Facility Location Strategy for Minimizing Cost in Edge-Based Mobile Crowdsensing

En Wang[1], Dongming Luan[1], Yongjian Yang[1] and Jie Wu[2], *IEEE Fellow*
[1]Department of Computer Science and Technology, Jilin University, Changchun, China
[2]Department of Computer and Information Sciences, Temple University, Philadelphia, USA.

***Abstract*—Mobile crowdsensing emerges as a powerful sensing paradigm which utilizes a group of users with mobile devices to perform sensing tasks cooperatively. The traditional mobile crowdsensing architecture is centralized and cloud-based, where the users upload the sensing data directly to the central server. Due to the fact that the amount of sensing data is very large, it will bring much burden to upload and process data collected by mobile users on the central server. In this paper, we propose an edge-based mobile crowdsensing architecture, which introduces a new intermediate layer for data storage, processing and aggregation through deploying mobile edge servers between the traditional cloud server and the user layer. Considering the limited budget, a decision-maker has to decide which server is activated to process each data type at minimum cost, where the cost consists of the facility cost for activating server and processing data, and the service cost measured by the distance of mobile users' movement during the process of uploading data. Since a user may have multiple types of data, this problem is formulated as a variant of the uncapacitated multi-commodity facility location problem. Furthermore, an approximation algorithm is proposed to solve it for minimizing cost, which is proved to have a bound to the optimal solution. We conduct extensive simulations based on the widely-used real-world datasets: roma/taxi set, epfl/mobility set and geolife trajectory set. The experiment results show that the proposed approximation algorithm outperforms other baseline algorithms and accords with the theoretical bound results.**

***Index Terms*—Facility Location, Mobile Edge Computing, Mobile Crowdsensing, Minimizing Cost.**

I. INTRODUCTION

In recent years, mobile devices have been widely used in people's daily life, being equipped with a multitude of embedded sensors. The sensors with powerful sensing ability enable mobile users to perform the sensing tasks. In view of this, a new sensing paradigm called Mobile CrowdSensing (MCS) is proposed [1] to leverage mobile users to collect urban sensing data. MCS has been used to solve many problems ranging from building radio environment map [2], road surface assessment [3] and roadside parking management [4] to digital map updating [5].

In traditional MCS, to complete the sensing tasks, mobile users move to the specified task location to collect data. Then, the generated sensor data is uploaded to the cloud server which is responsible for processing the data and providing the MCS service. In this way, the construction of traditional MCS architecture is centralized and cloud-based, which faces some limitations. The huge data volumes in large-scale MCS scenarios bring significant burden on the central server and mobile network. Moreover, all the sensor data is processed and stored on the central cloud server which may increase the risk of the user privacy disclosure. Fortunately, with the rapid development of Internet of Things and the proliferation of 4G/5G networks, mobile edge computing could be used to help solve the problems in centralized and cloud-based architecture.

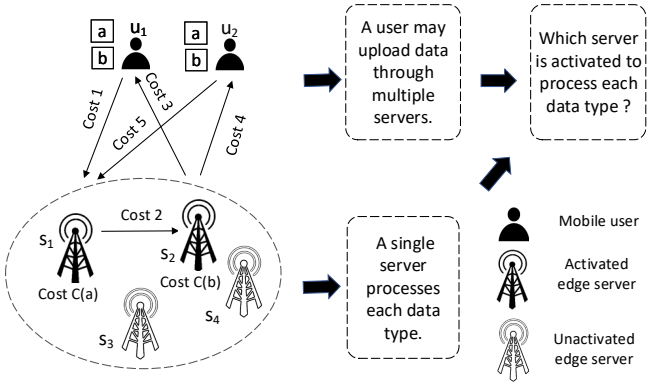


Fig. 1: Problem description in edge-based mobile crowdsensing. For example, user u_1 spends cost 1, cost 2 and cost 3 to upload data and the mobile edge server s_1 costs $C(a)$ to process 'a' type data.

Mobile Edge Computing (MEC) is a new network architecture proposed by the European Telecommunications Standards Institute [6]. It moves the computation tasks that are originally executed on the cloud server to the vicinity of the data source, which improves the performance of the network and reduces the computational load of the cloud server. In view of this, we propose an edge-based MCS architecture by deploying the mobile edge servers between mobile users and the central cloud server. The mobile edge server is responsible for storing, processing and aggregating the data uploaded by mobile users. Since there are multiple types of data carried by mobile users after performing the sensing tasks, the MCS platform will direct the users with the same data type to upload data through the same mobile edge server to reduce the transmission cost and facilitate data aggregation. So the edge-based MCS has the following advantages: it relieves the computational and storage burden of both mobile devices and the cloud server through moving the computation to the network edge. Since the edge servers can process and aggregate data, they can return the sensing result and provide MCS service directly

without contacting cloud servers in some real-time usage scenarios. This decreases the latency of data propagation and service provision.

In the edge-based MCS scenario, after collecting the sensing data, the mobile users will reach the proximity of the corresponding mobile edge servers to upload data. In order to facilitate data aggregation, each type of data must be uploaded on the same mobile edge server. The total cost is twofold: facility cost and service cost. The facility cost includes the mobile edge server activation cost and the cost for processing data. Because mobile users usually spend most time in 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 to corresponding mobile edge servers and return to the initiation. In Fig. 1, the service cost for user u_1 is the sum of cost 1, cost 2 and cost 3. The facility cost for mobile edge server s_1 is $C(a)$, which includes the cost for activating server and processing data. There is a trade-off between facility cost and service cost. If we activate mobile edge servers that are close to mobile users to minimize the service cost without considering facility cost, then the facility cost will definitely increase. Otherwise, if we only consider minimizing the facility cost, the facility cost will decrease but the service cost may increase. Hence, the MCS platform has to decide that for each data type, which server to activate in order to minimize both facility cost and service cost. It is worth noting that one mobile edge server can process multiple types of data and each data type must be served by a single mobile edge server.

To solve the above problem, we formulate this problem as a variant of the uncapacitated multi-commodity facility location problem in this paper. Our solution begins by performing user virtualization and solving LP relaxation of the formulated objective function. Then, we filter the solution and select a group of representative users. Next, we round the fractional solution to an integer solution. Finally, the remaining users are assigned to the edge servers which process their representatives. The above research ideas raise the following challenges: (1) the simplest facility location problem is NP-hard; (2) there are multiple data types and a single mobile edge server must serve each type of data, so this problem is more difficult than the classical facility location problem; (3) one mobile user may carry more than one type of data and it is difficult to find a solution that has a bound to the optimal solution.

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

- We propose an edge-based mobile crowdsensing architecture, where mobile edge servers are deployed at network edge between mobile devices and the central cloud server in the MCS scenario. It can relieve the computational and storage burden of the cloud server and mobile devices.
- The cost minimization problem is formulated as a variant of the uncapacitated multi-commodity facility location problem which is much more complex than the traditional facility location problem. Furthermore, we propose an approximation algorithm to determine the mobile edge servers to activate and the corresponding data types processed by each activated mobile edge server. We prove that the proposed approximation algorithm has a bound to the optimal solution.
- We conduct extensive simulations and the experiment results show that the algorithm proposed in this paper outperforms other baseline algorithms and achieves a bounded cost.

The remainder of the paper is organized as follows. We review the related works in Section II. In Section III, we present the system model and formulate the problem. An approximation algorithm is proposed in Section IV. In Section V, we prove that the approximation algorithm has a bound to the optimal solution. In Section VI, the extensive simulation is conducted to evaluate the performance of the proposed algorithm. We conclude the paper in Section VII.

II. RELATED WORKS

A. Edge-based Mobile Crowdsensing

There have been some research works focusing on MCS based on distributed architecture. Marjanovic $et\ al.$ [7] propose a mobile edge computing architecture for MCS that can reduce the complexity of data processing and increase the quality of MCS service. The authors in [8] propose two privacy preserving reputation management strategies in MCS based on edge computing to preserve privacy and handle malicious participants. In [9], the authors propose an edge-based context-aware crowdsourcing framework to perform instantaneous image sensing in a disaster environment. In [10], the authors 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. The authors in [11] propose an enhanced MCS system which combines the deep learning based data validation and edge computing based data processing.

Different from the research works mentioned above where mobile edge servers receive data from the data source in the close proximity, in edge-based MCS architecture proposed in this paper, each data type is processed by a single mobile edge server for the convenience of data aggregation. The decision-maker has to decide which servers to activate to process data.

B. Facility Location Problem

The facility location problem, also known as location analysis, has attracted many researchers' attention. In [12], the authors apply an evolutionary simulated annealing strategy to solve the large-scale uncapacitated facility location problem. It combines two well-known approaches with the purpose of avoiding local minima. In [13], the authors put forward an improved approximation algorithm to solve the capacitated facility location problem. In [14], the authors study the lower-bounded facility location problem and put forward a true approximation algorithm which respects the lower bound constraints for the problem. In addition to the

classical facility location problem, there are also many variant versions of the facility location problem and some works have appeared to solve them. R.Ravi *et al.* [15] put forward an approximation algorithm to solve the multi-commodity facility location problem with a bound to the optimal solution. The authors in [16] put forward a quasi-greedy algorithm to solve the classical uncapacitated 2-level facility location problem, which can approximate the problem in polynomial time with a ratio of 1.77. In [17], the authors propose an 8-approximation algorithm for minimizing total movement of the mobile facility location problem through rounding an LP relaxation in five phases. In [18], the authors aim to solve the dynamic facility location problem by modeling the problem as a 0-1 quadratic program and propose a heuristic solution which is proved to have a lower bound.

Since a single mobile edge server can process multiple types of data, 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 distance among mobile edge servers and a user can carry multiple types of data, which makes it more difficult.

III. SYSTEM OVERVIEW

A. System Model

We consider an edge-based MCS model that consists of a group of mobile users, signified by the set $U = \{u_1, u_2, \ldots, u_n\}$ and a set of mobile edge servers $S = \{s_1, s_2, \ldots, s_m\}$. Moreover, after collecting the sensing data, mobile users may carry multiple types of data. All the types of data are denoted as $B = \{b_1, b_2, \ldots, b_r\}$ and the data types carried by user u_i are $B_i \subseteq B$. The total cost is twofold: facility cost and service cost. Each mobile edge server s_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 s_i has an activation cost $C(s_i)$ and for each data type b, there is an incremental processing cost $C_i(b)$. Therefore, the facility cost for activating mobile edge server s_i in configuration β is $C_i(\beta) = C(s_i) + \sum_{b\in\beta} C_i(b)$.

The service cost for the mobile user is regarded as the mobile users' travel distance during the process of uploading data. Each user begins with an initial location and heads for the corresponding edge servers one by one and return to the initial location finally. In this paper, we assume that a mobile user carries at most two types of data. This case is very common because a person usually spends most of time in several places everyday (e.g. workplace and home). A user tends to collect a data when going to work (home to workplace), and collect another data on the way home (workplace to home). As shown in Fig. 2, 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

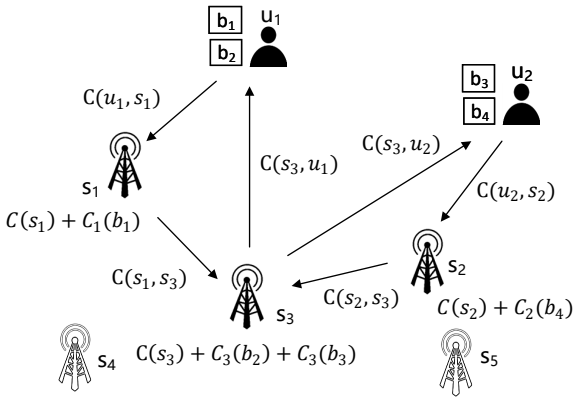


Fig. 2: An example of edge-based MCS system model. Servers s_1 and s_2 are selected to process data type b_1 and b_4 respectively. Server s_3 is selected to process data type b_2 and b_3.

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.

B. Problem Formulation

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 or not mobile edge server i is activated. 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:

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

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 MINIMIZING COST

In this section, we describe the facility location strategy in detail. Firstly, we relax the constraints and due to the reason that each user has multiple types of data, we transform a user into a set of virtual users where each virtual user has one data type. Then, we transform the objective function into a linear version and the fractional solution is obtained by solving LP relaxation. Next, we filter the solution so that each virtual user is assigned to mobile edge servers which 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 and assign the remaining virtual users to the edge servers which 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 u-s service cost instead of the total service cost and perform user virtualization in this step to transform the objective function into a linear function.

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

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 $\{u_j^b : \forall b \in B_j\}$. Specifically, 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 the 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. The objective function has been transformed into a linear function. After performing the linear relaxation, we will get some fractional solutions which are feasible for the relaxation of our objective function. 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 has a bound to the optimal solution in the next section.

B. Filtering Technique

Then, we apply the filtering technique used in [19] 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

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

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 big.

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 [15]. Specifically, we classify the virtual users who carry the same type of data into groups and select a representative from each of them. The process of representatives selection of different data types is independent.

The detailed progress 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 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 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 which serve their representatives and each mobile user will go to the mobile edge servers that are allocated to its virtual users.

V. BOUND PROOF

In this section, we prove that the proposed 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:

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

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 [15]. 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.

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 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 $\log k$ [20], 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 program 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, according to Lemma 2, 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, $c_{j,\varphi(j)} \leq 2c_{j,j_b} + c_{j,\varphi^{'}(j)}$. We denote the virtual user set as 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

TABLE I: 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 II: 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

cost, ignoring the constraint that each data type is processed by a single mobile edge server and it is proved in [15]. The total u-s service cost is equal to the sum of u-s service costs of all virtual users. Hence, the bound in the lemma is proved. □

Finally, we prove the bound of s-s service cost. Let d_{max} denote the maximum distance among mobile edge servers and d_{min} denote the minimum distance among mobile 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. 3 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. I and II. It is worth noting that b_0 denotes the activation costs for servers in Table. II. 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 $\{\frac{\log k}{\alpha}, \frac{3}{1-\alpha} + 4, \frac{d_{max}}{d_{min}}\}$.

VI. PERFORMANCE EVALUATION

A. Data Preparation

Three widely-used real-world traces, roma/taxi set[21], epfl/mobility set [22] and geolife trajectory set [23] were used to evaluate the performance of the proposed facility location strategy for minimizing the 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

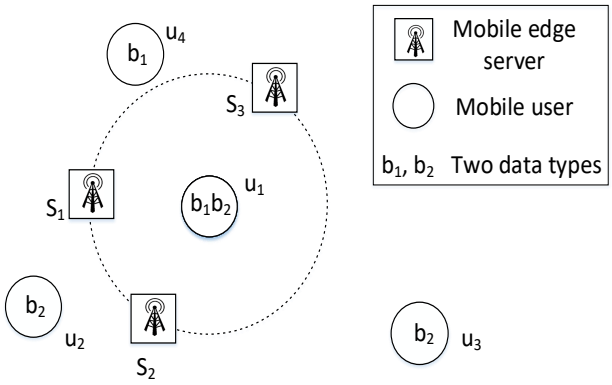


Fig. 3: The scenario for proving the bound of s-s service cost.

contains 17,621 GPS trajectories with a distance of 1.2 million kilometers. The first points of users' trajectories are set as the initial locations and we randomly select POI locations as the candidate mobile edge server locations.

We construct three different strategies: LF, DIS and RAN. The performance of the proposed approximation algorithm APX is 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 facility cost, $C_i(b)$. For each data type, DIS strategy selects the mobile edge server that 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 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.

B. Simulation Results

In this section, firstly, we give an example to illustrate the differences in the execution results of APX, DIS and LF strategies. Secondly, we evaluate the performances of the proposed approximation algorithm compared with the method LF, DIS and RAN along with changes in the number of candidate servers, number of mobile users and number of data types. The specific evaluation results on three datasets are demonstrated in Figs. 5-7. 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.

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. III, when data type is 0, it denotes the activation cost for each server. The configuration information including facility cost configuration, the distance between users and servers as well as the distance between servers are shown in detail in Table. III-V respectively.

Fig. 4 illustrates the scenarios for using three algorithms respectively. When using APX algorithm, server s_1 is activated

TABLE III: Facility cost.

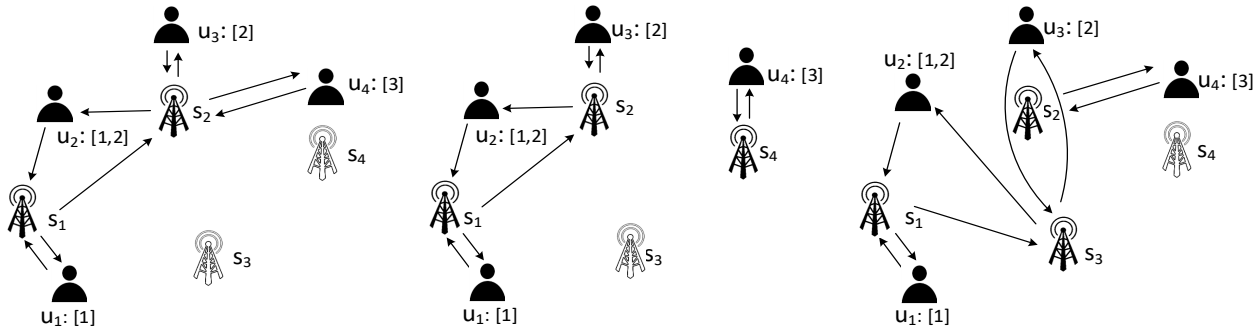
Server \ Type	0	1	2	3
s_1	2	1	3	3
s_2	5	3	2	1
s_3	2	3	1	2
s_4	4	4	2	5

TABLE IV: 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 V: 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. 4: 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.

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. Due to the fact that 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.

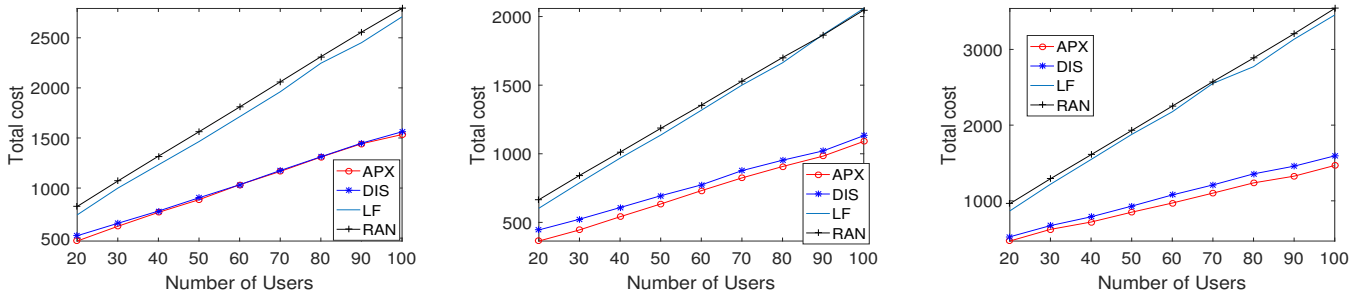
Secondly, we compare the total cost in terms of number of users in three datasets in Fig. 5 and the number of users varies from 20 to 90. The simulation results show that the total cost performances rank as follows:$APX < DIS < LF < RAN$ in all three data sets, though the cost of DIS is close to that of APX in roma/taxi set. The total costs of all four algorithms increase with the growth of the number of users.

Thirdly, we evaluate the performances of the four algorithms with the change of number of candidate servers in Fig. 6. It is easy to find out that the total cost of APX is lowest in all three datasets. The error bars in Fig. 6 measure the value of standard deviation and show that the simulation results are accurate. The total costs of all four 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.

Then, as illustrated in Fig. 7, the performances of four algorithms are evaluated with the change in the number of data types. Note 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 four algorithms in three datasets rank as follows: $APX < DIS < LF < RAN$ and the error bars show that the simulation results are accurate. The total costs of all four algorithms increase with the growth of the number of 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.

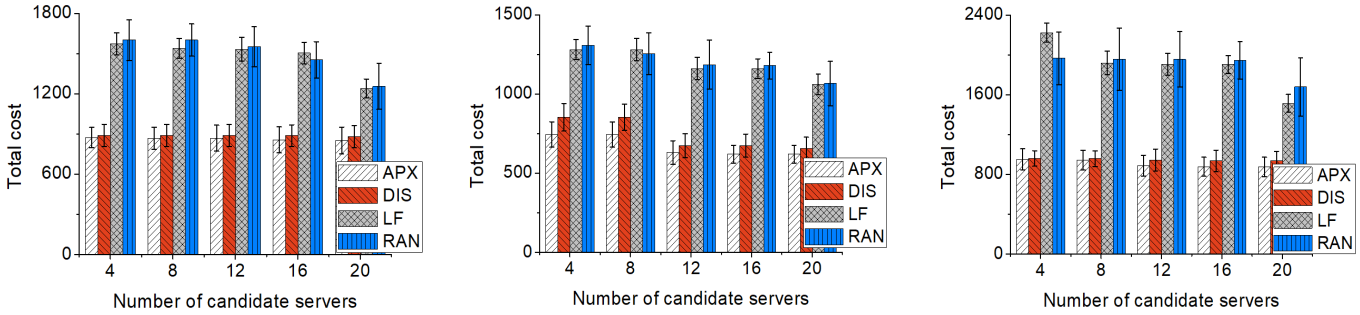
Furthermore, as shown in Fig. 8, we give an example of the simulation result of APX in roma/taxi set, where the number of candidate mobile edge servers is 10, the number of mobile users is 10 and the total number of data types carried by all mobile users is 5.

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



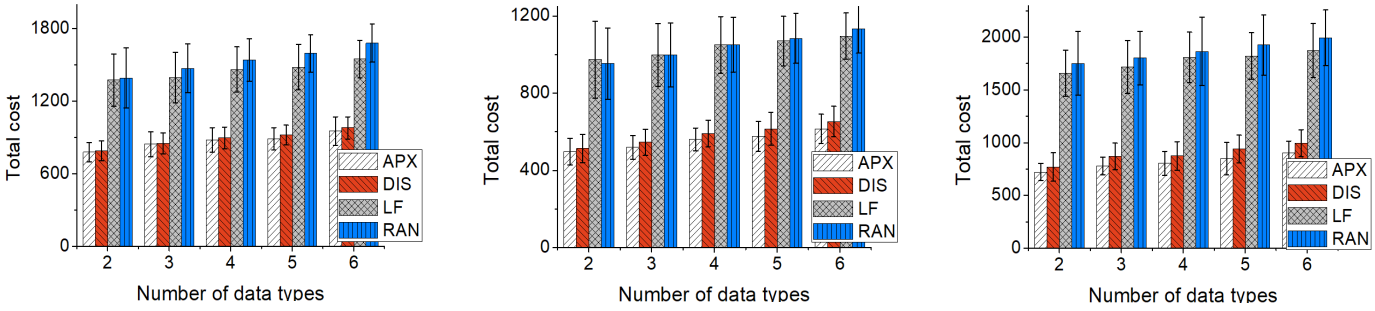
(a) roma/taxi set (b) epfl/mobility set (c) geolife trajectory set

Fig. 5: The simulation results in terms of number of users.



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

Fig. 6: The simulation results in terms of number of candidate servers.



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

Fig. 7: The simulation results in terms of number of data types.

users is 50. The detailed results are shown in Table VI-VII. In Table VI, 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 VII, 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 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. This matches the theoretical analysis.

TABLE VI: 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

VII. CONCLUSION

We study the facility location problem for minimizing cost in edge-based mobile crowdsensing in this paper. Firstly,

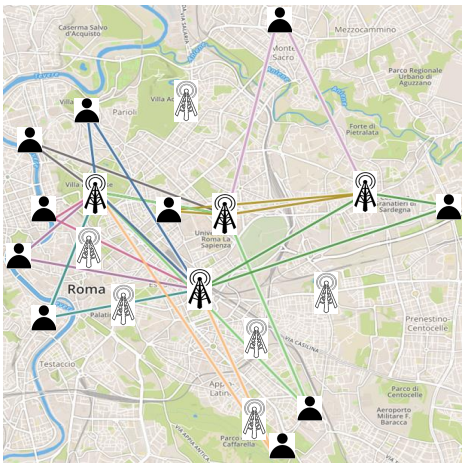


Fig. 8: A presentation of the simulation result of APX in roma/taxi set. The mobile edge servers in black color denote the activated servers and the rest are unactivated servers.

TABLE VII: 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

we propose an edge-based mobile crowdsensing architecture where mobile edge servers are deployed at network edge and each type of data is processed and aggregated by a single mobile edge server. This manner decreases the computation and transmission cost in the network. Then, the problem is formulated as a variant of the uncapacitated multi-commodity facility location problem and an approximation algorithm is proposed to solve it, which is proved to have a bound to the optimal solution. Finally, the extensive simulations are conducted based on widely-used real-world traces: roma/taxi set, epfl/mobility set and geolife trajectory set. The results of the simulation could match theoretical analysis and prove that the proposed approximation algorithm performs better than other baseline algorithms.

VIII. ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China under Grant No.61772230 and National Natural Science Foundation of China for Young Scholars No.61702215, China Postdoctoral Science Foundation No.2017M611322 and No.2018T110247 and this research is also supproted in part by NSF grants CNS 1824440, CNS 1828363, CNS 1757533, CNS 1618398, CNS 1651947, and CNS 1564128.

REFERENCES

[1] 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.

[2] 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.

[3] L. Xiao and D. W. Goldberg, "Toward a mobile crowdsensing system for road surface assessment," *Computers Environment and Urban Systems*, vol. 69, p. S0198971517301333, 2018.

[4] 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.

[5] Z. Peng, S. Gao, B. Xiao, S. Guo, and Y. Yang, "Crowdgis: Updating digital maps via mobile crowdsensing," *IEEE Transactions on Automation Science and Engineering*, vol. PP, no. 99, pp. 1–12, 2017.

[6] 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.

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

[8] 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.

[9] Z. Zhao, F. Liu, Z. Cai, and N. Xiao, "Edge-based content-aware crowdsourcing approach for image sensing in disaster environment," in *Proceedings of the 14th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 2017.

[10] 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.

[11] Z. Zhou, H. Liao, B. Gu, K. M. S. Huq, S. Mumtaz, and J. Rodriguez, "Robust mobile crowd sensing: When deep learning meets edge computing," *IEEE Network*, vol. 32, no. 4, pp. 54–60, July 2018.

[12] V. Yigit, M. E. Aydin, and O. Turkbey, "Solving large-scale uncapacitated facility location problems with evolutionary simulated annealing," *International Journal of Production Research*, vol. 44, no. 22, pp. 4773–4791, 2006.

[13] F. A. Chudak and D. P. Williamson, "Improved approximation algorithms for capacitated facility location problems," in *International Ipco Conference on Integer Programming and Combinatorial Optimization*, 1999, pp. 207–222.

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

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

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

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

[18] P. Chardaire, A. Sutter, and M. C. Costa, "Solving the dynamic facility location problem," *Networks*, vol. 28, no. 2, pp. 117–124, 1996.

[19] 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.

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

[21] 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.

[22] 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.

[23] 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.