PINOCCHIO: Probabilistic Influence-Based Location Selection over Moving Objects

Meng Wang, Hui Li, Jiangtao Cui, Ke Deng, Sourav S. Bhowmick, and Zhenhua Dong

Abstract—The *location selection* (LS) problem, which aims to mine the optimal location from a set of candidates to place a new facility such that a score (i.e., benefit or influence on some given objects) can be maximized, has drawn significant research attention in recent years. State-of-the-art LS techniques assume each object is static and can only be influenced by a single facility. However, in reality, objects (e.g., people, vehicles) are mobile and are influenced by multiple facilities, which prevents classical LS solutions from selecting accurate results. In this paper, we introduce a *generalized* LS problem called PRIME-LS which takes mobility and probability factors into consideration to address the aforementioned limitations. Specifically, given a set of candidate locations, PRIME-LS aims to mine the optimal location which can *influence* the most number of *moving objects*. Also, to address the problem we propose an efficient algorithm called PINOCCHIO that leverages two pruning rules based on a novel distance measure. These rules enable us to prune many inferior candidate locations prior to influence computation, paving the way to efficient and accurate solution. Furthermore, we extend PINOCCHIO (PINOCCHIO-VO) by incorporating two *optimization strategies* during candidate validation phase, which further reduce unnecessary computations. Experimental study over two real-world datasets demonstrates superiority of our framework in comparison to state-of-the-art LS techniques.

Index Terms—Moving objects, location selection, spatial database

1 Introduction

LOCATION selection (LS) problem has received considerable research attention due to the proliferation of GPS equipped mobile devices and location-based services (LBS) (e.g., *Gowalla*, *Foursquare*). Given a set of objects Ω with their positions and a set of candidate locations C, many existing LS techniques aim to mine a candidate location $c \in C$ such that c can *influence* the maximum number of objects [1]. Here, the *influence* of a candidate c is typically defined as the number of objects whose nearest neighbors is c [2]. Finding such optimal location from candidates to establish a new facility has a wide spectrum of applications such as marketing, urban planning, resource allocation, scientific research, etc. However, as classical LS assumes objects are static, some limitations appear in the application scenarios where objects are mobile.

1.1 Motivating Scenario

Consider the scenario where a company wishes to select an optimal location from a set of candidates to place a new outdoor advertising balloon such that it can be observed by as many potential customers as possible. To simplify the discussion, we assume a customer can observe an advertising balloon with a certain probability according to the Euclidean distance between the balloon and customer.[1] Intuitively, farther distance leads to lower probability. Moreover, the probability that a customer observes any advertising balloon is independent. That is, a customer may observe multiple balloons with different probabilities. Furthermore, many customers are not static as they have to travel from one point to another everyday. Typically, the mobility of a customer can be modeled as a set of positions [3]. Notably, it is possible that a customer may visit the nearest position from an advertising balloon occasionally, while appear frequently at farther positions. Consequently, whether a balloon is observed by her cannot be solely attributed to the nearest position as other positions may also play a role. Observe that at any position, the probability of a customer successfully observing an advertising balloon is independent. Hence, inspired by Influence Model [4], we define the probability for a moving object to observe an advertising balloon as cumulative probability, which is in fact the probability that a customer successfully observes the balloon in at least one of her positions. Consequently, whether an advertisement at a specific location attracts a customer can be reformulated as a problem that seeks to determine whether the cumulative probability on all positions is high enough.

It is not difficult to see there exist series of other potential applications similar to the aforementioned scenario. For

- *M. Wang and J. Cui are with the School of Computer Science and Technology, Xidian University, Xi'an 710071, China. E-mail: wamengit@sina.com, cuijt@xidian.edu.cn.*
- *H. Li is with the School of Cyber Engineering, Xidian University, Xi'an 710071, China. E-mail: hli@xidian.edu.cn.*
- *K. Deng is with the School of Computer Science and Information Technology, RMIT University, Melbourne, VIC 3001, Australia. E-mail: ke.deng@rmit.edu.au.*
- *S. S. Bhowmick is with the School of Computer Science and Engineering, Nanyang Technological University 639798, Singapore. E-mail: assourav@ntu.edu.sg.*
- *Z. Dong is with Huawei Noah's Ark Research Lab, Hongkong, China. E-mail: dongzhenhua@huawei.com.*

Manuscript received 26 Dec. 2015; revised 24 Apr. 2016; accepted 7 June 2016. Date of publication 13 June 2016; date of current version 3 Oct. 2016. Recommended for acceptance by R. Cheng.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2016.2580138

1. As we are studying a general problem that may also be used in domains other than outdoor advertisement, 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.).

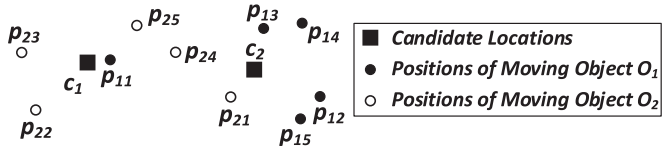


Fig. 1. Motivation example.

instance, an optimal location for a new retail shop or restaurant to be close to more mobile customers, a new monitoring station to track wild animals' migration, etc.

Consider an example in Fig. 1. For all positions of object O_1, the nearest candidate is c_1. In fact, nearest neighbor-based conventional LS techniques [2] will report c_1 influences O_1 but not c_2. However, the cumulative probability of O_1 being influenced by c_2 might be higher than c_1 as O_1 has four positions $(p_{12}, p_{13}, p_{14}, p_{15})$ near c_2. Similarly, the cumulative probability of O_1 being influenced by c_2 might be higher than that of O_2, even though O_2 has a position p_{21} that is a nearest neighbor of c_2 while her other positions are far away. *How can we select the optimal location that can potentially influence the largest number of objects given that they are mobile?*

1.2 Limitations of Classical Location Selection Techniques

Despite the significant progress made by state-of-the-art LS techniques [1], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], they are not effective in solving the aforementioned scenario due to at least one of the following drawbacks (detailed in Section 2).

- First, objects influenced by a facility are assumed to be static. However, objects in LBS are mobile and follow some mobility patterns [20]. Hence, considering only a single instead of a set of positions ignores the mobility of these objects, failing to provide a complete description of an object's activities.
- Second, the influence of a location on an object is considered binary. That is, an object is either influenced by a location or not, based on a distance metric between them [13]. However, the binary criterion may not be able to reflect the influence in real world, which is in fact a probabilistic event.
- Last but not the least, the strict assumption [12] that an object is influenced by only one facility (e.g., the nearest neighbor), ignores others which might also exhibit influence. Nevertheless, it is rational that an object can be influenced by multiple facilities simultaneously. Hence, excluding other facilities in selecting location may reduce the effectiveness of existing approaches.

In this paper, we study and provide solution to a more generalized LS problem called PRIME-LS (**PR**obabilistic **I**nfluence-based **M**obility-awar**E** **L**ocation **S**election), which takes mobility and probability factors into consideration to address the aforementioned limitations of traditional LS techniques.

1.3 Overview and Contributions

Informally, given a set of candidate locations C, a set of moving objects Ω, each of which consists of a set of positions O, a certain distance-based influence probability function PF and a user-specified influence probability threshold τ, the goal of PRIME-LS is to mine the optimal location which can influence the most number of moving objects, each of which may be influenced by multiple candidates simultaneously. In particular, we use the term *influence* to cover all instances, such as attraction, impact, signal intensity, coverage extent, visibility, etc., in diverse practical applications. By varying the probability functions (PF), PRIME-LS can be applied to various moving objects with distinct mobility patterns (e.g., distance-based check-in pattern [21], [22], feature pattern [23]), and even degenerate to traditional LS problems. For instance, it degenerates to *MaxBRNN* [1] if only NN locations are considered, or aggregate RNN [24] if the PF is associated with a distance metric. Empirical studies demonstrate that our proposed PRIME-LS solution can significantly improve the effectiveness of the mined optimal location compared to classical LS by around 10-35 percent.

A straightforward solution to the PRIME-LS problem is to exhaustively test all the object-candidate pairs. However, this naïve strategy is computationally expensive. This motivates us to explore effective pruning techniques to reduce the cost of the solution. Specifically, as a moving object's positions may scatter across a wide region, any solution to the PRIME-LS problem needs to tackle three grand challenges. First, moving objects' activity regions are highly overlapping in nature. Since majority of state-of-the-art pruning strategies for LS problem assume that activity regions do not overlap [5], [25], they cannot be adopted effectively. Furthermore, performances of these techniques degrade significantly when moving areas of objects largely overlap [5]. Second, candidates may fall into the minimum bounding rectangle (MBR) of an object's activity region, which is in fact an overlap, leading to the same challenge as above. As a result, neither *minDist* nor *minExistDist* [1] between a candidate and an object is available. Thus the corresponding pruning techniques [1], [5], [6] are not applicable as well. Third, in LS over uncertain objects, only one position contributes to the influence. In contrast, PRIME-LS needs to evaluate the influence of multiple positions to select the most influential candidate, which is much more complicated. Large number of positions of each moving object therefore incur substantial overhead for the validation of a candidate.

In this paper, we present a novel algorithm called PINOCCHIO (**P**robabilistic **IN**fluence-based L**OC**ation Sele**C**tion Tec**H**nique over Mov**I**ng **O**bjects) to address the PRIME-LS problem by employing two pruning rules based on a novel distance measure called *minMaxRadius*. Intuitively, *minMaxRadius* is a distance defined between a candidate and positions of a moving object. It enables us to quantify the cumulative influence that a candidate location exhibits on the moving object. The novel pruning rules therefore can largely prune unnecessary candidates. Furthermore, we extend PINOCCHIO to a more efficient solution (referred to as PINOCCHIO-VO) by optimizing its validation step based on upper-lower bounding and early stopping strategies. Additionally, we utilize R-tree [26] for candidates to further

TABLE 1
Notations

Notation	*Definition*
Ω	a set of moving objects
C	a set of candidate locations
r, m	cardinalities of Ω and C, respectively
O, O_k	a moving object with multiple positions
p, p_i	a position of an object
c, c_j	a candidate location
n, n_k	number of positions in O, O_k
$dist(p_1, p_2)$	distance between positions p_1 and p_2
$PF()$	probability function
$Pr_c(p)$, $Pr_c(O)$	cumulative influence probability of p or O being influenced by c
$Pr_c^{n-n'}(O)$	partial non-influence probability of $n - n'$ positions of O
τ	probability threshold
$inf(c)$	influence of candidate c
$MBR(O)$	minimal bounding rectangle of O

reduce the response time.[2] We theoretically and empirically show that our framework can avoid nearly 67 percent unnecessary position validation by adopting our pruning techniques. Along with our proposed optimization strategies, it is able to produce correct answer and significantly improve the efficiency by orders of magnitude. Interestingly, our experimental study also reveals that if we expect a certain number of objects to be influenced, the mined optimal locations do not vary much no matter how we select the threshold τ and number of positions of an object. In summary, our contributions in this paper are outlined as follows.

- To the best of our knowledge, this is the first effort to formalize the problem of probabilistic influence-based location selection over moving objects (i.e., PRIME-LS). Compared to existing work, PRIME-LS takes into account the influence of multiple positions, and each moving object can be influenced by multiple candidates. This improves the effectiveness of mined optimal location significantly.
- We develop two novel algorithms, PINOCCHIO and PINOCCHIO-VO, addressing PRIME-LS problem efficiently and accurately. A novel distance measure called *minMaxRadius* is designed to evaluate the probabilistic distance between the positions of a moving object and a candidate facility. Two new pruning rules based on *minMaxRadius* as well as corresponding optimization methods are designed to reduce the computational complexity.
- We conduct empirical studies on real-world datasets to demonstrate the effectiveness, efficiency, and superiority of our proposed algorithms compared to classical LS techniques.

The rest of this paper is organized as follows. We give a brief overview of related work in the next section. In Section 3, we formally define the PRIME-LS problem. We present the PINOCCHIO algorithm in Section 4 that utilizes the pruning rules based on *minMaxRadius* measure. Section 5 presents the PINOCCHIO-VO algorithm. Experimental results are reported in Section 6. Finally, the last section concludes this paper. Table 1 lists a set of frequently-used notations in this paper.

2. Other variations of R-tree and hierarchical spatial data structures can also be applied.

2 Related Work

In this section, we discuss related efforts in location selection and moving objects data management.

2.1 Location Selection Problem

There have been increasing research efforts in LS problem under various scenarios [1], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19]. These efforts can be broadly classified into two groups, namely, *maximum influence-based* (MAX-INF) and *minimum distance-based* (MIN-DIST).

MAX-INF. The term *influence* has been widely used in literature. For instance, in social influence study, it refers to the fact that one's emotions, opinions, or behaviors are affected by others. In LS problem, *influence* refers to the number (*resp.*, probability) of persons (*resp.*, objects) that may visit (*resp.*, be affected) if some facility is placed at a particular location.

Specifically, MAX-INF-based LS problem aims to find a location c with the maximum influence, defined as the number of objects whose nearest neighbor is c. Hence, MAX-INF is closely related to BRNN, which was introduced in [2]. Xia et al. [1] defined the influence of a location as the total weight of its RNNS (reverse nearest neighbors) and developed a distance metric called *minExistDNN* to prune search space using R-tree. In [6], the authors studied further with existing facilities. Two pruning algorithms are exploited to estimate the distance between a customer and its nearest facility utilizing the concept of *influential region*. Sun et al. [9] validated all clients and their corresponding BRNN sets and proposed three pruning techniques to tighten the search space. However, these efforts suffer from all the three limitations mentioned in Section 1.2.

Yan et al. [12] relaxed the assumption from NN facility to $(1 + \alpha) \cdot$ NN, where α is a user-specified value. By employing a grid-based algorithm, the approximate candidates are mined. Wong et al. [16] studied a similar problem, called *MaxBRkNN*, in which all the *k*NN facilities exhibit influence on objects. The authors took advantage of *region-to-point transformation* to tighten the continuous search space. Zhou et al. [17] proposed *MaxFirst* to solve *MaxBRkNN*. The solution partitions the space into quadrants iteratively and prunes the unpromising candidates using upper and lower bounds. Note that these studies suffer from the first two limitations in Section 1.2.

Choi et al. [18] studied the *MaxRS* problem, which is another variant of LS. Specifically, *MaxRS* finds an optimal region r with a given size, which maximizes the total weight of all the objects covered by r. The authors proposed *ExactMaxRS* and *ApproxMaxCRS* algorithms to handle the cases of rectangular and circular regions, respectively. They further proposed $(1 - \epsilon)$-*approximate MaxRS* algorithm to significantly improve its efficiency in [19]. Recently, Xu et al. [11] proposed *group location selection* (*GLS*) problem to find the minimum number of multiple locations with influence

regions, such that all the objects can be covered. However, in contrast to our problem setting, these efforts assume that the objects are static.

Shang et al. [8] extended RNN to *reverse path nearest neighbor* (*R-PNN*) in road networks. Dijkstra expansion and triangle inequality are adopted to estimate lower and upper bounds for candidates pruning. As the pruning strategies are based on NN and the effects of other points are ignored, it suffers from the third limitation in Section 1.2.

Cheema et al. [5] studied the problem of *probabilistic reverse nearest neighbor* (*PRNN*) based on the possible worlds semantics. They developed *Normalized Antipodal Half-Space* to solve the *PRNN* on multidimensional uncertain data. Following possible world semantics, Zhan et al. [13] aims to find top-k most influential facilities over uncertain objects. Zheng et al. [15] proposed partition-based algorithm and many pruning techniques to solve a similar problem. Although these studies model an object as "multiple position instances", they focus on uncertain data, which is inherently different from mobility data due to the cause of uncertainty [5]. Besides, in a possible world, each object is still represented by a single position and is limited to be influenced by only one facility based on NN metric. In contrast, in our problem setting, majority of moving objects may visit multiple locations, thus are influenced by these locations. Hence, these approaches cannot be easily adapted to address the PRIME-LS problem.

Yiu et al. [27] studied another kind of LS problem, in which they focus on the total qualities of surrounding facilities of query locations. By utilizing user-specific range and NN to restrict feature objects, they presented *Branch-and-Bound* and *Feature Join* algorithms to solve the problem. They further extended this problem with distance-weighted quality [23]. The authors used Gaussian density function to depict the decremental effect on distance. As these efforts are orthogonal to our problem setting, they cannot be applied to address the PRIME-LS problem.

MIN-DIST. MIN-DIST-based LS problem finds the optimal location to minimize the aggregate distance between locations and objects. Zhang et al. [14] proposed a method to find the MIN-DIST optimal-location for weighted objects in a given spatial region Q. Qi et al. [7] proposed an algorithm to find a location for new facility so that the average distance between an object and the corresponding nearest facility is minimized. Tang et al. [10] studied a converse problem to find *k-Nearest Neighboring Trajectories* (*k-NNT*) with the minimum distance to a set of given points. However, these MIN-DIST-based techniques suffer from all the three limitations discussed in Section 1.2. Hence, they are orthogonal to our problem setup.

2.2 Querying Moving Objects

There are a large body of work related to queries over moving objects, which are commonly represented by sampled positions on their trajectories. NN queries over moving objects [28] find the closest trajectory to a query trajectory or point at a particular timestamp or in a certain interval. On the other hand, *Continuous Aggregate* NN [29] aims to continuously retrieve a moving object whose aggregate distance to all facilities is the smallest. *Intersection* join over moving objects [30] finds every object pair at every timestamp, where the corresponding MBRS intersect. Moreover, in *continuous* RNN [31], both query points and objects are movable, the aim is to find the reverse NN of a query point over time. *Continuous maximal* RNN [32] is most related to LS, which continuously finds the optimal network location at every timestamp based on RNN. In a sense, all these efforts can be regarded as multiple queries over static objects at different time instances. Hence, they cannot be adopted effectively to address PRIME-LS.

3 Problem Definition

In this section, we formally define the PRIME-LS problem addressed in this paper. We begin by introducing some terminology that are necessary for the definition of the problem.

3.1 Terminology

A *position* p is a point in a two-dimensional Euclidian space, denoted by its geographical coordinates (i.e., latitude and longitude). Given any two positions p_1 and p_2, the distance between them is denoted by $dist(p_1, p_2)$. A moving object can be described either *discretely* or *continuously*. In the *discrete* case, the raw positions of a moving object are typically expressed as $\langle latitude, longitude \rangle$. In the latter case, any continuous moving object also can be discretized as a series of positions by sampling using the same time interval.[3] In this paper, we use a set of discrete positions $O = \{p_1, p_2, \ldots, p_n\}$ to denote a moving object O.[4] Moreover, we model each moving object O by an MBR [26], which encloses all its positions to represent her activity region and is denoted by $MBR(O)$.

We denote candidate locations of a new facility to deploy as $C = \{c_1, c_2, \ldots, c_m\}$. The probability that an object at position p is influenced by a facility at location $c \in C$ is denoted by $Pr_c(p)$. We use a pre-defined distance-based probability function PF to depict the behavior pattern of influence. Since the probability of a customer for a point-of-interest is inversely proportional to geographic distance [21], as remarked earlier (Section 1.1), we assume PF is monotonically decreasing to distance and the influence probability only depends on the distance between a facility and an object. That is, the influence probability $Pr_c(p)$ can be computed as $Pr_c(p) = PF(dist(c, p))$. For a moving object $O = \{p_1, p_2, \ldots, p_n\}$, the probability that O is influenced by candidate c at any position p_i ($i \in [1, n]$) is independent of those at other positions, i.e., $Pr_c(p_i) = PF(dist(c, p_i))$. O is influenced by c if and only if there is at least a position p_i of O that is influenced by c. The probability that O is influenced by c, namely *cumulative probability*, can be defined as follows.

Definition 1. *Given a candidate location c and a moving object O with n positions $\{p_1, p_2, \ldots, p_n\}$, the* ***cumulative influence***

3. To simplify our discussion, we assume all devices exhibit the same sampling rate.

4. Notably, for the effectiveness of problem result, we recommend a single type of mobility dataset (i.e., either discrete or discretized continuous dataset). Mixed datasets (e.g., mixing discrete check-ins with discretized trajectories) would lead to uncertain results due to the inherent diversity between data sources.

probability *of O being influenced by c, denoted by $Pr_c(O)$, is defined as: $Pr_c(O) = 1 - \prod_{i=1}^{n}(1 - Pr_c(p_i))$.*

Recall that in conventional LS, whether a candidate location $c \in C$ influences an object O can be directly determined by BRNN. That is, c can influence O if and only if the nearest neighbor of O in C is c. However, this assumption may not hold when an object consists of multiple positions (Fig. 1). Hence, we redefine the concept of *influence* using $Pr_c(O)$.

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

$Pr_c(O)$ measures the extent that O is influenced by c. Given $\Omega = \{O_1, O_2, \ldots, O_r\}$ and a user-specified probability threshold τ, we can evaluate $inf(c_j)$ ($c_j \in C$ and $j \in \{1, \ldots, m\}$) for every candidate location. The candidate with the maximum influence can be considered as the optimal location to deploy a new facility as it can influence the most potential customers.

3.2 The PRIME-LS Problem

We are now ready to define the location selection problem addressed in this paper.

Definition 3. *Given a set of candidate locations C, a set of moving objects Ω, a certain distance-based probability function PF and a user-specified influence threshold τ, the **PR**obabilistic **I**nfluence-based **M**obility-awar**E** **L**ocation **S**election (PRIME-LS) problem aims to mine the optimal candidate $c \in C$ such that $\forall c' \in C - \{c\}, inf(c) \geq inf(c')$.*

Example 1. Reconsider Fig. 1 with moving objects O_1, O_2 and candidates c_1, c_2. 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 0.8, c_1 only influences O_2 but not O_1, which even has the NN position p_{11}. Hence, $inf(c_1) = 1$. On the other hand, if $Pr_{c_2}(O_1) = 0.86$ and $Pr_{c_2}(O_2) = 0.83$, then c_2 obviously influences both O_1 and O_2. That is, $inf(c_2) = 2$. As $inf(c_2) > inf(c_1)$, c_2 is the selected location for PRIME-LS problem.

As remarked earlier, a naïve strategy to address the PRIME-LS problem is to exhaustively test all the object and candidate pairs. Unfortunately, it is computationally expensive (see Section 6). In the next section, we present a more efficient algorithm called PINOCCHIO to address this problem.

4 INFLUENCE-BASED LOCATION SELECTION

Intuitively, our PINOCCHIO algorithm comprises of two key phases, namely, *pruning* and *validation*. In the *pruning phase*, we propose a novel pruning strategy to filter out inferior candidate locations. In the *validation phase*, we validate the remaining candidates to select the final result.

As discussed in Section 2, existing LS efforts, which are based on static objects, solve inherently different problems from mobility-based PRIME-LS. Consequently, the widely-used NN-based pruning techniques cannot be applied here. Hence, a new pruning strategy is paramount for efficiently solving the PRIME-LS problem. In this section, we first design a novel measure called *minMaxRadius* to quantify the cumulative influence that a candidate location exhibits on a moving object with multiple positions. Based on that, we then design two pruning rules, namely *influence arcs rule* and *non-influence boundary rule*. The former utilizes *minMaxRadius* to determine a small area for each moving object such that any candidate location falling into the area definitely influences that object. The latter utilizes *minMaxRadius* to find a relatively larger area for each moving object such that any candidate outside the area definitely fails to influence that object. These rules enable us to avoid the computation and validation of cumulative influence for many object-candidate pairs. Finally, we present the PINOCCHIO algorithm to solve the PRIME-LS problem.

4.1 minMaxRadius

In this section, we introduce the novel *minMaxRadius* measure as follows. First, we present a lemma on the condition a candidate location c must satisfy to influence a moving object considering only one of its positions. Afterwards, we define a probabilistic model for the remaining positions. We then deduce the measure with regards to two positions of an object. Finally, the measure distance is derived by extending it to all positions.

Lemma 1. *Given a moving object $O = \{p_1\}$ (with only one position), a candidate location c and probability threshold τ, c can influence O if and only if $dist(c, p_1) \leq PF^{-1}(\tau)$.*

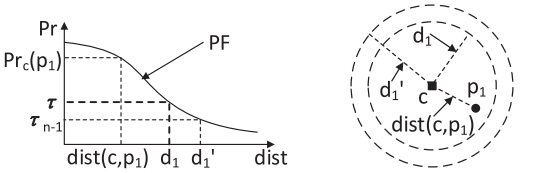
Proof. As $O = \{p_1\}$, $Pr_c(O) = 1 - (1 - Pr_c(p_1)) = Pr_c(p_1) = PF(dist(c, p_1))$. As discussed before, PF is monotonically decreasing with respect to the distance. That is, $Pr_c(p_1) = PF(dist(c, p_1)) \geq PF(PF^{-1}(\tau)) = \tau$. Hence, based on Definition 2, c can influence O. □

Definition 4. *Given a candidate location c and a moving object O with positions $\{p_1, \ldots, p_n\}$, which are ordered by distance from c, the **partial non-influence probability** of O to be not influenced by c, denoted by $Pr_c^{n-n'}(O)$, is computed as $Pr_c^{n-n'}(O) = \prod_{i=n'+1}^{n}(1 - Pr_c(p_i))$, where $n' \in N$, $n' < n$. Note that as there is no item in $Pr_c^{n-n}(O)$, we set it as 1.*

We now consider other positions of O. Let $Pr_c^{n-1}(O)$ be the partial non-influence probability of positions except p_1. Assuming moving object O is influenced by candidate c, then $Pr_c(O)$ should be no less than τ. Thus,

$$\begin{aligned} 1 - (1 - Pr_c(p_1)) \cdot Pr_c^{n-1}(O) &\geq \tau \\ Pr_c(p_1) &\geq 1 - \frac{1 - \tau}{Pr_c^{n-1}(O)}. \end{aligned}$$

Let $\tau_{n-1} = 1 - (1 - \tau)/Pr_c^{n-1}(O)$, then the inequality above can be rewritten as $Pr_c(p_1) \geq \tau_{n-1}$. Hence, the relationship between $Pr_c(O)$ and τ can be regarded as that between $Pr_c(p_1)$ and τ_{n-1}.



(a) Probability relationship for one position (b) Circle for one position

Fig. 2. Illustration of *minMaxRadius*.

As shown in Fig. 2a, there exists a distance d_1' such that $PF(d_1')$ equals to τ_{n-1}. As $\tau > \tau_{n-1}$ can be easily derived from the definition of τ_{n-1}, then $d_1' > d_1 = PF^{-1}(\tau)$. As shown in Fig. 2b, if and only if position p_1 of O lies in a circle that centers at c with radius d_1', $Pr_c(O) > \tau$. Thus, c influences O. Here, d_1' will decrease when $Pr_c^{n-1}(O)$ increases. If $Pr_c^{n-1}(O)$ is close to 1, d_1' is close to d_1. It implies that the farther a position is away from a candidate, the less it will contribute to the cumulative influence probability.

Now let us consider two positions p_1 and p_2 of O, where $dist(c, p_1) \leq dist(c, p_2)$. Ignoring $\{p_3, \ldots, p_n\}$, suppose c influences O. Then $1 - (1 - Pr_c(p_2))^2$, which is the lower bound of $Pr_c(O)$, is greater than τ, namely $Pr_c(p_2) \geq 1 - (1 - \tau)^{\frac{1}{2}}$. We take $\{p_3, \ldots, p_n\}$ into consideration again. Similar to the situation of one position, there exists a distance d_2' such that $d_2' > d_2 = PF^{-1}(1 - (1 - \tau)^{\frac{1}{2}}) > d_1$. Thus, if and only if p_1 and p_2 of O lie in a circle that centers at c with radius d_2', c influences O.

If we extend the situation to all n positions of O, i.e., $Pr_c^{n-n}(O) = 1$, the distance d_n' equals to d_n. This leads to the definition of the *minMaxRadius* measure.

Definition 5. *Given a moving object O with n positions and a probability function PF, the* ***minMaxRadius*** *of O based on a user-specified probability threshold τ, is defined as:* $minMaxRadius(\tau, n) = PF^{-1}(1 - (1 - \tau)^{\frac{1}{n}})$.

In the above definition, the term *min* indicates that it is the shortest radius and the term *Max* signifies the farthest distance between all positions and the center c. Notice that if n is fixed, $minMaxRadius(\tau, n)$ grows when τ decreases; if τ is fixed, $minMaxRadius(\tau, n)$ grows as n increases.

Theorem 1. *Given a moving object $O = \{p_1, \ldots, p_n\}$, if all n positions lie in a circle that centers at candidate location c with radius $minMaxRadius(\tau, n)$, c must influence O with a probability of at least τ.*

Proof. As $\{dist(c, p_1), \ldots, dist(c, p_n)\}$ are sorted in ascending order, $Pr_c(p_1) = PF(dist(c, p_1)) \geq \ldots \geq Pr_c(p_n) = PF(dist(c, p_n))$. Obviously, $1 - (1 - Pr_c(P_n))^n \leq Pr_c(O) \leq 1 - (1 - Pr_c(P_1))^n$. If $dist(c, p_n) \leq minMaxRadius(\tau, n) = PF^{-1}(1 - (1 - \tau)^{\frac{1}{n}})$, $Pr_c(p_n) \geq 1 - (1 - \tau)^{\frac{1}{n}}$. Hence, $1 - (1 - Pr_c(P_n))^n \geq \tau$. As a result, $Pr_c(O) \geq \tau$. That is, c can influence O whose positions lie in a circle that centers at c with radius $minMaxRadius(\tau, n)$. □

Theorem 2. *Given a moving object $O = \{p_1, \ldots, p_n\}$ and a threshold τ, if all n positions lie outside a circle that centers at candidate location c with radius $minMaxRadius(\tau, n)$, c must not influence O.*

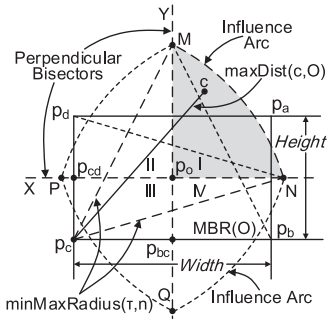


Fig. 3. Influence arcs pruning rule.

Proof. Similar to the proof of Theorem 1, if $dist(c, p_1) > minMaxRadius(\tau, n) = PF^{-1}(1 - (1 - \tau)^{\frac{1}{n}})$, $Pr_c(p_1) < 1 - (1 - \tau)^{\frac{1}{n}}$. Hence, $1 - (1 - Pr_c(P_1))^n < \tau$. As a result, $Pr_c(O) < \tau$. That is, c cannot influence O whose positions lie outside a circle that centers at c with radius $minMaxRadius(\tau, n)$. □

4.2 Pruning Rules

Based on Definition 5, given a certain τ, *minMaxRadius* varies with n. Let N denote the number of different n for all objects. For each candidate c, N range queries are required in order to determine using *minMaxRadius* whether these objects are influenced or not by c. However, N is large in reality [22]. Processing so many range queries for a large number of candidates is impractical, which motivates us to design two *minMaxRadius*-based pruning rules to reduce the number of candidates based on each moving object.

In this paper, we use two geometry metrics, *minDist* and *maxDist* [33], to indicate the lower and upper bounds of the distance between a candidate location and any possible position in a moving object.

4.2.1 Influence Arcs Pruning Rule

According to Theorem 1, if the farthest distance between all n positions and c is not larger than *minMaxRadius*, a moving object O can be influenced by candidate location c. However, we cannot predetermine the farthest distance as both range query and scan on all n positions are costly. Hence, to avoid probing the exact farthest distance between all positions and c, we employ $maxDist$ to obtain an upper bound on the distance. We adopt Cartesian coordinate[5] to facilitate the discussion in the sequel.

Definition 6. *Given a probability threshold τ and the $MBR(O)$ of O with n positions, let the center of $MBR(O)$ be the origin of Cartesian coordinate, where two axes are parallel with the sides of $MBR(O)$. For any corner of $MBR(O)$, draw an arc centering at the corner with radius $minMaxRadius(\tau, n)$, and the arc between two intersection points with the axes in the opposite direction of the corner is defined as an* ***influence arc****.*

For instance, p_a, p_b, p_c, p_d in Fig. 3 are the corners and p_o is the center of $MBR(O)$. Let p_o be the origin. Through p_o, let X-axis and Y-axis parallel with segments $\overline{p_b p_c}$ and $\overline{p_a p_b}$, respectively. The two axes are the perpendicular bisectors

5. In this paper, the distance is actually computed based on Geographic spherical distance.

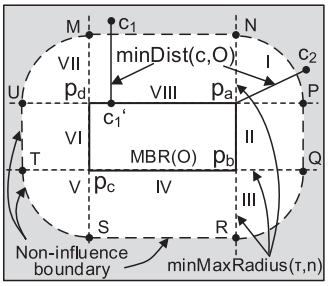


Fig. 4. Non-influence boundary pruning rule.

of $MBR(O)$, which divide the plane into four quadrants, I, ..., IV. With endpoints M and N on both axes, arc $\overset{\frown}{MN}$ is the influence arc of corner p_c, where the lengths of segments $\overline{Mp_c}$ and $\overline{Np_c}$ are therefore $minMaxRadius(\tau, n)$.

Lemma 2 (**Influence Arcs Pruning Rule**). *Any candidate location lying in the closed region bounded by the four influence arcs of the corners of $MBR(O)$ can influence O.*

Proof: As shown in Fig. 3, Y intersects $\overline{p_b p_c}$ at p_{bc}, the mid point of $\overline{p_b p_c}$. Hence, triangles $\Delta Mp_{bc}p_b$ and $\Delta Mp_{bc}p_c$ are congruent, and $\overline{Mp_b} = \overline{Mp_c} = minMaxRadius(\tau, n)$. That is, M is an endpoint of the influence arc of corner p_b. Similarly, X intersects $\overline{p_c p_d}$ at p_{cd}, the mid point of $\overline{p_c p_d}$. Thus, N is an endpoint of the influence arc of corner p_d. Likewise, adjacent points M, N, P, Q are pairwise the endpoints of influence arcs. Hence, the region bounded by influence arcs $\overset{\frown}{MN}$, $\overset{\frown}{NQ}$, $\overset{\frown}{QP}$ and $\overset{\frown}{PM}$ is closed.

On the other hand, as the $maxDist$ point of an MBR from any point p is defined as the farthest corner of the MBR away from p. As shown in Fig. 3, for a candidate inside any quadrant, its $maxDist$ point is the corner of $MBR(O)$ in the diagonal quadrant. Suppose c is in quadrant I, the $maxDist$ point is corner p_c. If c lies in the region bounded by $\overline{Mp_o}$, $\overline{Np_o}$ and influence arc $\overset{\frown}{MN}$, namely the shaded region in Fig. 3, the $maxDist$ between any position of O and c is no farther than $minMaxRadius(\tau, n)$. That is, c can influence O. Candidates in other quadrants exhibit the same property. □

4.2.2 Non-Influence Boundary Pruning Rule

By means of influence arcs, we can prune the candidates which influence a moving object definitively. We now discuss another rule that can prune candidates that cannot influence a moving object.

Definition 7. *Given a probability threshold τ and the $MBR(O)$ of a moving object O with n positions, draw quarter circle arcs centering at each corner of $MBR(O)$ with radius $minMaxRadius(\tau, n)$. For each arc, it intersects the extension lines of the adjacent sides of the corresponding corner. The closed curve formed by connecting the endpoints of adjacent arcs with line segments is defined as the* ***non-influence boundary*** *of moving object O.*

In Fig. 4, $p_a, \ldots, p_d$ are the corners of $MBR(O)$. Draw the quarter circle arc centering at p_a with radius $minMaxRadius(\tau, n)$, intersecting the extension lines of $\overline{p_b p_a}$ and $\overline{p_d p_a}$ at N and P. Similarly, draw $\overset{\frown}{QR}$, $\overset{\frown}{ST}$ and $\overset{\frown}{UM}$ centering at p_b, p_c and p_d with radius $minMaxRadius(\tau, n)$, respectively. Connect P and Q, R and S, T and U, M and N pairwise, the non-influence boundary of $MBR(O)$ is formed.

Lemma 3 (**Non-Influence Boundary Pruning Rule**). *Any candidate location that lies out of the region bounded by non-influence boundary of $MBR(O)$ cannot influence O.*

Proof. According to Definition 7, except $MBR(O)$, all the extension lines divide the space into areas I to VIII, as shown in Fig. 4. Based on the $minDist$ definition, for any candidate in areas I, III, V, VII, its $minDist$ point is the corresponding corner of the quarter circle arc (e.g., c_2 and corner p_a). In addition, for any candidate in areas II, IV, VI, VIII, the $minDist$ point is the perpendicular point of the candidate on the side of $MBR(O)$ in the area (e.g., c_1 and c_1'). For any point on the non-influence boundary, it is $minMaxRadius(\tau, n)$ the $minDist$ between the point and $MBR(O)$. Hence, if a candidate c locates out of the non-influence boundary region (i.e., inside the shaded region in Fig. 4), its $minDist$ is farther than $minMaxRadius(\tau, n)$. According to Theorem 2, c cannot influence O. □

Remark. If an $MBR(O)$ degenerates to a point (i.e., a moving object can be represented by a single position), the $minMaxRadius$ degenerates to *maxDist* or *minDist*. Hence, PRIME-LS degenerates to the conventional LS problem.

4.3 The PINOCCHIO Algorithm

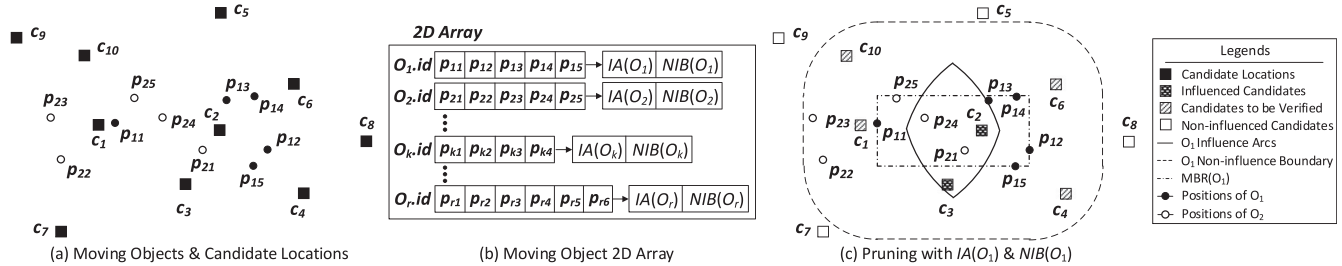
We now present algorithm PINOCCHIO to solve PRIME-LS by leveraging the aforementioned pruning rules. The key idea is as follows. For each moving object, we first calculate its $minMaxRadius$, based on which the corresponding influence arcs (IA) and non-influence boundary (NIB) can be obtained. With the help of IA, we identify and find the candidates that influence the object. For the remnant candidates, NIB is used to exclude the candidates that cannot influence the object. Lastly, the remnant candidates are verified using Definition 2.

Algorithm 1. Moving Objects 2D Array Initialization.

Input: moving objects set $\Omega = \{O_1, \ldots, O_r\}$, each object $O_k = \{p_1, \ldots, p_{n_k}\}$ ($k \in [1, r]$), PF and τ
Output: the moving object 2D array A_{2D}

```
1  HM = ∅, A_2D = ∅;
2  foreach O_k ∈ Ω do
3      if key n_k is found in HM then
4          retrieve minMaxRadius(τ, n_k);
5      else
6          compute minMaxRadius(τ, n_k);
7          insert pair ⟨n_k, minMaxRadius(τ, n_k)⟩ to HM;
8      initialize A_1D(O_k) = {p_1, ..., p_{n_k}};
9      compute IA(O_k) and NIB(O_k) based on MBR(O_k);
10     insert tuple ⟨A_1D(O_k), IA(O_k), NIB(O_k)⟩ to A_2D;
11 return A_2D;
```

Recall that moving objects' activity areas highly overlap with each other. For instance, in our experimental datasets a moving object covers nearly 55 percent of each dimension



(a) Moving Objects & Candidate Locations (b) Moving Object 2D Array (c) Pruning with $IA(O_1)$ & $NIB(O_1)$

Fig. 5. An example.

(the entire longitude and latitude cover 39.22 and 27.03 km, and on average each object covers 22.51 and 14.99 km, respectively). Moreover, we have to consider the cumulative influence probability of all positions of each object. Suppose we index the MBRs of moving objects' activity areas using an R-tree. Then in all tree levels the MBRs of intermediate nodes will also overlap significantly, which means the group-wise strategy is unavailable, and nearly every leaf still needs to be explored. This prevents us from applying such hierarchical data structures (e.g., R-tree-like or grid-like) as they fail to guarantee efficiency, besides extra construction time. Therefore, we present a *moving object two-dimensional (2D) array*, denoted by A_{2D}, for better performance.

A_{2D} contains multiple one-dimensional (1D) arrays, denoted as A_{1D}, to store positions of each object. As each object may have different number of positions, the 1D arrays are of different lengths. Given threshold τ and probability function PF, $minMaxRadius(\tau, n)$ of objects are different as well. Hence, to reduce the computation of each $minMaxRadius$, we store the values in a Hash-Map HM with n as keys. For pruning, each item of A_{2D} is associated with the corresponding IA and NIB data. Algorithm 1 outlines the procedure for initializing this A_{2D}. Note that we compute the influence arcs $IA(O_k)$ and the non-influence boundary $NIB(O_k)$ based on both $MBR(O_k)$ and $minMaxRadius(\tau, n)$ (Line 9). A tuple $\langle A_{1D}(O_k), IA(O_k), NIB(O_k)\rangle$ for O_k is built and inserted into A_{2D} with its ID $O_k.id$ (Line 10). Inspired by [7], we use the MBR of NIB (i.e., a simple rectangle to approximate NIB) to prune candidates in a more efficient way.

On the other hand, an R-tree is created to manage candidate locations to improve efficiency. Each leaf node of the R-tree contains the influence of the corresponding candidate.

Algorithm 2 outlines the PINOCCHIO algorithm. First, we initialize A_{2D} using Algorithm 1 (Line 1). We create an R-tree for the candidate locations, and initialize the influence $inf(c_j)$ of each candidate c_j (Lines 2-3). Next, for each object tuple in A_{2D}, we first utilize Lemma 2 to retrieve $C' \subseteq C$ candidates by the range query of $IA(O_k)$ on C R-tree. Each candidate in C' influences O_k, and thus its $inf(c_j)$ is increased (Lines 5-8). Similarly, C'' candidates, each of which is inside $NIB(O_k)$ but not in $IA(O_k)$, are retrieved using Lemma 3. As a result, C'' is the set of remnant candidates which needs to be verified (Line 9). In the validation phase, for each $c_j \in C''$, we perform a sequential scan over all positions in O_k (Lines 10-15) and finally return the candidate with the maximum influence.

Algorithm 2. The PINOCCHIO Algorithm

Input: moving objects set $\Omega = \{O_1, \ldots, O_r\}$, each object $O_k = \{p_1, \ldots, p_{n_k}\}$ ($k \in [1, r]$), candidate locations set $C = \{c_1, \ldots, c_m\}$, PF and τ
Output: the candidate location with maximum influence

1 initialize A_{2D} with Ω, PF and τ;
2 initialize C R-tree;
3 **foreach** c_j *leaf node in* C *R-tree* **do**
4 initialize $inf(c_j) = 0$;
5 **foreach** $A_{1D}(O_k) \in A_{2D}$ **do**
6 retrieve $C' \subseteq C$ in $IA(O_k)$;
7 **foreach** $c_j \in C'$ **do**
8 increase $inf(c_j)$ by 1;
9 retrieve $C'' \subseteq C$ inside $NIB(O_k)$ but not in $IA(O_k)$;
10 **foreach** $c_j \in C''$ **do**
11 **foreach** $p_i \in O_k$ **do**
12 $Pr_{c_j}(p_i) = PF(dist(c_j, p_i))$;
13 compute $Pr_{c_j}(O_k) = 1 - \prod_{i=1}^{n_k}(1 - Pr_{c_j}(p_i))$;
14 **if** $Pr_{c_j}(O_k) \geq \tau$ **then**
15 increase $inf(c_j)$ by 1;
16 **return** c_j *with the maximum* $inf(c_j)$;

Example 2. Consider Fig. 5a with objects O_1, O_2 and candidates c_1 to c_{10}. We create an A_{2D} to store all objects, each of which consists of a 1D array of positions with IA and NIB information. In Fig. 5b, the 1D array lengths of O_1 and O_2 are the same but differ from O_r or O_k. Take O_1 as an example in Fig. 5c. Using $IA(O_1)$ and $NIB(O_1)$, we identify that c_2, c_3 influence O_1 but c_5, c_7, c_8, c_9 do not. The remnant candidates c_1, c_4, c_6, c_{10} need further validation.

Theorem 3. *The time complexity of Algorithm 2 is* $O(m'nr)$, *where* m' *is the number of candidates after pruning.*

Proof. As there are r different $A_{1D}(O_k)$ within $A_{2D}(O_k)$, the number of iterations at line 5 is r in the worst case. As there are m' candidates left in C'', the number of iterations at line 10 is m'. Moreover, there are n different positions for each object, thus the number of iterations at line 11 is n. As a result, the time complexity of Algorithm 2 is $O(m'nr)$. □

Remark PINOCCHIO aims to largely reduce the number of candidate locations, denoted as m, using the pruning rules. By reducing m, the CPU cost decreases significantly as rn is huge. Let the width and height of $MBR(O)$ be w and h, and those of $MBR(C)$ for all candidates be δw and δh, respectively. Let μ denote *minMaxRadius* for brevity. Assume $\delta \gg 1$ and candidates are uniformly distributed.

Based on the pruning rules, the area S_I enclosed by the influence arcs is computed as $S_I = 4(\frac{\alpha-\beta}{360}\pi \cdot \mu^2 - \frac{w}{4}(\mu \cdot \sin\alpha - \frac{w}{2}\tan\beta) - \frac{h}{4}(\mu \cdot \cos\beta - w + \frac{w^2}{4\cdot\mu\cdot\cos\beta}))$. The area S_N enclosed by non-influence boundary can be computed as $S_N = \pi \cdot \mu^2 + wh + 2(w+h)\mu$. Therefore, the total number of candidates that need to be verified is $m' = \frac{S_N - S_I}{\delta^2 wh} m$. When $\delta \gg 1$, $m' \ll m$. As μ decreases, the candidates pruned by influence arcs reduces, while the effect of the non-influence boundary pruning is enhanced. As we shall see in Section 6, δ is much larger than 1 in reality. Therefore, $m'rn \ll mrn$.

5 Optimizing the Validation Phase

Note that the validation phase in Algorithm 2 performs a sequential scan on all the remnant candidates to find the final result. In this section, we optimize this phase by devising two strategies that enable us to ignore certain candidates during validation.

5.1 Optimization Strategies

Our first strategy leverages the upper and lower bounds of influence. Let $maxInf(c)$ be the maximum of the possible influence of candidate c and $minInf(c)$ be the identified influence of c. We also denote $maxminInf = \max_{c\in C} minInf(c)$. Initially, $maxInf(c)$ and $minInf(c)$ are assigned to $|\Omega|$ and 0, respectively. For each moving object, $maxInf(c)$ (*resp.*, $minInf(c)$) may decrease (*resp.*, increase) during pruning and validation. Intuitively, during the validation phase, it is possible that for some $c \in C$ we may find $maxInf(c) < maxminInf$. That is, $\exists c' \in C$ such that $minInf(c') = maxminInf$, and the identified influence of c' is larger than the maximum influence of c. Therefore, we do not need to validate the influence of c any more as it is dominated by c'.

Strategy 1 (Upper and Lower Bounds of Influence). *If $maxInf(c) < maxminInf$, then candidate c cannot influence the largest number of moving objects. Hence, c can be pruned and no further validation is needed. Consequently, $maxminInf$ is initialized to 0 and updated as the algorithm executes.*

On the other hand, note that if a candidate is not pruned by Strategy 1, we have to compute the exact cumulative influence probability. In this case, Definition 4 can be used to stop validation early based on the following lemma.

Lemma 4. *Given a candidate c, moving object O with n positions and τ, if $\exists n' < n$ such that partial non-influence probability $Pr_c^{n-n'}(O) \le 1-\tau$, then c can influence O.*

Proof. With $Pr_c^{n-n'}(O)$, $Pr_c(O)$ can be rewritten as $Pr_c(O) = 1 - \prod_{i=1}^{n'}(1 - Pr_c(p_i)) \cdot Pr_c^{n-n'}(O)$. As $Pr_c^{n-n'}(O) \le 1-\tau$, and $\prod_{i=1}^{n'}(1-Pr_c(p_i)) \in [0,1]$, then $\prod_{i=1}^{n'}(1 - Pr_c(p_i)) \cdot Pr_c^{n-n'}(O) \le Pr_c^{n-n'}(O) \le 1-\tau$. Hence, $Pr_c(O) \ge \tau$. That is, c influence O. □

Strategy 2 (Early Stopping Strategy). *Once Lemma 4 is satisfied, the validation of a moving object is accomplished by computing only independent influence probabilities for n' positions of O instead of all the n positions.*

Algorithm 3. The Pinocchio-vo Algorithm

Input: moving objects set $\Omega = \{O_1, \ldots, O_r\}$, each object $O_k = \{p_1, \ldots, p_{n_k}\}$ ($k \in [1, r]$), candidate locations set $C = \{c_1, \ldots, c_m\}$, PF and τ
Output: the candidate location with maximum influence

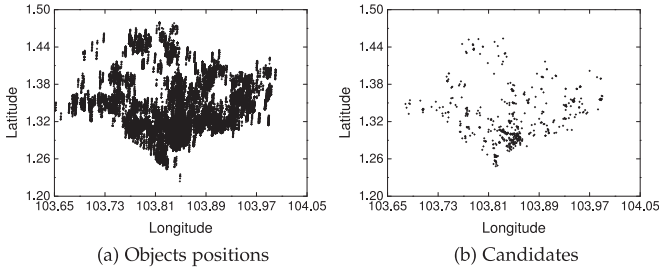
```
1  initialize A_2D with Ω, PF and τ;
2  initialize C R-tree;
3  foreach c_j leaf node in C R-tree do
4      initialize maxInf(c_j) = r, minInf(c_j) = 0;
5  initialize global maxminInf = 0;
6  foreach A_1D(O_k) ∈ A_2D do
7      retrieve C_IA ⊆ C in IA(O_k) and C_NIB ⊆ C outside NIB(O_k);
8      increase minInf(c_j) by 1 where c_j ∈ C_IA;
9      decrease maxInf(c_j) by 1 where c_j ∈ C_NIB;
10     retrieve C' ⊆ C in NIB(O_k) but not in IA(O_k);
11     foreach c_j ∈ C' do
12         insert O_k.id into VS(c_j);
13 initialize H from C ordered by maxInf and minInf;
14 while H ≠ ∅ do
15   c_j = Top(H);
16   if maxInf(c_j) < maxminInf then
17       break;
18   foreach O_k ∈ VS(c_j) do
19     foreach p_i ∈ A_1D(O_k) do
20       Pr_{c_j}(p_i) = PF(dist(c_j, p_i));
21       if Pr_c^{n-n'}(O_k) ≤ 1 − τ then
22         increase minInf(c_j) by 1;
23         continue for next object (goto Line 18);
24     decrease maxInf(c_j) by 1;
25     if maxInf(c_j) < maxminInf then
26       break;
27   update global maxminInf;
28   Pop(H);
29 return c_j whose minInf(c_j) = maxminInf;
```

5.2 The Pinocchio-vo Algorithm

We now extend the Pinocchio algorithm by incorporating the aforementioned optimization strategies during the validation phase. We refer to this enhanced algorithm as Pinocchio-vo, as shown in Algorithm 3. After constructing A_{2D}, it initializes the upper bound $maxInf(c_j)$ and lower bound $minInf(c_j)$ of each c_j node in C R-tree (Lines 1-4). The global $maxminInf$ is also initialized (Line 5). Unlike Algorithm 2, the pruning phase is realized independently from the validation phase. It increases $minInf(c_j)$ for each candidate in C_{IA}, and decrease $maxInf(c_j)$, if c_j is outside $NIB(O_k)$ (Lines 7-9). Additionally, it records the object O_k, which needs to be further verified with respect to each $c_j \in C'$, into the Verification Set of c_j, denoted by $VS(c_j)$ (Lines 10-12). In order to efficiently benefit from Strategy 1, we create a Max Heap H to organize C ordered first by $maxInf$ and second by $minInf$ (Line 13). In the validation phase, it iteratively checks the upper bound of the top candidate c_j in H. If $maxInf(c_j) < maxminInf$, validation is finished by Strategy 1 (Lines 15-17). Otherwise, objects in $VS(c_j)$ are verified by Strategy 2. The influence probability of every position in O_k and the latest partial non-influence probability are

TABLE 2
Description of Real-World Datasets

	Foursquare(F)	Gowalla(G)
user count	2,321	10,162
venue count	5,594	24,081
check-ins	167,231	381,165
avg. check-ins	72	37
min check-ins	3	2
max check-ins	661	780



(a) Objects positions (b) Candidates

Fig. 6. Geographical distributions of data samples.

computed iteratively. Once Lemma 4 satisfies, $minInf(c_j)$ increases by 1 and the next object is probed (Lines 19-23). Note that in Line 21, if $n' = n$, $Pr_c^{n-n'}(O_k) = 1$, therefore the inequality never holds (Definition 4). In this case, it replaces the checking of $Pr_c^{n-n'}(O_k) \leq 1 - \tau$ with that of $Pr_c(O_k) \geq \tau$. If it is verified that c_j cannot influence O_k, $maxInf(c_j)$ decreases by 1. After that, if $maxInf(c_j) < maxminInf$, c_j must not be the optimal one (Lines 24-26). Before probing the next top candidate, $maxminInf$ is updated accordingly (Lines 27-28). The candidate whose lower bound of influence equals to $maxminInf$ is returned as the result.

Theorem 4. *The time complexity of Algorithm 3 is $O(m''n'r'')$, where m'' (resp., r'') is the number of candidates (resp., objects) to be validated after applying pruning rules and Strategy 1, n' is the number of positions that has to be used for influence computation after applying Strategy 2.*

Proof. As discussed in Theorem 3, m' candidates needs to be verified. In other words, there are $r' = \frac{m'}{m} r$ objects to be verified in H iteration. Moreover, r' can be further reduced to r'' due to Lines 25 and 26 in Algorithm 3.

For candidates, the best case occurs when the top candidate in H is the optimal one, thus $m'' = 1$. In the worst case, if pruning has no effect (which is almost impossible), all candidates need to be scanned, i.e., $m'' = m$. Normally, assuming the actual maximum influence is M and the $(m''+1)$th candidate in H has its $maxInf < M$ exactly, then only m'' candidates need to be checked. As pruning effect is significant (detailed in Section 6.2), $m''r'' \ll mr$.

Moreover, due to Strategy 2, it is not necessary to validate all positions of an object, which means n is reduced to n' ($n' < n$). As a result, the complexity is $O(m''n'r'')$. □

6 Performance Study

We now investigate the performance of our proposed PRIME-LS framework over real-world datasets from a variety of aspects.

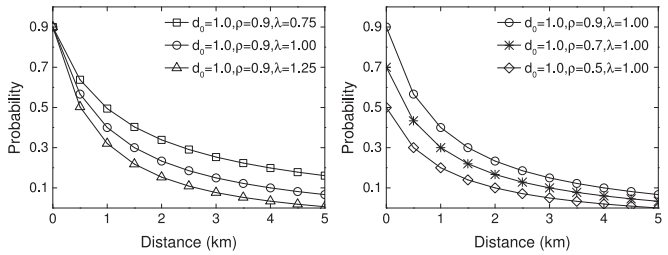
6.1 Experimental Setup

Datasets. Table 2 describes the two real-world datasets we use in the experiments. Both contain user check-in data of LBS and are available from [22]. We adopt check-in data here for two reasons: 1) The effectiveness can be verified by check-in ground-truth, which is the actual number of visitors for each POI; 2) The probability models of check-in with respect to distance have been justified [21], [22]. The positions of check-ins in *Foursquare* (abbr., F) are all located in Singapore, while those in *Gowalla* (abbr., G) are mainly in California. As the number of check-ins varies widely, the distribution of the numbers of positions in moving objects is skewed. Moreover, the geographical distribution of positions is also skewed as shown in Fig. 6a (check-ins in F). We choose 200, 400, 600, 800 and 1,000 positions from check-in coordinates as candidate locations by random uniform sampling. Fig. 6b shows the case of 600 candidates.

Algorithms. Following algorithms are tested in the experiments. They are implemented in C++, run on a 1.8 GHz machine with 4 GB RAM under Windows 8.1 (64 bit).

- NA: A baseline method that exhaustively computes the cumulative influence probabilities for all pairs of candidate location and moving object, based on which we retrieve the most influential candidate.
- BRNN*: We run MaxOverlap algorithm [16] to select for each object a location that influences the most positions. Finally, we return the location which has been selected by most objects.
- RANGE: We design a baseline where an object is influenced if at least a certain proportion of its positions lie within a given range of a candidate.
- PIN: PINOCCHIO algorithm described in Algorithm 2.
- PIN-VO: PINOCCHIO-VO algorithm described in Algorithm 3. The R-tree for candidates is loaded in main memory. The maximum number of elements in each R-tree node is 8.
- PIN-VO*: A variant of PINOCCHIO-VO algorithm without the pruning phase (using only validation optimization strategies).

Parameter Settings. As stated in [21], the probability of a user checking-in at a point-of-interest decays as the power-law of the distance between them. That is, $Pr_c(p) = \rho(d_0 + dist(c,p))^{-\lambda}$ where ρ is a factor to describe *behavior pattern* and d_0 is a distance factor, which is set to 1.0 (All parameters in this function are selected based on the empirical findings in [21].). We employ this model as the pre-defined probability function PF. As illustrated in Fig. 7a, we set $\rho = 0.9$, which means the independent influence probability is 0.9 if the distance between p and c is zero. In other words, ρ indicates the maximum influence probability between p and c. Additionally, λ is set to 0.75, 1.0 and 1.25 to change the downtrends, respectively. ρ is set to 0.5, 0.7 and 0.9. We vary probability thresholds τ to 0.1, 0.3, 0.5, 0.7 and 0.9. Unless specified otherwise, the default values of the number of candidates, probability threshold τ, behavior factor ρ, and power-law factor λ are 600, 0.7, 0.9, and 1.0, respectively.



(a) Effect of λ (b) Effect of ρ

Fig. 7. Probability functions.

6.2 Experimental Results

Comparison between Different Semantics. As discussed in Section 2, there are two primary semantics of influence in existing LS studies, namely NN-based and range-based. In this set of experiments, we investigate the performance of the optimal locations that are mined under PRIME-LS and these LS semantics. For NN semantics, since existing NN-based LS solutions are not designed for mobile scenario, we extend a state-of-the-art technique MAXBRNN [16] in order to apply it in such scenario. Specifically, we run MaxOverlap algorithm [16] to select for each object O the best location c, which influences the most positions in O. Afterwards, we choose the location that has been selected by the most objects. For range semantics, we design a baseline RANGE with a simple definition of influence, where an object is deemed to be influenced if at least some proportion of its positions lie within a given range of a candidate. We vary the proportions of positions among $25, 50$ and 75 percent as the minimum proportion thresholds. In light of [27], we follow its default range setting, namely 5‰ of the complete scale (e.g., 0.2 km for *Foursquare*) as the default range, and set half and twice of it as the lower and upper range bounds, respectively. Hence, there are nine parameter combinations in total, and we use the average value of the combinations for comparison.

We employ two measures, namely *Precision@K*[6] (abbr., $P@K$) and *Average Precision@K* (abbr., $AP@K$), to compare the effectiveness of the solutions. We treat the actual check-in logs at candidate locations, which have been assumed unknown in our framework, as the ground-truth. We rank *Top-K* ($K = 10, \ldots, 50$) of 200 candidates as the relevant locations according to their actual numbers of check-ins. The *Top-K* results of PRIME-LS, BRNN* and RANGE are regarded as recommended locations. We randomly choose 50 different groups of candidates, and compute the mean values for comparison. Tables 3 and 4 report the $P@K$ and $AP@K$ of these approaches for *Foursquare*, where both values increase as K grows. Since the experiment results on *Gowalla* are qualitatively similar, due to space constraint, we do not report them in detail. Observe that on average, PRIME-LS is significantly better than BRNN* with 20 ($P@K$)-35 ($AP@K$) percent improvement, indicating that other positions also play a role besides the nearest neighbor. Also, PRIME-LS is around 8 ($P@K$)-12 ($AP@K$) percent better than RANGE on average. Despite the ideas of NN and RANGE are simpler than PRIME-LS, their relatively poor performances may result in negative impacts (e.g., customer loss, economic loss, etc.) that cannot be ignored in practical location selection. Hence, PRIME-LS solution outperforms classical LS semantics in mobile scenarios, and we focus on PRIME-LS in the subsequent discussions.

6. As we use K for both relevant and recommended locations, Recall@K has the same value as Precision@K.

TABLE 3
Precision Comparison

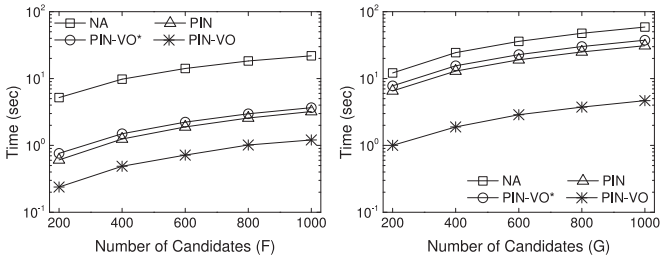
Precision	@10	@20	@30	@40	@50
PRIME-LS	0.072	0.113	0.185	0.238	0.293
Avg. RANGE	0.058	0.112	0.174	0.222	0.278
BRNN*	0.046	0.112	0.157	0.206	0.264

Comparison of NA, PIN, PIN-VO and PIN-VO.* We first study the scalability of these algorithms. Fig. 8 reports the running time varying the number of candidates. Clearly, the cost increases when the number of candidates grows. Compared to NA, PIN-VO has the best scalability, which significantly reduces the time by orders of magnitude. PIN is slightly better than PIN-VO*, which means the efficiencies of pruning rules and optimization strategies are close, if they are applied independently. Based on PIN, the efficiency of PIN-VO is improved due to the optimization strategies by around 60 (in F)-85 (in G) percent, which means Strategy 1 significantly benefits from the tightened upper-lower bounds by pruning, and Strategy 2 further reduces the number of positions to be verified. Observe that PIN-VO, PIN and PIN-VO* (especially the latter two) are more efficient in F than in G. The reason is twofold. First, the gap between upper and lower bounds in F is narrower than in G, which leads to lesser objects to be verified when iterating through each candidate in H. Second, as objects in F have more positions than in G, Strategy 2 is more efficient. Fig. 9 plots the scalability from 2k to 10k objects chosen randomly from *Gowalla*, with the same set of 600 candidates. The results are qualitatively similar to Fig 8. PIN-VO also demonstrates the best scalability, followed by PIN, PIN-VO* and NA.

Next, we study the effect of pruning rules. For each moving object, the area enclosed by IA and excluded by NIB determines the number of candidates that can be pruned. As shown in Fig. 10, nearly $2/3$ candidates are pruned on average. As τ increases (i.e., *minMaxRadius* declines based on Definition 5), the number of candidates inside IA decreases, while the number outside NIB grows. Observe that for F (*resp.*, G) the candidates pruned by IA (*resp.*, NIB) are much more than those by NIB (*resp.*, IA). This is because the candidates are scattered wider in F than in G with respect to the activity region of an object. Another interesting phenomenon is that the number of pruned candidates in F changes more significantly than in G when τ changes. The reason is twofold. First, compared to *minMaxRadius*, the size of activity region of each object is relatively close (*resp.*, much

TABLE 4
Average Precision Comparison

Avg. Precision	@10	@20	@30	@40	@50
PRIME-LS	0.022	0.032	0.055	0.081	0.110
Avg. RANGE	0.020	0.031	0.050	0.071	0.092
BRNN*	0.015	0.028	0.040	0.056	0.085



(a) *Foursquare* (b) *Gowalla*

Fig. 8. Scalability (#candidates).

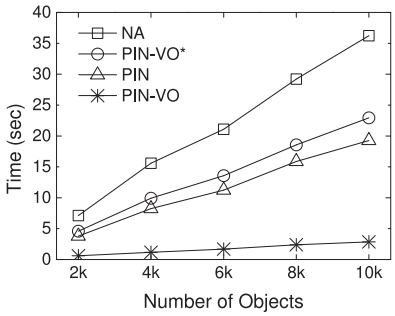


Fig. 9. Scalability (#objects).

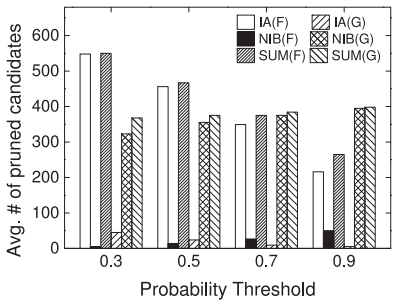


Fig. 10. Pruning effect.

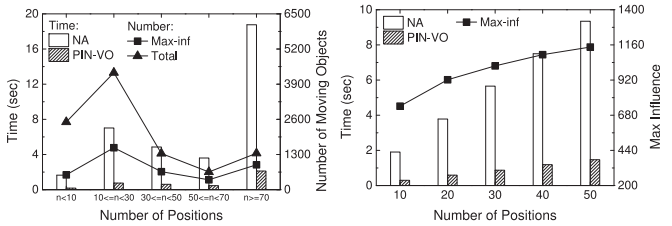
larger) in F (*resp.*, G), which means it is more (*resp.*, less) sensitive to *minMaxRadius*. Second, objects have more (*resp.*, less) positions in F (*resp.*, G), which leads to more (*resp.*, less) sensitivity to τ. As PIN-VO is superior to PIN and PIN-VO*, we only use NA and PIN-VO to study the impact of various parameters in subsequent experiments.

Effect of n. In this part, we test the effect of n over the algorithms. We divide objects in *Gowalla* into five groups according to their numbers of positions, as shown in Table 5.

Fig. 11a reports that PIN-VO has similar improvement in performance for different n with respect to NA. As *minMaxRadius* increases with n, less candidates are pruned by NIB, which results in larger $maxInf$. Then it is likely that more objects need to be verified by Strategy 1, which offsets the benefit brought in by Strategy 2. If n is greater than 70, the maximum influence is the closest to the total, which is more than 60 percent. In contrast, the group with the least positions ranks the last, where the maximum influence is only 20 percent of the total. This means any object with more positions has higher probability being attracted by candidates. Furthermore, the average distance between the five resulting optimal locations over different groups is only 0.22 km, where two results are identical and the maximum

TABLE 5
Five Groups

# *of positions*	$[1, 10)$	$[10, 30)$	$[30, 50)$	$[50, 70)$	$[70, 780]$
# *of objects*	2,501	4,325	1,337	655	1,344



(a) Different moving objects. (b) The same moving objects.

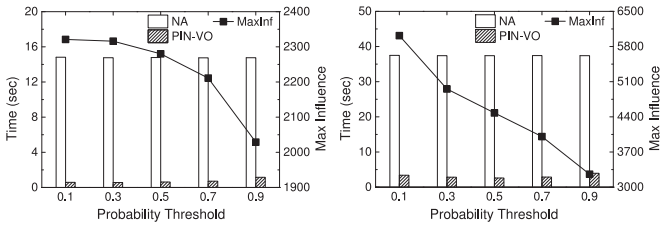
Fig. 11. Effect of n.

distance is 0.69 km. Hence, compared with the average distance between candidates, which is a few kilometers, the distance error is less than 6 percent.

We also study the case of the same objects with different number of positions. We first select 1,999 moving objects in *Gowalla*, each of which has more than 50 positions. For each object, we generate five different instances of it by choosing $10, \ldots, 50$ positions randomly from all its positions. Then we group all the instances into five sets, each has the same number of positions. In Fig. 11b, the cases of performance and maximum influence are similar to Fig. 11a. On the other hand, the average distance between the five result locations is 0.27 km, where two results are identical and the maximum distance is 0.78 km. Thus, the distance error is still less than 8 percent.

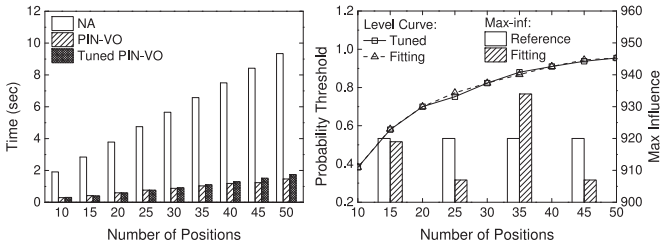
Intuitively, too small n may not describe mobility pattern and too large n causes expensive overhead. Therefore, we further investigate the reasonable range of n from two aspects, namely accuracy and time cost. Since the mobility patterns of both human beings and animals have periodicity [20], [34], proper samplings by the same temporal interval for long-term data still comply the regularity. For example, by identifying for each person the most visited positions within each hourly interval in a day, we can predict human mobility with around 93 percent accuracy [35]. Moreover, for different n, the precision of optimal locations is between 92 and 94 percent as described above. Hence, 24 hourly or 48 half-hourly positions are sufficient to achieve a satisfactory accuracy. On the other hand, although more positions lead to better accuracy, time cost increases linearly as n grows. To sum up, using 24-48 positions, we can achieve a tradeoff between accuracy and cost.

Effect of τ. Fig. 12 reports the running time and the maximum influence varying thresholds τ. The running time of PIN-VO falls and then rises when τ grows. Normally, as τ increases, more positions of an object need to be verified, which weakens the effect of Strategy 2. Moreover, influence values of candidates decline and differ widely between each other, which degrades Strategy 1. However, if τ is very small, many candidates may have similar influences, then Strategy 1 has to verify them all and performs poor. On the other hand, the maximum influence drops as τ grows.



(a) *Foursquare* (b) *Gowalla*

Fig. 12. Effect of τ.



(a) Time cost. (b) $\langle n, \tau \rangle$ level curve.

Fig. 13. Relationship between n and τ.

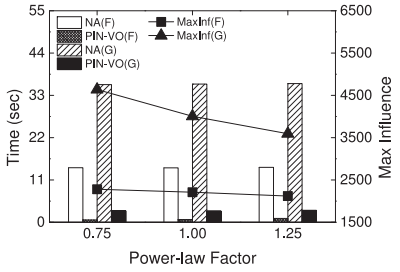


Fig. 14. Effect of λ.

Relationship between n and τ. We also explore the relationship between n and τ, as they both affect the maximum influence and result locations. Together with the five sets used in Fig. 11b, we construct four more with 15, 25, 35 and 45 positions in the same way. We first select the set with 20 positions and $\tau = 0.7$, based on which the maximum influence is set as the reference. Then we tune τ values for other sets until their maximum influences equal to the reference, which means these $\langle n, \tau \rangle$ pairs form a level curve with respect to the same maximum influence. In Fig. 13a, the error of execution time between tuned and original PIN-VO is less than 3 percent with respect to NA. It is noteworthy that four and two of the resulting optimal locations are respectively identical, and the average distance is only 0.16 km. This indicates the error of distance between any two resulting locations of $\langle n, \tau \rangle$ pairs on the level curve is very small. On the other hand, as shown in Fig. 13b, we fit the level curve by *Matlab's polyfit*, using the original $\langle n, \tau \rangle$ pairs with 10, 20, 30, 40 and 50 positions. The average error of the maximum influence between the fitting and original $\langle n, \tau \rangle$ pairs with 15, 25, 35 and 45 positions is less than 1.2 percent. *In summary, if we expect a certain number of objects to be influenced, the resulting locations are identical or very close with high accuracy, regardless of how the parameters n, τ are set.*

Effect of λ. Below, we test the impact of the power-law factor λ on both the execution time and maximum influence. As shown in Fig. 14, PIN-VO has similar running time for different λ, and the maximum influence grows when λ increases as cumulative probabilities of moving objects drop. For objects with more positions, the maximum influence (*resp.*, pruning and optimized validation) is less (*resp.*, more) sensitive to probability changes. As a result, the effect in *Gowalla* is not as significant as that in *Foursquare*, and the maximum influence in *Foursquare* have slower rate of decline than that in *Gowalla*.

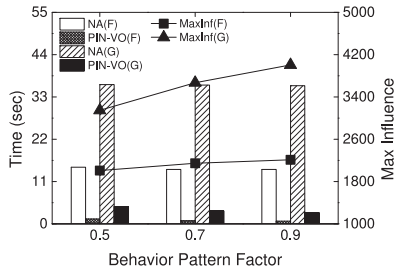
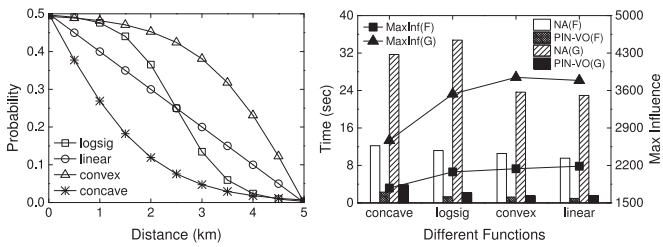


Fig. 15. Effect of ρ.



(a) Different PFs. (b) Efficiency and max influence.

Fig. 16. Effect of different PFs.

Effect of ρ. The effect of the parameter ρ, which describes behavior pattern and represents the maximum PF influence probability, is qualitatively similar to that of λ. In Fig. 15, the performance improves as ρ grows. The effect in *Gowalla*, where objects have less positions, is also less significant than that in *Foursquare*. The maximum influence decreases quickly when ρ declines. The reason is that nearer position of a moving object contributes larger probability to the cumulative influence probability. Compared to λ on the validation phase, ρ has lower efficiency as more validations are required.

Effect of Different PFs. It may be possible that in some specific scenarios, one may have to adopt PF functions. In fact, PINOCCHIO is a general framework and many other PF functions can also be adopted without any modification. In this part, we show the effectiveness and efficiency of our model under different PFs. As shown in Fig. 16a, we test four different PFs,[7] all of which are commonly used functions in data mining and machine learning areas. *Logsig* is a variation of the *Log-sigmoid transfer* function, i.e., $logsig(dist) = 1/(1 + e^{dist}) \cdot \rho$ where ρ is set to 0.5. *Convex* and *Concave* are the convex and concave parts of *Logsig*, respectively. The scales of them are normalized to the same as that of *Logsig*. *Linear* also has the same scales. Fig. 16b reports the effect in efficiency and maximum influence of these four PFs. Despite slight differences, the results demonstrate that our model can handle different PFs.

7. To simplify the discussion and show the effectiveness more intuitively, these functions may not be strictly pdfs.

7 Conclusions

In this paper, we introduce a generalized location selection problem called Prime-ls and provide an effective and efficient solution to it. Comparing to classical ls problem, Prime-ls assumes that objects are mobile and may be influenced by multiple candidate locations with different probabilities. This problem has many real-world applications such as facility allocation, urban planning, wild-life monitoring, location-based services, etc. We present a novel algorithm called Pinocchio that exploits two pruning rules, built on top of the *minMaxRadius* measure, to prune inferior candidate locations effectively. Additionally, Pinocchio-vo further improves the performance of Pinocchio with two optimization strategies that reduce the candidate validation cost. Extensive experiments on real datasets demonstrate the superiority of our approaches. As part of future work, we plan to study incremental solution towards Prime-ls in dynamic scenarios, where candidate locations, objects as well as their positions keep on changing.

Acknowledgments

The authors wish to thank the anonymous reviewers for the comments and suggestions that have greatly improved the paper. Hui Li and Jiangtao Cui are supported by the National Nature Science Foundation of China (Nos. 61202179, 61173089, and 61472298), the China 111 Project (No. B16037), and the Huawei Innovation Research Program.

References

[1] T. Xia, D. Zhang, E. Kanoulas, and Y. Du, "On computing top-t most influential spatial sites," in *Proc. 31st Int. Conf. Very Large Data Bases*, 2005, pp. 946–957.

[2] F. Korn and S. Muthukrishnan, "Influence sets based on reverse nearest neighbor queries," in *Proc. 2000 ACM SIGMOD Int. Conf. Manage. Data*, 2000, pp. 201–212.

[3] L. Chen, M. T. Özsu, and V. Oria, "Robust and fast similarity search for moving object trajectories," in *Proc. 2005 ACM SIGMOD Int. Conf. Manage. Data*, 2005, pp. 491–502.

[4] D. Kempe, J. M. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *Proc. 9th ACM SIGKDD Int. Conf. Knowl. Discovery and Data Min. Press*, 2003, pp. 137–146.

[5] M. A. Cheema, X. Lin, W. Wang, W. Zhang, and J. Pei, "Probabilistic reverse nearest neighbor queries on uncertain data," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 4, pp. 550–564, Apr. 2010.

[6] J. Huang, Z. Wen, J. Qi, R. Zhang, J. Chen, and Z. He, "Top-k most influential locations selection," in *Proc. 20th ACM Int. Conf. Inf. Knowl. Manage.*, 2011, pp. 2377–2380.

[7] J. Qi, R. Zhang, L. Kulik, D. Lin, and Y. Xue, "The min-dist location selection query," in *Proc. IEEE 28th Int. Conf. Data Eng.*, 2012, pp. 366–377.

[8] S. Shang, B. Yuan, K. Deng, K. Xie, and X. Zhou, "Finding the most accessible locations: Reverse path nearest neighbor query in road networks," in *Proc. 19th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, 2011, pp. 181–190.

[9] Y. Sun, J. Huang, Y. Chen, R. Zhang, and X. Du, "Location selection for utility maximization with capacity constraints," in *Proc. 21st ACM Int. Conf. Inf. Knowl. Manage.*, 2012, pp. 2154–2158.

[10] L. A. Tang, Y. Zheng, X. Xie, J. Yuan, X. Yu, and J. Han, "Retrieving k-nearest neighboring trajectories by a set of point locations," in *Proc. 12th Int. Conf. Adv. Spatial Temporal Databases*, 2011, pp. 223–241.

[11] C. Xu, Y. Gu, R. Zimmermann, S. Lin, and G. Yu, "Group location selection queries over uncertain objects," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 12, pp. 2796–2808, Dec. 2013.

[12] D. Yan, R. C.-W. Wong, and W. Ng, "Efficient methods for finding influential locations with adaptive grids," in *Proc. 20th ACM Int. Conf. Inf. Knowl. Manage.*, 2011, pp. 1475–1484.

[13] L. Zhan, Y. Zhang, W. Zhang, and X. Lin, "Finding top k most influential spatial facilities over uncertain objects," in *Proc. 21st ACM Int. Conf. Inf. Knowl. Manage.*, 2012, pp. 922–931.

[14] D. Zhang, Y. Du, T. Xia, and Y. Tao, "Progressive computation of the min-dist optimal-location query," in *Proc. 32nd Int. Conf. Very Large Data Bases*, 2006, pp. 643–654.

[15] K. Zheng, Z. Huang, A. Zhou, and X. Zhou, "Discovering the most influential sites over uncertain data: A rank-based approach," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 12, pp. 2156–2169, Dec. 2012.

[16] R. C.-W. Wong, M. T. Özsu, P. S. Yu, A. W.-C. Fu, and L. Liu, "Efficient method for maximizing bichromatic reverse nearest neighbor," in *Proc. Very Large Data Bases Endowment*, vol. 2, no. 1, pp. 1126–1137, 2009.

[17] Z. Zhou, W. Wu, X. Li, M. L. Lee, and W. Hsu, "Maxfirst for maxbrknn," in *Proc. IEEE 27th Int. Conf. Data Eng.*, pp. 828–839, 2011.

[18] D. Choi, C. Chung, and Y. Tao, "A scalable algorithm for maximizing range sum in spatial databases," in *Proc. Very Large Data Bases Endowment*, vol. 5, no. 11, pp. 1088–1099, 2012.

[19] Y. Tao, X. Hu, D.-W. Choi, and C.-W. Chung, "Approximate maxrs in spatial databases," in *Proc. Very Large Data Bases Endowment*, vol. 6, no. 13, pp. 1546–1557, 2013.

[20] Z. Li, B. Ding, J. Han, R. Kays, and P. Nye, "Mining periodic behaviors for moving objects," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery and Data Min.*, 2010, pp. 1099–1108.

[21] B. Liu, Y. Fu, Z. Yao, and H. Xiong, "Learning geographical preferences for point-of-interest recommendation," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery and Data Min.*, 2013, pp. 1043–1051.

[22] Q. Yuan, G. Cong, Z. Ma, A. Sun, and N. Magnenat-Thalmann, "Time-aware point-of-interest recommendation," in *Proc. 36th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2013, pp. 363–372.

[23] M. L. Yiu, H. Lu, N. Mamoulis, and M. Vaitis, "Ranking spatial data by quality preferences," in *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 3, pp. 433–446, Mar. 2011.

[24] D. Papadias, Y. Tao, K. Mouratidis, and C. K. Hui, "Aggregate nearest neighbor queries in spatial databases," *ACM Trans. Database Syst.*, vol. 30, no. 2, pp. 529–576, 2005.

[25] F. Aurenhammer, "Voronoi diagrams— survey of a fundamental geometric data structure," *ACM Comput. Surveys*, vol. 23, no. 3, pp. 345–405, 1991.

[26] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *Proc. 1984 ACM SIGMOD Int. Conf. Manage. Data*, 1984, pp. 47–57.

[27] M. L. Yiu, X. Dai, N. Mamoulis, and M. Vaitis, "Top-k spatial preference queries," in *Proc. IEEE 29th Int. Conf. Data Eng.*, 2007, pp. 1076–1085.

[28] J. Niedermayer, et al., "Probabilistic nearest neighbor queries on uncertain moving object trajectories," in *Proc. Very Large Data Bases Endowment*, vol. 7, no. 3, pp. 205–216, 2013.

[29] H. G. Elmongui, M. F. Mokbel, and W. G. Aref, "Continuous aggregate nearest neighbor queries," *GeoInformatica*, vol. 17, no. 1, pp. 63–95, 2013.

[30] R. Zhang, J. Qi, D. Lin, W. Wang, and R. C.-W. Wong, "A highly optimized algorithm for continuous intersection join queries over moving objects," *The VLDB J. – Int. J. Very Large Data Bases.*, vol. 21, no. 4, pp. 561–586, 2012.

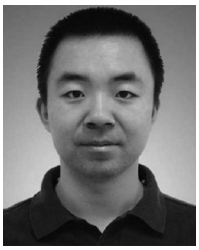
[31] J. M. Kang, M. F. Mokbel, S. Shekhar, T. Xia, and D. Zhang, "Continuous evaluation of monochromatic and bichromatic reverse nearest neighbors," in *Proc. IEEE 23rd Int. Conf. Data Eng.*, pp. 806–815, 2007.

[32] P. Ghaemi, K. Shahabi, J. P. Wilson, and F. B. Kashani, "Continuous maximal reverse nearest neighbor query on spatial networks," in *Proc. 20th Int. Conf. Adv. Geographic Inf. Syst.*, 2012, pp. 61–70.

[33] N. Roussopoulos, S. Kelley, and F. Vincent, "Nearest neighbor queries," in *Proc. 1995 ACM SIGMOD Int. Conf. Manage. Data*, 1995, pp. 71–79.

[34] Z. Li, J. Han, B. Ding, and R. Kays, "Mining periodic behaviors of object movements for animal and biological sustainability studies," *Data Min. Knowl. Discov.*, vol. 24, no. 2, pp. 355–386, 2012.

[35] C. Song, Z. Qu, N. Blumm, and A.-L. Barabasi, "Limits of predictability in human mobility," *Sci.*, vol. 327, no. 5968, pp. 1018–1021, 2010.



Meng Wang received the MS degree in software engineering from Xi'an Jiaotong University, China, in 2012. He is currently working toward the PhD degree from the School of Computer Science and Technology, Xidian University, China. His research interests include data and knowledge engineering and spatio-temporal database.

Hui Li received the BEng degree from the Harbin Institute of Technology in 2005 and the PhD degree from Nanyang Technological University, Singapore in 2012, respectively. He is an associate professor in School of Cyber Engineering, Xidian University, China. His research interests include data mining, knowledge management and discovery, privacy-preserving query, and analysis in big data.

Jiangtao Cui received the MS and PhD degrees both in computer science from Xidian University, Xi'an, China, in 2001 and 2005, respectively. Between 2007 and 2008, he was with the Data and Knowledge Engineering group working on high-dimensional indexing for large scale image retrieval, in the University of Queensland, Australia. He is currently a professor in the School of Computer Science and Technology, Xidian University, China. His current research interests include data and knowledge engineering, data security, and high-dimensional indexing.

Ke Deng received the master's degree in information and communication technology from Griffith University, Australia, in 2001, and the PhD degree in computer science from the University of Queensland, Australia, in 2007. His research background include high-performance database system, spatio-temporal data management, data quality control, and business information system. His current research interest include big spatio-temporal data management and mining.

Sourav S. Bhowmick is an associate professor in the School of Computer Science and Engineering, Nanyang Technological University. His current research interests include data management, data analytics, computational social science, and computational systems biology. He has published more than 150 papers in major venues in these areas such as Special Interest Group on Management of Data, Very Large Data Base, International Conference on Data Engineering, Special Interest Group on Knowledge Discovery and Data Mining, *Modern Machinery*, the *IEEE Transactions on Knowledge and Data Engineering*, *Very Large Data Base Journal*, *Bioinformatics*, and *Biophysical Journal*.

Zhenhua Dong received the BEng degree from Tianjin University in 2006 and the PhD degree in computer science and technology from Nankai University, China, in 2012. He was a research assistant at the University of Minnesota during 2010-2011. His research interests include web search, knowledge discovery, and recommender systems. He is currently a researcher at Huawei Noah's Ark Research Lab.