Minimum-Cost Edge-Server Location Strategy in Mobile Crowdsensing

Dongming Luan, En Wang, Wenbin Liu*, Yongjian Yang, Jie Wu, *Fellow, IEEE*

***Abstract*—Mobile crowdsensing has become a significant sensing technique which takes advantage of mobile devices to collect information about the surrounding. The traditional cloud-based centralized mobile crowdsensing architecture generates significant traffic on networks and computation burden on the cloud. In this paper, we investigate the edge-based mobile crowdsensing architecture, where a group of mobile edge servers is deployed at network edge as the bridge between the central server and mobile users for data filtering and aggregation. Each user may collect multiple types of data in mobile crowdsensing. To facilitate data aggregation, the same type of data carried by different users is supposed to be uploaded to the same mobile edge server. In this scenario, a problem emerges: which server should be activated for processing each type of data in order to minimize the total cost? The cost consists of the facility cost (activating server and processing data) and the service cost (the users' movement cost for uploading data). Furthermore, the problem is formulated as a variant of the uncapacitated multi-commodity facility location problem. In particular, two situations of the problem are studied in our work: (1) for the situation where each user carries at most two types of data, we propose a relaxation based approximation algorithm, which is proved to have a bound to the optimal solution; (2) for a more generalized situation where each user can carry multiple types of data, we propose a connected multi-agent simulated annealing algorithm. Finally, we conduct extensive simulations based on the widely-used real-world datasets: roma/taxi, epfl/mobility and geolife trajectory. The simulation results show that the proposed algorithms demonstrate their superiority over baseline methods and are consistent with the theoretical analysis.**

***Index Terms*—Facility location, mobile crowdsensing, linear relaxation, simulated annealing.**

I. Introduction

In recent years, the proliferation of mobile devices with powerful sensing capabilities in everyday life has led to an appealing sensing paradigm, named Mobile CrowdSensing (MCS) [2], [3], [4], [5]. MCS exploits mobile devices to collect sensing data over urban environments [6], [7]. The collected data gives rise to diverse services ranging from constructing radio environment map [8] and road surface assessment [9] to roadside parking management [10].

The traditional MCS architecture is centralized where mobile users directly upload the sensing data to the central server. In large-scale MCS scenarios, the central server is expected to receive huge data volumes from mobile users, which brings much burden to the central server and networks. Besides, all the sensing data is stored on the central server, which increases the risk of the user privacy leak. Fortunately, driven by the rapid development of Internet of Things and 5G communications, the emergence of mobile edge computing [11], [12], [13] is helpful to compensate for the deficiency of the traditional centralized MCS architecture.

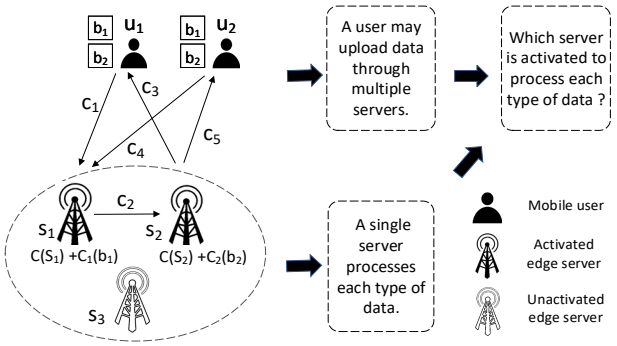


Fig. 1: Problem description in edge-based mobile crowdsensing. For user u_1, it spends movement costs C_1, C_2 and C_3 to upload data. For server s_1, it costs $C(S_1)$ to activate server and $C_1(b_1)$ to process data b_1.

Mobile Edge Computing (MEC) [14] moves the computation tasks that are originally executed on the central server to the vicinity of the data source [15]. In view of this, we propose an edge-based MCS architecture which introduces a new intermediate layer by deploying the mobile edge servers as the bridge between mobile users and the central server. In this way, mobile users upload the sensing data through the mobile edge servers instead of directly uploading to the central server. Then, the intermediate layer processes and aggregates the uploaded data. Each type of data is aggregated on a single mobile edge server. In other words, the MCS platform will guide users to move to different edge servers for data uploading according to the types of data carried by users. Subsequently, the processed and aggregated data is sent to the central server to provide MCS services. Aggregating the same type of data on a single mobile edge server can filter the redundant sensing data. Compared to the raw sensing data uploaded by the users, the data aggregation removes the redundant and erroneous data. This process reduces the amount of data that is sent to the cloud, which decreases the network traffic and the computation load of the cloud.

In order to collect crowdsensing data in the edge-based

A conference version of the paper has appeared in Proceedings of IEEE MASS 2019 [1].

Dongming Luan, En Wang, Wenbin Liu and Yongjian Yang are with the Department of Computer Science and Technology, Jilin University, Changchun, Jilin 130012, China. (E-mail: luandm17@mails.jlu.edu.cn; wangen@jlu.edu.cn; liuwenbin@jlu.edu.cn; yyj@jlu.edu.cn)

Jie Wu is with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122, USA. (E-mail:jiewu@temple.edu)

*The corresponding author is Wenbin Liu.

MCS scenario, we need to cover the following two costs: facility cost and service cost. The facility cost includes the server activation cost and the data processing cost. The service cost is the users' movement costs for uploading data. Since mobile users usually spend most time in only a few places such as home and workplace in daily life, they tend to leave their home or workplace to upload data and return to the initiation. Hence, the service cost is the total distance for the user traveling from the initiation, sequentially passing by corresponding mobile edge servers and returning to the initiation. As shown in Fig. 1, the service cost for user u_1 is the sum of costs C_1, C_2 and C_3. The facility cost for mobile edge server s_1 is $C(S_1) + C_1(b_1)$, which includes the cost for activating server and processing b_1 type of data, respectively. In this scenario, a problem emerges: which server should be activated for processing each type of data in order to minimize the total cost.

The edge-aided MCS architecture has been studied within different fields, such as task allocation [16], user recruitment [17], [18], and vehicular crowdsensing [19]. However, none of them studies the data offloading cost minimization problem in the edge-based MCS scenario. The previous research regarding task offloading in MEC mainly focused on minimizing the makespan [20], [21] or the overhead [22], [23] of task execution. They did not consider the user movement cost during the process of data offloading, and thus can not be directly applied to the case studied in this paper. On the other hand, we should note that the previous research in MCS was devoted to minimizing the data uploading cost from the user perspective [24], [25]. They did not take the data processing overhead into consideration. Different from the aforementioned research, we propose an edge server location strategy to minimize the data offloading cost from both server (facility cost) and user perspective (service cost).

In this paper, we formulate the problem as a variant of the uncapacitated multi-commodity facility location problem. Furthermore, we design the facility location strategies for the following two situations: two-type scenario and multi-type scenario. For the two-type scenario, each user carries no more than two types of sensing data. This case is very common because a person usually spends most of the time in several places every day (e.g., workplace and home). A user tends to collect a type of data when going to work (home to workplace) and collect another type of data on the way home (workplace to home). Therefore, each user carries at most two types of data in this situation. For the multi-type scenario, each user can carry multiple types of data without constraints, which is a generalized extension of the two-type scenario.

The above research ideas raise the following challenges: (1) the simplest facility location problem is NP-hard; (2) different from the traditional facility location problem (there is one kind of data), there are multiple data types and the travelling distance among the edge servers is taken into consideration in the paper, so it is more difficult to decide a minimum cost facility location strategy; (3) for the two-type scenario, it is difficult to find a solution with a bound of total cost to the optimal solution; (4) for the multi-type scenario, the problem becomes more complex and the solution space is so large that the traditional combination optimization techniques could not work well any more.

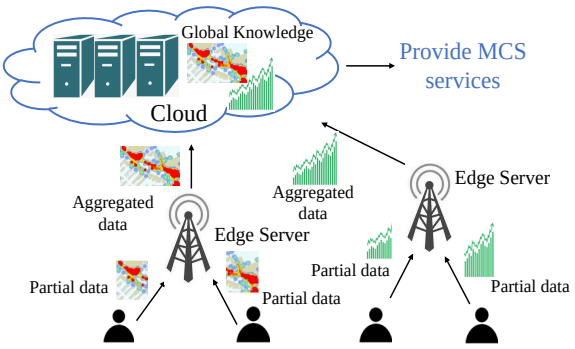


Fig. 2: An example of the edge-based MCS architecture. The edge servers receive the uploaded sensing data from the user. Then, they send the aggregated data to the cloud.

For the two-type scenario, we propose a relaxation-based approximation algorithm. Firstly, we relax the constraints. Then, we solve the relaxed problem and obtain the fractional solution. Finally, we filter the solution and round the fractional solution into integer solution. The relaxation-based algorithm is proved to have a bound to the optimal solution. For the generalized multi-type scenario, we propose an improved connected multi-agent simulated annealing algorithm. The algorithm utilizes multiple agents to search for the solution in parallel. Each agent applies a different search method, and the useful information is passed among the parallel search agents.

The main contributions of this paper are briefly summarized as follows:

- We formally define the edge-server location problem in the edge-based MCS scenario, and formulate it as a uncapacitated multi-commodity facility location problem.
- We propose a relaxation based approximation algorithm and prove that it has a bound to the optimal solution.
- Furthermore, we propose an improved multi-agent simulated annealing algorithm for a generalized case where each user carries multiple types of data.

The remainder of the paper is organized as follows. We review the related works in Section II. In Section III, we present the model and problem. A relaxation based approximation algorithm is proposed in Section IV. In Section V, we prove that the relaxation based algorithm has a bound to the optimal solution. We propose the simulated annealing based algorithm in Section VI. In Section VII, the simulation is conducted to evaluate the performances of the proposed algorithms. We conclude the paper in Section VIII.

II. RELATED WORKS

A. Edge-based Mobile Crowdsensing

There have been some research works focusing on MCS based on distributed architectures. Marjanovic *et al*. [26] propose a mobile edge computing architecture for MCS that can increase the quality of MCS service. The authors in [27] propose two privacy preserving reputation management strategies in MCS based on edge computing to preserve privacy and handle malicious participants. In [19], the authors

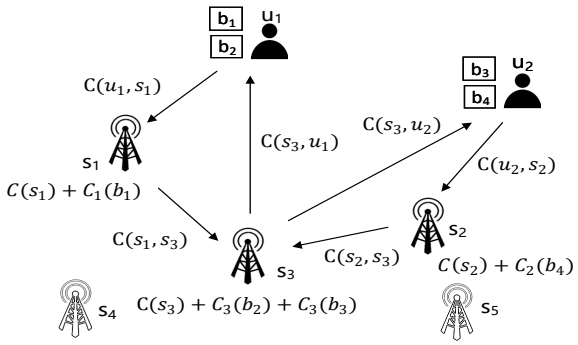


Fig. 3: An example of edge-based MCS system model. Servers s_1 and s_2 are selected to process data type b_1 and b_4 respectively. Server s_3 is selected to process data type b_2 and b_3.

propose an energy-efficient edge-based framework for large-scale vehicular crowdsensing applications, which aims to minimize the energy consumption of vehicles participating in heterogeneous crowdsensing applications. In [28], the authors propose an edge-based network selection scheme in vehicular crowdsensing. Specifically, they formulate the problem as a double objective optimization problem considering maximizing user satisfaction. In [29], in order to support decentralized incentives, the authors propose a distributed ledger framework based on edge computing in MCS scenario. In [30], the authors propose an incentive mechanism for the edge-assisted MCS system, which can achieve the truthfulness and individual rationality. In [16], the authors propose a fog-assisted task allocation method for MCS. Furthermore, a fog-assisted secure data deduplication scheme is proposed to improve communication efficiency. In addition, there are also some research focusing on user recruitment in edge-aided MCS [17], [18], [31]. In [17], the authors investigate the user recruitment for sparse data collection. In [18] and [31], the authors propose the incentive-aware recruitment mechanism for edge-aided MCS. The aforementioned research investigate the edge-based MCS from different aspects. However, none of them studies the edge-server location problem in MCS scenario. To address this problem, we propose an edge-server location strategy to minimize the crowdsensing cost.

B. Facility Location Problem

The facility location problem has attracted many researchers' attention. According to whether considering the capacity of the facility, the facility location problem can be classified into the uncapacitated facility location [32], [33], [34], [35] and capacitated facility location[36], [37], [38]. Moreover, there are many variants of the classical facility location. The k-median problem is a kind of facility location problem, where there is the restriction on the number of facilities opened. The authors in [39] formulate the problem of minimizing the total movement of facilities and clients a k-median problem. The authors in [40] propose a greedy local search algorithm to solve the k-median algorithm. In the k-level uncapacitated facility location problem, the demands must be routed among the facilities in a hierarchical order. There is some research [32], [33] proposing the approximation algorithm for the 2-level facility location problem. Furthermore, [41] gives logarithmic approximation algorithms for the multilevel facility location problem. Different from above research, a client can demand a subset of commodities. We call this multi-commodity facility location (MFL). The authors in [34], [42] propose approximation algorithms for uncapacitated MFL and the authors in [37] propose a large-scale model for capacitated MFL. In addition, some research takes the facility disruption into consideration, where some facilities may be subject to failures [43], [44], [45].

The facility location problem in this paper can be seen as a variant of the multi-commodity facility location problem. However, different from the classical multi-commodity facility location problem, we consider the traveling distance among different facilities and each commodity can be served only on a single facility. These constraints make the problem more difficult than the classical MFL, thus the existing methods cannot be directly applied to solve this problem.

TABLE I: Main notations

Symbol	Meaning
U, S, B	the sets of mobile users, mobile edge servers, and data types.
B_j	the set of data types carried by user j.
C_i	the activation cost for edge server i.
$C_i(b)$	the processing cost for edge server i processing data type b.
β	a combination of data types.
$C_i(\beta)$	the facility cost for activating edge server i in configuration β.
C_j	the service cost of user j.
u_j^b	the virtual user of user j carrying data type b.
c_{ij}	the distance between edge server i and user j.
D_b	the set of virtual users carrying data type b.
j_b	the representative of b type of data.
R	the representative set of all data types.

III. MODEL AND PROBLEM

A. Model

We consider an edge-based MCS architecture in Fig. 2. After collecting the sensing data, the users upload the data to the edge servers. Then, the edge servers perform the data filtering and aggregation. Finally, the aggregated data is sent to the cloud. The cloud analyzes the data and generates the global knowledge that will be used to provide the MCS service.

The group of mobile users is denoted by the set $U = \{1, 2, \ldots, n\}$ and a set of mobile edge servers $S = \{1, 2, \ldots, m\}$. Moreover, after collecting the sensing data, mobile users may carry multiple types of data. All the types of data are denoted as $B = \{1, 2, \ldots, r\}$ and the data types carried by user j are $B_j \subseteq B$. Each mobile edge server i can operate in any configuration $\beta(i) \in 2^B$, specifying the combination of data types it processes with the cost $C_i(\beta)$. Each mobile edge server i has an activation cost $C(i)$ and for each data type b, there is an incremental processing cost $C_i(b)$. Therefore, the facility cost for activating mobile edge server i in configuration β is $C_i(\beta) = C(i) + \sum_{b \in \beta} C_i(b)$.

The service cost for the mobile user is regarded as the mobile users' travel distance in the process of uploading data. Each user begins with an initial location, then heads for the corresponding edge servers one by one and returns to the initial location in the end. As shown in Fig. 3, for user u_1 that will go to server s_1, s_3 for uploading data, u_1 will consume the cost $C(u_1, s_1) + C(s_1, s_3) + C(s_3, u_1)$, which is equal to the total distance u_1 travels. Specifically, the cost for traveling between server and initiation is named as *u-s* service cost such as $C(u_1, s_1) + C(s_3, u_1)$ and the cost for traveling between servers is named as *s-s* service cost such as $C(s_1, s_3)$. The facility cost for server s_1 is $C(s_1) + C_1(b_1)$. Specifically, $C(s_1)$ is the activation cost for s_1 and $C_1(b_1)$ is the processing cost for s_1 to process data type b_1. The main notations used throughout the paper are in Table I.

B. Problem

In this paper, we aim to find a solution to determine which mobile edge servers to activate and which data types are assigned to the activated mobile edge servers for minimizing the total cost. Let C_j denote the service cost of user j. Variable y_i^0 indicates whether mobile edge server i is activated or not. It is 1 when activated and 0 otherwise. Variable $y_i^b = 1$ indicates that mobile edge server i processes b type data and it is 0 otherwise. Variable x_{ij}^b is 1 if user j with b type data is assigned to server i to upload data. Note that when $b = 0$, $C_i(b)$ denotes the cost for activating server i. Hence, our purpose is to find the best facility location strategy for the following optimal problem:

$$
\begin{aligned}
\text{Minimize} \quad & \sum_{i=1}^{m} \sum_{b=0}^{r} C_i(b) y_i^b + \sum_{j=1}^{n} C_j && (1)\\
\text{s.t.} \quad & \sum_{i=1}^{m} x_{ij}^b = 1 && \forall b \in B_j, \forall j \in U \\
& \sum_{i=1}^{m} y_i^b = 1 && \forall b \in B \\
& x_{ij}^b \le y_i^b && \forall b \in B, \forall i \in S, \forall j \in U \\
& y_i^b \le y_i^0 && \forall b \in B, \forall i \in S \\
& x, y \in \{0, 1\}
\end{aligned}
$$

The first constraint ensures that there exists a mobile edge server that can process each data type carried by each user. The second constraint ensures that each data type is processed by a single mobile edge server. The third constraint means that only when the mobile edge server has the ability to process the corresponding data, can the user upload data through it. The fourth constraint guarantees the mobile edge server has the ability to process data only when it is activated. We aim at finding a strategy to minimize the total cost satisfying the above constraints.

IV. FACILITY LOCATION STRATEGY FOR TWO-TYPE SCENARIO

In this section, we propose the relaxation-based algorithm for the scenario where users carry two types of data in detail.

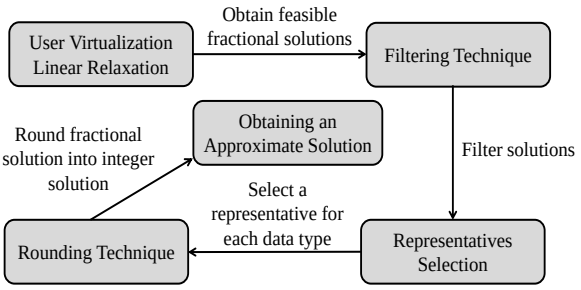


Fig. 4: Workflow of the relaxation-based algorithm for two-type scenario.

The workflow of the algorithm is shown in Fig. 4. Since the optimization problem (1) is difficult to solve directly, we first relax the constraints and transform a user into a set of virtual users where each virtual user has one data type. Then, we solve the relaxation version of the problem (1) and obtain the fractional solution. Next, we filter the solution so that each virtual user is fractionally assigned to the mobile edge servers that are relatively close to it. Furthermore, we select a group of representatives from virtual users and assign the remaining virtual users to the corresponding representatives. Finally, we round the fractional solution to the integer solution. More specifically, we first assign the representatives to the edge servers and then assign the remaining virtual users to the servers that serve their representatives. Each mobile user will upload data through the mobile edge servers that are allocated to its virtual users.

A. User Virtualization and Linear Relaxation

Due to the fact that the objective function (1) is hard to solve directly in polynomial time, we relax the constraint as shown in function (2). Moreover, we only consider minimizing the *u-s* service cost instead of the total service cost temporarily and perform the user virtualization in this step.

$$
\begin{aligned}
\text{Minimize} \quad & \sum_{i=1}^{m} \sum_{b=0}^{r} C_i(b) y_i^b + \sum_{j=1}^{n} C_j && (2)\\
\text{s.t.} \quad & \sum_{i=1}^{m} x_{ij}^b \ge 1 && \forall b \in B_j, \forall j \in U \\
& \sum_{i=1}^{m} y_i^b \ge 1 && \forall b \in B \\
& x_{ij}^b \le y_i^b && \forall b \in B, \forall i \in S, \forall j \in U \\
& y_i^b \le y_i^0 && \forall b \in B, \forall i \in S \\
& 0 \le x, y \le 1
\end{aligned}
$$

The process of user virtualization is as follows: we replicate a set of virtual users for each mobile user so that each virtual user has one data type. The virtual user set for user j is defined as $\{u_j^b : \forall b \in B_j\}$. It is worth noting that, for the user with only one data type, we will transform the user into two virtual users with the same data type. The *u-s* service cost for the virtual user is the distance between the mobile edge server and the initiation. So the *u-s* service cost for the user is equal to the

sum of its virtual users' *u-s* service costs. For a user with only one data type, the *u-s* service cost is the roundtrip from the initiation to the mobile edge server, which is equal to the sum of its two virtual users' *u-s* service costs.

Then, we aim to find a solution to minimize the sum of facility costs and virtual users' *u-s* service costs. After performing the linear relaxation, we will get some fractional solutions where the user is permitted to be splittable and assigned to several edge servers. Although *s-s* service cost is not considered in this step, according to the triangle inequality, *s-s* service cost is lower than the sum of the *u-s* service costs. Furthermore, we will prove that *s-s* service cost of this solution also has a bound to the optimal solution in the next section.

B. Filtering Technique

Then, we apply the filtering technique used in [46] to filter the solution and obtain a new fractional solution, where the new solution satisfies the property that the user is fractionally assigned to mobile edge servers which are not too far away from it.

We fix a constant $0 < \alpha < 1$ and define the $\alpha-$point, $p_j^b(\alpha)$, for each virtual user u_j^b. Then, we order the mobile edge servers which serve u_j^b according to non-decreasing distance to j. Let c_{ij} denote the distance between mobile edge server i and virtual user u_j^b. Let ϕ be a permutation of servers that serve u_j^b such that $c_{\phi(1)j} \leq c_{\phi(2)j} \leq \cdots \leq c_{\phi(k)j}$. Then, $p_j^b(\alpha) = c_{\phi(i^*)j}$, where $i^* = \min\{i' : \sum_{i=1}^{i'} x_{\phi(i)j}^b \geq \alpha\}$. For each virtual user u_j^b, let $\alpha_j^b = \sum_{i:c_{ij} \leq p_j^b(\alpha)} x_{ij}^b$. Obviously, $\alpha_j^b \geq \alpha$. We merely set

$$\overline{x}_{ij}^b = \begin{cases} x_{ij}^b/\alpha_j^b, & c_{ij} \leq p_j^b(\alpha); \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

And for each $i \in S$, we set $\overline{y}_i^b = \min\{1, y_i^b/\alpha\}$. After the filtering process, we will obtain a new fractional solution, which has the property that when a virtual user u_j^b is fractionally assigned to a mobile edge server s_i, the corresponding cost c_{ij} is not too large.

C. Representatives Selection

After filtering the fractional solution, we select a set of virtual users as representatives. The process of representatives selection is similar to [42]. Specifically, For each data type, we classify the virtual users who carry that type of data into a group and select a representative from each group. The process of representatives selection of different data types is independent.

The detailed process is described as follows: for a data type b, let D_b denote the set of virtual users who carry b type data. Let the selection cost of a virtual user $j \in D_b$ be $\hat{c}_j = \sum_{j' \in D_b} c_{jj'}$, where $c_{jj'}$ is the distance between user j and j'. Variable j_b signifies the user with the minimum selection cost among all users in D_b and j_b is selected to be the representative of b type of data. The representative set of all data types is denoted as R.

D. Rounding Technique

The rounding technique is used to round the fractional solution into the integer solution. The algorithm executes iteratively and it keeps a feasible fractional solution $(\hat{x}, \hat{y})$. Initially, let $(\hat{x}, \hat{y}) = (\overline{x}, \overline{y})$. In the process of the algorithm, we denote $\hat{S}$ as the set of partially activated mobile edge servers, specifically, $\hat{S} = \{i \in S : \exists b, \hat{y}_i^b > 0\}$. For each $j_b \in R$, we define S' as the set of mobile edge servers for which $\hat{x}_{ij}^b > 0$, that is, $S' = \{i \in \hat{S} : \hat{x}_{ij}^b > 0\}$. Then the algorithm will find the server $i \in S'$ so that $C_i(b)$ is the smallest and let i' denote this server. Following this, assign j_b to i' and round the value $\hat{y}_{i'}^b = 1$ and $\hat{y}_i^b = 0$ for each $i \in S - i'$. Accordingly, $\hat{x}_{i'j}^b$ is set to be 1 and $\hat{x}_{ij}^b$ is 0 for each $i \in S - i'$.

The algorithm executes iteratively until all the representatives are assigned to a mobile edge server. Note that this rounding process guarantees that each data type must be processed by a single mobile edge server. Finally we assign the other virtual users to the servers that serve their representatives and each mobile user will go to the mobile edge servers that are allocated to its virtual users.

V. THEORETICAL ANALYSIS

In this section, we prove the relaxation based approximation algorithm has a bound to the optimal solution.

Firstly, we use the filtered fractional solution and representatives to construct a k-set cover instance and prove that $\overline{y}$ is a feasible fractional solution of the constructed k-set cover instance. The k-set cover problem is a special case of the weighted set cover problem in which each set has no more than k elements. We use the following IP formulation for the instance of k-set cover problem. Here we define a server-configuration pair (i, β) where $i \in S$ and $\beta \in 2^B$. There is a cost $C_i(\beta)$ for each server-configuration pair (i, β). Note that we only consider the virtual users in R and a virtual user u_j^b is covered by (i, β) if and only if $x_{ij}^b > 0$ and $b \in \beta$. Let variable z_i^β be 1 if the server-configuration pair (i, β) is included in the solution. The IP formulation for the instance of k-set cover is as follows:

$$\begin{aligned} \text{Minimize} \quad & \sum_{i,\beta} C_i(\beta) z_i^\beta \\ \text{s.t.} \quad & \sum_{(i,\beta): x_{ij}^b > 0, b \in \beta} z_i^\beta \geq 1 \quad \forall j_b \in R \end{aligned} \tag{4}$$

The fractional solution $\overline{y}$ can be transformed as a fractional solution of an instance of k-set cover problem so that only polynomially many server-configuration pairs have non-zero values [42]. Given a mobile edge server $i \in S$, sort the data types in the non-decreasing order of $\overline{y}_i^b$, that is, $\overline{y}_i^0 \geq \overline{y}_i^1 \geq \overline{y}_i^2 \geq \ldots \geq \overline{y}_i^k$. Let $[b]=\{1, 2, \ldots, b\}$. We activate mobile edge server i in configuration $[b]$ to extent $z_i^{[b]} = \overline{y}_i^b - \overline{y}_i^{b+1}$ for $b = 1, 2, \ldots, k-1$ and $z_i^{[k]} = \overline{y}_i^k$. Hence, for each mobile edge server s_i, there are at most k configurations that $z_i^\beta > 0$.

Then, we prove the bound of the proposed approximation algorithm using the following lemmas.

TABLE II: User-server distance.

s \ u	u_1	u_2	u_3	u_4
s_1	5	6	18	6
s_2	5	6	13	10
s_3	5	13	13	6

TABLE III: Facility cost.

s \ b	b_0	b_1	b_2
s_1	3	8	11
s_2	3	10	5
s_3	3	5	11

Lemma 1. *Function (4) is an instance of k-set cover and $z = \overline{y}$ is a feasible fractional solution of the linear relaxation version for the instance, the cost of which is no more than $\sum_{i,\beta} C_i(\beta)\overline{y}_i^{\beta}$.*

Proof. Since we select only one representative for each data type, the total number of representatives is equal to the total number of data types and the cardinality of each set in the formulated k-set cover instance is no more than k (let k be the total number of data types r). Since $(\overline{x}, \overline{y})$ is a feasible solution of the linear relaxation of the formulated objective function, there exists $\sum_i \overline{x}_{ij}^b \geq 1$ for each $j_b \in R$ and $z_i^{\beta} \geq \overline{x}_{ij}^b$ for each $b \in \beta$, which ensures that z is a feasible fractional solution of the formulated k-set cover instance and bounds the cost of the fractional solution. □

Lemma 2. *There is an integer solution $(\hat{x}, \hat{y})$ satisfying the following properties: (1)$\hat{x}_{ij}^b \leq \hat{y}_i^{\beta}, \forall b \in \beta$; (2) $\hat{x}_{ij}^b = 1$ only if $\overline{x}_{ij}^b > 0$; (3) $\hat{y}_i^{\beta} = 1$ only if $\overline{y}_i^{\beta} > 0$; (4) $\sum_{i,\beta} C_i(\beta)\hat{y}_i^{\beta} \leq \log k \sum_{i,\beta} C_i(\beta)\overline{y}_i^{\beta}$.*

Proof. Due to the fact that the integrality gap of k-set cover problem is no more than $\log k$ [47], there is an integer solution $\hat{z}$ for the formulated k-set cover instance and its cost is no more than $\log k \sum_{i,\beta} C_i(\beta)\overline{y}_i^{\beta}$ and let $\hat{y} = \hat{z}$. Hence, property (3) and (4) are proved. It is clear that for each representative $j_b \in R$, there must exist $y_i^{\beta} = 1, b \in \beta$ such that $\overline{x}_{ij}^b > 0$. Let $\hat{x}_{ij}^b$ be 1 and 0 otherwise. So property (1) and (2) are proved. □

Finally we activate the mobile edge server and fix the data type for which $\hat{y}_i^{\beta} = 1$. When considering the constraint that each data type is processed by a single mobile edge server, the solution is a special case of the k-set cover problem where each element is covered by only one set. Due to the fact that $\overline{y}_i^b = \min\{1, y_i^b/\alpha\}$, hence $\overline{y}_i^b \leq y_i^b/\alpha$. Due to the fact that the optimal solution of integer programming problem is not superior to the optimal solution of its relaxation, we use the optimal solution of the linear relaxation for the objective function as our lower bound. Hence, the facility cost is bounded within $\frac{\log k}{\alpha}$ of the optimal facility cost.

Then we prove that u-s service cost has a bound to the optimal. Let $\varphi(j)$ be the server that is assigned to process virtual user j in the solution. Variable $\varphi^*(b)$ denotes the server that processes data type b in the optimal solution. Variable $c_{j,\varphi(j)}$ is u-s service cost of virtual user j.

Lemma 3. *The u-s service cost of the proposed solution is no more than $(\frac{3}{1-\alpha}+4)\cdot C_{opt}$ in which C_{opt} is the u-s service cost of the optimal solution.*

Proof. Let j_b^* be the virtual user that minimizes $c_{j,\varphi^*(b)}$ in D_b. Since the inequality $c_{j_b^*,\varphi^*(b)} \leq c_{j,\varphi^*(b)}$ for all $j \in D_b$, there exists $\hat{c}_{j_b^*} \leq 2\sum_{j\in D_b} c_{j,\varphi^*(b)}$. Due to the fact that the representative j_b minimizes $\hat{c}_j$ in D_b, we have $\hat{c}_{j_b} \leq \sum_{j\in D_b} 2c_{j,\varphi^*(b)}$. Then, consider a virtual user $j \in D_b$ and $\varphi'(j)$ is the server that processes j while ignoring the constraint that each data type is processed by a single server. Because of the triangle inequality, we have $c_{j_b,\varphi'(j)} \leq c_{j,j_b} + c_{j,\varphi'(j)}$. Furthermore, $c_{j,\varphi(j)} \leq 2c_{j,j_b} + c_{j,\varphi'(j)}$. The virtual user set is V. As $\hat{c}_{j_b} = \sum_{j\in D_b} c_{j,j_b}$, by summing $c_{j,\varphi(j)}$ over all virtual users, we have $\sum_{j\in V} c_{j,\varphi(j)} \leq (\frac{3}{1-\alpha}+4)\cdot C_{opt}$, where $\frac{3}{1-\alpha}$ is the approximation ratio of u-s service cost, ignoring the constraint that each data type is processed by a single mobile edge server and it is proved in [42]. The total u-s service cost is equal to the sum of u-s service costs of all virtual users.

□

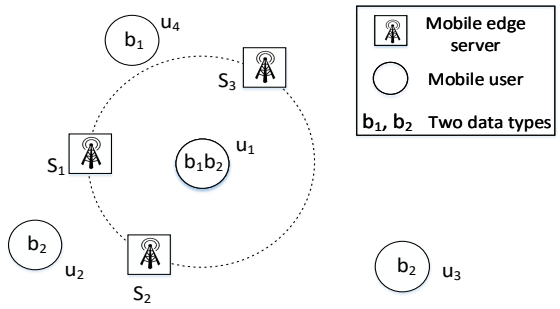


Fig. 5: The scenario for proving the bound of s-s service cost.

Finally, we prove the bound of s-s service cost. Let d_{max} denote the maximum distance among edge servers and d_{min} denote the minimum distance among edge servers.

Lemma 4. *The s-s service cost of the proposed approximation solution is bounded within $\frac{d_{max}}{d_{min}}$ of the optimal.*

Proof. There is a situation as shown in Fig. 5 in which there are four mobile users u_1, u_2, u_3, u_4 and three candidate mobile edge servers s_1, s_2, s_3. There are two data types: b_1 and b_2. The distance between server s_1 and s_2 is 6, which is the minimum distance among servers and denoted as d_{min}. The distance between s_2 and s_3 is 10, which is the maximum distance and denoted as d_{max}. The distance between s_1 and s_3 is 8. The distance between users and servers and facility cost configuration are shown in Table. II and III. It is worth noting that b_0 denotes the activation costs for servers in Table. III. So in this case, the proposed solution will configure server s_2 and s_3 to process data b_2 and b_1 respectively. The s-s service cost is d_{max}. However, the s-s service cost of the optimal solution is the distance between s_1 and s_2, which is d_{min}. In other situations, the ratio between s-s service cost of the proposed algorithm and the optimal solution is no more than $\frac{d_{max}}{d_{min}}$. Hence, the s-s service cost is bounded within $\frac{d_{max}}{d_{min}}$ of the optimal. The lemma is proved. □

Hence, in conclusion, our proposed algorithm is a constant-factor approximation algorithm and has a bound to the optimal solution. The approximation ratio is the maximum value of $\{\frac{\log k}{\alpha}, \frac{3}{1-\alpha}+4, \frac{d_{max}}{d_{min}}\}$.

Algorithm 1 The construction algorithm to produce the initial solution.

Input: the data type set B, the candidate server set S.
Output: the initial solution h.

while $|B| \neq 0$ **do**
  for all data type $b \in B$ **do**
    $s_b = \arg\min_{s \in S} C(b, s)$.
  end for
  $b^{'} = \arg\min_{b \in B} C(b, s_b)$
  assign data type $b^{'}$ to server $s_{b'}$ for constructing h.
  $B \leftarrow B - b^{'}$.
end while
return h.

VI. FACILITY LOCATION STRATEGY FOR MULTI-TYPE SCENARIO

For the situation where mobile users can carry multiple types of data without limitations, the problem becomes more intractable. Compared with the situation where each mobile user has at most two data types, its solution space is much larger. Specifically, when each user has at most two types of data, it will go to at most two mobile edge servers to upload data. For the same combination of servers assigned to a user, the service cost is unique. However, when each user carries multiple data types, it tends to go to multiple servers. The route planning problem should be considered. The same combination of servers may correspond to different service costs, because of the different sequences that the mobile user visits. Hence, the problem for this situation is very complex.

The computational complexity and the large solution space make it difficult for the traditional combination optimization technique to work well. Hence, we propose a Connected Multi-agent Simulated Annealing algorithm (CMSA) to address this problem. The traditional simulated annealing algorithm only utilizes one thread to search for the solution. To improve the searching capability and convergence rate, CMSA uses multiple threads, named agents, to search for the solution in parallel. To diversify the searching direction, each agent applies a different method to modify its current state. Moreover, the useful information is passed among the parallel search agents at regular intervals. Then, the agents abandon the unfruitful searches and start the search from the state where the best agent has reached.

A. The Construction Algorithm

We use a heuristic algorithm to construct the initial solution. As shown in Algorithm. 1, the algorithm assigns the data types to the mobile edge server iteratively. We define $C(b, s)$ as the change of the current total cost after assigning data type b to server s. The construction algorithm selects one data type and assigns it to the corresponding server in each iteration. It firstly selects the server s_b that minimizes $C(b, s)$ from the candidate server set for each unassigned data type b. Then, we select data type $b^{'}$ that minimizes cost $C(b, s_b)$ and assign it to the corresponding server $s_{b'}$. The initial solution is constructed by this means iteratively.

To reduce the computational complexity, the service cost for a user in $C(b, s)$ is computed as the sum of the distance from the user to each of the corresponding servers. According to the triangle inequality, it is easy to find out that the real service cost of the final solution is lower than the double of the service cost computed in the algorithm. It is worth noting that the service cost used in the successor search algorithm is computed in the same way as in $C(b, s)$.

Algorithm 2 Successor search algorithm of one agent.

Input: the initial solution h, T, T_{max}, γ, δ.
Output: the best solution h^*.

1: initialize the best solution $h^* \leftarrow h$;
2: $j \leftarrow 0$.
3: **while** stopping condition not met **do**
4: generate a new successor $h^{'}$ of current solution h.
5: **if** $C(h^{'}) < C(h)$ **then**
6: $h \leftarrow h^{'}$.
7: **else**
8: $h \leftarrow h^{'}$ with probability $\exp((C(h) - C(h^{'}))/T)$.
9: **end if**
10: **if** $C(h) < C(h^*)$ **then**
11: $h^* \leftarrow h$, $T_r \leftarrow T$.
12: **end if**
13: **if** $j\%\delta = 0$ **then**
14: obtain the current global best solution $\hat{h}$ among all agents.
15: **if** $C(h^*) > C(\hat{h})$ **then**
16: $h \leftarrow \hat{h}$, $h^* \leftarrow \hat{h}$
17: **end if**
18: **end if**
19: $T \leftarrow \gamma \times T$.
20: **if** $T < 0.01$ **then**
21: $T_r \leftarrow 2 \times T_r, T \leftarrow \min\{T_r, T_{max}\}$.
22: **end if**
23: $j \leftarrow j + 1$.
24: **end while**
25: **return** h^*.

B. Successor Search Algorithm

Since the combination of the mobile edge servers is not considered in the construction algorithm, the initial solution is merely a local optimal solution. Therefore, we propose a successor search algorithm to modify the initial solution and get the successor of the current solution to search for a better approximate solution.

As presented in Algorithm. 2, the algorithm aims to optimize the initial solution by searching the successor solution until the stopping condition is met. It first initializes the best solution h^* to be the initial solution h. The main loop of lines 3-24 consists of the main part of the algorithm. In each iteration of the loop, a new successor $h^{'}$ of the current solution is generated by removing some data types from activated servers and reinserting them into other servers (line 4). It is worth noting that different agents apply different methods to modify the current solution in this paper. Then, the algorithm determines whether the solution $h^{'}$ is accepted or not (lines 5-12). Specifically, if the cost of the new solution $h^{'}$ is lower than that of the current solution h, $h^{'}$ is accepted. Otherwise,

TABLE IV: Facility cost.

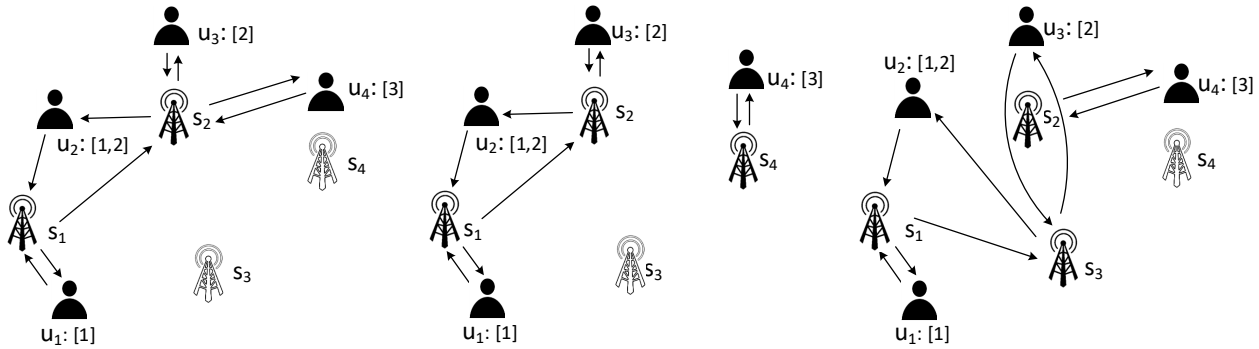
Server \ Type	0	1	2	3
s_1	2	1	3	3
s_2	5	3	2	1
s_3	2	3	1	2
s_4	4	4	2	5

TABLE V: User-server distance.

Server \ User	u_1	u_2	u_3	u_4
s_1	1	3	5	7
s_2	5	3	1	3
s_3	5	5	4	3
s_4	7	5	3	1

TABLE VI: Server-server distance.

Server \ Server	s_1	s_2	s_3	s_4
s_1	0	5	5	7
s_2	5	0	4	3
s_3	5	4	0	3
s_4	7	3	3	0



(a) The scenario for using APX method. (b) The scenario for using DIS method. (c) The scenario for using LF method.

Fig. 6: An example for the comparison of APX, DIS and LF algorithms. The mobile edge servers in black color denote the activated servers and the rest are unactivated servers.

the probability of accepting $h^{'}$ is $\exp((C(h)-C(h^{'}))/T)$ (lines 5-9), where $C(h)$ is the cost of the solution h and T is the temperature. Subsequently, if the cost of the current solution is lower than that of the current best solution h^*, the current best solution is set to be h and T_r is set as T (lines 10-12). T_r is used to record the temperature when the best solution is updated. The agent compares its current local best solution h^* with the current global best solution $\hat{h}$ among all agents at a predefined interval δ. If $C(h^*) > C(\hat{h})$, h is updated to be $\hat{h}$ and h^* is set to be $\hat{h}$ (lines 13-18). In each iteration, the temperature T is updated as $\gamma \times T$ where $0 < \gamma < 1$ (line 19). To avoid the search getting stuck in a local minimum, the temperature will increase when the temperature is lower than 0.01, which can increase the probability of escaping from the trap [48]. The temperature is limited to T_{max} to limit the probability of accepting a solution that is not good. The stopping condition is satisfied when the iteration number is equal to the predefined maximum iteration number. Then the searching process stops and each agent returns its current best solution. Finally, CMSA algorithm returns the best solution among all agents.

In the successor search algorithm, each agent modifies the current solution by applying the data type removal method and insertion method. We introduce two data type removal methods and two data type insertion methods in this paper. The different agents can apply different combinations of the removal method and insertion method to modify the current solution. Besides, the agent can also randomly select a mobile edge server and remove all the data types on it. Then, the agent randomly inserts each removed data type to a server. The random method can help diversify the solution.

C. Data Type Removal Method

In successor search algorithm, in order to modify the current solution, we first select some data types and then remove them from the current solution. This subsection gives a description of some heuristics for data type removal.

1) One Server Removal (OSR): We define C_s as the total cost of all data types processed by server s and N_s as the number of data types assigned on server s. It is worth noting that C_s includes the sum of facility cost and service cost. OSR method calculates the average cost $\frac{C_s}{N_s}$ for each activated server. Then OSR selects the server with the highest average cost and removes all its assigned data types.

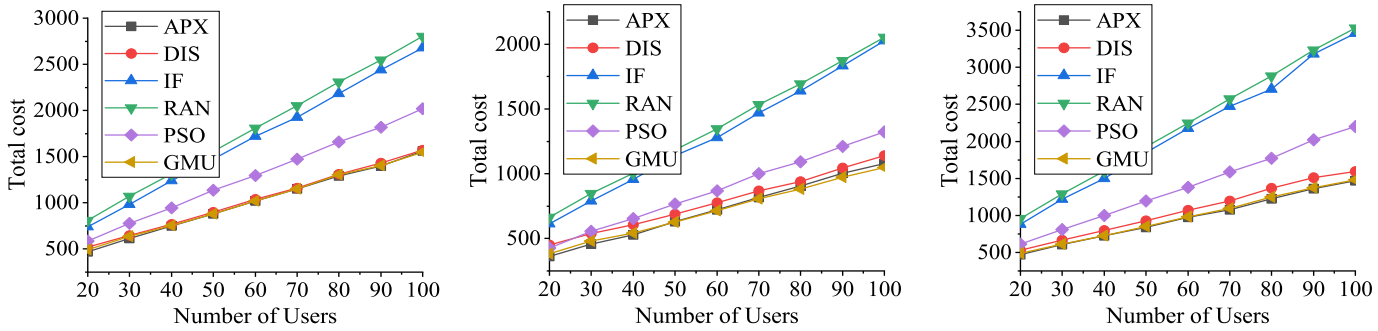
2) Largest Change Removal (LCR): LCR removes q data types iteratively. In each iteration, LCR calculates the change of total cost when removing each unremoved data type and it removes the data type with the largest change of total cost.

D. Data Type Insertion Method

After removing some data types from the current solution, we have to reinsert them to servers for constructing a new successor solution. This subsection introduces some heuristics to reinsert data types.

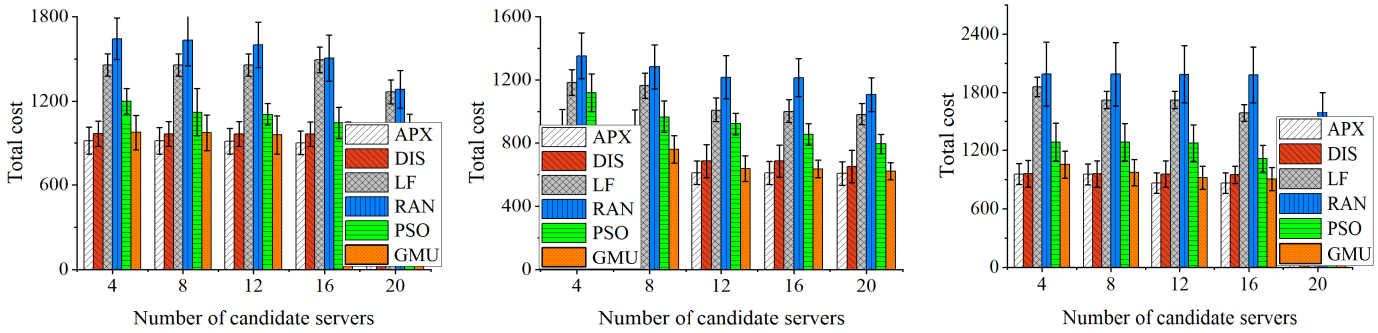
1) One Server Insertion (OSI): Given some data types to be inserted, OSI inserts them into a single server. Specifically, OSI selects the server that minimizes the total cost and assigns all the data types on it.

2) Lowest Change Insertion (LCI): LCI is similar to the construction algorithm and it inserts the data type iteratively. In each iteration, LCI selects the data type with the minimal insertion cost and inserts it into the corresponding server. Specifically, for each data type, the insertion cost is the



(a) roma/taxi set (b) epfl/mobility set (c) geolife trajectory set

Fig. 7: The simulation results in terms of number of users.



(a) roma/taxi set (b) epfl/mobility set (c) geolife trajectory set

Fig. 8: The simulation results in terms of number of candidate servers.

minimum change of total cost after inserting the data type into the candidate server set.

VII. PERFORMANCE EVALUATION

A. Data Preparation

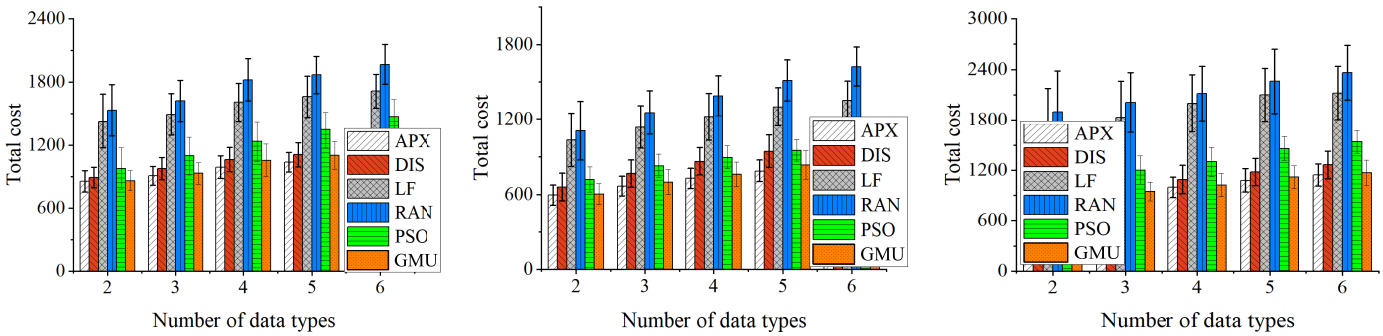
Three widely-used real-world traces, roma/taxi set[49], epfl/mobility set [50] and geolife trajectory set [51] were used to evaluate the performances of the proposed facility location strategies for minimizing the sensing cost. The roma/taxi set contains about 320 taxis' GPS coordinate mobility traces in Rome, Italy collected over 30 days. The epfl/mobility set records the GPS trajectories of approximately 500 taxis in the San Francisco Bay Area, USA, which are collected over 30 days. The geolife trajectory set was collected in Geolife project by 182 users. It contains 17,621 GPS trajectories with a distance of 1.2 million kilometers.

We construct five strategies LF, DIS, RAN, PSO, and GMU. The performances of the proposed algorithms are compared with these strategies in the paper. For each data type b, LF strategy selects the mobile edge server i which processes the data type with the lowest cost, $C_i(b)$. For each data type, DIS strategy selects the mobile edge server which has the minimum average distance to the set of mobile users with this data type. RAN strategy randomly selects a mobile edge server for each data type. We modify the algorithms in [45] and [31] to apply to our scenario, and the algorithms are denoted as PSO and GMU respectively. We take the total cost as the evaluation metric, which is the sum of the facility cost and service cost for all activated mobile edge servers and mobile users. In the simulation, the facility costs are generated randomly following a uniform distribution and the service costs for the users are set as the travel distance when uploading the sensing data in the real-world dataset. We select the first GPS location of the user's trajectory as the initial location, and we randomly select POI locations as the candidate mobile edge server locations.

B. Simulation Results for Two-Type Scenario

Firstly, we give an example to illustrate the differences in the execution results of APX, DIS and LF strategies, where APX is our proposed strategy. Secondly, we evaluate the performances along with the changes in the number of candidate servers, number of mobile users and number of data types. The specific evaluation results are demonstrated in Figs. 7-9. Then, an example of simulation result of APX in roma/taxi set is presented. Finally, the optimal results are compared with the proposed approximation algorithm. The value of parameter α used in Figs 4-7 is set as 0.4.

Firstly, we give an example to illustrate the difference among APX, DIS and LF algorithms. The example consists of four candidate mobile edge servers s_1, s_2, s_3, s_4, four mobile users u_1, u_2, u_3, u_4 and three data types from 1 to 3. Note that in Table. IV, when the data type is 0, it denotes the activation cost for each server. The configuration information are shown in detail in Table. IV-VI respectively.



(a) roma/taxi set (b) epfl/mobility set (c) geolife trajectory set

Fig. 9: The simulation results in terms of number of data types.

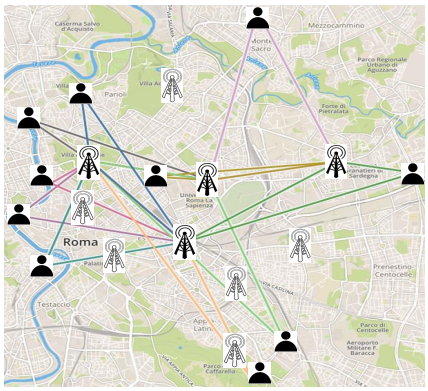


Fig. 10: A presentation of the simulation result of APX in roma/taxi set. The mobile edge servers in black color denote the activated servers and the rest are unactivated servers.

Fig. 6 illustrates the scenarios for using three algorithms respectively. When using APX algorithm, server s_1 is activated to process type 1 of data and server s_2 is activated to process type 2 and 3 of data. The service cost and facility cost are 21 and 11 respectively. The total cost is 32. When using DIS algorithm, server s_1, s_2 and s_4 are activated to process type 1, 2 and 3 of data respectively. The service cost and facility cost are 17 and 19 respectively. The total cost is 36. When using LF algorithm, server s_1, s_2 and s_3 are activated to process type 1, 3 and 2 of data respectively. The service cost and facility cost are 29 and 12 respectively. So the total cost is 41. In this case, although the service cost of DIS is lowest, the performance of APX is best. Since each mobile edge server has an activation cost and LF algorithm only focuses on minimizing the processing cost, it may have more activation cost than that of APX. Furthermore, this results in the fact that the facility cost of LF is more than that of APX sometimes. In conclusion, the APX algorithm minimizes the total cost by considering the facility cost and service cost comprehensively, which performs better than DIS and LF algorithms, though DIS and LF may have relatively low service cost and facility cost sometimes.

Secondly, we compare the total cost in terms of number of users in Fig. 7. The simulation results show that the total costs of all six algorithms increase with the growth of the number of users. More specifically, the total cost performance ranks as follows in most cases: APX<GMU<DIS<PSO<LF<RAN, though the performance of GMU and APX is close.

TABLE VII: The comparison of facility cost between APX and the optimal solution.

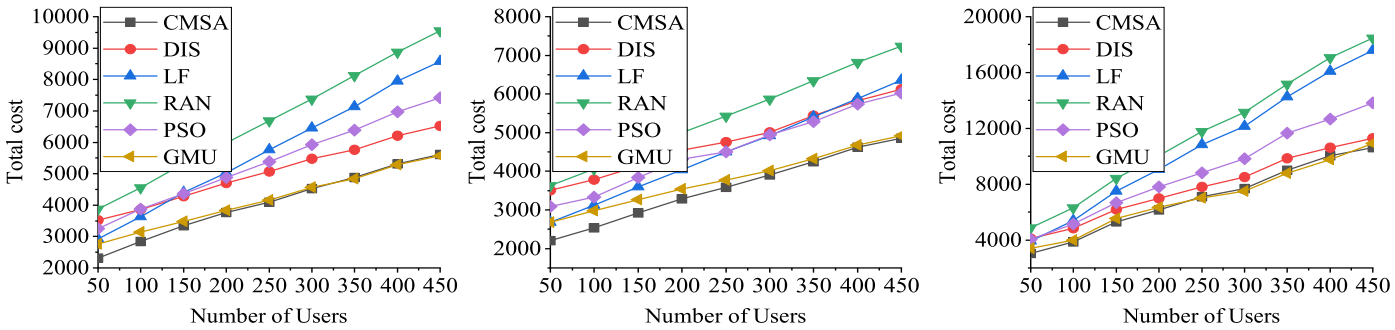
Data type number	APX	Optimal	Ratio	Bound
2	109	79	1.37	1.38
3	154	140	1.10	2.19
4	227	200	1.14	2.77
5	270	185	1.46	3.21
6	439	253	1.74	3.58

TABLE VIII: The comparison of *u-s* service cost between APX and the optimal solution.

α	APX	Optimal	Ratio	Bound
0.2	712	678	1.05	7.50
0.3	712	678	1.05	8.28
0.4	712	678	1.05	9.00
0.5	712	678	1.05	10.00
0.6	712	678	1.05	16.50

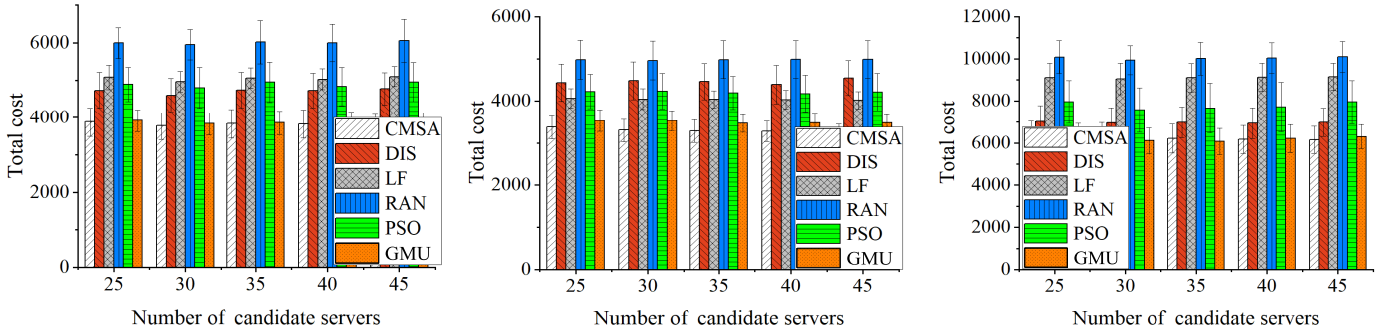
Thirdly, we evaluate the performances of the five algorithms with the change of number of candidate servers in Fig. 8. It is easy to find out that the total cost of *APX* is lowest in all three datasets. The error bars in Fig. 8 measure the value of standard deviation and show that the simulation results are accurate. The total costs of all algorithms decrease along with the increase of number of candidate servers. The reason is that when the number of candidate servers increases, there will be more chance to activate the proper servers to minimize the total cost.

Then, as illustrated in Fig. 9, the performances of all five algorithms are evaluated with the change of the number of data types. It is worth noting that the number of data types here means the total number of data types carried by all mobile users. The simulation results show that the total cost performances of all six algorithms rank as follows: APX<GMU<DIS<PSO<LF<RAN. Since APX, GMU and PSO consider minimizing both facility cost and service cost, they perform better than other algorithms. The total costs of all algorithms increase with the growth of the number of



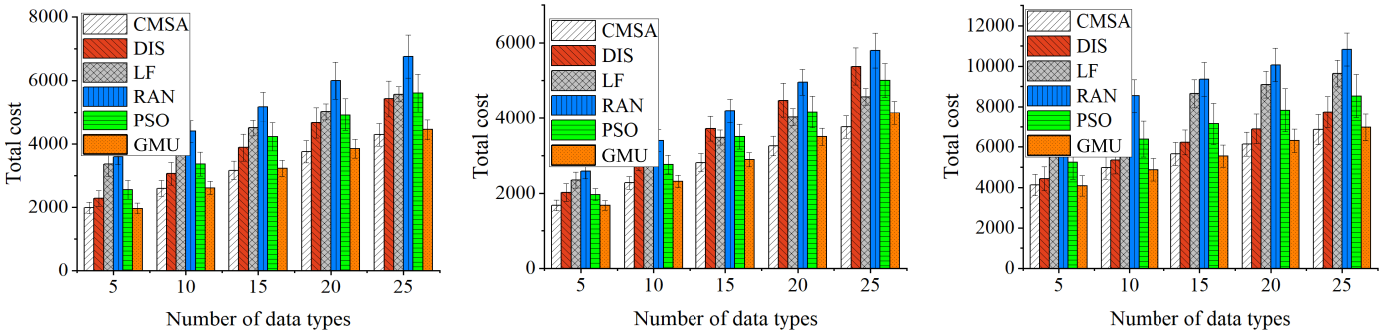
(a) roma/taxi set (b) epfl/mobility set (c) geolife trajectory set

Fig. 11: The simulation results in terms of number of users.



(a) roma/taxi set (b) epfl/mobility set (c) geolife trajectory set

Fig. 12: The simulation results in terms of number of candidate servers.



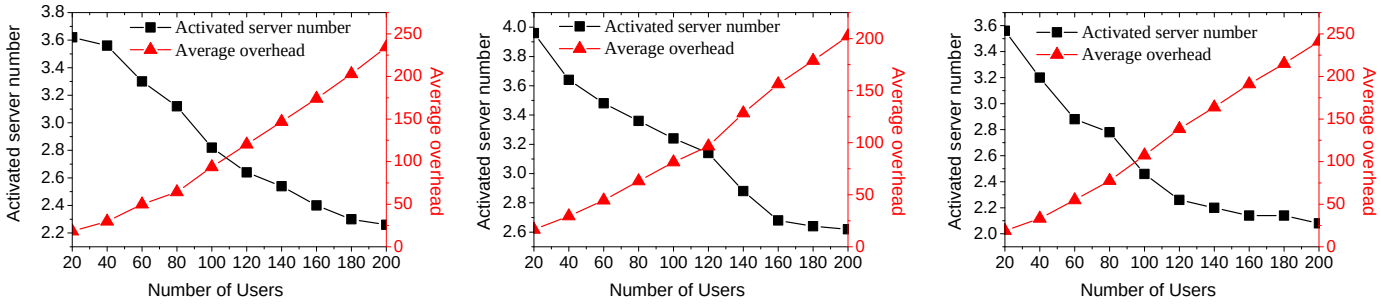
(a) roma/taxi set (b) epfl/mobility set (c) geolife trajectory set

Fig. 13: The simulation results in terms of number of data types.

data types. The reason is that when the number of data types increases, each user may carry more data types and they may travel to more mobile edge servers to upload data.

Furthermore, as shown in Fig. 10, we give an example of the simulation result of APX in roma/taxi set, where the number of candidate mobile edge servers is 10, the number of mobile users is 10 and the total number of data types carried by all mobile users is 5.

Finally, we conduct some simulations to compare the results of the proposed approximation algorithm with the optimal results in roma/taxi set. Specifically, we compare the facility cost and *u-s* service cost between APX algorithm and the optimal results. Besides, we calculate the ratio between the cost of APX and the optimal solution. We set $\alpha = 0.6$, the number of candidate servers is 15 and the number of mobile users is 50. The detailed results are shown in Table VII-VIII. In Table VII, we compare the facility cost of APX and the optimal in terms of the change of number of data types. Due to the fact that the value of data type number changes little, the value of ratio fluctuates slightly. But the ratio is always less than the bound. In Table VIII, we compare the *u-s* service cost between APX and the optimal results. Since many of the fractional solutions obtained in roma/taxi set are close to 1, the filtering technique will filter few fractional solutions and



(a) roma/taxi set (b) epfl/mobility set (c) geolife trajectory set

Fig. 14: The trend of activated server number and average overhead in terms of number of users.

TABLE IX: The promotion ratio in terms of number of users.

Number of users	Ratio	Number of users	Ratio
30	7.6%	60	5.4%
40	7.0%	70	4.0%
50	5.4%	80	4.5%

TABLE X: The promotion ratio in terms of number of types.

Number of types	Ratio	Number of types	Ratio
14	6.1%	17	5.8%
15	6.3%	18	6.2%
16	6.2%	19	5.4%

the impact of the value of α to the cost is limited. The cost of APX is the same when changing α. However, the ratio is also less than the bound. The experimentation results show that the proposed algorithm performs much better than what its worst-case bound suggests.

C. Simulation Results for Multi-Type Scenario

To evaluate the performance of the proposed facility location strategy for the situation where each user can carry multiple types of data without limitations. Firstly, we compare the total costs of the proposed CMSA algorithm with DIS, LF, RAN, PSO and GMU algorithms with the change of number of users, number of candidate servers and number of data types. Then, we investigate the trend of activated server number and average overhead with the change of the number of users. Finally, we evaluate the promotion ratio of the final solution of CMSA against the initial solution. In the simulation, the CMSA parameters are set as follows: The input of algorithm 2, $(T, T_{max}, \gamma, \delta)$, is set as (10000, 5000, 0.99, 1000).

Firstly, we compare the total cost of the CMSA with DIS, LF, RAN, PSO and GMU algorithms with the growth of number of users. The number of users varies from 50 to 450. As shown in Fig. 11, it is obvious that GMU and CMSA perform better than other baseline algorithms. More specifically, CMSA achieves a lower cost than GMU when the user number is small and GMU outperforms CMSA with the growth of number of users.

Secondly, we evaluate the performances of algorithms with the change of number of candidate servers on three datasets. The simulation results are shown in Fig. 12. Since the user may go to a nearer edge server when the number of servers increases, the total cost decreases slightly with the growth of the number of candidate servers. It is easy to find out that CMSA and GMU perform better than other baseline algorithms. Although GMU achieves a lower cost than CMSA on geolife trajectory set when the server number is low, CMSA performs better than GMU on other two datasets.

Thirdly, as shown in Fig. 13, we compare the total cost with the change of number of data types. The number of data types varies from 5 to 25. It is worth noting that the number of data types here refers to the total number of data types carried by all mobile users. GMU performs better than CMSA in the beginning on geolife trajectory set, but it achieves a higher cost than CMSA with the growth of the number of data types. Moreover, CMSA has the best performance on other two datasets.

Then, we investigate the trend of activated server number and average overhead with the change of number of users. The average overhead here is the average number of users served by each activated server. We set the candidate server number as 40 and the total data type number as 20. We repeat the simulation 100 times and obtain the average results. As shown in Fig. 14, the activated server number decreases and the average overhead increases with the growth of number of users in all three datasets. As the number of users increases, the service cost outnumbers the facility cost. Hence, the proposed strategy tends to guide users to upload data through one server for minimizing users' movement costs. Each activated server will serve more users when the number of activated servers decreases, which matches the theoretical analysis.

Finally, we evaluate the promotion ratio of the solution of CMSA against the initial solution in terms of number of users. We denote the total cost of the initial solution as C_{init} and the total cost of the final solution of CMSA as C_{final}. Therefore, the promotion ratio is computed as $\frac{C_{init}-C_{final}}{C_{init}}$. As shown in Table. IX, the promotion ratio is no less than 4.0%. Moreover, we evaluate the promotion ratio in terms of number of data types. The simulation results are demonstrated in Table. X. It is easy to find out that the promotion ratio fluctuates between 5.4% and 6.1%.

VIII. CONCLUSION

We investigate the edge server location problem for minimizing cost in mobile crowdsensing. Firstly, we investigate the edge-based mobile crowdsensing architecture and formally define the edge server location problem. Then, the problem is formulated as a variant of the multi-commodity facility location problem. For the two-type scenario, a relaxation based approximation algorithm is proposed, which is proved to have a bound to the optimal solution. Furthermore, we extend the problem to a generalized case. The computational complexity and the large solution space make it difficult for the traditional combination optimization technique to work well. Hence, we propose the connected multi-agent simulated annealing algorithm. Finally, the simulation results match the theoretical analysis and prove that the proposed algorithms perform better than other baseline algorithms.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundations of China under Grant No. 61772230, No. 61972450 and No. 62072209, Natural Science Foundations of Jilin Province No. 20190201022JC, National Science Key Lab Fund Project No. 61421010418, Innovation Capacity Building Project of Jilin Province Development and Reform Commission No. 2020C017-2, Changchun Science and Technology Development Project No.18DY005, Key Laboratory of Defense Science and Technology Foundations No. 61421010418, Jilin Province Young Talents Lifting Project No. 3D4196993421, and in part by NSF grants CNS 1824440, CNS 1828363, CNS 1757533, CNS 1629746, CNS 1651947, and CNS 1564128.

REFERENCES

[1] D. Luan, Y. Yang, E. Wang, and J. Wu, "Facility location strategy for minimizing cost in edge-based mobile crowdsensing," in *The 16th International Conference on Mobile Ad-hoc and Smart Systems*, 2019.

[2] R. K. Ganti, F. Ye, and H. Lei, "Mobile crowd sensing: Current state and future challenges," *IEEE Communications Magazine*, vol. 49, no. 11, pp. 32–39, 2011.

[3] E. Wang, Y. Yang, J. Wu, W. Liu, and X. Wang, "An efficient prediction-based user recruitment for mobile crowdsensing," *IEEE Transactions on Mobile Computing*, vol. 17, no. 1, pp. 16–28, 2018.

[4] W. Liu, Y. Yang, E. Wang, and J. Wu, "Dynamic user recruitment with truthful pricing for mobile crowdsensing," in *IEEE INFOCOM 2020*, April 2020.

[5] E. Wang, M. Zhang, X. Cheng, Y. Yang, W. Liu, H. Yu, L. Wang, and J. Zhang, "Deep learning-enabled sparse industrial crowdsensing and prediction," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2020.

[6] J. Wang, L. Wang, Y. Wang, D. Zhang, and L. Kong, "Task allocation in mobile crowd sensing: State-of-the-art and future opportunities," *IEEE Internet of Things Journal.*, vol. 5, no. 5, pp. 3747–3757, 2018.

[7] Y. Yang, W. Liu, E. Wang, and J. Wu, "A prediction-based user selection framework for heterogeneous mobile crowdsensing," *IEEE Transactions on Mobile Computing*, vol. 18, no. 11, pp. 2460–2473, 2019.

[8] Z. Han, J. Liao, Q. Qi, H. Sun, and J. Wang, "Radio environment map construction by kriging algorithm based on mobile crowd sensing," *Wireless Communications and Mobile Computing*, vol. 2019, pp. 1–12, 02 2019.

[9] L. Xiao and D. W. Goldberg, "Toward a mobile crowdsensing system for road surface assessment," *Computers Environment and Urban Systems*, vol. 69, 2018.

[10] K. Banti, M. Louta, and G. Karetsos, "Parkcar: A smart roadside parking application exploiting the mobile crowdsensing paradigm," in *International Conference on Information, Intelligence, Systems and Applications*, 2017, pp. 1–6.

[11] D. Sabella, A. Vaillant, P. Kuure, U. Rauschenbach, and F. Giust, "Mobile-edge computing architecture: The role of mec in the internet of things," *IEEE Consumer Electronics Magazine*, vol. 5, no. 4, pp. 84–91, 2016.

[12] K. Taik Kim, C. Joe-Wong, and M. Chiang, "Coded edge computing," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 2020, pp. 237–246.

[13] J. Zhang, L. Zhou, F. Zhou, B. C. Seet, H. Zhang, Z. Cai, and J. Wei, "Computation-efficient offloading and trajectory scheduling for multi-uav assisted mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 2114–2125, 2020.

[14] X. Li, W. Li, Q. Yang, W. Yan, and A. Y. Zomaya, "Edge computing enabled unmanned module defect detection and diagnosis system for large-scale photovoltaic plants," *IEEE Internet of Things Journal*, 2020.

[15] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2018.

[16] J. Ni, K. Zhang, Y. Yu, X. Lin, and X. S. Shen, "Providing task allocation and secure deduplication for mobile crowdsensing via fog computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 3, pp. 581–594, 2020.

[17] X. Xia, Y. Zhou, J. Li, and R. Yu, "Quality-aware sparse data collection in mec-enhanced mobile crowdsensing systems," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 5, pp. 1051–1062, 2019.

[18] J. Xiong, X. Chen, Q. Yang, L. Chen, and Z. Yao, "A task-oriented user selection incentive mechanism in edge-aided mobile crowdsensing," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 4, pp. 2347–2360, 2020.

[19] L. Pu, X. Chen, G. Mao, Q. Xie, and J. Xu, "Chimera: An energy-efficient and deadline-aware hybrid edge computing framework for vehicular crowdsensing applications," *IEEE Internet of Things Journal*, vol. PP, no. 99, pp. 1–1, 2018.

[20] G. Zhao, H. Xu, Y. Zhao, C. Qiao, and L. Huang, "Offloading dependent tasks in mobile edge computing with service caching," in *IEEE INFOCOM 2020*.

[21] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *IEEE INFOCOM 2016*.

[22] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Transactions on Wireless Communications*, 2017.

[23] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Transactions on Vehicular Technology*, 2017.

[24] W. Gong, X. Huang, G. Huang, B. Zhang, and C. Li, "Data offloading for mobile crowdsensing in opportunistic social networks," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6.

[25] L. Wang, D. Zhang, Z. Yan, H. Xiong, and B. Xie, "effsense: A novel mobile crowd-sensing framework for energy-efficient and cost-effective data uploading," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 12, pp. 1549–1563, 2015.

[26] M. Marjanovic, A. Antonic, and I. P. Zarko, "Edge computing architecture for mobile crowdsensing," *IEEE Access*, vol. PP, no. 99, pp. 1–1, 2018.

[27] L. Ma, X. Liu, Q. Pei, and X. Yong, "Privacy-preserving reputation management for edge computing enhanced mobile crowdsensing," *IEEE Transactions on Services Computing*, vol. PP, no. 99, pp. 1–1, 2018.

[28] L. Liu, L. Wang, and X. Wen, "Joint network selection and traffic allocation in multi-access edge computing-based vehicular crowdsensing," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2020, pp. 1184–1189.

[29] P. Bellavista, M. Cilloni, G. Di Modica, R. Montanari, P. Carlo Maiorano Picone, and M. Solimando, "An edge-based distributed ledger architecture for supporting decentralized incentives in mobile crowdsensing," in *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*, 2020, pp. 781–787.

[30] C. Ying, H. Jin, X. Wang, and Y. Luo, "Chaste: Incentive mechanism in edge-assisted mobile crowdsensing," in *2020 17th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, 2020, pp. 1–9.

[31] L. Liu, X. Wen, L. Wang, Z. Lu, W. Jing, and Y. Chen, "Incentive-aware recruitment of intelligent vehicles for edge-assisted mobile crowdsensing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 10, pp. 12 085–12 097, 2020.

[32] J. Zhang, "Approximating the two-level facility location problem via a quasi-greedy approach," *Mathematical Programming*, vol. 108, no. 1, pp. 159–176, 2006.

[33] D. B. Shmoys, V. Tardos, and K. Aardal, "Approximation algorithms for facility location problems," in *Twenty-ninth Acm Symposium on Theory of Computing*, 1997.

[34] A. M. Nezhad, H. Manzour, and S. Salhi, "Lagrangian relaxation heuristics for the uncapacitated single-source multi-product facility location problem," *International Journal of Production Economics*, vol. 145, no. 2, pp. 713–723, 2013.

[35] Z. Svitkina, "Lower-bounded facility location," *Acm Transactions on Algorithms*, vol. 6, no. 4, pp. 1–16, 2010.

[36] F. A. Chudak and D. B. Shmoys, "Improved approximation algorithms for capacitated facility location problems," in *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms, 17-19 January 1999, Baltimore, Maryland*, 1999.

[37] M. T. Melo, S. Nickel, and F. S. Gama, "Largescale models for dynamic multicommodity capacitated facility location," 2003.

[38] S. S. R. Shariff, N. H. Moin, and M. Omar, "Location allocation modeling for healthcare facility planning in malaysia," *Computers and Industrial Engineering*, vol. 62, no. 4, pp. 1000–1010, 2012.

[39] Z. Friggstad and M. R. Salavatipour, "Minimizing movement in mobile facility location problems." in *IEEE Symposium on Foundations of Computer Science*, 2008.

[40] M. Charikar and S. Guha, "Improved combinatorial algorithms for facility location problems," *SIAM J. Comput.*, vol. 34, no. 4, pp. 803–824, 2005.

[41] R. Fleischer, J. Li, S. Tian, and H. Zhu, "Non-metric multicommodity and multilevel facility location," in *International Conference on Algorithmic Aspects in Information and Management*, 2006.

[42] R. Ravi and A. Sinha, "Multicommodity facility location," in *Fifteenth Acm-siam Symposium on Discrete Algorithms*, 2004.

[43] E. L. Mooney, Y. Almoghathawi, and K. Barker, "Facility location for recovering systems of interdependent networks," *IEEE Systems Journal*, vol. 13, no. 1, pp. 489–499, 2019.

[44] A. A. Abin, "Querying beneficial constraints before clustering using facility location analysis," *IEEE Transactions on Cybernetics*, vol. 48, no. 1, pp. 312–323, 2018.

[45] L. P. Meng, Q. Kang, C. F. Han, and M. C. Zhou, "Determining the optimal location of terror response facilities under the risk of disruption," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–11, 2018.

[46] D. B. Shmoys, E. Tardos, and K. Aardal, "Approximation algorithms for facility location problems," in *Annual ACM Symposium on Theory of Computing*, vol. 3, no. 3, 1997, pp. 265–274.

[47] V. V. Vazirani, *Approximation Algorithms*, 2001.

[48] Y. Liu, B. Guo, C. Chen, H. Du, Z. Yu, D. Zhang, and H. Ma, "Foodnet: Toward an optimized food delivery network based on spatial crowdsourcing," *IEEE Transactions on Mobile Computing*, vol. 18, no. 6, pp. 1288–1301, June 2019.

[49] L. Bracciale, M. Bonola, P. Loreti, G. Bianchi, R. Amici, and A. Rabuffi, "CRAWDAD dataset roma/taxi (v. 2014-07-17)," Downloaded from https://crawdad.org/roma/taxi/20140717, Jul. 2014.

[50] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, "CRAWDAD dataset epfl/mobility (v. 2009-02-24)," Downloaded from https://crawdad.org/epfl/mobility/20090224, Feb. 2009.

[51] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma, "Mining interesting locations and travel sequences from gps trajectories," in *Proceedings of the 18th International Conference on World Wide Web*, ser. WWW '09. New York, NY, USA: ACM, 2009, pp. 791–800.

Dongming Luan received his B.E.degree in Software Engineering and M.E.degree in computer science and technology from Jilin University, Changchun, Jilin, China, in 2017 and 2020. He is currently a Ph.D. candidate in College of Computer Science and Technology in Jilin University, Changchun, Jilin, China. His current research interest is Mobile Crowdsensing and Mobile Computing.

En Wang received his B.E. degree in software engineering from Jilin University, Changchun, in 2011, his M.E. degree in computer science and technology from Jilin University, Changchun, in 2013, and his Ph.D. in computer science and technology from Jilin University, Changchun, in 2016. He is currently a Professor in the Department of Computer Science and Technology at Jilin University, Changchun. He is also a visiting scholar in the Department of Computer and Information Sciences at Temple University in Philadelphia. His current research focuses on the efficient utilization of network resources, scheduling and drop strategy in terms of buffer-management, energy-efficient communication between human-carried devices, and mobile crowdsensing.

Wenbin Liu the corresponding author, received the B.S. degree in physics and the Ph.D. degree in computer science and technology from Jilin University, China, in 2012 and 2020. He was also a joint Ph.D. student in the Wireless Networks and Multimedia Services Department, Telecom SudParis/Institut Mines-Telecom, Evry, France. He is currently a Postdoctoral Researcher in Dingxin Scholar Program with the College of Computer Science and Technology, Jilin University, China. His research interests include Mobile CrowdSensing, Mobile Computing, and Ubiquitous Computing.

Yongjian Yang received his B.E. degree in automatization from Jilin University of Technology, Changchun, Jilin, China in 1983; his M.E. degree in computer communication from Beijing University of Post and Telecommunications, Beijing, China in 1991; and his Ph.D. in software and theory of computer from Jilin University, Changchun, Jilin, China in 2005. He is currently a professor and a PhD supervisor at Jilin University, the Vice Dean of the Software College of Jilin University, Director of Key lab under the Ministry of Information Industry, Standing Director of the Communication Academy, and a member of the Computer Science Academy of Jilin Province. His research interests include: network intelligence management, wireless mobile communication and services, and wireless mobile communication.

Jie Wu is the Associate Vice Provost for International Affairs at Temple University. He also serves as Director of the Center for Networked Computing and Laura H. Carnell professor in the Department of Computer and Information Sciences. Prior to joining Temple University, he was a program director at the National Science Foundation and was a distinguished professor at Florida Atlantic University. His current research interests include mobile computing and wireless networks, routing protocols, cloud and green computing, network trust and security, and social network applications. Dr. Wu regularly publishes in scholarly journals, conference proceedings, and books. He serves on several editorial boards, including IEEE Transactions on Service Computing and the Journal of Parallel and Distributed Computing. Dr. Wu was general cochair/chair for IEEE MASS 2006, IEEE IPDPS 2008, IEEE ICDCS 2013, and ACM MobiHoc 2014, as well as program co-chair for IEEE INFOCOM 2011 and CCF CNCC 2013. He was an IEEE Computer Society Distinguished Visitor, ACM Distinguished Speaker, and chair for the IEEE Technical Committee on Distributed Processing (TCDP). Dr. Wu is a CCF Distinguished Speaker and a Fellow of the IEEE. He is the recipient of the 2011 China Computer Federation (CCF) Overseas Outstanding Achievement Award.