2019 20th IEEE International Conference on Mobile Data Management (MDM)

k-Collective Influential Facility Placement over Moving Object

Dan Li†, Hui Li†‡, Meng Wang§, Jiangtao Cui*
†*State Key Laboratory on Integrated Services Networks, Xidian University, Xi'an, China*
Email: i.danli@outlook.com
‡*School of Cyber Engineering, Xidian University, Xi'an, China*
Email: hli@xidian.edu.cn
§*School of Computer Science, Xi'an Polytechnic University, Xi'an, China*
Email: wamengit@sina.com
**School of Computer Science and Technology, Xidian University, Xi'an, China*
Email: cuijt@xidian.edu.cn

***Abstract*—In this paper we propose and study the problem of k-Collective influential facility placement over moving object. Specifically, given a set of candidate locations, a group of moving objects, each of which is associated with a collection of reference points, as well as a budget k, we aim to mine a group of k locations, the combination of whom can influence the most number of moving objects. We show that this problem is NP-hard and present a basic hill-climb algorithm, namely GreedyP. We prove this method with $(1-\frac{1}{e})$ approximation ratio. One core challenge is to identify and reduce the overlap of the influence from different selected locations to maximize the marginal benefits. Therefore, the GreedyP approach may be very costly when the number of moving objects is large. In order to address the problem, we also propose another GreedyPS algorithm based on FM-sketch technique, which maps the moving objects to bitmaps such that the marginal benefit can be easily observed through bit-wise operations. Through this way, we are able to save more than a half running time while preserving the result quality. Experiments on real datasets verify the efficiency and effectiveness for both algorithms we propose in this paper.**

***Keywords*-moving objects, location selection, submodular, approximate algorithm**

I. Introduction

Location Selection (LS) problem has always received great attention due to the value of application in many aspects. Given a set of moving objects Ω, each of which is represented using a set of reference positions, and a set of candidate locations C, many methods have been proposed to detect an optimal $c \in C$, such that c can influence (*i.e.,* affect/cover) the maximum number of moving objects [1]. Finding such an optimal location from candidates to establish a new facility has a wide spectrum of applications such as marketing, urban planning [2], monitoring wildlife [3], scientific research, *etc.* In LS problem, *influence* refers to the number (*i.e.,* probability) of persons (*i.e.,* moving objects) that may visit (*i.e.,* be influenced) if a facility is placed at a particular location.

There exists several different criteria for evaluating the *influence*. For instance, according to BRNN [4], the influence of a candidate c is defined as the number of objects

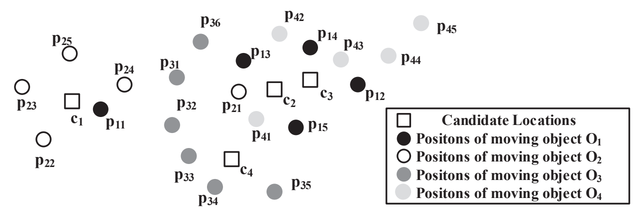


Figure 1. Motivating example.

whose nearest neighbors are c. Recently, Wang *et al.* [5] introduced a generalized LS problem called PRIME-LS which takes into account mobility and probability factors in location selection. The authors employed the cumulative probability to judge whether an object is influenced by a particular location or not. We compare both influence criteria using an example in Fig. 1. On one hand, nearest neighbor based conventional LS techniques [4] will report c_1, but not c_2, influences O_1. On the other hand, the cumulative probability (according to [5]) of O_1 being influenced by c_2 might be higher than c_1 as O_1 has four positions, namely $p_{12}, p_{13}, p_{14}, p_{15}$, which are close to c_2. In light of that, we select to follow the influence model of [5] and focus on the cumulative probability settings in this work.

1) Motivation: The proposed algorithm in [5], namely PINOCCHIO, is substantially efficient in finding only one location. However, if a user asks to set up a group of homogeneous facilities and aims to cover as many people as possible, this method can not be directly and effectively applied. Reconsider Fig. 1, assume that following the accumulative probability influence criteria, c_2, c_3 can both influence O_1, O_3, O_4; and c_1 can influence O_2; and c_4 influences O_4. Suppose we are selecting 2 locations to place some facilities, directly applying PINOCCHIO [5] and select the best two candidates will produce a results set $\{c_2, c_3\}$. However, $\{c_2, c_1\}$ or $\{c_3, c_1\}$ can eventually influence more objects than $\{c_2, c_3\}$. That is, PINOCCHIO is not suitable to such collective LS problem.

Instead of finding the optimal location, this scenario requires to find a group of k locations. The problem of mining a set of k locations from candidates to influence the maximal persons is also widely required in practice, *e.g.,* setting up k billboards, running k new restaurants in a city, opening k stores to sell mobile phones, *etc.*. Mining k locations problem has appeared in literature [6]–[9]. In these works, they extract the positions in a moving object as a representative position, and if the distance between this position and candidate c is lower than a certain value, then it is considered that the candidate c can influence the moving object. For example, if candidate c overlaps the moving object, c can influence this object [8]. What's more, these methods judging whether candidate c can influence a moving object or not for mining k locations problem are tightly coupled with a few specific applications and can't be directly applied to other scenarios. For example, [8] illustrates moving a object must traverse candidate c, thus, candidate c can influence moving object. However, in the case of setting up billboard (*i.e.,* monitoring wildlife), the user (*i.e.,* animals) only need to see the billboards (*i.e.,* cameras) within a range. Therefore, as discussed before, we employ a more general rule of [5] to determine whether candidate c can influence moving object.

Unfortunately, given a set C of n candidate locations and m moving objects, the time complexity of calculating the number of moving objects influenced (following the cumulative probability model of [5]) by each candidate is $O(mn)$. The time complexity of finding all subsets that each subset contains k elements from candidate set C, and calculating the moving objects set influenced by candidate should be $C_n^k O(km)$. Afterwards, we need to select a set from C_n^k subsets which can influence the maximum number of moving objects. This whole process is exponential and the time consumption is unacceptable. Specifically, we define it as the **k-Collective Influential Facility Placement problem** and shall theoretically show that it is NP-Hard. To address the problem, we propose a pair of algorithms, namely GreedyP and GreedyPS, which solve the k-Collective Influential Facility Placement problem under the same cumulative influence probability criteria with [5]. GreedyP is an approximated solution that is guaranteed to provide an $(1-\frac{1}{e})$ approximation ratio for the k-Collective Influential Facility Placement problem. In order to reduce the time consumption of the algorithm, we further propose GreedyPS utilizing FM sketch techniques, and theoretically prove its effectiveness.

2) Contributions: The contributions of this paper can be summarized as follows:

- We introduce a novel location selection task, namely the k-Collective Influential Facility Placement problem, and theoretically prove this problem is NP-Hard.
- We present a greedy algorithm with an $(1-\frac{1}{e})$ approximation ratio.
- We propose another algorithm by employing FM Sketch to further improve the efficiency and provide the corresponding theoretical study.
- Experimental evaluations on real-world datasets show that our methods are effective and efficient.

The rest of the paper is organized as follows. We review the related work in Section II. The formalized problem definition is given in Section III. Afterwards, we present our solutions and conduct theoretical studies in Section IV. The experimental results are demonstrated in Section V. Lastly, we conclude our work in Section VI.

II. Related Work

In this section, we discuss related efforts in location selection as well as the recent maximum coverage problems.

A. Location selection

There have been increasing research efforts in LS problem under various applications [1], [5], [10]–[16]. Most of these studies assume that user's locations are static and only the most influential location is retrieved. Xia *et al.* [1] defined the influence of a location as the total weight of its reverse nearest neighbors (RNNS). Sun *et al.* [10] validated all clients and their corresponding BRNN sets and proposed three pruning techniques to tighten the search space. Yan *et al.* [11] further relaxed the criterion from NN facility to $(1+\alpha)*NN$, where α is a user-specified value. Wong *et al.* [12] studied a similar problem, called MaxBRkNN, in which all kNN facilities exhibit influence on objects. Zhou *et al.* [13] proposed MaxFirst to solve MaxBRkNN. The solution partitioned the space into quadrants iteratively and pruned the unpromising candidates using upper and lower bounds. Recently, Wang *et al.* [5] introduced a generalized LS problem called PRIME-LS, which utilizes mobility and probability factors. In this work, they presented a rule that uses cumulative probability for all positions along the moving object to judge the impact. As this rule is more relevant to real scenarios, we will adopt that rule to judge that whether a candidate location $c \in C$ impacts a moving object.

B. Maximum coverage problems

Maximum coverage problem has great utility for several real-world applications [6]–[9], [17]–[21]. In these methods, every user is modeled as a moving object. Xu *et al.* [17] proposed group locations selection problem to find the minimum number of multiple locations with influence regions, such that all the objects can be coverd. Mitra *et al.* [6] proposed three different applications, namely TOPS, TUMP and TIPS, respectively. TOPS [7] mainly showed a multi-resolution clustering based indexing framework called NETCLUS. It exhibits practical response times and low memory footprints. TUMP focused on providing good quality of experience (QoE), which differs from our problem. TIPS [18] is closely related to the TOPS problem, which aims to minimize the maximum inconvenience, *i.e.,* minimizing the extra distance travelled by a commuting user in order to avail a service at her nearest service location. Li *et al.* [8] aimed to find k locations set, reversed by the the maximum number of unique trajectories, in a given spatial region. In brief, if a candidate c is in the object, it can influence this moving object. Zhang *et al.* [9] proposed and studied the problem of trajectory-driven influential billboard placement, which finds a set of billboards within the budget to influence the largest number of trajectories. As long as the position in a trajectory falls within a certain radius of the candidate, it is believed that candidate can influence the trajectory. In these works, they extract the positions in the trajectory as a representative position, and if the distance between this position and candidate c is lower than a certain value, then it is considered that the candidate c influence the trajectory. Guo *et al.* [20] and Zhang *et al.* [21] illustrated that given candidate set (bus trajectories) and trajectories with longitude, latitude, timestamp and interest. In these scenes, k bus trajectories carrying advertisement to influence maximum users are returned, which are different from our work.

Reconsider Fig. 1, we illustrate the different result if the influence criteria varies. For instance, candidate c_1, c_2 can't influence any of the objects according to [8]. The rules for determining the impact in [7], [9] are similar, and c_1, c_2 can affect both O_1 and O_2. Comparing with these works, the cumulative probability proposed in [5], c_1 only influences O_1, c_2 influences O_1 and O_2. In this paper, we employ the influence setting of [5] and present a pair of algorithms to find k locations in candidate set which can affects the largest number of moving objects.

III. Preliminary and Problem Definition

In this section, we begin by introducing some terminology that is necessary for the definition of the problem as well as the influence criteria that decides whether a candidate affects a moving object (user).

A. Preliminary

A location p is a point in a two-dimensional Euclidean space, denoted by latitude and longitude. Given two locations p_1 and p_2, the distance between them is denoted by $dist(p_1, p_2)$. In this paper, we use a set of discrete positions $O = \{p_1, p_2, ..., p_r\}$ to represent a moving object. We denote candidate locations for new facilities to deploy as $C = \{c_1, c_2, ..., c_n\}$. The probability that an object at location p is influenced by a facility $c \in C$ is denoted by $Pr_c(p)$. As we are studying a general problem that may also be used in domains including all types of facility placement applications, where distance is the common factor among all these domains, we select to focus on distance here although other factors may also play a role in specific scenarios (*e.g.,* content of an advertising balloon, altitude of a relay station, *etc.*). Therefore, $Pr_c(p)$ can be computed as $Pr_c(p) = PF(dist(c, p))$. Hereby, $PF(\cdot)$ is a kernel function that monotonically decreases. As a result, the influence probability only depends on the distance. O is influenced by c if and only if there is at least a position p_i of O influenced by c. The probability that O is influenced by c, namely cumulative probability, can be defined as follows.

Definition 1: Given candidate location c and a moving object O with r positions $\{p_1, p_2, ..., p_r\}$, the **cumulative influence probability** of O being influenced by c, denoted by $Pr_c(O)$, is defined as: $Pr_c(O) = 1 - \prod_{i=1}^{r}(1 - Pr_c(p_i))$ [5].

Definition 2: Given a moving object O, a candidate location c and a probability threshold τ, c can influence O if and only if $Pr_c(O) \geq \tau$. Further, given a set of mobile object Ω, the influence value of c, denoted as $inf(c)$, is the number of mobile objects in Ω that are influenced by c [5].

$Pr_c(O)$ measures the extent to which O is influenced by c. Given a set of objects $\Omega = \{O_1, O_2, ..., O_m\}$ and a user-specified probability threshold τ, we can evaluate $inf(c_j)(c_j \in C)$ for every candidate location.

Example 1: (See Fig. 1) We only use two moving objects O_1, O_2 and candidate c_1, c_2 as examples. Assume the independent influence probabilities of c_1 at positions p_{11}, p_{12}, p_{13}, p_{14} and p_{15} are 0.5, 0.1, 0.2, 0.15 and 0.12, respectively. Then $Pr_{c_1}(O_1) = 1 - (1 - 0.5)(1 - 0.1)(1 - 0.2)(1 - 0.15)(1 - 0.12) = 0.73$. Similarly, since the probabilities of c_1 influencing positions $p_{21}, p_{22}, p_{23}, p_{24}$ and p_{25} are 0.25, 0.35, 0.33, 0.3 and 0.38, respectively. $Pr_{c_1}(O_2) = 0.86$. If τ is set to 0.75, c_1 only influences O_2 but not O_1, although O_1 even has the NN position p_{11}. Hence, $inf(c_1) = 1$. On the other hand, if $Pr_{c_2}(O_1)$=0.8 and $Pr_{c_2}(O_2)$=0.79, then c_2 obviously influences both O_1 and O_2. That is, $inf(c_2) = 2$.

B. Problem Definition

We are now ready to define the k-Collective Influential Facility Placement problem to be addressed in this paper.

Firstly, we extend Definition 2 in order to evaluate the number of objects influenced by a set of candidates.

Definition 3: Given a candidate set S, $S = \{c_1, c_2, ..., c_k\}$. $\sigma(S) = |\{O | Pr_{c_i}(O) \geq \tau, c_i \in S, O \in \Omega\}|$. $\sigma(S)$ denotes the total number of moving objects that are influenced by candidate set S.

Then, we are ready to formally present the definition of our problem.

Definition 4: Given a set of candidate locations $C = \{c_1, c_2, ...c_n\}$, a set of moving objects $\Omega = \{O_1, O_2, ...O_m\}$ where $O_i = \{p_1, p_2, ..., p_r\}$, the budget number of new facilities k ($k \leq n$). The **k-Collective Influential Facility Placement problem** aims to mine $\exists S \subseteq C$ ($|S| = k$) to maximize $\sigma(S)$.

Example 2: Consider Table I as an example, which lists the information that every location c can influence a set of

Table I
THE OBJECTS INFLUENCED BY CANDIDATES

Candidate	The objects influenced by c_i
c_1	O_2, O_3
c_2	O_1, O_2, O_4
c_3	O_4

objects[1]. Assume we need to find two locations from C, *i.e.,* $k = 2$. According to the table, O_2 and O_3 are influenced by c_1. O_1, O_2 and O_4 are influenced by c_2. Therefore, when k is set as 2, $S = \{c_1, c_2\}$ is the best choice as it can influence all the objects of O_1, O_2, O_3 and O_4.

IV. SOLUTIONS TO k-COLLECTIVE INFLUENTIAL FACILITY PLACEMENT PROBLEM

Intuitively, a brute-force approach to address the problem in Definition 4 can be described as follows. Find all subsets containing k elements from C, compute the number of objects influenced by every subset, *i.e.,* $\sigma(\cdot)$, and finally return the subset with the maximum $\sigma(\cdot)$. However, the time complexity of this process is obviously exponential. As we shall prove immediately, the problem in Definition 4 is NP-Hard. Therefore, one practical way to address the problem is to find an approximation algorithm that runs in polynomial time. To this end, we firstly propose a basic greedy algorithm to address this problem. In order to further reduce the running time, we also provide an more efficient solution utilizing FM Sketch technique [22], [23].

Before presenting our basic solution, we firstly provide a theoretical study showing that the problem in Definition 4 is NP-Hard. It is not hard to prove that our target, *i.e.,* k-Collective Influential Facility Placement problem with respect to $\sigma(\cdot)$, is equivalent to the well-known Max k-cover problem.

Definition 5: $R = \{a_1, a_2, ..., a_n\}$, R_i represents a subset of R, $P(R)$ is the collection of R_i, $P(R) = \{R_1, R_2, ..., R_l\}$. **Max k-cover** is the problem of selecting k subsets from $P(R)$ such that their union set contains as many points as possible [24].

Theorem 1: The k-Collective Influential Facility Placement problem in Definition 4 is NP-hard.

Proof: Given $\Omega = \{O_1, O_2, ..., O_m\}$, let $Tr(c_i)$ denote the user sets influenced by c_i and $Tr(c_i) \subseteq \Omega$, and let $Q = \{Tr(c_1), Tr(c_2), ..., Tr(c_l)\}$. Selecting a group of k locations from C to affect the most objects is equivalent to extracting k subsets from Q to influence the maximum number of elements from Ω. Thus, k-Collective Influential Facility Placement problem in Definition 4 is the same as the Max k-cover problem in Definition 5. As mentioned in [24], the Max k-cover problem has been proven to be NP-hard. Therefore, the problem in Definition 4 is NP-hard. ■

[1]In the following of this paper, we shall use user and object interchangeably for ease of presentation.

Algorithm 1 GreedyP Algorithm.

Input: The set of candidates C; The set of Object; The number of new facilities k;
Output: The set of selected locations S which k elements;
1: Calculate the object set influenced by every candidate, $Tr(C)$;
2: $S = \varnothing$;
3: **for** $i = 1$ to k **do**
4: find $s_i \in C - S$, where the value of $inf(s_i)$ is maximum;
5: $S = S \cup s_i$
6: each $s_j \in C - S$
7: delete $Tr(s_j) \cap Tr(S)$ from $Tr(C)$;
8: **end for**
9: **return** S;

A. GreedyP Algorithm

1) Algorithm design: As the target problem is NP-hard, we shall seek for an approximated solution that can address the task in polynomial time. Intuitively, a popular and easy way to address Max k-cover is greedy approach. Inspired by that, we design a basic greedy solution, namely GreedyP (short for Greedy PRIME-LS) towards the k-Collective Influential Facility Placement problem. The procedure of GreedyP algorithm is outlined in Algorithm 1. The algorithm begins by computing the sets $Tr(C) = \{Tr(c_i)|c_i \in C\}$ via PINOCCHIO Algorithm [5](Line 1). Besides, we initialize the target set S as an empty set (Line 2). Afterwards, we perform k iterations to select the locations one after another. In each iteration, it selects the site s_i which can influence the maximum number of objects. If a site s_i is selected, we immediately delete the object influenced by s_i from $Tr(C)$ (Lines 3-8). Finally, after k iterations, it returns the target set S (Line 9).

Example 3: Reconsider Table I as an example, assuming $k = 2$. In the first iteration, candidate c_2 is selected as it can influence the most number of moving objects, *i.e.,* O_1, O_2 and O_4. Thus, the value of $inf(c_2)$ is the maximum. We merge c_2 into set S. Then, we delete the moving objects influenced by c_2. Afterwards, we perform the second iteration. In this round, c_1 can influence O_3, and c_3 influences no object. Therefore, we shall select c_1 into set S. Thus, c_2 and c_1 can influence four moving objects in total.

2) Theoretical study: In this part, we shall theoretically prove that the results quality of our GreedyP algorithm is guaranteed. To prove that, we shall firstly introduce a group of definitions and lemmas.

Definition 6: Consider an arbitrary function $\sigma(\cdot)$ that maps subsets of a finite ground set U to non-negative real numbers. We say that σ is **submodular** if it satisfies a natural "diminishing returns" property: the marginal gain from adding an element to a set is at least as high as the marginal gain from adding the same element to superset. Formally, a submodular function satisfies $\sigma(A \cup \{v\}) - \sigma(A) \geq \sigma(B \cup \{v\}) - \sigma(B)$, for all elements v and all pairs of sets $A \subseteq B \subseteq U$.

Lemma 1: For a non-negative, monotone submodular function σ, let S be a set of size k obtained by selecting elements one at a time, each time choosing an element that provides the largest marginal increase in the function value. Let S be a set that maximizes the value of σ over all k-element sets. Then $\sigma(S) \geq (1 - \frac{1}{e}) \cdot \sigma(S^*)$, where S^* is the optimal solution; in other words, S provides an $(1 - \frac{1}{e})$-approximation ratio. [25]

Afterwards, we shall show that the evaluated function $\sigma(\cdot)$ in our problem definition is also submodular in Lemma 2. Combined with Lemma 1, we can further illustrate the approximation rate guarantee of the greedy algorithm.

Lemma 2: The function $\sigma(\cdot)$ defined in Definition 3 is non-negative, monotone, and submodular.

Proof: The proof of this Lemma is in Appendix. ■

Theorem 2: GreedyP algorithm as shown in Algorithm 1 can achieve $(1 - \frac{1}{e})$ approximation ratio.

Proof: It can be directly proved according to Lemma 1 and Lemma 2. ■

Theorem 3: The time complexity of Algorithm 1 is $O(n'mr') + O(knm^2)$, where k is the budget number of locations required, m is the number of moving objects and n is the number of candidates.

Proof: The time complexity of calculating object set for every candidate is $O(n'mr')$, where n' is the number of candidates to be validate after apply pruning rules , r' is the number of positions that has to be used for influence computation after applying Strategy 2 , m is the number of moving objects. In our work, we only use the pruning rules and Strategy 2 [5]. The time complexity of deleting a process that already affects the object of S is $O(knm^2)$ in the worst case. This process takes a lot of time and reduces the efficiency of the algorithm. Thus, the total time complexity is $O(knm^2) + O(n'mr')$. ■

Notably, the time consumption in Algorithm 1 is mainly affected by two aspects. On the one hand, it takes a lot of time to calculate a set of each candidate that affects moving objects in Line 1. We utilize an efficient algorithm called PINOCCHIO that leverages on two pruning rules based on a distance measure. These rules enable us to prune many inferior candidate locations prior to influence computation, paving the way to efficient and accurate solution. On the other hand, we need to delete the moving objects influenced by S in Line 7. Although PINOCCHIO provides a good solution for the first aspect, the running time of the algorithm maybe costly in large dataset as the second aspect also takes much time. In order to address this problem, we further present a more efficient solution in next part.

B. GreedyPS Algorithm

In order to reduce the time cost of the second aspect aforementioned, *i.e.,* recognizing the moving objects that shall be deleted after current iteration (Line 7 in Algorithm 1), we propose to utilize FM sketch strategy. FM algorithm proposed by Flajolet and Martin [23] is a bitmap based algorithm that can efficiently estimate the number of distinct elements (data points). Let F be a bitmap of length L with subindexed $[0, L-1]$, and all bits are initialized as 0 (*i.e.,* $F[j] = 0$ for $0 \leq j \leq L-1$). Suppose the $h(\cdot)$ is a randomly generated hash function which maps the identification of each object into an integer in $[0, L-1]$. An *FM sketch* on $P = \{O_1, O_2, ..., O_l\}$, denoted as $F^{(P)}$, is a bitmap with length L which is defined as:

$$F^{(P)} : \forall 0 \leq j \leq L-1, \quad F^{(P)}[j] = 1$$
$$iff \quad \exists O_i \in P, h(O_i) = j, 1 \leq i \leq l.$$

As $h(\cdot)$ is a randomly generated hash function, a single FM sketch may not accurately accomplish the task. In order to improve the accuracy of FM algorithm, multiple copies (say w) of FM sketches are constructed based independently generated hash functions. Let $f(P)$ represent the set of w FM sketches generated over P. That is, $f(P) = \{F_1^{(P)}, F_2^{(P)}, ..., F_w^{(P)}\}$, where each element $O_i \in P$ is hashed into these FM sketches, respectively, as described above.

Suppose f is applied over two sets of objects, *e.g.,* $f(P)$ and $f(Q)$, generated by the same L and the same set of hashing functions. We define the bit-union of both sets in terms of the bitwise-or operator (denotes by $\vee$) as follows.

Definition 7: Let $f(P) = \{F_i^{(P)} : 1 \leq i \leq w\}$, $f(Q) = \{F_i^{(Q)} : 1 \leq i \leq w\}$, we define the **bit-union** operation of $f(P)$ and $f(Q)$, denoted using $f(P) \oplus f(Q)$, as $\{F_i^{(P)} \vee F_i^{(Q)} : 1 \leq i \leq w\}$, where each $F_i^{(P)} \vee F_i^{(Q)}$ is also a bitmap with subindexes $[0, L-1]$, such that for $1 \leq i \leq w$: $\forall 0 \leq j \leq L-1$, $(F_i^{(P)} \vee F_i^{(Q)})[j] = 1 \quad iff \quad F_i^{(P)}[j] = 1 \quad or \quad F_i^{(Q)}[j] = 1$.

An important feature of FM sketch is that the sketch for the union of a pair of arbitrary sets P and Q can be expressed as the bit-union operation between their corresponding sketches, which can be easily interpreted using bitwise-or operation in bitmaps. Given a set of w hash functions and two collections, P and Q, we have $f(P \cup Q) = f(P) \oplus f(Q)$. This can be easily justified based on Definition 7.

The FM sketch can be used to speed up the update stage of GreedyP Algorithm. The marginal utility of P and Q can be denoted as $\Delta = f(P) \oplus f(Q) - f(P)$, where "$-$" is the bitwise-minus operation for each bitmap F. The GreedyP algorithm shown in Algorithm 1 can be changed as follows. The procedure of deciding whether a candidate location influences the largest number of users can now be easily interpreted as finding the bitmap which has the largest count of "1". Algorithm 2, namely GreedyPS (short for Greedy

PRIME-LS with Sketches), details the modified algorithm using FM Sketch. The algorithm begins by computing the sets $Tr(C)$ and converts moving object information to 1 or 0 in bitmap (Line 1). Then, similar to Algorithm 1, it initializes the target set S as an empty set (Line 2). In each of k iterations, it selects the site s_i that can influence the maximum number of moving objects, and deletes the moving objects influenced by s_i immediately. During this process, we update the corresponding bitmaps using bitwise-or operation in order to delete the moving objects influenced by S. Finally, we return the target set S (Line 9).

Example 4: Reconsider Table I as an example, and suppose we adopt only one FM sketch, *i.e.,* a bitmap for each location candidate. Without loss of generality, we design a simple bitmap with 4 bits, each of which corresponds to a moving object. As c_1 can influence O_2, O_3, the corresponding bitmap is 0110. As c_2 can influence O_1, O_2, O_4, its bitmap is 1011. c_3 can influence O_4, its bitmap is 1000. In the first iteration, we put c_2 into S as it has the most "1". The current bitmap becomes 1011, and the bitmaps of c_2 and c_3 are accordingly changed to 0100 (1011 $\vee$ 0110 -1011) and 0000 (1011 $\vee$ 1000 -1011), respectively. In the second iteration we can choose c_1 directly without recomputing the influenced moving objects. Finally, we return $S = \{c_1, c_2\}$.

Only using one bitmap in algorithm, the resulting set is not ideal. The reason is that multiple moving objects are mapped to the same bit in bitmap during the execution of the algorithm, which causes a large deviation. Therefore, we map moving objects to multiple bitmaps using different hash functions to reduce the bias. Later, we show that increasing the number of bitmaps can improve accuracy.

Theorem 4: Given two sets of moving objects A and B, where $|A| \geq |B|$ and let $\phi^{(w)}(\cdot)$ denote the number of "1" after $\cdot$ mapped into w bitmaps, then the following holds:

$$\forall w > 1, Pr[\phi^{(w)}(A) \geq \phi^{(w)}(B)] > Pr[\phi^{(1)}(A) \geq \phi^{(1)}(B)].$$

Proof: When $w = 1$, the probability that $\phi^{(1)}(A)$ is larger than that $\phi^{(1)}(B)$ can be recorded as $Pr[\phi^{(1)}(A) \geq \phi^{(1)}(B)] = \rho$.

The probability that $\phi^{(w)}(A)$ is larger than that $\phi^{(w)}(B)$ is at least $1-(1-\rho)^w$, denoted as $Pr[\phi^{(w)}(A) \geq \phi^{(w)}(B)] \geq 1-(1-\rho)^w$.

$Pr[\phi^{(w)}(A) \geq \phi^{(w)}(B)] - Pr[\phi^{(1)}(A) \geq \phi^{(1)}(B)] = 1-(1-\rho)^w - \rho \geq 0$.

Specifically, if and only if $w = 1$, $1-(1-\rho)^w - \rho = 0$. That is, $\forall w > 1$, $Pr[\phi^{(w)}(A) \geq \phi^{(w)}(B)] > Pr[\phi^{(1)}(A) \geq \phi^{(1)}(B)]$. ■

Remark 1: $Pr[\phi^{(w)}(A) \geq \phi^{(w)}(B)]$ is monotonically increasing. When w is close to infinity, the value is close to 1. In that case, $Pr[\phi^{(w)}(A) \geq \phi^{(w)}(B)] - Pr[\phi^{(1)}(A) \geq \phi^{(1)}(B)]$ is close to $1-\rho$.

Based on the above theorem, we can also observe that the results quality of GreedyPS algorithm is similar to GreedyP when enough number of bitmaps are adopted.

Algorithm 2 GreedyPS Algorithm.

Input: The set of candidates C; The set of Object; The number of new facilities k;
Output: The set of locations S with k elements;

1: Calculate the object set influenced by every candidate, $Tr(C)$, and compute FM sketch sets for each candidate, $f(Tr(c_i)), c_i \in C$;
2: $S = \varnothing$, $f(current) = \varnothing$;
3: **for** $i = 1$ to k **do**
4: find s_i, where the count of '1' in bitmaps is the maximum;
5: $S = S \cup s_i, f(current) = f(current) \oplus f(Tr(s_i))$;
6: each $s_j \in C - S$
7: $f(Tr(s_j)) = f(current) \oplus f(Tr(s_j)) - f(current)$;
8: **end for**
9: **return** S;

Theorem 5: The time complexity of Algorithm 2 is $O(n'mr') + O(nmw) + O(knw)$, where m is the number of moving objects, n is the number of candidates and w is the number of bitmaps.

Proof: The time complexity of calculating object set for every candidate is $O(n'mr')$, as mentioned in [5]. The time complexity of mapping moving objects into w bitmaps is $O(nmw)$. The time complexity of updating bitmaps by utilizing bitwise-or is $O(knw)$. Therefore, the complexity of the algorithm is $O(n'mr') + O(nmw) + O(knw)$. ■

According to the time complexity analysis, when the number of bitmaps is very large, the efficiency brought by the bitwise-or operation is reduced. Therefore, there exist a tradeoff between the efficiency and accuracy in the algorithm. In light of that, the algorithm can be improved in the following way. The upper bound of the marginal utility for any location s_j is its own utility. Thus, if the current best marginal utility of another location s_i is already greater than that, it is not required to do the union operation with s_j. If the locations are sorted according to their marginal utility in descending order, the scan can stop as soon as the first such site s_j is encountered.

In our implementation, the FM sketches are stored 32 bits. This allows handling of roughly 2^{32} number of moving objects. The length 32 is chosen since the bitwise-or operation of two bitmaps is extremely fast in modern operating systems.

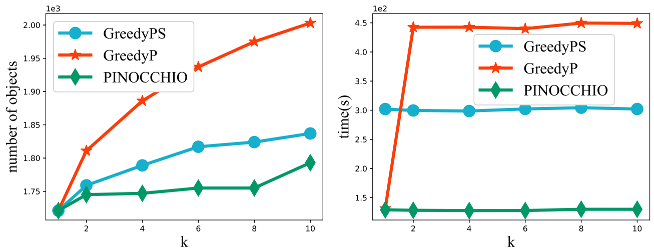
V. Experiment

A. Experiment Setup

1) Datasets: Table II describes the two real-world datasets we use in the experiments. We adopt check-in data here for two reasons: the effectiveness can be compared with check-in ground-truth, which is actual number of visitors for each place of interest; the probability models of check-in

Table II
DESCRIPTION OF REAL-WORLD DATASETS.

	Foursquare	*Gowalla*
number of users (objects)	2,321	10,162
number of check-ins (positions)	167,231	381,165



(a) (b)

Figure 2. Effect of k (*Foursquare*, 20 bitmaps)

with respect to distance have been justified. The position of check-ins in *Foursquare* are all located in Singapore, while those in *Gowalla* are mainly in California.

2) Experimental settings: GreedyP and GreedyPS algorithms are both tested in the experiments. They are implemented in C++, runing on a 3.3 GHz machine with 8 GB RAM under Windows 7 (64 bit).

In line with the settings in paper [5], the default values of probability threshold τ in *Foursquare* and *Gowalla* are set as 0.99 and 0.7, respectively.

The source code of this work can be found in our project homepage[2].

3) Algorithms:

- PINOCCHIO: It refers to the solution in [5]. We evaluate the $inf(\cdot)$ for all candidates, and select the top k candidates with the maximum $inf(\cdot)$ as the results.
- GreedyP: The GreedyP algorithm in Algorithm 1.
- GreedyPS: The GreedyPS algorithm in Algorithm 2.

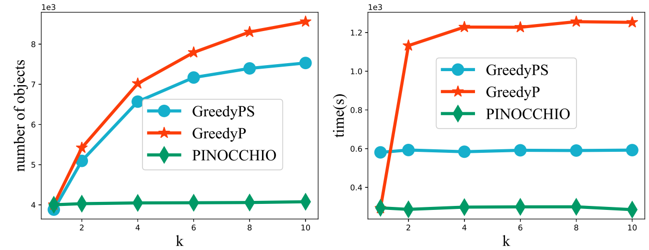
In the following, we evaluate the performances for all methods in the aspects of the number of objects influenced by candidates as well as the time cost for returning the results.

B. Experiment Results

In this section, experiments about the effectiveness for GreedyPS are averaged after 10 groups of experiments. In each of the following experiments, we randomly select 600 positions from the corresponding dataset as the candidate locations to place the facilities.

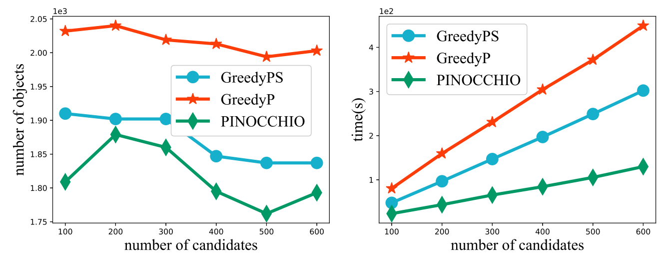
First, we fixed the number of bitmaps and candidates. When the value of k is constantly changing, the following experimental results are obtained.

[2]https://lihuixidian.github.io/malos/



(a) (b)

Figure 3. Effect of k (*Gowalla*, 30 bitmaps)



(a) (b)

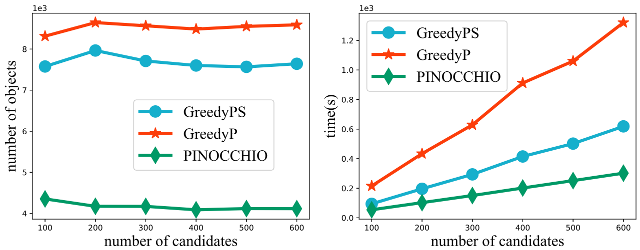
Figure 4. Number of candidates (*Foursquare*, 20 bitmaps)

Fig. 2 shows the results when the number of bitmaps is fixed at 20, the number of candidates is 600 in *Foursquare* dataset. The Fig. 2(a) shows the number of objects as the value of k varies. Fig. 2(b) illustrates the time cost for PINOCCHIO, GreedyP and GreedyPS. GreedyP returns the maximum number of objects and its time consumption is the worst. Although PINOCCHIO takes the least time, the number of objects influenced by candidates is also small.

Fig. 3 shows the results when the number of bitmaps is fixed at 30, the number of candidates is 600 in *Gowalla* dataset. Fig. 3(a) shows the number of objects as the value of k varies, while Fig. 3(b) illustrates the time consumption. Generally, The number of objects using GreedyPS is over 90% for GreedyP. Moreover, GreedyPS takes only half the running time for GreedyP. However, the time consumption for PINOCCHIO is the least and the number of objects using PINOCCHIO algorithm is only a half of GreedyP. We can find a phenomenon that the time does not change significantly as the value of k varies. The reason for this phenomenon may be that the time consumption is the longest to remove the overlap of trajectory sets influenced by candidates in the first iteration.

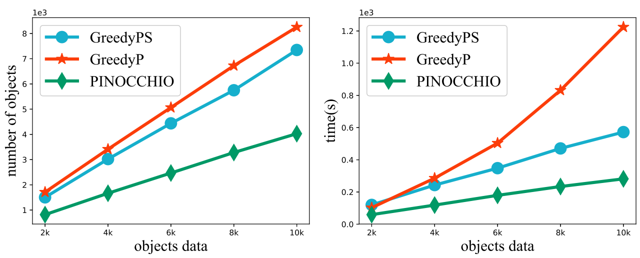
The following part explains the number of moving objects influenced by candidates and time consumption as the number of candidates varies.

Fig. 4 displays the results, the value of k is 10 and the number of bitmaps is fixed at 20 in Foursquare. Fig. 4(a) shows the number of objects influenced by candidates.



(a) (b)

Figure 5. Number of candidates (*Gowalla*, 30 bitmaps)



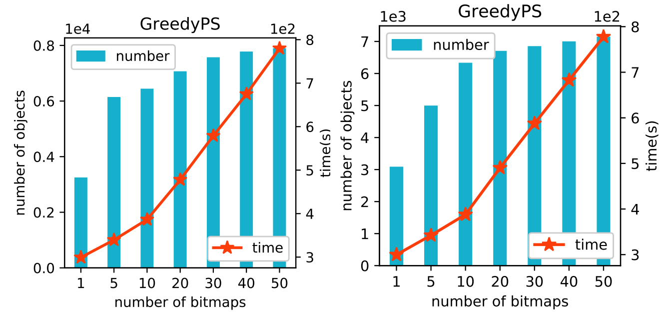
(a) (b)

Figure 6. Number of objects (*Gowalla*, 30 bitmaps)

Fig. 4(b) illustrates the time consumption. GreedyP algorithm returns the maximum number of objects and time consumption is the worst. Although the PINOCCHIO algorithm takes the least time, the number of objects influenced by candidates is also the least.

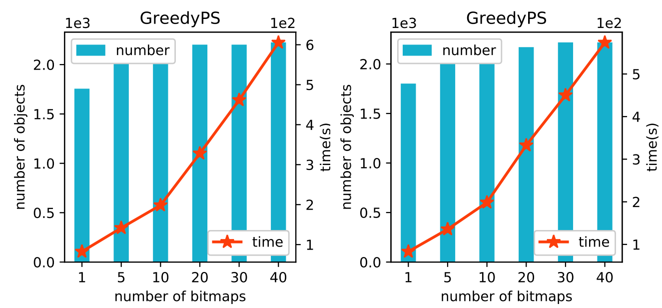
Fig. 5 illustrates the results when the value of k is 10 and the number of bitmaps is fixed at 30 in *Gowalla* dataset. Fig. 5(a) displays that the number of moving objects using GreedyPS can achieve 90% compared with GreedyP algorithm. The PINOCCHIO algorithm only reaches half of the number of using GreedyP algorithm. Fig. 5(b) shows the time consumption for the three algorithms. The time consumption of PINOCCHIO is the least, followed by GreedyPS, and GreedyP algorithms.

Fig. 6 displays the comparison of the three algorithms when the number of objects is varied. As the number of objects in *Foursquare* dataset is too limited, hereby we only test the scalability with respect to the number of objects in *Gowalla*. In all experiments, the value of k is 10, the number of candidate locations is 600 and the number of bitmaps is 30. Fig. 6(a) illustrates the number of objects influenced by 10 candidate locations. The number of objects influenced using the PINOCCHIO algorithm is only half of that of GreedyP algorithm. Compared with GreedyP algorithm, GreedyPS algorithm can obtain nearly 90% objects. Fig. 6(b) shows the time consumption. The time consumption of PINOCCHIO is the least, followed by GreesyPS, and GreedyP algorithm.



(a) $k = 10$ (b) $k = 5$

Figure 7. Number of bitmaps (*Gowalla*)



(a) $k = 10$ (b) $k = 5$

Figure 8. Number of bitmaps (*Foursquare*)

Fig. 7 shows the results for the scenario when the number of bitmaps changes in *Gowalla* dataset. As the number of bitmaps increases, the number of objects influenced by candidates and the time consumption are both increasing. Fig. 7(a) shows the trend of number of objects and time consumption as the number of bitmaps changes, when $k = 10$. The GreesyP algorithm can totally return 8561 objects. The time consumption for GreedyP is 1,228s. When the number of bitmap is one, the number of objects using GreedyPS is only 3,277, but the time cost is extremely limited, i.e., less than 1/10 that of GreedyP. If there are 40-50 bitmaps, precision of GreedyPS can achieve over 90% compared with GreedyP and time consumption is only half of the GreedyP algorithm. Fig. 7(b) illustrates the results for the scenario when $k = 5$ in *Gowalla* dataset, when the number of bitmaps increases. The GreedyP algorithm can totally influence 7427 objects, and the time consumption is 1,213s. When there is only one bitmap, tprecision of GreedyPS is only 40% compared with GreedyP algorithm, but the time consumption is dramatically reduced. If there are 40-60 bitmaps, the precision of GreedyPS can achieve about 95% and the time consumption is only half of the GreedyP algorithm.

Furthermore, we conduct another group of experiments to show how the effectiveness and efficiency will be affected when we employ more bitmaps in GreedyPS algorithm. Fig. 8 displays the results for the scenario when the number of bitmaps changes in *Foursquare* dataset. Fig. 8(a) shows The GreedyP algorithm can totally return 2311 objects influenced by 10 candidates, and the time consumption is 716s. When there is only one bitmap, precision of GreedyPS is only 75% compared with GreedyP, but time consumption is extremely low. If there are 5-10 bitmaps, precision of GreedyPS can achieve about 90% and time consumption is only one-third of the GreedyP algorithm. Fig. 8(b) illustrates the GreedyP algorithm which mines 5 candidates can totally influence 2287 objects, and the time consumption is 713s. When there is only one bitmap, precision of GreedyPS is only 78% compared with GreedyP, but the time consumption is extremely low. If there are 10-30 bitmaps, precision of GreedyPS can achieve about 95% and the time consumption is only half of the GreedyP algorithm.

VI. Conclusion

In this paper, we have introduced a k-Collective Influential Facility Placement problem based on the cumulative influence probability criteria defined in [5]. We prove that the proposed problem is NP-hard. Due to that, we present a basic greedy algorithm called GreedyP with a provable approximation bound $1-\frac{1}{e}$. Considering the time cost of the algorithm may be large in huge dataset, we futher present a more efficient algorithm, namely GreedyPS, using FM sketch to dramatically speed up the moving object update process of GreedyP. We also theoretically justify that, by varying the number of bitmaps adopted in GreedyPS, we are able to control the tradeoff between efficiency and accuracy. Empirical study over two real-world datasets justifies our theoretical study and demonstrates that GreedyP can achieve the best effectiveness with relatively longer running time; while GreedyPS can solve the problem more efficiently with a satisfied accuracy.

Acknowledgment

The work is supported by National Natural Science Foundation of China (No. 61672408), Fundamental Research Funds for the Central Universities (No. JB181505), Natural Science Basic Research Plan in Shaanxi Province of China (No. 2018JM6073) and China 111 Project (No. B16037).

References

[1] Tian Xia, Donghui Zhang, Evangelos Kanoulas, and Yang Du. On computing top-t most influential spatial sites. In *Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005*, pages 946–957, 2005.

[2] Xiang Zhang, David Rey, and S. Travis Waller. Multitype recharge facility location for electric vehicles. *Comp.-Aided Civil and Infrastruct. Engineering*, 33(11):943–965, 2018.

[3] Julien Kritter, Mathieu Brévilliers, Julien Lepagnot, and L-hassane Idoumghar. On the optimal placement of cameras for surveillance and the underlying set cover problem. *Appl. Soft Comput.*, 74:133–153, 2019.

[4] Flip Korn and S. Muthukrishnan. Influence sets based on reverse nearest neighbor queries. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA.*, pages 201–212, 2000.

[5] Meng Wang, Hui Li, Jiangtao Cui, Ke Deng, Sourav S. Bhowmick, and Zhenhua Dong. PINOCCHIO: probabilistic influence-based location selection over moving objects. In *33rd IEEE International Conference on Data Engineering*, pages 21–22, 2017.

[6] Shubhadip Mitra. Identifying top-k optimal locations for placement of large-scale trajectory-aware services. In *Proceedings of the VLDB 2016 PhD Workshop co-located with the 42nd International Conference on Very Large Databases (VLDB 2016), New Delhi, India, September 9, 2016.*, 2016.

[7] Shubhadip Mitra, Priya Saraf, Richa Sharma, Arnab Bhattacharya, Sayan Ranu, and Harsh Bhandari. Netclus: A scalable framework for locating top-k sites for placement of trajectory-aware services. In *33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, April 19-22, 2017*, pages 87–90, 2017.

[8] Yuhong Li, Jie Bao, Yanhua Li, Yingcai Wu, Zhiguo Gong, and Yu Zheng. Mining the most influential k-location set from massive trajectories. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS 2016, Burlingame, California, USA, October 31 - November 3, 2016*, pages 51:1–51:4, 2016.

[9] Ping Zhang, Zhifeng Bao, Yuchen Li, Guoliang Li, Yipeng Zhang, and Zhiyong Peng. Trajectory-driven influential billboard placement. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 2748–2757, 2018.

[10] Yu Sun, Jin Huang, Yueguo Chen, Rui Zhang, and Xiaoyong Du. Location selection for utility maximization with capacity constraints. In *21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012*, pages 2154–2158, 2012.

[11] Da Yan, Raymond Chi-Wing Wong, and Wilfred Ng. Efficient methods for finding influential locations with adaptive grids. In *Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011, Glasgow, United Kingdom, October 24-28, 2011*, pages 1475–1484, 2011.

[12] Raymond Chi-Wing Wong, M. Tamer Özsu, Philip S. Yu, Ada Wai-Chee Fu, and Lian Liu. Efficient method for maximizing bichromatic reverse nearest neighbor. *PVLDB*, 2(1):1126–1137, 2009.

[13] Zenan Zhou, Wei Wu, Xiaohui Li, Mong-Li Lee, and Wynne Hsu. Maxfirst for maxbrknn. In *Proceedings of the 27th International Conference on Data Engineering, ICDE 2011, April 11-16, 2011, Hannover, Germany*, pages 828–839, 2011.

[14] Man Lung Yiu, Xiangyuan Dai, Nikos Mamoulis, and Michail Vaitis. Top-k spatial preference queries. In *Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, The Marmara Hotel, Istanbul, Turkey, April 15-20, 2007*, pages 1076–1085, 2007.

[15] Tai-Hsi Wu and Jen-Nan Lin. Solving the competitive discretionary service facility location problem. *European Journal of Operational Research*, 144(2):366–378, 2003.

[16] Sheng Wang, Zhifeng Bao, J. Shane Culpepper, Timos Sellis, and Gao Cong. Reverse k nearest neighbor search over trajectories. *IEEE Trans. Knowl. Data Eng.*, 30(4):757–771, 2018.

[17] Chuanfei Xu, Yu Gu, Roger Zimmermann, Shukuan Lin, and Ge Yu. Group location selection queries over uncertain objects. *IEEE Trans. Knowl. Data Eng.*, 25(12):2796–2808, 2013.

[18] Shubhadip Mitra, Priya Saraf, and Arnab Bhattacharya. TIPS: mining top-k locations to minimize user-inconvenience for trajectory-aware services. *CoRR*, abs/1709.02343, 2017.

[19] Guoliang Li, Shuo Chen, Jianhua Feng, Kian-Lee Tan, and Wen-Syan Li. Efficient location-aware influence maximization. In *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, pages 87–98, 2014.

[20] Long Guo, Dongxiang Zhang, Gao Cong, Wei Wu, and Kian-Lee Tan. Influence maximization in trajectory databases. *IEEE Trans. Knowl. Data Eng.*, 29(3):627–641, 2017.

[21] Dongxiang Zhang, Long Guo, Liqiang Nie, Jie Shao, Sai Wu, and Heng Tao Shen. Targeted advertising in public transportation systems with quantitative evaluation. *ACM Trans. Inf. Syst.*, 35(3):20:1–20:29, 2017.

[22] Xuemin Lin, Yidong Yuan, Qing Zhang, and Ying Zhang. Selecting stars: The k most representative skyline operator. In *Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, The Marmara Hotel, Istanbul, Turkey, April 15-20, 2007*, pages 86–95, 2007.

[23] Philippe Flajolet and G. Nigel Martin. Probabilistic counting algorithms for data base applications. *J. Comput. Syst. Sci.*, 31(2):182–209, 1985.

[24] Uriel Feige. A threshold of ln *n* for approximating set cover. *J. ACM*, 45(4):634–652, 1998.

[25] David Kempe, Jon M. Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24 - 27, 2003*, pages 137–146, 2003.

APPENDIX

Proof: According to Definition 3, $\sigma(C)$ denotes the number of moving objects influenced by C. Let $Tr(c_i)$ be the moving object set influenced by c_i, *i.e.,* $Tr(c_i) = \{O_1, O_2, ..., O_m\}$.

$\forall A \in C$, $\sigma(A) \geq 0$.
Thus, the function $\sigma(\cdot)$ is non-negative.
$\forall A \in C$, $\sigma(A) \geq 0$.
$\forall e, e \in C - A, \sigma(e) \geq 0$, $\sigma(A \cup e) \geq \sigma(A)$.
Thus, the function $\sigma(\cdot)$ is monotone.
Suppose $A \subseteq B \subseteq C, \sigma(A) \geq 0, \sigma(B) \geq 0$.
$\forall e, e \in C - B$, we can get:

1) If $Tr(A) = \varnothing$ and $Tr(e) \cap Tr(B) = \varnothing$.
   $\sigma(A \cup e) - \sigma(A) = inf(e)$.
   $\sigma(B \cup e) - \sigma(B) = inf(e)$.
   Thus, $\sigma(A \cup \{e\}) - \sigma(A) \geq \sigma(B \cup \{e\}) - \sigma(B)$.
2) If $P = Tr(e) \cap Tr(A) = \{O_{11}, O_{12}, ..., O_{1j}\}$.
   Assume:
   $Q = Tr(e) \cap Tr(B) = \{O_{11}, O_{12}, ..., O_{1j}, ..., O_{1i}\}$,
   then $\|Q - P\| \geq 0$.
   $\sigma(A \cup \{e\}) - \sigma(A) = \sigma(B \cup \{e\}) - \sigma(B) + \|Q - P\|$.
   Thus, $\sigma(A \cup \{e\}) - \sigma(A) \geq \sigma(B \cup \{e\}) - \sigma(B)$.

According to 1) and 2), the function $\sigma(\cdot)$ is submodular. ■