Distributed and Parallel Databases
https://doi.org/10.1007/s10619-020-07311-0

Influential facilities placement over moving objects

Ying Zhou[1,2] · Hui Li[1,2] · Dan Li[1] · Meng Wang[3] · Jiangtao Cui[4]

Accepted: 14 September 2020

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. We further present a pair of extensions to the problem, namely k-Additional and k-Eliminative Influential Facility Placement problems. We also present corresponding approximate solutions towards both extensions and theoretically show that results of both algorithms are guaranteed. Experiments on real datasets verify the efficiency and effectiveness for all these algorithms comparing with baselines.

Keywords Moving objects · Location selection · Submodular · Approximate algorithm

A short version of this paper appeared in [16], which received the Best Paper Award Runner-up in MDM 2019.

✉ Hui Li
hli@xidian.edu.cn

[1] State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an, China

[2] School of Cyber Engineering, Xidian University, Xi'an, China

[3] School of Computer Science, Xi'an Polytechnic University, Xi'an, China

[4] School of Computer Science and Technology, Xidian University, Xi'an, China

Published online: 20 September 2020

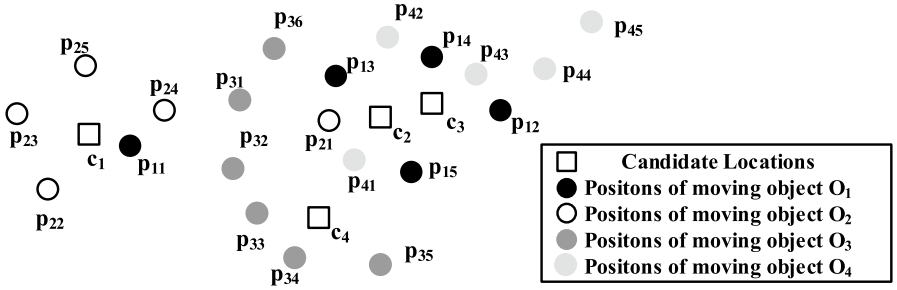


Fig. 1 Motivating example

1 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 [29]. Finding such an optimal location from candidates to establish a new facility has a wide spectrum of applications such as marketing, urban planning [35], monitoring wildlife [14], 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 [13], the influence of a candidate c is defined as the number of objects whose nearest neighbors are c. Recently, Wang et al. [25] 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 [13] will report c_1, but not c_2, influences O_1. On the other hand, the cumulative probability (according to [25]) 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 [25] and focus on the cumulative probability settings in this work.

1.1 Motivation

The proposed algorithm in [25], 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 [25] 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 [18, 20, 22, 34]. 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 [18]. 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, [18] 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 [25] 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 [25]) 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 [25]. 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.

In fact, it is not limited to the previously mentioned scenarios in real life. In some cases, a merchant may have set up some facilities in a city and now need to expand its business scope. On the premise that some facilities have been deployed, additional k facilities need to be added to maximize the number of moving objects. We define it as the k **-Additional Influential Facility Placement problem** and show that this problem can be degenerated into k-Collective Influential Facility problem. Therefore, we only need to make a simple modification to previous algorithms, and the approximation ratio is still $(1 - \frac{1}{e})$. Another possible scenario is that there are

too many facilities(*i.e.,* θ facilities) currently deployed by the merchant, which not only causes a waste of resources but also increases the operation costs. At this time, the merchant needs to remove k facilities from the existing facility network, so as to maximize the number of objects influenced by $(\theta - k)$ facilities. We define it as the k-**Eliminative Influential Facility Placement problem** and clarify that it is not feasible to delete the location with the least impact from existing ones using the greedy strategy. Instead, we use the solutions in k-Collective Influential Facility Placement problem to find $\theta - k$ locations.

1.2 Challenge and algorithm

Given the k**-Collective Influential Facility Placement problem**, we are requested to select the desired number k of locations to set up service facilities so that these facilities can serve as many users as possible. This problem can be attributed to a max k-cover problem. From an algorithmic perspective, the max k-cover problem is NP-Hard and cannot be solved in polynomial time. To address that, we proposed a GreedyP algorithm, using the idea of the hill-climb strategy to iteratively select the points that influences the most users. This process is continued for k times to obtain the most influential k facilities required.

However, in the actual execution, we found that the calculation of whether the user trajectory is influenced by a certain candidate is costly. In this regard, we apply the pruning strategy proposed by [25] to avoid calculating some users who's moving trajectories are far from the candidate, thus reduce time in impact calculation. At the same time, after selecting a candidate point, we should also delete the users' trajectories affected by the selected candidate point, so as to avoid recalculating the affected user trajectories in the next location selection. This task is also a time-consuming operation. What's more, when the number of user trajectory is huge, recording and calculating the user's impact of each candidate point requires a lot of space and time.

To solve this problem, we proposed to use the FM-sketch method to unevenly hash the user trajectory into a 0–1 array of length L, which means that when the user influences the candidate, a certain position in the 0–1 array of the candidate will change into 1. When calculating the number of users affected by the candidate, you only need to count the number of 1 in the array. Besides, after selecting a candidate point, to eliminate the influence of the candidate, we only need to perform 0–1 bit operation. It will largely reducing the time cost. What's more, to alleviate the collision problem of the hash operation, in order to improve the accuracy of the calculation, we propose to use multiple (*i.e.,* w) hash arrays to improve the accuracy.

1.3 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.
- We extensively present two more general problems, and propose approximate solutions correspondingly.

The rest of the paper is organized as follows. We review the related work in Sect. 2. The formalized problem definition is given in Sect. 3. Afterwards, we present our solutions and conduct theoretical studies in Sect. 4. In Sect. 5, we present two other general extensions to the k-Collective Influential Facility Placement problem and give the corresponding solutions. The experimental results are demonstrated in Sect. 6. Lastly, we conclude our work in Sect. 7.

2 Related work

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

2.1 Location selection

There have been increasing research efforts in LS problem under various applications [1, 10, 24–29, 31, 32, 36]. Most of these studies assume that user's locations are static and only the most influential location is retrieved. Xia et al. [29] defined the influence of a location as the total weight of its reverse nearest neighbors (RNNS). Sun et al. [24] validated all clients and their corresponding BRNN sets and proposed three pruning techniques to tighten the search space. Yan et al. [31] further relaxed the criterion from NN facility to $(1 + \alpha) * NN$, where α is a user-specified value. Wong et al. [27] studied a similar problem, called MaxBRkNN, in which all kNN facilities exhibit influence on objects. Zhou et al. [36] proposed MaxFirst to solve MaxBRkNN. The solution partitioned the space into quadrants iteratively and pruned the unpromising candidates using upper and lower bounds. Gao et al. [7] aims to find mutual nearest neighbor of users and facilities, it focus on the nearest neighbor to provide services such as taking a taxi and so on. Our work only concerned about whether the candidate location can affect the user in order to select the most influential facility point. Chen et al. [3] aims to find the co-movement pattern to achieve trajectory compression and prediction of future motion trajectories, which is different from our work. Recently, Wang et al. [25] 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.

Notably, in this work, we consider only the distance factor in testing whether a facility will influence some trajectories. In fact, there may exist many other criteria to evaluate the influence, especially social influence, which has been extensively studied recently. For instance, direct social information metrics adopt the number of followers [2] or entropy [23] as the measure for social influence; hyperlink-based metrics may adopt influence rank [9] or variations of PageRank [11] to evaluate the influence; machine-learning-based metrics employ learning models to predict the influence [5, 15] of an arbitrary object.

2.2 Maximum coverage problems

Maximum coverage problem has great utility for several real-world applications [8, 17, 18, 20–22, 30, 33, 34]. In these methods, every user is modeled as a moving object. Xu et al. [30] proposed group locations selection problem to find the minimum number of multiple locations with influence regions, such that all the objects can be covered. Mitra et al. [20] proposed three different applications, namely TOPS, TUMP and TIPS, respectively. TOPS [22] 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 [21] 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. [18] 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. [34] 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. [8] and Zhang et al. [33] 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 [18]. The rules for determining the impact in [22, 34] are similar, and c_1, c_2 can affect both O_1 and O_2. Comparing with these works, the cumulative probability proposed in [25], c_1 only influences O_1, c_2 influences O_1 and O_2. In this paper, we employ the influence setting of [25] and present a pair of algorithms to find k locations in candidate set which can affects the largest number of moving objects.

3 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).

3.1 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: $Pr_c(O) = 1 - \prod_{i=1}^{r}(1 - Pr_c(p_i))$ [25].

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 [25].

$Pr_c(O)$ 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,

Table 1 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

$inf(c_1) = 1$. On the other hand, if $Pr_{c_2}(O_1)$=0.8 and $Pr_{c_2}(O_2)$=0.79, then c_2 influences both O_1 and O_2. That is, $inf(c_2) = 2$.

3.2 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 1 as an example, which lists the information that every location c can influence a set of 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.

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

4 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 [6, 19].

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 [4].

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 [4], the Max k-cover problem has been proven to be NP-hard. Therefore, the problem in Definition 4 is NP-hard. □

4.1 GreedyP algorithm

4.1.1 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 [25] (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).

Algorithm 1: GreedyP Algorithm.

```
   Input: The set of candidates C;
   Input: The set of Object Ω;
   Input: 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 = ∅;
3  foreach i to k do
4      find s_i ∈ C − S, where the value of inf(s_i) is maximum;
5      S = S ∪ s_i
6      each s_j ∈ C − S
7          delete Tr(s_j) ∩ Tr(S) from Tr(C);
8  return S;
```

Example 3 Reconsider Table 1 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.

4.1.2 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.* [12]

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 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.,* $Tr(c_i) = \{O_1, O_2, ..., O_m\}$.

Firstly, $\forall A \in C, \sigma(A) \geq 0$.

Thus, the function $\sigma(\cdot)$ is non-negative.

Secondly, $\forall A \in C, \sigma(A) \geq 0$.

$\forall e, e \in C - A, \sigma(e) \geq 0, \sigma(A \cup e) \geq \sigma(A)$.

Thus, the function $\sigma(\cdot)$ is monotone.

Lastly, Suppose $A \subseteq B \subseteq C, \sigma(A) \geq 0, \sigma(B) \geq 0$.

$\forall e, e \in C - B$, we can get:

Case 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, $\sigma(A \cup \{e\}) - \sigma(A) \geq \sigma(B \cup \{e\}) - \sigma(B)$.

Case 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. □

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 [25]. 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.

4.2 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 [6] 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 $0 \le j \le L-1$). Suppose the $h(\cdot)$ is a randomly generated hash function[2] 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:

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

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

[2] In fact, FM sketch contains a series of other techniques, we only employ the hash strategy herein.

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 iterating each bit of $f(P)$ (w arrays with length L) and finding the bitmap that has the largest number 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 1 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 $S = \{c_1, c_2\}$.

However, if only one bitmap is adopted in our solution, the resulting set will be definitely poor. 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;
Input: The set of Object Ω;
Input: 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 **foreach** i *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 $s_j \in C - S$ $f(Tr(s_j)) = f(current) \oplus f(Tr(s_j)) - f(current)$;
7 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 [25]. 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.

5 Two extensions

In this part, we discuss a pair of extensions to the k-Collective Influential Facility Placement problem for more general scenarios. Two algorithms are presented to address these general problems; theoretical guarantees for the result quality are also provided for both algorithms.

5.1 *k*-Additional influential facility placement problem

In practice, the network of facilities may not be built overnight, instead the facilities are always incrementally built in batches. For instance, hotel groups always plan and announce new hotels within a country in batches;[3] a set of new gas stations within a region are planned at the same time, in addition to the existing service network.[4] In these cases, it can be regarded as, that there are θ existing facilities, an additional k facilities need to be added to maximize the number of moving objects that are influenced. It can be formalized as the followings.

Definition 8 Given a set of exiting locations $E = \{e_1, e_2, ...e_\theta\}$, a set of moving objects $\Omega = \{O_1, O_2, ...O_m\}$ where $O_i = \{p_1, p_2, ..., p_r\}$, a set of candidate locations $C = \{c_1, c_2, ...c_n\}$, the budget number of new facilities k $(0 < k \leq n)$. The k-**Additional Influential Facility Placement problem** aims to mine $\exists S \subseteq C$, subject to $|S| = k$, such that $\sigma(E \cup S)$ is maximized, hereby $\sigma(\cdot)$ follows Definition 3.

In fact, the problem mentioned in Definition 8 will degenerate to the k-Collective Influential Facility Placement problem when θ is fixed as 0. The previous methods mentioned in Sect. 4 can be applied to this new problem, and our solution has the same theoretical guarantee.

Theorem 6 *The k-Additional Influential Facility Placement problem in Definition 8 is NP-hard.*

[3] https://www.hotelmanagement.net/franchising/marriott-announces-7-new-hotels-across-3-brands-for-china.

[4] https://www.dallasnews.com/business/local-companies/2017/12/07/exxon-to-open-eight-mobil-gas-stations-in-mexico-with-plans-for-50-by-early-2018/.

Proof For the existing moving objects set Ω, there are already existing locations set E. The purpose of adding k new candidates from C is to maximize the number of objects covered in the union of E and $S \subseteq C(|S| = k)$. It is equivalent to finding k locations in the set $S \subseteq C$, which influence the most moving objects in the new set $\Omega' = \Omega - Tr(E)$. That is, this new problem can be degenerated to the k-Collective Influential Facility Placement problem. Following Theorem 1, the problem in Definition 8 is also NP-hard. □

Now, we are well prepared to solve new problem. For the k-Additional Influential Facility Placement problem, similar to GreedyP, we also give a specific solution called $(\theta + k)$-Greedy algorithm outlined in Algorithm 3. Above all, the algorithm begins by calculating the set $Tr(E)$ (Line 1). Then, the moving objects that are already influenced by E, namely $Tr(E)$, are removed from Ω (Line 2). Finally, utilize GreedyP (*resp.,* GreedyPS) algorithm to extract k locations as results, which are denoted as S (Lines 3-5).

Obviously, given Theorem 6 and Algorithm 3, we can easily derive that Algorithm 3 achieves $(1 - \frac{1}{e})$ approximation ratio with respect to k-Additional Influential Facility Placement problem. The proof straightforwardly follows that of Theorem 2.

Notably, in Line 4 of Algorithm 3, GreedyPS can be adopted as an alternative choice other than GreedyP. In that case, the result quality will be sacrificed for better efficiency according to the discussion in Sect. 4.2.

Theorem 7 *The time complexity of Algorithm 3 is* $O(\theta' mr') + O(\theta m^2) + O(n'm'r'') + O(knm'^2)$, *where* m *is the number of moving objects,* n *is the number of candidates and* θ *is the number of existing locations.*

Proof The time complexities of calculating object sets for existing locations and candidates are $O(\theta' mr')$ and $O(n'm'r'')$, respectively, as mentioned in [25]. m' is the number of deleting the moving objects influenced by existing E. The meaning of r'' is the same as r', with different values. The time complexities of deleting the objects influenced by E and S are $O(\theta m^2)$ and $O(knm'^2)$, respectively. Therefore, the complexity of the algorithm is $O(\theta' mr') + O(\theta m^2) + O(n'm'r'') + O(knm'^2)$. □

Algorithm 3: k-Addition Algorithm.

Input: The set of existing locations E, the set of candidates C, the set of moving object Ω, the number of additional facilities k;
Output: The set of locations S with k elements;
1 Calculate the objects set influenced by every candidate, $Tr(E)$;
2 Delete the moving objects influenced by set E, $\Omega' = \Omega - Tr(E)$;
3 $S = \varnothing$;
4 S=*GreedyP*(C,Ω',k) (or *GreedyPS*(C,Ω',k));
5 return S;

Table 2 Description of real-world datasets

	Foursquare	*Gowalla*
Number of users (objects)	2321	10,162
Number of check-ins (positions)	167,231	381,165

5.2 *k*-Eliminative influential facility placement problem

From a different perspective, converse to the examples discussed in Sect. 5.1, there are also cases that existing facilities need to be closed in a batch due to unsatisfied economic returns or redundant coverage. This scenario further extends the problem in Definition 8, as it can be regarded as the case for $k < 0$. In this case, k facilities with inferior influences need to be removed from existing service network. Such a problem can be formalized as follows.

Definition 9 Given a set of exiting locations $E = \{e_1, e_2, ...e_\theta\}$, a set of moving objects $\Omega = \{O_1, O_2, ...O_m\}$ where $O_i = \{p_1, p_2, ..., p_r\}$, the budget number k of facilities to be eliminated, $(k \leq \theta)$. The ***k*-Eliminative Influential Facility Placement problem** aims to find $S \subseteq E$, subject to $|S| = k$, such that $\sigma(E \setminus S)$ is maximized, hereby $\sigma(\cdot)$ follows Definition 3.

Intuitively, this new scenario cannot seem to be directly convertible to either of the two problems proposed above, which perform addition of facilities while this new scenario focuses on elimination. We can not simply greedily select a facility with the minimal marginal influence iteratively, as this will definitely introduce much errors. For instance, suppose there are three existing facilities $E = \{e_1, e_2, e_3\}$, four moving objects $\Omega = \{O_1, O_2, O_3, O_4\}$ and budget $k = 1$; and e_1 individually influences $\{O_2, O_3, O_4\}$, e_2 individually influences $\{O_2, O_3\}$ and e_3 individually influences $\{O_1\}$. If we select an existing facility with the minimal influence, e_3 has to be chosen here, leading to a result with $\sigma(E \setminus S) = 3$ as $\{e_2, e_3\}$ influences $\{O_2, O_3, O_4\}$. However, this choice is definitely inferior to eliminating e_2, which will result in $\sigma(E \setminus S) = 4$ as $\{e_1, e_3\}$ influences $\{O_1, O_2, O_3, O_4\}$.

In spite that greedy elimination cannot be applied in the problem, we can also present effective solutions based on the framework presented above. Instead of directly eliminating facilities from E, we consider the problem from the opposite way. Deleting k locations from set E to maximize the influence of the rest facilities, is in fact equivalent to the problem of finding $\theta - k$ facilities from E, such that the influence of these selected facilities with respect to Ω is maximized. In this regard, the problem in Definition 9 is the same as the k-Collective Influential Facility Placement problem in Definition 4 where E is set as the candidate set and $\theta - k$ corresponds to the budget number of new facilities.

Therefore, the k-Eliminative Influential Facility Placement problem is also NP-hard and can be solved by GreedyP and GreedyPS by setting the candidate set as

E and budget number as $\theta - k$. Consequently, the GreedyP solution is also guaranteed to achieve $(1 - \frac{1}{e})$ approximation ratio in this problem.

6 Experiment

6.1 Experiment setup

6.1.1 Datasets

Table 2 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 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.

6.1.2 Experimental settings

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

In line with the settings in paper [25], 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[5].

6.1.3 Algorithms

- PINOCCHIO: It refers to the solution in [25]. 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.

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

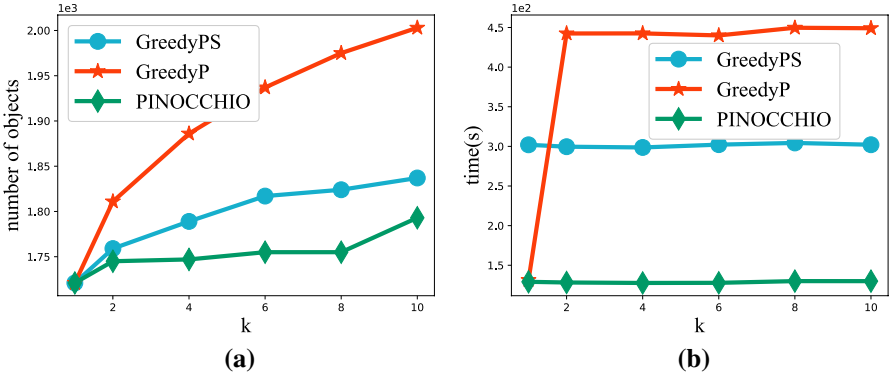


Fig. 2 Effect of k (*Foursquare*, 20 bitmaps)

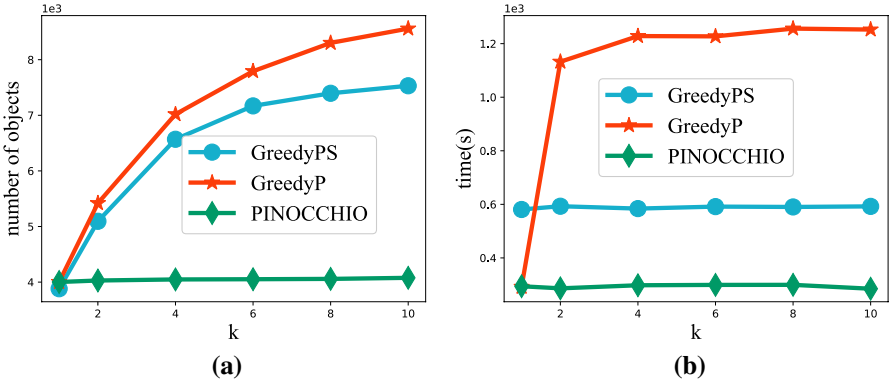


Fig. 3 Effect of k (*Gowalla*, 30 bitmaps)

6.2 Experiment results

6.2.1 Experimental study of *k*-collective problem

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.

Figure 2 shows the results when the number of bitmaps is fixed at 20, the number of candidates is 600 in *Foursquare* dataset. Figure 2a shows the number of objects as the value of k varies. Figure 2b 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.

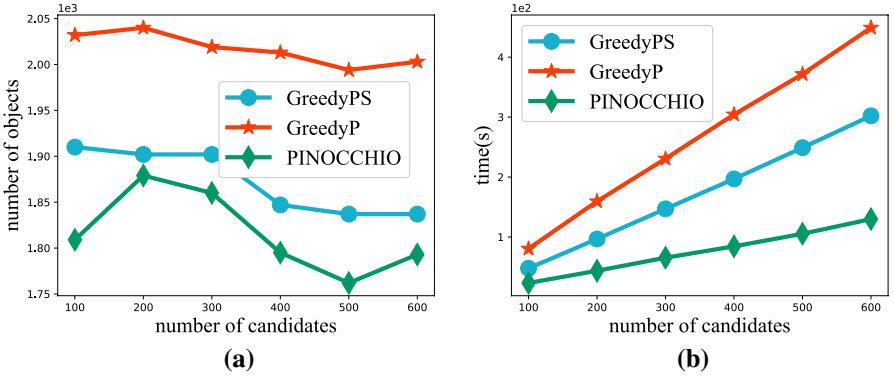


Fig. 4 Number of candidates (*Foursquare*, 20 bitmaps)

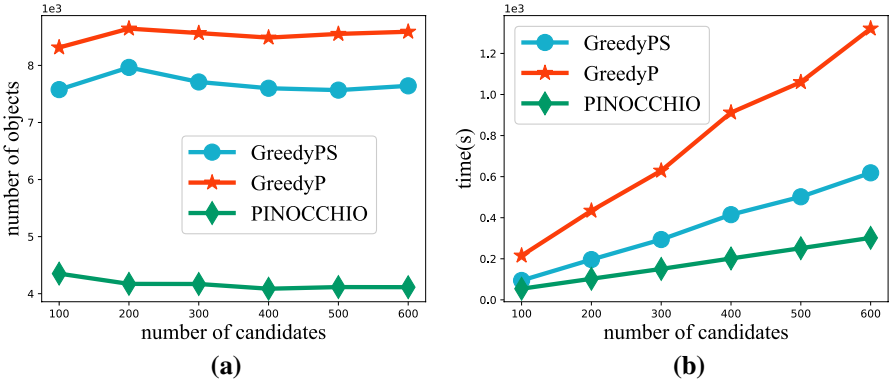


Fig. 5 Number of candidates (*Gowalla*, 30 bitmaps)

Figure 3 shows the results when the number of bitmaps is fixed at 30, the number of candidates is 600 in *Gowalla* dataset. Figure 3a shows the number of objects as the value of k varies, while Fig. 3b 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.

Figure 4 displays the results, the value of k is 10 and the number of bitmaps is fixed at 20 in Foursquare. Figure 4a shows the number of objects influenced by candidates. Figure 4b illustrates the time consumption. GreedyP algorithm returns

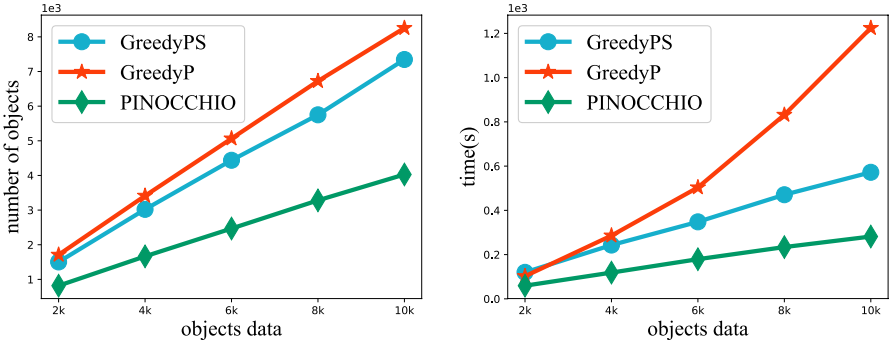


Fig. 6 Number of objects (*Gowalla*, 30 bitmaps)

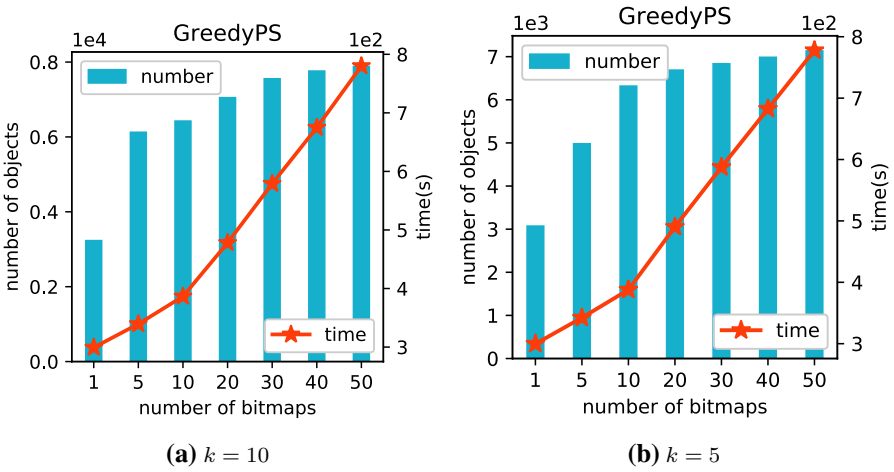
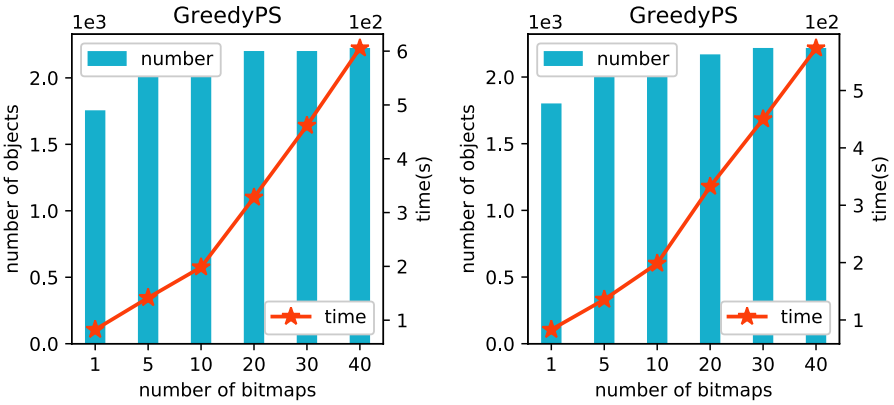


Fig. 7 Number of bitmaps (*Gowalla*)

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.

Figure 5 illustrates the results when the value of k is 10 and the number of bitmaps is fixed at 30 in *Gowalla* dataset. Figure 5a 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. Figure 5b shows the time consumption for the three algorithms. The time consumption of PINOCCHIO is the least, followed by GreedyPS, and GreedyP algorithms.

Figure 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*.



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

Fig. 8 Number of bitmaps (*Foursquare*)

In all experiments, the value of k is 10, the number of candidate locations is 600 and the number of bitmaps is 30. Figure 6a 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. Figure 6b shows the time consumption. The time consumption of PINOCCHIO is the least, followed by GreesyPS, and GreedyP algorithm.

Figure 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. Figure 7a 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 1228s. 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. Figure 7b 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. Figure 8 displays the results for the scenario when the number of bitmaps changes in *Foursquare* dataset. Figure 8a shows The GreedyP algorithm can totally return 2311 objects influenced by ten candidates, and the time consumption is 716s.

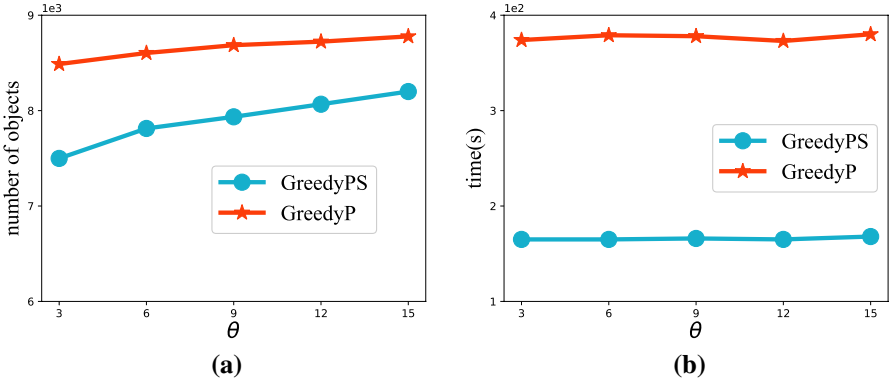


Fig. 9 Number of θ (*Gowalla, bitmaps = 40*)

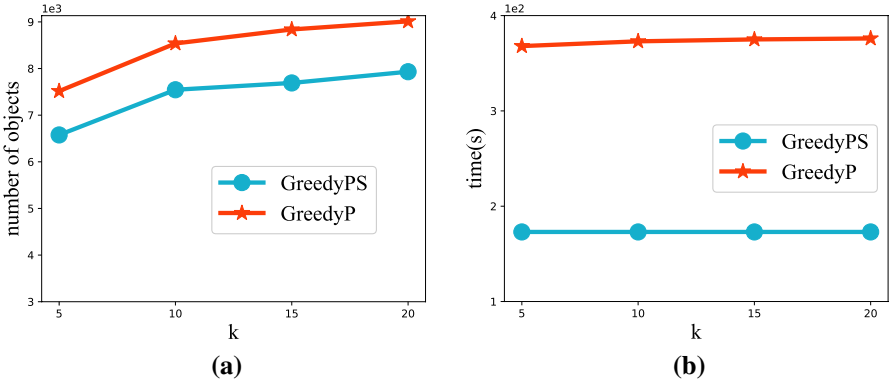


Fig. 10 Number of k (*Gowalla, bitmaps = 40*)

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. Figure 8b 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.

6.2.2 Experimental study on *k*-additional and *k*-eliminative problems

In this part, we conduct experiments to test our solutions towards two extension problems proposed in Sect. 5.

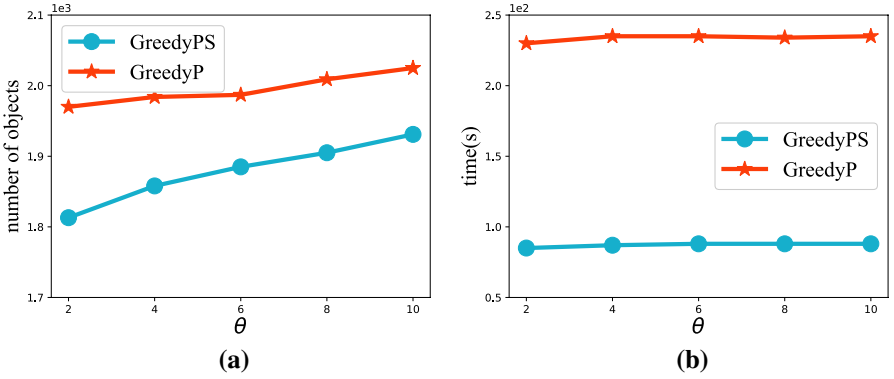


Fig. 11 Number of θ (*Foursquare, bitmaps = 30*)

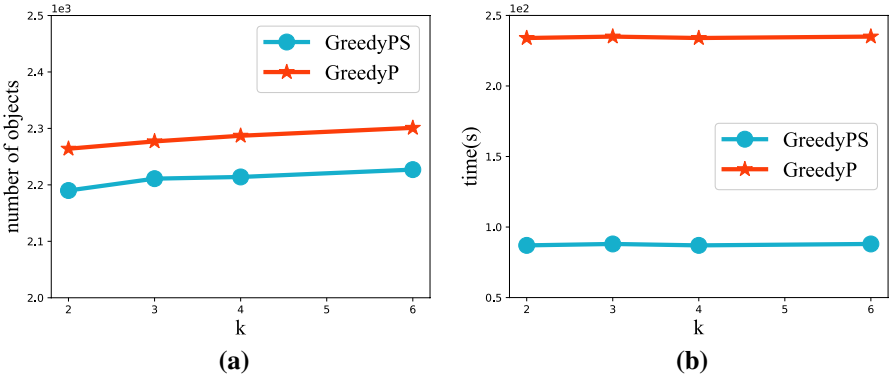


Fig. 12 Number of k (*Foursquare, bitmaps = 30*)

For the k-Additional Influential Facility Placement problem, we mainly conduct experiments on Algorithm 3 and compare the two methods of GreedyP and GreedyPS algorithms on the Gowalla and Foursquare datasets, in terms of the overall number of influenced objects (*i.e.,* influence by $\theta + k$) as well as the response time.

For Gowalla dataset, Fig. 9 displays the results of the number of moving objects and the time consumption when k is set to 10 and the value of θ varies. Notably, as θ increases from 3 to 15 (*i.e.,* $\theta + k$ increases from 13 to 25), the overall number of influenced objects does not increases obviously. The reasons for this phenomenon are as follows. Firstly, as θ increases, the marginal influence for k locations that are selected by GreedyP (*resp.,* GreedyPS) eventually decreases. It is easy to understand, there is smaller improvement space for a bigger service network. Secondly, although the increase in θ may enlarge the service network of E, the improvement exhibits a long-tail phenomenon, that is, after a certain threshold, further increase in the number of facilities may not improve the service effect significantly. Figure 10 shows the case where the value of θ is fixed at 10

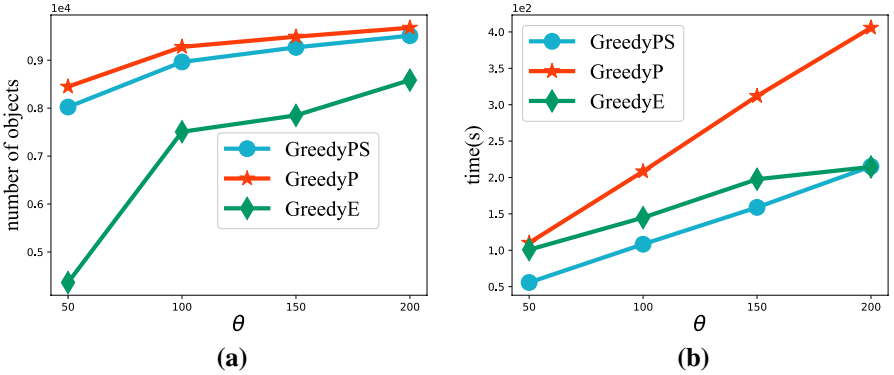


Fig. 13 Number of θ (*Gowalla, bitmaps = 40*)

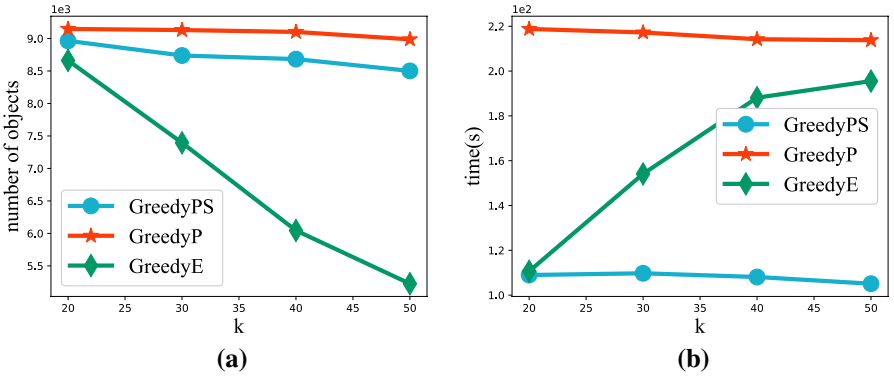


Fig. 14 Number of k (*Gowalla, bitmaps = 40*)

and k varies. The results in Figs. 9 and 10 justify that GreedyP provides higher results quality but requires much more running time (with more than 2 times).

For Foursquare dataset, Fig. 11 shows the results by varying θ when k is fixed as 5. Figure 12 shows the experimental results when the value of k varies and θ is fixed at 5. Unsurprisingly, the results in Foursquare exhibit the same phenomenon as that of Gowalla.

Afterwards, we conduct experiments over the k-Eliminative Influential Facility Placement problem, where the following three algorithms are adopted for experimental comparison.

- GreedyE: Greedily eliminate objects from E over k iterations. During each iteration, it removes from E the one that influences the minimum number of moving objects.
- GreedyP: Following our approach proposed in Sect. 5.2, looking for the $\theta - k$ locations that collectively have the maximum influence via GreedyP.

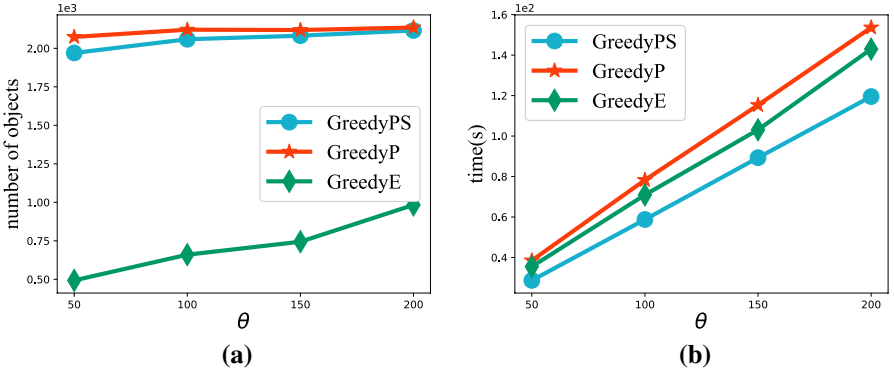


Fig. 15 Number of θ (*Foursquare, bitmaps = 30*)

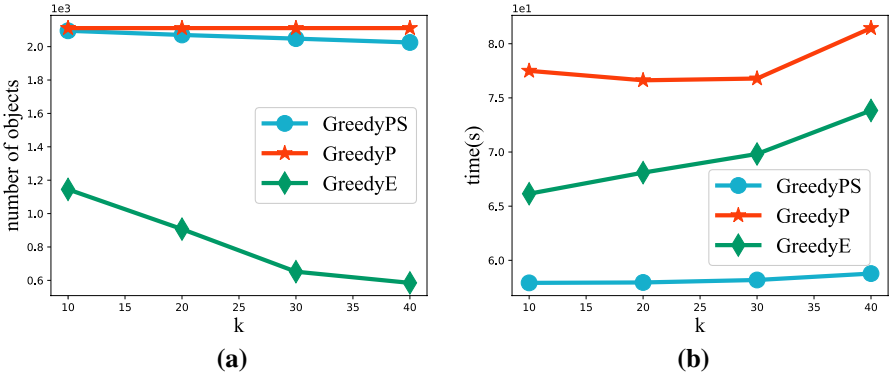
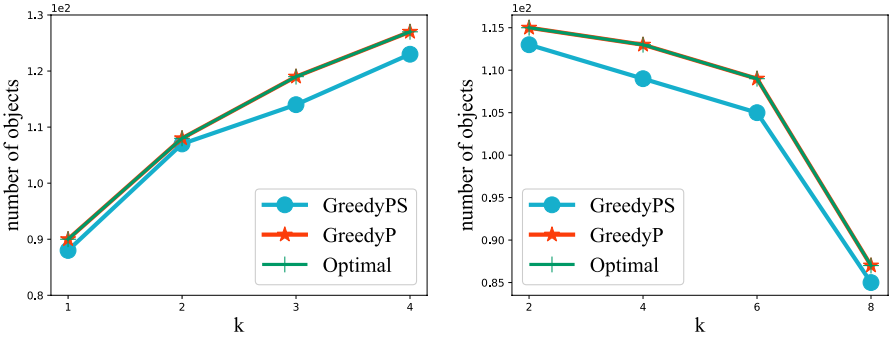


Fig. 16 Number of k (*Foursquare, bitmaps = 30*)

– GreedyPS: Following our approach proposed in Sect. 5.2, looking for the $\theta - k$ locations that collectively have the maximum influence via GreedyPS.

The results in Fig. 13 show the difference among the algorithms for Gowalla dataset, where θ varies from 50 to 200 and the value of k is 30. It is clear that the numbers of moving objects that are obtained using the GreedyP and GreedyPS algorithms are greater than that of the GreedyE algorithm. It is consistent with our discussion of GreedyE approach just below Definition 9. Interestingly, similar to GreedyP and GreedyPS, whose complexity are directly affected by $\theta - k$ (n, k in Theorem 3 are replaced by $\theta, \theta - k$ in k-Eliminative scenario, respectively), and the running time for GreedyE also increases as θ varies from 50 to 200. The reason is that, as θ increases, E is enlarge, the complexity for GreedyE to select the one with the minimum influence from E also increases.

Figure 14 shows performance of different algorithms as k changes. The number of moving objects influenced by the GreedyP algorithm is the largest. As we clarified in Sect. 5.2, GreedyE has the worst effectiveness with the growth of k,



(a) k-additional **(b)** k-eliminative

Fig. 17 Comparison with the *Optimal* on small dataset (*bitmaps* = *30*)

which indicates that it is ineffective to remove the inferior facility in each round of greedy iterations. The time consumption of GreedyPS is far less than the other two algorithms, while the number of moving objects is very close (more than 90%) to that of the GreedyP algorithm.

Figure 15 reports the results of the algorithms in Foursquare. Similar to Fig. 13, θ varies and k stays unchanged as 30. Obviously, GreedyP and GreedyPS can obtain significantly larger number of influenced objects, while GreedyE has wide gaps for all θ values. On the other hand, the running time of GreedyPS is still the shortest, followed by GreedyE and GreedyP.

Figure 16 illustrates the number of moving objects influenced by $\theta - k$ locations and the time consumption for the algorithms by varying k. The number of moving objects obtained by GreedyP is slightly larger than that of GreedyPS. GreedyE is significantly inferior to the other two algorithms, and the phenomenon is more obvious than that in Fig. 14. The efficiency of GreedyPS is the best, followed by GreedyE and GreedyP.

To better understand the performance gap between the approximate solutions and the *Optimal* solutions. We extracted 200 users from the *Gowalla* dataset, and generated a small candidate dataset with a few (less than 10) candidate points. Through that, we are able to enumerate all possible solutions to find the optimal one, which we refer to as *Optimal*. We compare the performance gap between our approximate approaches and the *Optimal* in Figs. 17a and b.

Figure 17a shows the results when the number of moving objects θ is set to 2 and k varies from 1 to 4 for k-additional problem. We can find the *Optimal* solution has the identical result with GreedyP. Besides, GreedyPS achieves more than 95% accuracy comparing with the *Optimal* solution.

Figure 17b shows the results when θ is set to 10 and the value of k varies from 2 to 8 for k-eliminative problem. Similar with Fig. 17a, *Optimal* solution has the identical result with GreedyP. GreedyPS achieves more than 95% accuracy comparing with the *Optimal* solution.

7 Conclusion

In this paper, we have introduced a k-Collective Influential Facility Placement problem based on the cumulative influence probability criteria defined in [25]. 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 further 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. Moreover, we further present two extension problems, namely k-Additional and k-Eliminative. We theoretically show that k-Additional (*resp.,* k-Eliminative) degenerates (*resp.,* is equivalent) to k-Collective problem. We also theoretically discuss the hardness of both problems, and present solutions with guaranteed approximation ratio. As part of our future work, we shall explore ways to further improve the efficiency while preserving high result quality.

Funding It is supported by National Natural Science Foundation of China (No. 61672408, 61972309, 61976168), CCF-Huawei Database System Innovation Research Plan (No. 2020010B), Natural Science Basic Research Program of Shaanxi (No. 2020JM-575) and China 111 Project (No. B16037).

References

1. Abdullatif, A., Masulli, F., Rovetta, S.: Tracking time evolving data streams for short-term traffic forecasting. Data Sci. Eng. **2**(3), 210–223 (2017)
2. Cha, M., Haddadi, H., Benevenuto, F., Gummadi, P.K.: Measuring user influence in twitter: The million follower fallacy. In: Cohen WW, Gosling S (eds) Proceedings of the Fourth International Conference on Weblogs and Social Media, ICWSM 2010, Washington, DC, USA, May 23–26, 2010, The AAAI Press (2010)
3. Chen, L., Gao, Y., Fang, Z., Miao, X., Jensen, C.S., Guo, C.: Real-time distributed co-movement pattern detection on streaming trajectories. Proc. VLDB Endow. **12**(10), 1208–1220 (2019)
4. Feige, U.: A threshold of ln n for approximating set cover. J. ACM **45**(4), 634–652 (1998)
5. Feng, S., Cong, G., Khan, A., Li, X., Liu, Y., Chee, Y.M.: Inf2vec: Latent representation model for social influence embedding. In: 34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, April 16–19, 2018, IEEE Computer Society, pp. 941–952 (2018)
6. Flajolet, P., Martin, G.N.: Probabilistic counting algorithms for data base applications. J. Comput. Syst. Sci. **31**(2), 182–209 (1985)
7. Gao, Y., Zheng, B., Chen, G., Li, Q., Chen, C., Chen, G.: Efficient mutual nearest neighbor query processing for moving object trajectories. Inf. Sci. **180**(11), 2176–2195 (2010)
8. Guo, L., Zhang, D., Cong, G., Wu, W., Tan, K.: Influence maximization in trajectory databases. IEEE Trans. Knowl. Data Eng. **29**(3), 627–641 (2017)
9. Hajian, B., White, T.: Modelling influence in a social network: Metrics and evaluation. In: PASSAT/SocialCom 2011, Privacy, Security, Risk and Trust (PASSAT), 2011 IEEE Third International Conference on and 2011 IEEE Third International Conference on Social Computing (SocialCom), Boston, MA, USA, 9–11 October 2011, IEEE Computer Society, pp. 497–500 (2011)

10. Huang, W., Yu, J.X.: Investigating TSP heuristics for location-based services. Data Sci. Eng. **2**(1), 71–93 (2017)
11. Jabeur, L.B., Tamine, L., Boughanem, M.: Active microbloggers: Identifying influencers, leaders and discussers in microblogging networks. In: Calderón-Benavides L, González-Caro CN, Chávez E, Ziviani N (eds) String Processing and Information Retrieval-19th International Symposium, SPIRE 2012, Cartagena de Indias, Colombia, October 21–25, 2012. Proceedings, Springer, Lecture Notes in Computer Science, vol. 7608, pp. 111–117 (2012)
12. Kempe, D., Kleinberg, J.M., Tardos, É.: 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, pp. 137–146 (2003)
13. Korn, F., Muthukrishnan, S.: 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, pp. 201–212 (2000)
14. Kritter, J., Brévilliers, M., Lepagnot, J., Idoumghar, L.: On the optimal placement of cameras for surveillance and the underlying set cover problem. Appl. Soft. Comput. **74**, 133–153 (2019)
15. Lampos, V., Aletras, N., Preotiuc-Pietro, D., Cohn, T.: Predicting and characterising user impact on twitter. In: Bouma G., Parmentier Y., (eds) Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2014, April 26–30, 2014, Gothenburg, Sweden, The Association for Computer Linguistics, pp. 405–413. https://doi.org/10.3115/v1/e14-1043 (2014)
16. Li, D., Li, H., Wang, M., Cui, J.: k-collective influential facility placement over moving object. In: 20th IEEE International Conference on Mobile Data Management, MDM 2019, Hong Kong, SAR, China, June 10–13, 2019, pp. 191–200 (2019)
17. Li, G., Chen, S., Feng, J., Tan, K., Li, W.: Efficient location-aware influence maximization. In: International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22–27, 2014, pp. 87–98 (2014)
18. Li, Y., Bao, J., Li, Y., Wu, Y., Gong, Z., Zheng, Y.: 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, pp. 51:1–51:4 (2016)
19. Lin, X., Yuan, Y., Zhang, Q., Zhang, Y.: 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, pp. 86–95 (2007)
20. Mitra, S.: 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)
21. Mitra, S., Saraf, P., Bhattacharya, A.: TIPS: mining top-k locations to minimize user-inconvenience for trajectory-aware services. CoRR arXiv:1709.02343 (2017a)
22. Mitra, S., Saraf, P., Sharma, R., Bhattacharya, A., Ranu, S., Bhandari, H.: 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, pp 87–90 (2017b)
23. Peng, S., Yang, A., Cao, L., Yu, S., Xie, D.: Social influence modeling using information theory in mobile social networks. Inf. Sci. **379**, 146–159 (2017)
24. Sun, Y., Huang, J., Chen, Y., Zhang, R., Du, X.: 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, pp. 2154–2158 (2012)
25. Wang, M., Li, H., Cui, J., Deng, K., Bhowmick, S.S., Dong, Z.: PINOCCHIO: probabilistic influence-based location selection over moving objects. In: 33rd IEEE International Conference on Data Engineering, pp. 21–22 (2017)
26. Wang, S., Bao, Z., Culpepper, J.S., Sellis, T., Cong, G.: Reverse k nearest neighbor search over trajectories. IEEE Trans. Knowl. Data Eng. **30**(4), 757–771 (2018)
27. Wong, R.C., Özsu, M.T., Yu, P.S., Fu, A.W., Liu, L.: Efficient method for maximizing bichromatic reverse nearest neighbor. PVLDB **2**(1), 1126–1137 (2009)
28. Wu, T., Lin, J.: Solving the competitive discretionary service facility location problem. Eur. J. Oper. Res. **144**(2), 366–378 (2003)

29. Xia, T., Zhang, D., Kanoulas, E., Du, Y.: 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, pp. 946–957 (2005)
30. Xu, C., Gu, Y., Zimmermann, R., Lin, S., Yu, G.: Group location selection queries over uncertain objects. IEEE Trans. Knowl. Data Eng. **25**(12), 2796–2808 (2013)
31. Yan, D., Wong, R.C., Ng, W.: Efficient methods for finding influential locations with adaptive grids. In: Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011, Glasgow, UK, October 24–28, 2011, pp. 1475–1484 (2011)
32. Yiu, M.L., Dai, X., Mamoulis, N., Vaitis, M.: 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, pp. 1076–1085 (2007)
33. Zhang, D., Guo, L., Nie, L., Shao, J., Wu, S., Shen, H.T.: Targeted advertising in public transportation systems with quantitative evaluation. ACM Trans. Inf. Syst. **35**(3), 1–20 (2017)
34. Zhang, P., Bao, Z., Li, Y., Li, G., Zhang, Y., Peng, Z.: 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, pp. 2748–2757 (2018a)
35. Zhang, X., Rey, D., Waller, S.T.: Multitype recharge facility location for electric vehicles. Comp. Aid. Civ. Infrastruct Eng. **33**(11), 943–965 (2018b)
36. Zhou, Z., Wu, W., Li, X., Lee, M., Hsu, W.: Maxfirst for maxbrknn. In: Proceedings of the 27th International Conference on Data Engineering, ICDE 2011, April 11–16, 2011, Hannover, Germany, pp. 828–839 (2011)