then**
11 | Divide this coalition by k-means
12 **end**

Figure 4 illustrates crowds of people gathering in a circular area in Stanford Campus [28, 29]. Edge server is in this area as well. Figure 5 illustrates the users' trajectories of Fig. 4. It is clear to see a hotspot easily emerges in such scene. Service migrations are in high need to counter the emergence of hotspots caused by the gathering of massive users. We consider that all ESs construct a graph G as shown in (9).

$$G = (P, E) \tag{9}$$

where P is the set of vertices as $P = \{e_1, e_2, e_3...e_N\}$. N is the number of all ESs in the whole area. E is the set of euclid distances between any two ESs. The euclid distance $Dist(a, b)$ between any two edge servers: e_{j1}, e_{j2} can be computed as:

$$\begin{aligned} E &= \{Dist(j1, j2) \ \mid e_{j1}, e_{j2} \in P\} \\ Dist(j1, j2) &= \sqrt{(ELO_{j1} - ELO_{j2})^2 + (ELA_{j1} - ELA_{j2})}^2 \end{aligned} \tag{10}$$

ELO_{j1}, ELA_{j1} are the longitude and latitude of e_{j1}, and the same to e_{j2}. According to E, close ESs are divided into a group as a coalition. To partition the ESs in a large area into several small units, the locations of ESs are the key factor. We use the k-means method to divide the ESs and employ the locations of ESs as the deciding factor for partition. As shown in Algorithm 1, the partition takes hotspot areas, i.e., H, $H = \{h_1, h_2, h_3...h_k\}$ of human crowds as the input, where k is both the number of hotspot areas and also the number of ES coalitions. The partition algorithm takes H, P as the inputs as well and generates the set of coalitions as C. n_c is the number of ESs in a single coalition c and it's bounded by J. J is flexible, as it varies according to the size of the hotspot area. If the hotspot area is wide, it needs more ESs to cover. Coalitions may share the borders as shown in pseudocode lines 7–8 of *Algorithm 1*. k is decided by the number of hotspot areas. d is the number of iteration, and thus the time complexity of pseudocode lines 1–6 is $O(k * N * d)$. For the modified operations for each coalition in pseudocode lines 7–11, if the total modification times is g, then the final time complexity is $O(k * N * d + g)$. As shown in Fig. 6, 100 base stations (BSs) colocated with ESs are presented, and users are distributed around them. Users are marked in blue, and ESs in red. To make the coalitions fine-grained, the k-means-based clustering analysis process is iterated until n_c fits the population distribution. As shown in Fig. 7, all BSs are divided into a number of coalitions according to Algorithm 1.

4.3 Coalitional Game Model for Workload Allocation

A coalitional game Γ consists of two essential elements [10]: 1) a set of players $N = \{1, 2...\}$, in this paper (ESs are modelled as players); 2) a characteristic value ν that specifies the value created by different subsets of the players. i.e., the payoff of a coalition c. Here maximizing the payoff $\nu(c)$ means maximizing the coalition's load fairness.

$$\Gamma = (N, \nu) \tag{11}$$

Every edge server is modelled as a player. Players are assumed to be rational to join a coalition c. As a participant, each ES wants to keep a moderate utility.

Service migration is the strategy for them to adjust the workloads distribution among ESs in the coalition.

$$c = \{e_j \mid e_j \in \Gamma\} \tag{12}$$

The coalition's load variance $D(c)$ should be bounded by:

$$D(c) = \frac{1}{n-1}\sum_{j=1}^{n}(W_j - \bar{W}_c)^2 \tag{13}$$

For each coalition, it should keep a low level of variance $D(c)$ to avoid imbalance of workloads.

$$\begin{aligned} &Min\ D(c) \\ &s.t.\ F^j > \sum_{i=1}^{n} F_i^j \end{aligned} \tag{14}$$

where $\bar{W}_c$ is the average workload of all ESs in a coalition c. Std is the standard deviation based on $D(c)$. As the constraint, workloads of the services placed on the j^{th} ES should not exceed its capacity F^j. Here, we stipulate that the payoff of a coalition is $\nu(c)$, and it can be obtained as:

$$\begin{aligned} &\nu(c) = \frac{1}{Std} \\ &s.t.\ d_{i,j} = 1,\ 0 < R_j < z_j \end{aligned} \tag{15}$$

where $d_{i,j}$ is a boolean variable to indicate whether the i^{th} service is placed on the j^{th} server. Edge servers in a coalition communicate with each other and migrate the services among the coalition members to balance the workloads on them. The resulting optimization object is thus to maximize ν with constraint of B_m^i:

$$\begin{aligned} &Max\ \ \nu(c) \\ &s.t.\ \nu \neq 0,\ B_m^i > 0 \end{aligned} \tag{16}$$

A coalition $c = \{e_1, e_2 ... e_{n_c}\}$ includes edge servers grouped by *Algorithm 1*. Users' locations change with time, and they may form a hotspot at any time slice. Pseudocodes line 1–9 in *Algorithm 2* describe three criteria for judging whether a hotspot in the ES's coverage area is discovered or an ES is qualified to be a source ES, namely: 1) any ES e_j in the coalition is over utilized, namely, the utilization rate of it is higher than its threshold value R^h, which is set according to the edge server itself; 2) any ES receives the most workloads among all ESs in the coalition; 3) any ES's workload exceeds the average workload of ESs in the coalition. Based on these criteria, an ES could be regarded as a source ES. It is constrained that any service must be migrated to another ES which covers its user. Pseudocodes in line of 9–17 in *Algorithm 2* illustrate the service migration operation. As a consequence, the subsequent payoff of the coalition, i.e., $\nu'(c)$, should exceed $\nu(c)$. $\nu'(c)$ is the payoff of the coalition after one service migration is performed.

$$\begin{aligned}
&\forall\ e_{j1}\ e_{j2} \in c,\ u_i \in e_{j1},\ W_{j1} > W_{j2}\\
&\textbf{If } \nu(c) < \nu^{'}(c),\ \textbf{after service migration}\\
&\textbf{Then do } u_i \xrightarrow{move} e_{j2}\\
&s.t.\ traj_i \in Cov(e_{j1}, e_{j2}), B_m^i > 0
\end{aligned} \tag{17}$$

In *Algorithm 2*, $Cov(e_{j1}, e_{j2})$ denotes the coverage area of e_{j1} and e_{j2}. In a coalition, to match the destination ES and the source one, let $ord(j)$ denote the ascending order of the j_{th} ES according to the workload. A highly-loaded ES is matched with a lowly-loaded ES with $ord(n_c - ord(j))$. If the ES with $ord(n_c - ord(j))$ does not cover the user, then consider the ES with $ord(n_c - ord(j) - 1)$ or $ord(n_c - ord(j) + 1)$. The match operation is in pseudocode line 11 in *Algorithm 2*. m_s denotes all service migrations performed during the whole process. Fairness improvement is calculated according to the decrease of the standard deviation of the workloads distributed on ESs in a coalition after the migration is conducted, i.e., $\varDelta Std$.

Algorithm 2: CGL-SM
(Service Migration in a coalition c)

```
    Input: U, S, c, traj, B_m
    Output: Updated match of U, S, c, B_m^i
 1  Step 1: Hotspot or source ES detection
 2  for all ESs in c do
 3      if 1. any ES's utilization rate exceeds its highest threshold value R^h ;
 4         2. any ES in c receives the most workloads ;
 5         3. any ES receives workload that exceeds the average workload in c. then
 6          Take this ES as a source ES according to the criterion: 1, 2, 3.
 7      end
 8  end
 9  Step 2: Perform service migrations
10  for all ESs in c do
11      if a source ES e_j1 and e_j2 can be matched then
12          for services on e_j1 and in Cov(e_j1, e_j2) do
13              if any s_i can be migrated based on (17) and its B_m^i > 0 then
14                  migrate s_i from e_j1 to e_j2
15              end
16              else if no services are qualified to migrate then
17                  Stop Service Migration
18              end
19          end
20      end
21      else if ESs in c could not handle workloads from the hotspot then
22          Transfer services to neighbor coalitions or central cloud by backbone
            network
23      end
24  end
```

4.4 Complexity Analysis

The complexity of *Algorithm 2* can be examined as several steps. n_c is the number of ESs in a coalition. The time complexity of the match step is $O(n_c/2)$. The process of service migration depends on the number of service migrations. If the maximum number of service migrations performed in a round of match operation is m, then the time complexity of it is $O(m)$. Finally, the overall time complexity is $O(m * n_c/2)$.

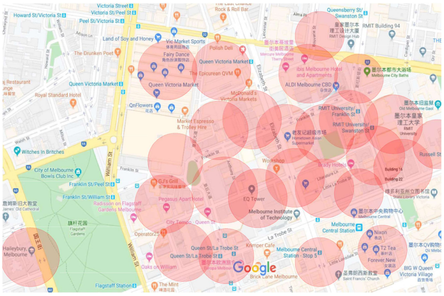

Fig. 8. Experimental Area in Melbourne

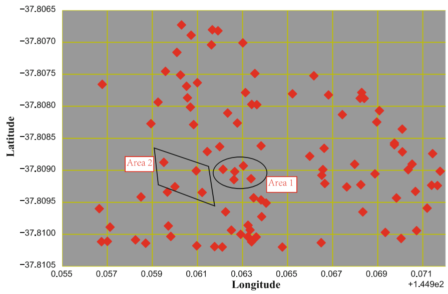

Fig. 9. ESs in Some Areas of Melbourne

5 Experiment Setting and Evaluation

5.1 Benchmark Policies

We compare our CGL-SM with existing user allocation algorithms and a no coalition formed algorithm: 1) EUAGame [19], an incooperative game-based approach applied in for user allocation in MEC; 2) Greedy [27], a proximity-priority-based migration method; 3) No-Cos, a non-coalitional variant of CGL-SM for showing the performance gain by the coalitional model.

5.2 Experimental Settings

In Fig. 8, base stations (BS) are distributed based on Google Map [26]. We depict the locations of BS in part of Melbourne from dataset [25] as shown in Fig. 9. According to the modified k-means algorithm, we select two typical crowded areas based on ES coalitions, i.e., *area 1* and *area 2* as illustrated in Fig. 9. The total area is around 0.06 km^2. Here J is set as 7 according to the distribution of these BS. R^h of any ES is set to 80%. The workload capacity of each ES is set to 1800–2000 according to the service workloads as shown in Fig. 10, which is based on a public workload dataset CoMon [6]. Based on workloads, B_m is set to 30. E_m is set to 2 mJ. To evaluate the effectiveness of our approach, we conducted experiments on a real-world crowded-scene mobile user trajectory

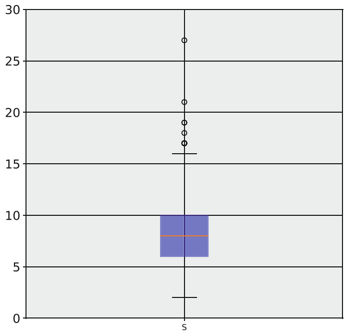

Fig. 10. Workloads of user services

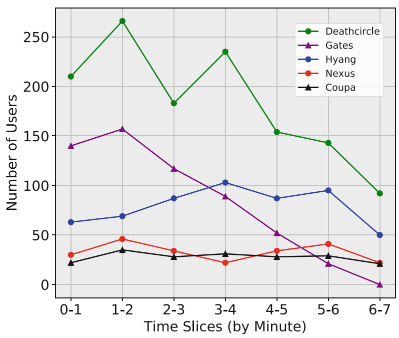

Fig. 11. The number of users by time

dataset for crowded scenes [28], i.e., The Stanford Drone dataset [29]. We choose five scenes: Gates, Deathcircle, Coupa, Hyang, and Nexus as shown in Table 1. As shown in Fig. 11, the number of users in each scene varies by time. We consider the first four rush-hour time slices $T = \{T_1, T_2, T_3, T_4\}$, in our experiment as shown in Table 2.

Table 1. Scenes in two areas

Scenarios	*Area 1*	*Area 2*	Total users	Time (min)
Scene 1	Deathcircle	Deathcircle	871	6:56
Scene 2	Gates	Gates	409	5:03
Scene 3	Hyang	null	326	6:19
Scene 4	Nexus	Nexus	129	6:22
Scene 5	Coupa	Coupa	117	6:39

Table 2. Four time slices

Time Slices	T_1	T_2	T_3	T_4
Period (min)	00:00–1:00	1:00–2:00	2:00–3:00	3:00–4:00

5.3 Experiment Evaluation and Analysis

We evaluated all approaches by: 1) m_s, B_r and E_S, which evaluate quality of service migrations; 2) ΔStd, which evaluates the load fairness; 3) the frequency of using backbone network by the impact of N_S.

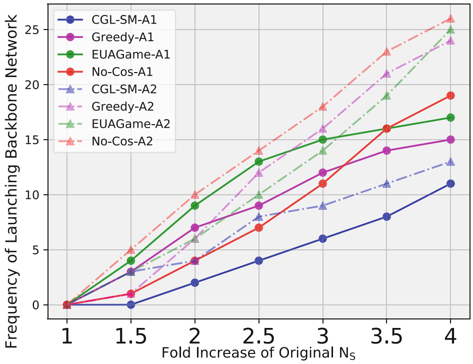

Fig. 12. Using backbone network by fold change of N_S

Impact of N_S on Using Backbone Network: When the number of services rises rapidly, the ESs in a coalition are allowed to transfer the workloads to other ESs by using backbone network. As shown in Fig. 12 (suffix $A1$ and $A2$ denote *Area 1* and *Area 2*), the occurrence rate of using backbone network of CGL-SM is clearly lower than those of its competitors due to the fact that CGL-SM achieves a better load distribution among coalition members.

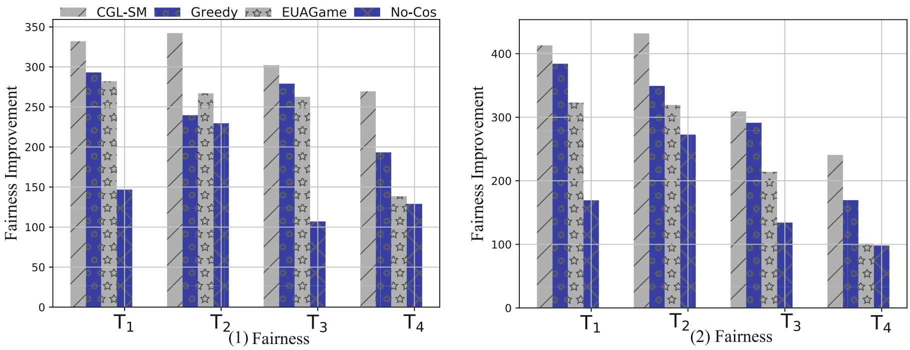

Fig. 13. Δ Std in *Area 1*, *Area 2*

Load Fairness ($\boldsymbol{\Delta Std}$): As shown in Fig. 13, all four migration methods improve the fairness of workload among all ESs, but our approach achieves the highest amount of fairness improvement at all time slices in both areas. The average advantages of CGL-SM over EUAGame, Greedy, No-Cos in *Area 1* are 16.9%, 8.7%, 35.6% in terms of ΔStd, and 24.3%,7.6%, 45.4% in *Area 2*, respectively. As can be seen, the advantage of our proposed method is achieved due to the fact that it chooses ESs appropriately while EUAGame tends to choose the ES with lowest load and Greedy tends to choose the nearest one. No-Cos does not balance loads so well, as workloads are gathering at the boundary ESs between *Area 1* and *Area 2*, other ESs gathers much fewer workloads.

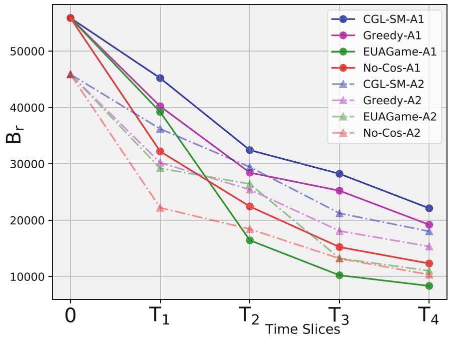

Fig. 14. B_r in two areas at different time slices

Service Migrations ($\boldsymbol{B_r}$, $\boldsymbol{m_s}$, $\boldsymbol{E_S}$): As shown in Fig. 14 (suffix $A1$ and $A2$ denote *Area 1* and *Area 2*), B_r in *Area 1* and *Area 2* are depicted. CGL-SM also costs least budget in both cases (in *Area 1*, CGL-SM is 8.3%, 10.6%, 19.3% less than Greedy, EUAGame, No-Cos, and in *Area 2*, it is 5.1%,12.8%, 13.5%, respectively). It is due to the fact that CGL-SM migrates services according to their workloads. Namely, a service with larger workload is migrated in a lower frequency. EUAGame uses the most budget, as it selfishly migrates services for a lower system cost, which inversely creates more burdens in a crowded scenario. As shown in Fig. 15, for both areas, CGL-SM takes fewer migrations than EUAGame and Greedy. Averagely, in *Area 1*, CGL-SM is 5.6% fewer than Greedy, and 16.3% fewer than EUAGame, 11.5% fewer than No-Cos. And in *Area 2*, CGL-SM is 6.1% fewer than Greedy, 9.8% fewer than EUAGame,10.5%

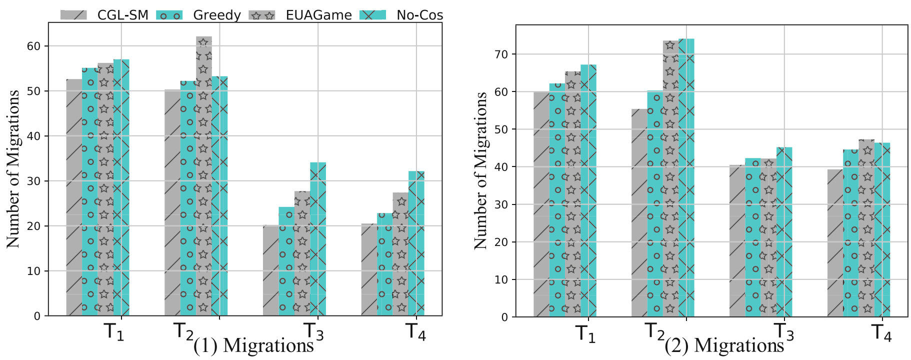

Fig. 15. m_s in *Area 1*, *Area 2*

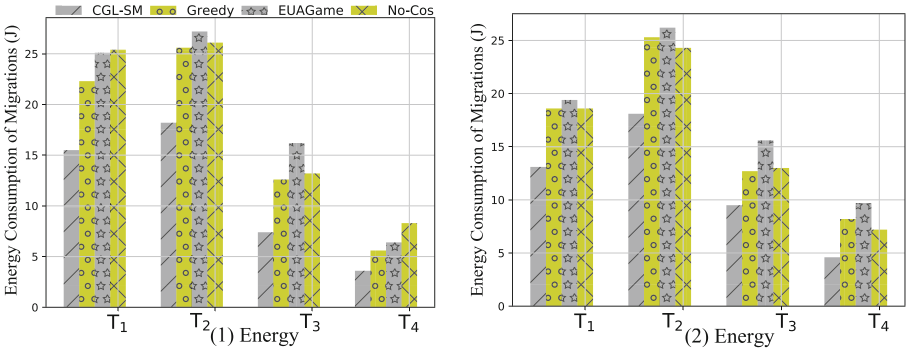

Fig. 16. E_S in *Area 1*, *Area 2*

fewer than No-Cos. As shown in Fig. 16, the energy consumption of migrations in two areas are depicted. We see that CGL-SM has the least energy consumption for migrations compared to other approaches (in *Area 1*, CGL-SM is 15.6%, 28.3%, 21.5% less than Greedy, EUAGame, No-Cos, and in *Area 2*, it is 13.1%, 25.8%, 22.3%, respectively). The advantage of our approach is achieved due to the fact that: 1) the objective model of CGL-SM aims at decreasing the variance of ES workloads in a coalition. Meanwhile, the mobility of users and the user-server proximity constraint are appropriately exploited in balancing workloads of ESs; 2) as CGL-SM performs fewer migrations, then it consumes less migration-overhead. 3) some baseline algorithm (e.g. EUAGame) aims to decrease system cost by using more service migrations, which is not effective in a crowded scenario.

6 Conclusion

In this paper, we propose a location-aware MEC service migration approach for mobile user reallocation in crowded scenes. The proposed method leverages a modified-k-means-based strategy for the formation of coalitions, a workload-based method for the detection of hotspots, and a load-fairness-based strategy for ES workload allocation. Additionally, coalitions are connected by backbone network in case that massive workloads of services could not be sustained by a single coalition. A case study based on real-world datasets of edge-user distribution and trajectory traces demonstrates that our proposed method beats its peers in terms of load-fairness, the number of service migrations required, and energy consumption of migrations. In future work, we will explore a hybrid approach, namely, D2D and edge-cloud mode.

References

1. Niu, X., et al.: Workload allocation mechanism for minimum service delay in edge computing-based power Internet of Things. IEEE Access **7**, 83771–83784 (2019)

2. Puthal, D., et al.: Secure and sustainable load balancing of edge data centers in fog computing. IEEE Commun. Mag. **56**, 60–65 (2018)
3. Li, T., et al.: Crowded scene analysis: a survey. IEEE Trans. Circuits Syst. Video Technol. **25**, 367–386 (2015)
4. Jameel, F., et al.: A survey of device-to-device communications: research issues and challenges. IEEE Commun. Surv. Tutor. **20**, 2133–2168 (2018)
5. Wang, W., et al.: Virtual machine placement and workload assignment for mobile edge computing. In: IEEE International Conference on Cloud Networking. IEEE (2017)
6. Park, K.S., Pai, V.S.: CoMon: a mostly-scalable monitoring system for PlanetLab. ACM SIGOPS Oper. Syst. Rev. **40**(1), 65–74 (2006)
7. Pang, A.C., Chung, W.H., Chiu, T.C., Zhang, J.: Latency-driven cooperative task computing in multi-user fog-radio access networks. In: Proceedings of the IEEE 37th International Conference on Distributed Computing Systems (ICDCS), pp. 615–624, June 2017
8. Mohiuddin, I., Almogren, A.: Workload aware VM consolidation method in edge/cloud computing for IoT applications. J. Parallel Distrib. Comput. **123**, 204–214 (2019)
9. Myerson, R.B.: Game Theory: Analysis of Conflict. Harvard Univ. Press, Cambridge (1991)
10. Saad, W., Han, Z., Debbah, M., Hjorungnes, A., Basar, T.: Coalitional game theory for communication networks: a tutorial. IEEE Signal Process. Mag. **26**(5), 77–97 (2009)
11. He, Y., Ren, J., Yu, G., Cai, Y.: D2D communications meet mobile edge computing for enhanced computation capacity in cellular networks. IEEE Trans. Wirel. Commun. **18**, 1750–1763 (2019)
12. Mehmood, Y., et al.: Internet-of-Things-based smart cities: recent advances and challenges. IEEE Commun. Mag. **55**(9), 16–24 (2017)
13. Baccour, E., Erbad, A., Mohamed, A., Guizani, M.: CE-D2D: dual framework chunks caching and offloading in collaborative edge networks with D2D communication. In: 2019 15th International Wireless Communications and Mobile Computing Conference, IWCMC 2019, pp. 1550–1556. (Institute of Electrical and Electronics Engineers Inc.) (2019)
14. Huang, X., Yu, R., Pan, M., Shu, L.: Secure roadside unit hotspot against eavesdropping based traffic analysis in edge computing based internet of vehicles. IEEE Access **6**, 62371–62383 (2018)
15. Wang, S., et al.: Dynamic service migration in mobile edge computing based on Markov decision process. IEEE/ACM Trans. Netw. **27**, 1272–1288 (2019)
16. Liu, L., Liu, X., Zeng, S.: Research on virtual machines migration strategy based on mobile user mobility in mobile edge computing. J. Chongqing Univ. Posts Telecommun. (Nat. Sci. Ed.) **31**(2) (2019)
17. He, H., Qiao, Y., Gao, S., Yang, J., Guo, J.: Prediction of user mobility pattern on a network traffic analysis platform. In: International Workshop on Mobility in the Evolving Internet Architecture, pp. 39–44 (2015)
18. Guo, Q., Huo, R., Meng, H.: Research on reinforcement learning-based dynamic power management for edge data center. In: 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS) (2018)
19. He, Q., et al.: A game-theoretical approach for user allocation in edge computing environment. IEEE Trans. Parallel Distrib. Syst. **31**, 515–529 (2020)

20. Wu, Q., Chen, X., Zhou, Z., Chen, L.: Mobile social data learning for user-centric location prediction with application in mobile edge service migration. IEEE Internet of Things J. **6**, 7737–7747 (2019)
21. Le Tan, C.N., et al.: Location-aware load prediction in Edge Data Centers. In: 2017 Second International Conference on Fog and Mobile Edge Computing (FMEC) (2017)
22. Yin, H., et al.: Edge provisioning with flexible server placement. IEEE Trans. Parallel Distrib. Syst. **28**(4) (2017)
23. Anchuri, P., Sumbaly, R., Shah, S.: Hotspot detection in a service-oriented architecture. In: Proceedings of the 2014 ACM International Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, pp. 1749–1758 (2014)
24. Zhao, H., Deng, S., Zhang, C., Du, W., He, Q., Yin, J.: A mobility-aware cross-edge computation offloading framework for partitionable applications. In: 2019 IEEE International Conference on Web Services (ICWS), Milan, Italy, pp. 193–200 (2019)
25. He, Q.: Swinbine University of Technology EUA Dataset. https://sites.google.com/site/heqiang/eua-repository
26. Google Map. https://www.google.com/maps/@-37.8081158,144.9622256,17z?hl=zh-CN
27. Peng, Q., et al.: Mobility-aware and migration-enabled online edge user allocation in mobile edge computing. In: Proceedings - 2019 IEEE International Conference on Web Services, ICWS 2019, Part of the 2019 IEEE World Congress on Services, pp. 91–98. Institute of Electrical and Electronics Engineers Inc. (2019)
28. Robicquet, A., Sadeghian, A., Alahi, A., Savarese, S.: Learning social etiquette: human trajectory understanding in crowded scenes. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9912, pp. 549–565. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46484-8_33
29. Stanford Drone Dataset (2018). http://cvgl.stanford.edu/projects/uav_data/. Accessed 26 Aug 2018
30. Xia, X., Chen, F., He, Q.: Cost-effective app data distribution in edge computing. IEEE Trans. Parallel Distrib. Syst. **32**(1), 31–44 (2020). https://doi.org/10.1109/TPDS.2020.3010521