Information Sciences 463–464 (2018) 1–20

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Place Your Next Branch with MILE-RUN: Min-dist Location Selection over User Movement

Jiangtao Cui[a], Meng Wang[a], Hui Li[b,*], Yang Cai[a]

[a] School of Computer Science and Technology, Xidian University, Xi'an 710071, China
[b] School of Cyber Engineering, Xidian University, Xi'an 710071, China

ARTICLE INFO

Article history:
Received 3 August 2017
Revised 8 April 2018
Accepted 12 June 2018
Available online 13 June 2018

Keywords:
Location selection
User movement
Spatial database
Road network

ABSTRACT

Due to the wide spectrum of applications, the *Min-dist location selection* problem has drawn much research attention in spatial database studies. Given a group of existing locations of a specific kind of facility, Min-dist problem aims to find the optimal location from series of candidate places to establish a new facility such that the average distance between users and their respective nearest facilities can be minimized. Although plenty of efforts have been proposed to address Min-dist problems, they all assume that users are stationary. Unfortunately, in practice, objects (*e.g.,* people, animals) are mobile in various scenarios. Due to the movements of users, it is non-trivial to identify an optimal candidate, where none of existing solutions is applicable. Motivated by that, in this paper we take into account the mobile factor and present the first effort on a *generalized* Min-dist problem, called **M**in-d**I**st **L**ocation S**E**lection ove**R** **U**ser Moveme**N**t (MILE-RUN). To address the efficiency issue caused by user movement and road network, based on a *reference location* transformation, we present two groups of algorithms, index-based and index-free ones. The first group answers MILE-RUN efficiently with the help of spatial locality based index structures, and fits the case where facilities and users are known apriori, while the second group solves the problem from scratch. Extensive experiments are conducted on both real-world and synthetic datasets, the results of which demonstrate that our algorithms are more efficient compared to the baseline method by orders of magnitude.

1. Introduction

Min-dist location selection, as discussed in [14], is an important spatial decision problem that has a wide spectrum of applications, such as marketing, urban planning, logistics, location-based services, etc. Informally, given a set U of users and a set F of existing locations for a specific kind of facility, the Min-dist problem selects a location c from a set of candidates, namely C, for establishing a new facility so that the average distance between users and their respective nearest facilities is minimized.

Several techniques [14,15,26,30] have been proposed for addressing Min-dist location selection problems, and all these works assume that each user is stationary such that one can be represented as a single point in a two-dimensional space. Unfortunately, with the proliferation of mobile devices, this assumption may not hold in various applications where users

* Corresponding author.
E-mail addresses: cuijt@xidian.edu.cn (J. Cui), wamengit@sina.com (M. Wang), hli@xidian.edu.cn, lihu0006@e.ntu.edu.sg (H. Li), wolfcyang@163.com (Y. Cai).

https://doi.org/10.1016/j.ins.2018.06.036
0020-0255/

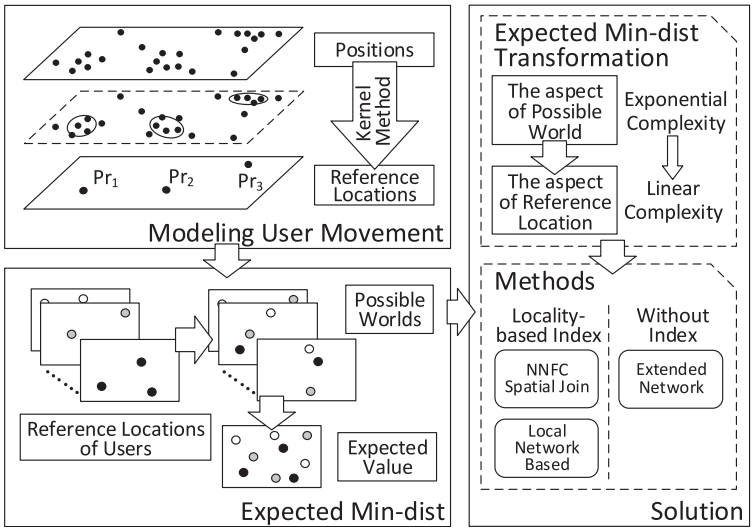


Fig. 1. MILE-RUN framework.

are not static and traveling from one place to another every day. In these scenarios, where a moving user can be modeled as a set of spatial positions instead of a single one [21], none of existing efforts in this field can be applied.

Consider the scenario of sharing economy which is taking off in recent years. A car-sharing company (*e.g., Autolib, Car2Go*) wishes to improve its market share by optimizing the existing service network, namely the parking and charging stations. In order that vehicles can be more accessible to users for competitive purposes, the company needs to choose from a collection of candidate places an optimal location which, if a new station is built there, minimizes the average pick-up distance for users. Suppose that a user is mobile, then she would probably take a car close to where she were present at. As a result, for a more effective solution, the evaluation of the average distance cannot be solely attributed to a single location as all of the positions that represent a user's movement play a role.

Another example is franchise chain (*e.g., McDonald's, 7-eleven*). In order to optimize customer experience, the company plans to build a new store on the basis of existing service branches, so as to minimize the average travel cost to attract more customers. As mobile customers would probably purchase in stores nearby where they were present at, similar to the previous example, their movements have to be taken into consideration in the distance evaluation.

To address the Min-dist problem in the mobile setting, in this paper we present a novel problem, namely ***M****in-d****I****st* ***L****ocation S****E****lection ove****R*** ***U****ser Moveme****N****t* (MILE-RUN), which addresses the aforementioned limitation that existing Min-dist techniques suffer from. As users' travels between spatial locations in the real-world are usually confined to road network, we focus on the road network distance. Accordingly, we assume both facilities and users are located on a road network graph $G(V, E)$, where V and E denote the sets of vertices and edges, respectively. Additionally, as stated in [14,15], in many real applications, companies can only choose from a finite number of candidate places for rent or sale in a region or on a road, and we follow this setting in the paper. As a result, the MILE-RUN problem can be intuitively defined as follows. Given a set U of mobile objects (*e.g.,* users, customers), a set F of existing locations for a specific kind of facility (*e.g.,* stations, chain stores) and a network graph $G(V, E)$, the MILE-RUN problem returns the optimal location among a set C of candidate ones for establishing a new facility so that the average network distance between all objects and their respective nearest facilities is minimized.

Comparing with existing Min-dist location selection solutions, there are three challenges in the MILE-RUN problem. Firstly, a moving object is described by a set of positions. In addition to the large amount of positions, which may lead to costly computation, taking every position into account would be inappropriate as some positions are not/poorly contributive for finding the optimal place to establish the new branch. For instance, there may be positions with GPS errors or visited by a user purely occasionally, etc. Hence, it is vital to the evaluation of average distance by eliminating these noisy positions. Secondly, even if we have the knowledge of the worthwhile positions for each moving user, it is non-trivial to determine which facility a specific user will visit for service as she is probable to be present at any one of these positions. As existing solutions to the Min-dist problem are single point based, they cannot handle the multiple point scenario. Accordingly, the average distance is difficult to be evaluated under the Min-dist criteria. Finally, the computational overhead is extremely expensive due to the massive shortest network distance computations over users, existing facilities and candidate locations, which makes the Euclidean distance based Min-dist techniques unavailable.

In this work, we present a systematic solution, illustrated in Fig. 1, to address the above challenges introduced by the MILE-RUN problem. The framework consists of three major components as follows.

In the first component, we model user movement with the concept of *reference locations* [11]. Each reference location is one of the activity places that a user frequently shows up every day and thus are worth considering. With the help of *kernel method* [24], we can identify for each user multiple reference locations with different present probabilities.

The second component formalizes the Min-dist problem in the mobile scenarios. In light of the uncertainty of movements, users that are probably present at any one of the reference locations can be modeled following *possible world* semantics [2]. Based on the criteria of existing Min-dist problem (*i.e.,* stationary objects), the optimal candidate with respect to a specific possible world can be obtained. As any possible world may occur, all possible worlds have effects on the result and the average distance is in fact a random variable accordingly. Hence, candidates can be ranked according to the expected average distance over all possible worlds.

In the third component, we systematically solve the proposed MILE-RUN problem in two steps. (1) Notably, the amount of possible worlds is huge due to the massive number of users, such that it is impractical to explore all of them, which ends up with exponential complexity. To avoid the exhaustive enumeration, a novel method from the aspect of reference location is designed, which can be effectively leveraged to reduce the ranking process from exponential complexity to linear one. (2) Furthermore, as shown in Section 7, city road networks usually have large volumes of vertices and edges, which results in costly network traversals in finding the optimal candidate for massive users. To address this problem, with the help of spatial locality index techniques, we propose two solutions, *network nearest facility circle* (NNFC) based and *local network* based, both of which provide superior efficiency. The former method focuses on only those reference locations affected by certain candidates. For the latter, not any network traversal is needed for the selection of candidates. On the other hand, we present an alternative solution based on *network extension* scheme for addressing MILE-RUN from scratch, in the case that existing facilities and users are unknown in advance. In summary, our major contributions are outlined as follows.

- To the best of our knowledge, this is the first effort to formalize the problem of Min-dIst Location SElection oveR User MovemeNt (MILE-RUN). Compared to existing works, MILE-RUN is a generalized Min-dist problem, which takes into account the movements of objects.
- Depending on whether index can be used, three novel solutions are developed to address the MILE-RUN problem efficiently and accurately under different cases.
- Comprehensive experiments are conducted on both real-world and synthetic data over two real road networks. The results demonstrate that, comparing to a baseline method, our proposed solutions significantly improve the efficiency by orders of magnitude.

The rest of this paper is organized as follows. We give a brief overview of related work in next section. In Section 3, we discuss user movement and model mobile objects using reference locations. Section 4 formally gives the definition of the MILE-RUN problem. We propose our solutions to MILE-RUN in Section 5. Section 6 analyzes the cost of the proposed methods. Section 7 reports the empirical study and the last section concludes this paper. A set of frequently-used notations and acronyms are listed in Table 1.

2. Related work

There has been increasing research efforts to address the location selection (LS) problem due to its importance in real-life applications. They are generally classified into two major categories: *Min-dist* and *Max-inf*. The former problem aims to find a location, if a new facility is built there, minimizing the aggregate distance from each user to her closest facility. The latter one returns an optimal location for establishing a new facility such that it can attract the most number of users. In this section, we briefly review these LS efforts over the spaces of different distance metrics, *e.g.,* L_p-norm space and road network.

2.1. Min-dist location selection

Zhang et al. [30] introduced the Min-dist LS problem in continuous L_1-norm space. Using lower-bound estimators for pruning, they first identified a set of candidate locations from the given region and then divided them progressively until the answer is found. Qi et al. [14] studied a similar problem given an extra set of candidate locations for establishing a new facility. They developed several methods, with or without additional index structure, to solve the problem on L_2 distance metric. In [15], the authors further extended the Min-dist problem in which a new facility is set up elsewhere to replace an existing one. Unlike the case for L_p-norm distance, where the aim is to decrease the amount of candidates to be calculated, the key challenge for road network distance is to reduce the number of users that require network traversals, then these works are not suitable for solving the proposed MILE-RUN problem in this paper.

Xiao et al. [26] proposed a unified framework utilizing the divide-and-conquer paradigm to address both Min-dist and Max-inf LS problems on continuous road edges. They first computed Attraction Set based on Attractor Distances of users. Road network was then split into sub-networks, among whose local optima the global answer can be derived. Chen et al. [3] studied MaxSum (*i.e.,* Max-inf) and MinMax (different from Min-dist) LS problems following the continuous setting. They presented a pruning technique based on Nearest Location Component (an alias of Local Network [7]). They further studied two extensions, including multiple-location and on three-dimensional road networks. As discussed in [15], however, in many real applications, we can only choose from some candidate locations for rent or sale rather than anywhere on roads.

Table 1
Notations and Acronyms.

Notation & Acronym	*Definition*
$G(V, E)$	road network with vertices V and edges E
$d(\ell_1, \ell_2)$	shortest network distance from location ℓ_1 to ℓ_2
$d_e(\ell_1, \ell_2)$	Euclidean distance between locations ℓ_1 and ℓ_2
U, F, C, W	sets of mobile users, existing facilities, candidate locations and all possible worlds
u, f, c, w	a user, a facility, a candidate and a possible world, respectively
$nn(F, \ell)$	network nearest facility of ℓ chosen from F
$nnd(F, \ell)$	the distance between ℓ and $nn(F, \ell)$
r	a reference location
$L(u)$	the reference locations of u
$l(\cdot)$	the reference location where "$\cdot$" currently appears at
$\|F\|, \|C\|, \|W\|, n$	cardinalities of F, C, W and $L(u)$, respectively
$m, \|U\|$	interchangeably used as the cardinality of U
$Pr[l(u) = r]$	the probability that u presents at r
$Pr[l(U) = w]$	the probability that w occurs
$E[nnd(F, L(u))]$	the expected nearest facility distance of u
$\Delta_w(c)$	reduction of total distance in w if c is built
$\Delta(c)$	expected reduction of total distance for c
$NNFC(F, r)$	the network nearest facility circle of r with F
$\delta_r(c)$	expected reduction of nearest facility distance of r if c is built
$\delta_r^+(c),\ \Delta^+(c)$	upper bounds of $\delta_r(c)$ and $\Delta(c)$
$R_c^+,\ R_c^-$	set of reference locations (not) affected by c
$v_iv_j, r\overrightarrow{v_iv_j}$	an edge with endpoints v_i and v_j, and a directed edge with respect to r
$\sigma^+(r\overrightarrow{v_iv_j}), \sigma^-(r\overrightarrow{v_iv_j})$	maximum and minimum reductions of distance on $r\overrightarrow{v_iv_j}$ for any candidate
$\Delta^+(v_iv_j)$	*ERD* upper bound of edge v_iv_j
LS	Location Selection
ERD	Expected Reduction of total Distance with respect to a candidate c
NSJ	the Network nearest facility circle based Spatial Join method
SJ	the Spatial Join process of the NSJ method
MBR	Minimum Bounding Rectangle
LNB	the Local Network Based method
LNT	Local Network Table used in the LNB method
EN	the Extended Network method
CAT	Candidate Accumulative Table used in the EN method
AP	Average Precision

This means that in general the answers of these approaches may not even be contained in given candidate locations. In addition, based on the empirical comparison with our proposed algorithms (detailed in Section 7), the solutions for continuous setting are not as efficient as for specific candidate locations. Hence, they are not applicable in our case.

Papadias et al. [12] focused on another kind of Min-dist problem, called Aggregate Nearest Neighbor (ANN), which considers only existing facilities without building any new one. The authors proposed methods and the corresponding extensions for processing both memory-resident and disk-resident ANN queries on L_2 distance metric. There exist more efforts [17,27,29,32] in road network. To solve the problem, Yiu et al. [29] presented techniques utilizing network connectivity information and spatial locality. In [32], network Voronoi diagram (NVD) was firstly applied for addressing ANN query. Using the look-up tables of NVD generated in advance, network retrieval is then avoided. Yan et al. [27] extended the problem where all the locations on roads are candidates. They designed a two-phase convex-hull-based pruning technique for both exact and approximate solutions. Sun et al. [17] studied Merged ANN query, which takes into account an extra set of candidate users. As these efforts are orthogonal to our problem setting, they cannot be applied to address MILE-RUN.

2.2. Max-inf location selection

Xia et al. [25] studied a typical Max-inf problem in L_2 space. With a novel distance metric called minExistDNN, the authors developed a branch and bound method to find top-k optimal facilities. Huang et al. [9] investigated further with existing facilities. Two pruning algorithms were exploited utilizing the concept of influential region. Du et al. [5] worked on weighted objects under L_1 distance metric. They presented plane-sweep algorithms based on aSB-tree and Virtual OL-tree for query efficiency. Wong et al. [23] studied MaxBRkNN problem that finds an optimal region instead of a location. They took advantage of region-to-point transformation to tighten the L_2 continuous search space. In [22], the authors extended the problem in three-dimensional and L_p-norm metric spaces. Gao et al. [6] studied the problem on metric space, given a region and a distance as additional constraints.

Besides [3,26] discussed above, Ghaemi et al. [7] also solved a Max-inf problem (MaxOSN) in road network based on a novel spatial locality concept called Local Network. Specifically, the local network of a user u is a sub-network expanded from u (*i.e.,* by Dijkstra algorithm [4]) that contains all points on the network with a distance less than or equal to that between u and her nearest facility.

Apparently, Max-inf based techniques have a completely different optimization objective from our problem setting.

2.3. Efforts on mobile objects

Bespamyatnikh et al. [1] studied a Min-dist problem on mobile users. As they aimed to build a new facility without existing ones in L_2 plane, the presented algorithms, both for exact and approximate answers, cannot be applied for addressing the MILE-RUN problem.

Cheema et al. [2] modeled users under possible world semantics in L_2 space. They developed Normalized Antipodal Half-Space to solve the PRNN problem. Wang et al. [21] presented a generalized Max-inf problem, called PRIME-LS, taking movements of users into account. They proposed pruning rules on a novel distance measure to address the problem. As these efforts focus on Max-inf, they are not suitable for our problem.

Shang et al. [16] studied R-PNN problem in road network, which finds an optimal candidate with the most reverse nearest trajectories. Tang et al. [18] studied an opposite problem, k-NNT, that retrieves top-k trajectories with the minimum aggregated distance to a set of locations. This kind of queries only consider the nearest sample point of a trajectory instead of the whole movement, then they are also orthogonal to our problem setting.

3. Preliminaries

In this section, we first discuss the representation of mobile objects, and then introduce kernel method, which is employed to model the movements of users in our framework.

3.1. Mobile objects

In general, mobile objects (*e.g.,* human beings, animals, etc.) are ubiquitous in many real-life applications. The movement of a user is commonly represented in two ways: continuous case (*e.g.,* trajectory [18]) and discrete case (*e.g.,* check-ins data [28]). The former case explicitly records every move of an object, while the latter one works in an implicit way, which reflects activities at POIs (Point-of-interests) and movements between locations. For both cases, a moving user is modeled as a set of spatial positions [21] (*e.g.,* sampled points of a trajectory or check-ins at POI locations). Nevertheless, in addition to the costly computation, taking every position into account in addressing LS problems is also inappropriate for effectiveness issue, due to three kinds of valueless points: noisy, passing-by and outlier points. Noisy points result from data or GPS errors, whereas the latter two are relevant to mobile characteristics. Compared to places where daily activities are conducted, passing-by points are those places where a user does not spend any time except passing by. For outlier points, a user visits there purely occasionally. Apparently, these positions are not/poorly contributive for deciding where is optimal to establish the new branch. As a result, it is more reasonable to focus on the representative points, which can describe the frequent activity areas of a user.

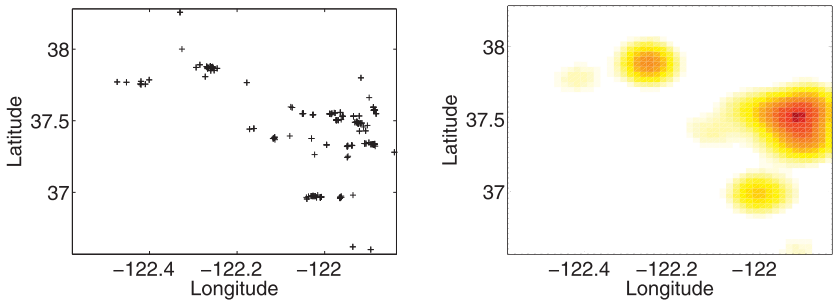
On the other hand, as discussed in [11], moving users are closely associated with two major concepts: frequently appearance at some place and staying for a duration. Incorporating the idea with Tobler's First Law of Geography "Everything is related to everything else, but near things are more related than distant things" [20], two indications can be derived in LS problems. Firstly, a user usually correlates to some reference locations [11]. Secondly, a user tends to conduct her activities nearby the reference locations [28].

In summary, it makes sense to identify reference locations from raw movement data, which help us get rid of the aforementioned limitations and pave the way to capture daily activity places for handling MILE-RUN problems.

3.2. Capturing reference locations

In order to find reference locations, we employ *kernel method* [24]. The original purpose of kernel method is to detect home ranges of animals, and it has been widely used in a variety of domains [19]. In finding activity places for humans, kernel method outperforms most traditional algorithms (*e.g.,* time-distance cluster detection) in accuracy (the accuracy is up to 92.3%) [19], thus can provide more effective solutions to the MILE-RUN problem. Additionally, kernel method is of flexible form, thus can be used where simple parametric models are difficult to specify [24]. Accordingly, kernel method is more suitable to our case where massive users are with different distributions in location, compared to the parametric methods. In this study, we use the standard bivariate normal density kernel [24] to capture reference locations.

To capture reference locations of each user, in line with [11], we also discretize the continuous space into grids and evaluate the density for each of them. The top-p% (5% in this paper) grids with the highest density value are selected. Inspired by reference spot [11], in this paper, we aggregate the adjacent grids among the selected ones together to form a series of separate grid groups. In each grid group, the peak grid with highest density is intuitively regarded as a reference location. Moreover, the density accumulation of each group is normalized and further viewed as the probability that a user appears nearby the corresponding reference location. For instance, Fig. 2(a) and (b) illustrate positions (from check-in data) of a user and the probabilities of grids calculated using kernel method. Based on kernel method and grid group strategy, the valueless points discussed in Section 3.1 can be avoided and three of her reference locations are captured.



(a) Positions of a user. (b) Kernel distribution.

Fig. 2. Capturing reference locations.

Notably, in practice, reference locations usually can be reached by road network, which means they locate on some edges in a given road network graph G, otherwise, we assume a reference location locates at its nearest point in G. This assumption is reasonable, as residential and office buildings may not exactly locate on roads in G.

4. MILE-RUN problem

In this section, we formally define the MILE-RUN problem. Based on reference location, mobile objects are modeled following *possible world* semantics. Then candidate locations are ranked by *expected distance* over all possible worlds. We begin by introducing some terminologies that are necessary for the formal definition.

A location ℓ in this paper is a two-dimensional position on an edge in a given directed road network $G(V, E)$, with a geographical coordinate (*i.e.,* latitude and longitude). Each directed edge in E is associated with a positive weight, which represents travel distance or time cost, etc. Given any two locations ℓ_1 and ℓ_2, the directed network distance (*i.e.,* shortest path distance) from ℓ_1 to ℓ_2 is denoted by $d(\ell_1, \ell_2)$, and $d(\ell_1, \ell_2)$ may not be equal to $d(\ell_2, \ell_1)$. Since the locations of existing facilities (*resp.,* candidates to deploy a new facility) are usually static and thus can be obtained precisely, we denote existing facilities (*resp.,* candidates) as a set of locations $F = \{f_1, f_2, \ldots, f_{|F|}\}$ (*resp.,* $C = \{c_1, c_2, \ldots, c_{|C|}\}$), where $|F|$ (*resp.,* $|C|$) is the cardinality of F (*resp.,* C). Given a location ℓ and a set of facilities F, the network nearest facility of ℓ with respect to F is denoted as $nn(F, \ell)$, and the network distance between them is defined as $nnd(F, \ell) = d(nn(F, \ell), \ell)$. For ease of discussion, in the following, directed network distance and network nearest facility are referred to as distance and nearest facility for simplicity, respectively. If a new facility is built at candidate location c, the distance between ℓ and her nearest facility is $nnd(F \cup \{c\}, \ell)$. Obviously, $nnd(F \cup \{c\}, \ell) \leq nnd(F, \ell)$.

To minimize the average distance between users and their respective nearest facilities, we have to identify her location ℓ first. However, as discussed in Section 3, a mobile object u may be present at a set of n reference locations $L(u) = \{r_1, r_2, \ldots, r_n\}$. Let $l(\cdot)$ denote a reference location(s) where "$\cdot$" is(are) present at, then $l(u) \in L(u)$ and $\sum_{i=1}^{n} Pr[l(u) = r_i] = 1$. This differs from the setting of classical Min-dist criteria, where each object has only a single location. *How can we select the Min-dist optimal location given that objects are mobile?*

Considering the uncertain movements, users can be described following *possible world* semantics. Given a set of users $U = \{u_1, u_2, \ldots, u_m\}$, where m is the cardinality of U, each user u_i is associated with reference locations $L(u_i)$, then a possible world $w = (r_1^w, r_2^w, \ldots, r_m^w)$ is a list of location instances with one instance for each user, where $r_i^w \in L(u_i)$. Assume the reference locations of any user are independent with each other, then $Pr[l(U) = w] = \prod_{i=1}^{m} Pr[l(u_i) = r_i^w]$. Let W be all the possible worlds, then $|W| = \binom{|L(u_1)|}{1} \ldots \binom{|L(u_m)|}{1} = \prod_{i=1}^{m} |L(u_i)|$. Obviously, $\sum_{w \in W} Pr[l(U) = w] = 1$.

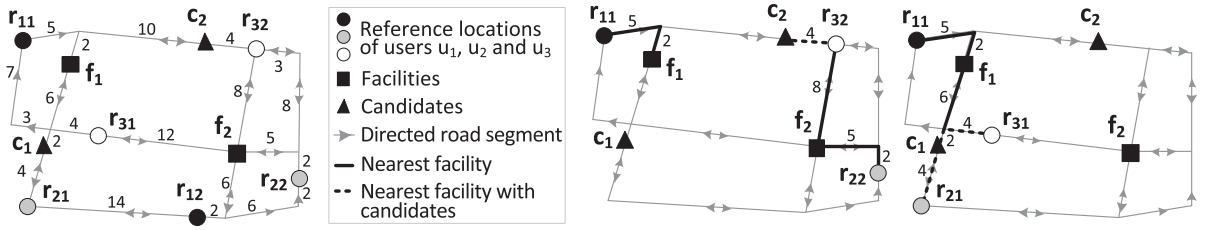
For a particular possible world, each user is associated with only a single reference location, which is in line with the Min-dist problem setting. Hence, we can derive the reduction of average distance between users and their respective nearest facilities, if a new facility is established on candidate location $c \in C$ in a specific possible world.

Definition 1. Given a set of facilities F, a possible world w and a candidate location c, the **reduction of total distance** between all the users U and their respective nearest facilities is defined as $\Delta_w(c) = \sum_{i=1}^{m} (nnd(F, r_i^w) - nnd(F \cup \{c\}, r_i^w))$.[1]

Thus the reduction of average distance is $\Delta_w(c)/m$. Since the denominator m is the cardinality of users and is consistent for every $c \in C$, $\Delta_w(c)$ can be an alternative to evaluate the reduction of average distance. Hence, the candidate location with the maximal $\Delta_w(c)$ is the optimal one in w with respect to the Min-dist criteria.

As illustrated in Fig. 3(a), there are two existing facilities (*i.e.,* $F = \{f_1, f_2\}$), two candidates (*i.e.,* c_1, c_2) and three users (*i.e.,* $U = \{u_1, u_2, u_3\}$), each of which has two reference locations. The numbers and arrows on the edge segments indicate

[1] For clarity and brevity, we omit F and U in the representation, as they are regarded as default settings in definitions in this paper.



(a) Users, facilities & candidates. (b) $w = (r_{11}, r_{22}, r_{32})$. (c) $w' = (r_{11}, r_{21}, r_{31})$.

Fig. 3. Reduction of total distance.

the weights and directions. In this paper, the proposed problem and methods apply to directed road networks, including bidirectional edges that have different weights in the opposite directions. Nevertheless, for ease of presentation, the both directions of a bidirectional edge are assumed the same weight in our examples. Obviously, there exist $|W| = 8$ possible worlds in total. Fig. 3(b) shows a possible world instance $w = (r_{11}, r_{22}, r_{32})$. According to Definition 1, $\Delta_w(c_1) = 0$ and $\Delta_w(c_2) = 4$, thus candidate c_2 can be found as the optimal solution to Min-dist problem in w. Unfortunately, there are other 7 possible worlds, and c_2 may not always be the optimal one. For example, in Fig. 3(c), c_1 is the optima in $w' = (r_{11}, r_{21}, r_{31})$ with $\Delta_{w'}(c_1) = 12$. Accordingly, we need a mechanism to rank candidates taking all the possible worlds into account. As the distribution of Δ_w for any candidate is a random variable, it is reasonable to evaluate the expected value over all possible worlds.

Definition 2. Given a set F of existing facilities, a set U of users, for all possible worlds W, the **expected reduction of total distance** (**ERD**) with respect to a candidate c is defined as $\Delta(c) = \sum_{w \in W}(\Delta_w(c) \times Pr[l(U) = w])$.

Incorporating the concept of reference location and the expected Min-dist criteria following possible world semantics, we are now ready to define the location selection problem proposed in this paper.

Definition 3. Given a spatial network graph $G(V, E)$, a set of existing facilities F and a set of users U, each of whose movement can be modeled as a set of reference locations, the **Min-dIst Location SElection oveR User MovemeNt** (**MILE-RUN**) problem aims to find the optimal location c among a set C of query candidates such that $\forall c' \in C \backslash \{c\}$, $\Delta(c) \geq \Delta(c')$.

To find the optimal candidate with highest *ERD*, we need to calculate $\Delta_w(c)$ for every $w \in W$, which requires enumerating all possible worlds. Unfortunately, $|W|$ is increasing exponentially with respect to $|U|$. Suppose that each user has only two reference locations, then for m users, the complexity of the *ERD* calculation is $O(2^m)$. Hence, directly computing *ERD* as Definition 2 is unpractical. In order to solve this, in the following we show how the computation of *ERD* can be transformed and addressed from the aspect of reference locations, which can be completed polynomially.

Definition 4. Let r^w be the reference location of a mobile user u in possible world w, then the **expected nearest facility distance** of u with respect to F in all possible worlds W is defined as $E[nnd(F, L(u))] = \sum_{w \in W} nnd(F, r^w) \times Pr[l(U) = w]$.

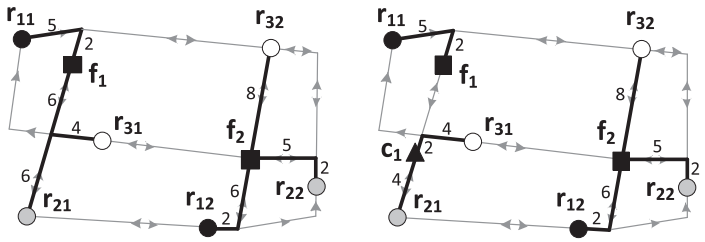
Lemma 1. $E[nnd(F, L(u))] = \sum_{r \in L(u)} nnd(F, r) \times Pr[l(u) = r]$.

Proof. Observe that a user u presents at $r \in L(u)$ with probability $Pr[l(u) = r]$, and for all $|W|$ possible worlds, r occurs in $Pr[l(u) = r] \times |W|$ possible worlds exactly. Then for the possible worlds $W' \subseteq W$ in which a specific r occurs, namely $r^w = r$ if $r^w \in W'$, $\sum_{r^w \in W'} nnd(F, r^w) \times Pr[l(U) = w] = nnd(F, r) \times Pr[l(u) = r]$. Hence, taking into account all $r \in L(u)$, $E[nnd(F, L(u))] = \sum_{r \in L(u)} nnd(F, r) \times Pr[l(u) = r]$. □

Theorem 1. *Given a set of facilities F, a candidate c, and a set of users U, the* ***expected reduction of total distance*** *in Definition 2 can also be computed as* $\Delta(c) = \sum_{i=1}^{m} E[nnd(F, L(u_i))] - E[nnd(F \cup \{c\}, L(u_i))]$.

Proof. Obviously, if we take into account a new facility at candidate location c, $E[nnd(F \cup \{c\}, L(u))] = \sum_{r_i \in L(u)} nnd(F \cup \{c\}, r_i) \times Pr[l(u) = r_i]$. According to Lemma 1, we can transform the computation of *ERD* from the perspective of possible worlds to reference locations as follows.

$$\begin{aligned}
\Delta(c) &= \sum_{j=1}^{|W|} \Delta_{w_j}(c) \times Pr[l(U) = w_j] \\
&= \sum_{j=1}^{|W|} \sum_{i=1}^{m} nnd(F, r_i^{w_j}) \times Pr[l(U) = w_j] \\
&\quad - \sum_{j=1}^{|W|} \sum_{i=1}^{m} nnd(F \cup \{c\}, r_i^{w_j}) \times Pr[l(U) = w_j]
\end{aligned}$$



(a) ERD. (b) ERD with c_1.

Fig. 4. Total expected nearest facility distance.

$$= \sum_{i=1}^{m} E[nnd(F, L(u_i))] - \sum_{i=1}^{m} E[nnd(F \cup \{c\}, L(u_i))]$$
$$= \sum_{i=1}^{m} E[nnd(F, L(u_i))] - E[nnd(F \cup \{c\}, L(u_i)]$$

□

According to Theorem 1, we no longer need to enumerate all possible worlds, and the complexity significantly drops to $O(m)$, which is linearly correlated with the number of users. Suppose that each reference location of a user in Fig. 3(a) occurs with the same probability, say 0.5. As illustrated in Fig. 4(a) and (b), $\sum_{i=1}^{3} E[nnd(F, L(u_i))] = 26$ and $\sum_{i=1}^{3} E[nnd(F \cup \{c_1\}, L(u_i))] = 20$ (if candidate c_1 is built), and thus $\Delta(c_1) = 6$. Similarly, $\Delta(c_2) = 2$. Hence, c_1 has higher expected value and will be selected as the optimal location for the MILE-RUN problem.

5. Solutions to MILE-RUN

In light of Theorem 1 and Definition 3, a straightforward solution to the MILE-RUN problem is to exhaustively check all candidate locations. It works as follows. For each $c \in C$, we compute its $\Delta(c)$. Then the candidate with the greatest *ERD* is the optimal answer. Despite the avoidance of enumerating all possible worlds for expected Min-dist, this method is still expensive due to the massive number of network traversals that occur in repeatedly finding the nearest facility of each reference location for every candidate. In this section, we shall present a series of techniques for solving MILE-RUN problem with superior efficiency, where two methods are based on spatial locality index structures and another answers the problem from scratch.

5.1. Network nearest facility circle based method

Intuitively, we observe that a new established facility may only *affect* (*i.e.,* be the new nearest facility of) a small fraction of reference locations, in light of series of existing facilities. Furthermore, the *ERD* of this new facility only depends on those reference locations that are affected, as the nearest facility distances of other ones remain the same. Motivated by the observations, we present a solution called *Network Nearest Facility Circle*, which focuses on only those affected reference locations for computing *ERD*.

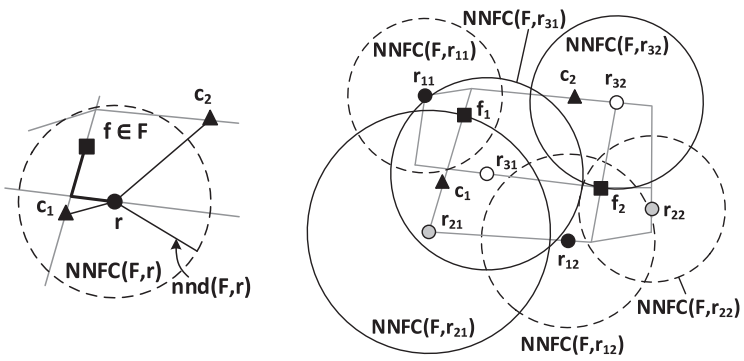
Lemma 2. *Let* $\delta_r(c) = (nnd(F, r) - nnd(F \cup \{c\}, r)) \times Pr[l(u) = r]$ *be the expected reduction of nearest facility distance of r with respect to c, then* $\Delta(c) = \sum_{r \in R_c^+} \delta_r(c)$, *where* $R_c^+ = \{r | r \in L(u) \wedge u \in U \wedge nn(F \cup \{c\}, r) = c\}$.

Proof. For users in U, we denote R_c^- as the set of all their reference locations that are not affected by the newly established c, then all reference locations $R = \bigcup_{u \in U} L(u) = R_c^+ \cup R_c^-$. If $r \in R_c^-$, then $nnd(F, r) = nnd(F \cup \{c\}, r)$ and $\sum_{r \in R_c^-} \delta_r(c) = 0$. Hence, $\Delta(c) = \sum_{r \in R_c^+} \delta_r(c)$. □

As an example in Fig. 4(b), when c_1 is built, only r_{21} and r_{31} change their nearest facility into c_1. Besides, $\Delta(c_1)$ exactly equals to $\delta_{r_{21}}(c_1) + \delta_{r_{31}}(c_1) = (12 - 4) \times 0.5 + (10 - 6) \times 0.5 = 6$. Motivated by this, the computation cost of MILE-RUN problem would be significantly reduced, if we could focus on only those reference locations affected by candidates. The *Incremental Euclidean Restriction* paradigm, which was originally presented for nearest neighbor queries in road networks [13], can be adapted for filtering out reference locations which cannot be affected by candidates.

Definition 5. Given a network graph $G(V, E)$, a set of existing facilities F, the **network nearest facility circle** (**NNFC**) of a user u at reference location $r \in L(u)$, denoted by *NNFC*(F, r), is a circle centered at r with radius $nnd(F, r)$.

In Fig. 5(a), the bold edge segments denote the actual network distance from r to its nearest facility $f \in F$, namely $nnd(F, r)$, and the dashed circle is exactly the *NNFC*(F, r).



(a) NNFC of r. (b) NNFCs.

Fig. 5. NNFC.

Lemma 3. *Given a NNFC(F, r), any facility lies outside it cannot be the nearest facility of r.*

Proof. Let $d_\varepsilon(\ell, \ell')$ represent the Euclidean distance. Fig. 5(a) illustrates that a candidate facility c_2 lies outside *NNFC(F, r)*, and $d_\varepsilon(r, c_2) > nnd(F, r)$. As the network distance between two locations is at least as large as the Euclidean one, *i.e.*, $d(r, c_2) \geq d_\varepsilon(r, c_2)$, then c_2 cannot be the nearest facility of r. □

In light of Lemmas 2 and 3, only the reference locations whose NNFCs enclose a certain candidate c may contribute to $\Delta(c)$, therefore, the others can be pruned in the *ERD* computation. In Fig. 5(b), as c_1 locates inside *NNFC(F, r_{21})* and *NNFC(F, r_{31})*, we can derive $\Delta(c_1)$ from $\delta_{r_{21}}(c_1)$ and $\delta_{r_{31}}(c_1)$, which is in line with the case in Fig. 4(b). Similarly, $\Delta(c_2)$ only correlates to r_{32}. As the enclosure relation is on the basis of Euclidean distance, existing spatial index techniques, such as *R-tree* [8], can be applied for better efficiency. In particular, for all the reference locations, we precompute their corresponding nearest facility distances, and then utilize an R-tree RT_r to index the *minimum bounding rectangle* (*MBR*) of every NNFC. We organize candidate locations C by another R-tree RT_c. Given a query with series of candidate locations C, we can quickly identify all NNFCs that enclose each specific candidate with the help of RT_r. Since all candidate locations in RT_c need to identify the corresponding NNFCs, a *spatial join* between the two R-trees is exactly suitable, which finds out all intersected NNFC-candidate pairs.

Furthermore, based on spatial join, we have the following two observations. Firstly, as a reference location could be affected by multiple candidates, computing distances for all NNFC-candidate pairs results in redundant network traversals. Secondly, candidates probably have skewed *ERD* distribution, then the distance computations for candidates with relatively lower *ERD*s might be unnecessary. Hence, in light of the fact that the computation cost of network distance in a large network is more expensive than that of Euclidean distance, we employ a *delay computation* policy, based on Euclidean *ERD* upper bound and Max Heap, to avoid both redundant and unnecessary computations.

Lemma 4. *If candidate c locates inside NNFC(F, r), the upper bound of $\delta_r(c)$, denoted by $\delta_r^+(c)$, is $(nnd(F, r) - d_\varepsilon(r, c)) \times Pr[l(u) = r]$.*

Proof. As c locates inside *NNFC(F, r)*, the actual reduction of distance is $nnd(F, r) - d(r, c)$. Obviously, $d(r, c) \geq d_\varepsilon(r, c)$, then $nnd(F, r) - d_\varepsilon(r, c) \geq nnd(F, r) - d(r, c)$, and thus $(nnd(F, r) - d_\varepsilon(r, c)) \times Pr[l(u) = r]$ is the upper bound of $\delta_r(c)$. □

Definition 6. Given a candidate c and a set R_c of reference locations whose *NNFC*s enclose c, the **ERD upper bound** of c is denoted as $\Delta^+(c) = \sum_{r \in R_c} \delta_r^+(c)$.

ERD upper bound, by which we use a *Max Heap CH* to order candidates, avoids computing distance for every NNFC-candidate pair. Each entry of *CH* is in the form of $\langle c, \Delta^+(c), R_c \rangle$. Utilizing *CH*, we can easily prune inferior candidates whose $\Delta^+(c)$ are less than the currently found *ERD*, and thus the network traversals for reference locations correlated to the candidates are no longer needed. By incorporating spatial join and delay computation, we develop an algorithm, called *NNFC Spatial Join*, for solving MILE-RUN. Algorithms 1 and 2 show the detailed steps.

We first initialize the optimal location $o\ell$, $\Delta(c)$s for all candidates, *CH* and a HashMap *HM* (lines 1–3). For each reference location *r*, *HM* records the number of candidates enclosed by *NNFC(F, r)*, namely the maximum number of candidates that could affect *r*. Then, we begin the spatial join process from the root nodes of RT_c and RT_r (line 4). Algorithm 2 (SJ) recursively retrieves nodes on both R-trees, until all candidate entries are checked. Specifically, for any non-leaf node N, if one of its entry $e \in N$ intersects a node in the other R-tree, the child nodes of e (*i.e.*, e_c.*child*, e_r.*child*) will be recursively explored (SJ, lines 1–12). If nodes in RT_c and RT_r are both leaves, we update *CH* entry of c and increase the number of candidates enclosed by *NNFC(F, r)* (SJ, lines 13–17). When spatial join is done, facilities in F and candidates in C are added to network $G(V, E)$ to augment existing vertices for subsequent network traversals (line 5). Specifically, for edges on which some facilities and/or candidates locate, each of them is split into edge segments by the facilities and/or candidates, thus an augmented network

Algorithm 1: NNFC Spatial Join (NSJ).

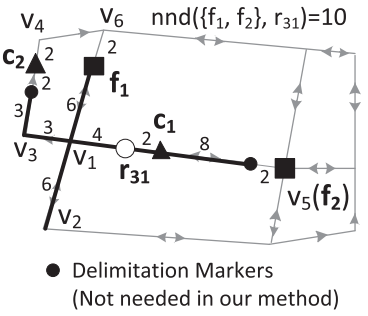
Input: road network $G(V, E)$, candidates R-tree root node N_c, NNFCs R-tree root node N_r
Output: the optimal location $o\ell$

initialize $o\ell := NULL$, $\Delta(o\ell) := 0$ and $\Delta(c) := 0$ ($\forall c \in C$);
initialize a global CH that will be ordered by $\Delta^+(c)$;
initialize a HashMap $HM := \emptyset$ of reference locations;
SJ(N_c, N_r, CH, HM);
augment $G(V, E)$ with F and C as extra vertices, *i.e.*, $G'(V', E')$ where $V' = V \cup F \cup C$;
while $CH \neq \emptyset$ **do**
    heap entry $\langle c_t, \Delta^+(c_t), R_{c_t})\rangle := Pop(CH)$;
    if $\Delta^+(c_t) \leq \Delta(o\ell)$ **then**
        return $o\ell$;
    foreach $r_i \in R_{c_t}$ **do**
        if $HM(r_i) = 0$ **then**
            continue;
        while do
            traverse G' from r_i (by *Dijkstra algorithm*) until a candidate or facility vertex $v \in V'$ is found;
            if v *is a candidate vertex* **then**
                $\delta_{r_i}(v) := (nnd(F, r_i) - d(r_i, v)) \times Pr[l(u) = r_i]$;
                $\Delta(v) := \Delta(v) + \delta_{r_i}(v)$;
                $HM(r_i) := HM(r_i) - 1$;
                if $HM(r_i) = 0$ **then**
                    break;
            else if v *is a facility vertex* **then**
                $HM(r_i) := 0$;
                break;
    if $\Delta(c_t) > \Delta(o\ell)$ **then**
        update $o\ell := c_t$ and $\Delta(o\ell) := \Delta(c_t)$;
return $o\ell$;

Algorithm 2: Spatial Join (SJ).

Input: candidates R-tree node N_c, NNFCs R-tree node N_r, CH and HM

if N_c *and* N_r *are non-leaf nodes* **then**
    foreach $\langle e_c, e_r\rangle \in N_c \times N_r$ **do**
        if $e_c.MBR \cap e_r.MBR \neq \emptyset$ **then**
            invoke SJ with $e_c.child$, $e_r.child$;
else if N_c *is a leaf node* $\wedge$ N_r *is a non-leaf node* **then**
    foreach $e_r \in N_r$ **do**
        if $N_c.MBR \cap e_r.MBR \neq \emptyset$ **then**
            invoke SJ with N_c, $e_r.child$;
else if N_c *is a non-leaf node* $\wedge$ N_r *is a leaf node* **then**
    foreach $e_c \in N_c$ **do**
        if $e_c.MBR \cap N_r.MBR \neq \emptyset$ **then**
            invoke SJ with $e_c.child$, N_r;
else if N_c *and* N_r *are leaf nodes* **then**
    foreach $e_c \in N_c \wedge e_c \cap N_r \neq \emptyset$ **do**
        foreach $e_r \in N_r \wedge e_c \cap e_r \neq \emptyset$ **do**
            update $\Delta^+(c) := \Delta^+(c) + \delta_r^+(c)$ and $R_c := R_c \cup \{r\}$ of heap entry c in CH;
            $HM(r) := HM(r) + 1$;



(a) Local network of r_{31}.

E	Δ^+(E)	Pr[l(u)=r]	σ^+	σ^-	offset
v_5v_1	5 + ...	Pr[l(u_3)=r_{31}]=0.5	10	6	0
		...	...	...	...
v_1v_5	5 + ...	Pr[l(u_3)=r_{31}]=0.5	10	-2	0
		...	...	...	...
v_1v_6	3 + ...	Pr[l(u_3)=r_{31}]=0.5	6	-2	0
		...	...	...	...
v_1v_2	3 + ...	Pr[l(u_3)=r_{31}]=0.5	6	0	0
		...	...	...	...
v_1v_3	3 + ...	Pr[l(u_3)=r_{31}]=0.5	6	3	0
		...	...	...	...
v_3v_4	1.5 + ...	Pr[l(u_3)=r_{31}]=0.5	3	-4	0
		...	...	...	...

(b) Local network table (LNT).

Fig. 6. Local network based method.

$G'(V', E')$ is obtained, where $V' = V \cup F \cup C$, E' consists of the undivided edges and the split edge segments. In other words, the road network is now composed of road intersection vertices, existing facility vertices and candidate vertices. Afterwards, we iteratively check the top candidate c_t in CH (lines 6 and 7). Once $\Delta^+(c_t) \leq \Delta(o\ell)$ holds, no candidate can be better than $o\ell$, then $o\ell$ is the answer (lines 8 and 9). Otherwise, we verify each r_i, whose NNFC covers c_t, by traversing $G'(V', E')$ (*e.g.,* by Dijkstra algorithm) until all candidates that can affect r_i are found or the nearest facility of r_i is found. Before the traversal, we first check whether $HM(r_i)$ equals to zero, which means no candidate is enclosed by $NNFC(F, r_i)$ or r_i has been explored earlier by another candidate c where $r_i \in R_c$. If so, we can immediately probe the next reference location in R_{c_t} (lines 11 and 12). During the traversal from r_i, if a candidate (vertex) v is encountered, with the distance $d(r_i, v)$, we update its $\Delta(v)$ by adding $\delta_{r_i}(v)$. Then we decrease $HM(r_i)$ by 1 to mark the number of remaining candidates that could affect r_i (lines 13–18). After that, if $HM(r_i) = 0$, which means all the candidates that can affect r_i are visited, it is unnecessary to find the nearest facility (lines 19 and 20). On the other hand, if the nearest facility is met, the traversal from r_i is complete and we mark r_i as explored (*i.e.,* set $HM(r_i)$ to 0) (lines 21–23). After the iteration of R_{c_t}, the ERD of c_t is derived. We then update $o\ell$ if necessary and proceed to the next top candidate, and the optimal candidate is finally obtained (lines 24–26).

5.2. Local network based method

With the help of Euclidean spatial locality (*i.e.,* NNFC), it only needs to take the reference locations affected into consideration. Similar to the underlying idea, we expand the NNFC concept to its counterpart in network locality, namely *local network*, such that we can only take into account the reference locations in whose local networks a new facility locates (*i.e.,* affects). Local network concept was proposed to model the network locality of a user with respect to her nearest facility [7]. Specifically, it is a sub-network expanded from user location $l(u) = r$ with a distance less than or equal to $nnd(F, r)$. As shown in Fig. 6(a), the bold edge sections are the local network of r_{31}, where the nearest facility of r_{31} is f_1 and $nnd(F, r_{31}) = 10$. According to Ghaemi et al. [7], which aims to handle a Max-inf problem (MaxOSN), local networks do not contain distance information. As a result, the concept cannot be directly applied to solve our problem. Moreover, in [7], users (reference locations in our case) are viewed as vertices in road network, which results in more complicated network traversals as the number of users is very large. Additionally, in order to delimit the local networks of users, ending points of local networks (referred to as delimitation markers in [7]) are marked and further complicates the network. Fig. 6(a) shows two delimitation markers.

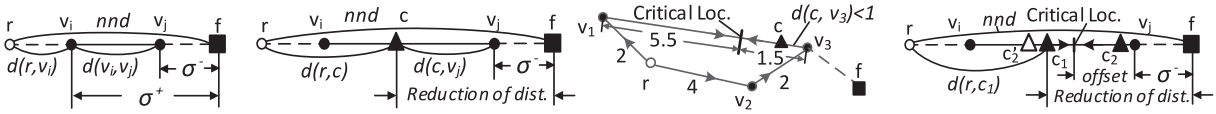
In this section, by incorporating distance factor to the network locality, we present a Local Network based method to address MILE-RUN. Differing from [7], only existing facilities are settled on the road network. Also, we introduce the *expanding direction* of an edge to replace delimitation markers, such that complexity in the evaluation of distance, from a reference location to any point on the edges of its local network, can be significantly reduced.

Definition 7. Given a reference location r and an undirected edge v_iv_j, if a network path expended from r traverses the edge or part of it from v_i towards v_j, we denote the **directed edge with respect to r** as $r\overrightarrow{v_iv_j}$.

For example, v_1v_3 and v_1v_5 are two undirected edges in Fig. 6(a). Expanding local network of r_{31}, v_1v_3 is traversed from v_1 towards v_3, and thus the directed edge with respect to r_{31} is $r_{31}\overrightarrow{v_1v_3}$. The case of v_1v_5 is more complicated. The network path expanded from r_{31} to f_1 traversing v_1v_5 is from v_5 towards v_1, then we denote it as $r_{31}\overrightarrow{v_5v_1}$. To the contrary, for any candidate that locates between r_{31} and f_2 (*e.g.,* c_1), the directed edge is $r_{31}\overrightarrow{v_1v_5}$.

Definition 8. Given $r\overrightarrow{v_iv_j}$, on which a candidate is built, the **maximum and minimum reductions of distance on** $r\overrightarrow{v_iv_j}$ are denoted as $\sigma^+(r\overrightarrow{v_iv_j})$ and $\sigma^-(r\overrightarrow{v_iv_j})$.

As shown in Fig. 7(a), where r accesses the nearest facility f traversing $r\overrightarrow{v_iv_j}$ (the general case that r expands its local network traversing $r\overrightarrow{v_iv_j}$ also holds), $\sigma^+(r\overrightarrow{v_iv_j})$ and $\sigma^-(r\overrightarrow{v_iv_j})$ can be easily computed as $nnd(F, r) - d(r, v_i)$ and $nnd(F, r) -$



(a) σ^+ and σ^-. (b) Compute reduction of distance. (c) Critical location. (d) Reduction of distance with critical location.

Fig. 7. Compute reduction of distance via σ^+ and σ^-.

$d(r, v_j)$. Based on $\sigma^+(r\overrightarrow{v_iv_j})$, we can derive the *ERD* upper bound of any candidate to be built on v_iv_j, which overlaps with the local networks of a set $R_{v_iv_j}$ of reference locations, as $\Delta^+(v_iv_j) = \sum_{r_k \in R_{v_iv_j}} \sigma \times Pr[l(u) = r_k]$, where

$$\sigma = \begin{cases} \sigma^+(r_k\overrightarrow{v_iv_j}) & \text{if } r_k\overrightarrow{v_iv_j}, \\ \sigma^+(r_k\overrightarrow{v_jv_i}) & \text{if } r_k\overrightarrow{v_jv_i}. \end{cases} \tag{1}$$

As there is no road of retrogression when expanding a local network, either of the cases in Eq. (1) holds. On the other hand, based on $\sigma^-(r\overrightarrow{v_iv_j})$, if a candidate c is deployed on $r\overrightarrow{v_iv_j}$, as shown in Fig. 7(b), we can calculate $\delta_r(c)$ as

$$\begin{aligned} \delta_r(c) &= (nnd(F, r) - d(r, c)) \times Pr[l(u) = r] \\ &= (nnd(F, r) - (d(r, v_j) - d(c, v_j))) \times Pr[l(u) = r] \\ &= (\sigma^-(r\overrightarrow{v_iv_j}) + d(c, v_j)) \times Pr[l(u) = r]. \end{aligned} \tag{2}$$

Eq. (2) has three special cases: Firstly, if $\sigma^-(r\overrightarrow{v_iv_j}) + d(c, v_j)$ is negative, which means c locates outside the local network of r, and thus we set $\delta_r(c)$ to zero. Secondly, if $\sigma^-(r\overrightarrow{v_iv_j}) + d(c, v_j)$ is greater than $\sigma^+(r\overrightarrow{v_iv_j})$, namely $d(c, v_j) > d(v_i, v_j)$, which is impossible, then c must not be accessed from v_i to v_j. This indicates the reverse direction $r\overrightarrow{v_jv_i}$ exists and r must locate on v_iv_j. Hence, c takes effect on the direction from v_j to v_i, therefore $\delta_r(c) = (\sigma^-(r\overrightarrow{v_jv_i}) + d(c, v_i)) \times Pr[l(u) = r]$ instead. In this case, v_iv_j can be called a *bidirectional* directed edge with respect to r, and $\sigma^+(r\overrightarrow{v_iv_j}) = \sigma^+(r\overrightarrow{v_jv_i}) = nnd(F, r)$, as a candidate would be exactly built overlapping r. For σ in Eq. (1), its value is accordingly $nnd(F, r_k)$.

For instance, for candidate location c_2 in Fig. 6(a), $\sigma^-(r_{31}\overrightarrow{v_3v_4}) + d(c_2, v_4) = -4 + 2 < 0$. It indicates that c_2 is outside the local network of r_{31}, then $\delta_{r_{31}}(c_2)$ is set to 0. The situation of candidate c_1 is more complex, as $r_{31}\overrightarrow{v_1v_5}$ and $r_{31}\overrightarrow{v_5v_1}$ both exist. As $\sigma^-(r_{31}\overrightarrow{v_5v_1}) + d(c_1, v_1) = 6 + 6 > \sigma^+(r_{31}\overrightarrow{v_5v_1}) = 10$, c_1 brings the reduction of distance from the reverse direction $r_{31}\overrightarrow{v_1v_5}$ instead of $r_{31}\overrightarrow{v_5v_1}$. Hence, $\delta_{r_{31}}(c_1) = (\sigma^-(r_{31}\overrightarrow{v_1v_5}) + d(c_1, v_5)) \times Pr[l(u_3) = r_{31}] = (-2 + 10) \times 0.5 = 4$.

The third special case can be described in Fig. 7(c). Supposing v_1v_3 is a bidirectional road, the shortest path from r to c is $r \to v_2 \to v_3 \to c$ instead of $r \to v_1 \to c$. This is another kind of *bidirectional* directed edge with respect to r. The reason is that, except the shortest path (via v_2v_3), the other paths from r to v_3 (*e.g.*, via v_1v_3) have longer distances (*e.g.*, $d(r, v_3) < d(r, v_1) + d(v_1, v_3)$). Then c is reached via the shortest path from v_3 (*i.e.*, $r\overrightarrow{v_3v_1}$) instead of from v_1 (*i.e.*, $r\overrightarrow{v_1v_3}$). Obviously, there exists one and only one location ℓ_c, called *critical location*, on v_1v_3 such that $d(r, v_1) + d(v_1, \ell_c) = d(r, v_3) + d(v_3, \ell_c)$. If an edge v_iv_j has a critical location ℓ_c, the farthest distance from r to any location of v_iv_j is onto ℓ_c instead of its endpoints, and thus Eq. (2) needs to be adjusted, if c locates on v_iv_j.

Lemma 5. *Given candidate c on a bidirectional v_iv_j with critical location ℓ_c, where $d(r, v_i) < d(r, v_j) < d(r, v_i) + d(v_i, v_j)$. Let* ***offset*** *be $d(\ell_c, v_j)$. If $d(c, v_j) \geq$ offset, Eq. (2) is applicable; otherwise, it is adjusted as $\delta_r(c) = (\sigma^-(r\overrightarrow{v_iv_j}) + 2 \times \text{offset} - d(c, v_j)) \times Pr[l(u) = r]$.*

Proof. According to the definition of critical location, we have $d(r, v_i) + d(v_i, \ell_c) = d(r, v_j) + d(\ell_c, v_j)$ and $d(v_i, \ell_c) = d(v_i, v_j) - d(\ell_c, v_j)$, then $offset = \frac{1}{2}(d(r, v_i) + d(v_i, v_j) - d(r, v_j))$. When $d(c, v_j) \geq offset$, as c_1 in Fig. 7(d), it locates on $r\overrightarrow{v_iv_j}$, then Eq. (2) holds. If $d(c, v_j) < offset$, as c_2 in Fig. 7(d), it is reached from v_j, namely $r\overrightarrow{v_jv_i}$. As the distances from r to locations on v_iv_j are symmetrical with ℓ_c, we can find the symmetry location c_2' of c_2, and $d(r, c_2') = d(r, c_2)$. As Eq. (2) holds for c_2', we can compute $\delta_r(c_2) = \delta_r(c_2') = (\sigma^-(r\overrightarrow{v_iv_j}) + 2 \times offset - d(c, v_j)) \times Pr[l(u) = r]$. □

Based on the above discussion, we design a *Local Network Table* (*LNT*) to organize directed edges. Each entry of LNT is associated with a directed edge and consists of its *ERD* upper bound and a set of reference location entries, each of which is in the form of $\langle Pr[l(u) = r], \sigma^+(r\overrightarrow{v_iv_j}), \sigma^-(r\overrightarrow{v_iv_j}), offset\rangle$. Fig. 6(b) illustrates part of the LNT that is correlated with r_{31} for simplicity. We construct LNT in four steps. Firstly, from a reference location r, we expand network paths via Dijkstra algorithm. For each $r\overrightarrow{v_iv_j}$ traversed, we record $d(r, v_i)$ and $d(r, v_i) + d(v_i, v_j)$. Secondly, after the nearest facility of r is found, we compute for each recorded $r\overrightarrow{v_iv_j}$ the σ^+, σ^- and *offset* based on the correlated distances and $d(r, v_j)$. For $r\overrightarrow{v_iv_j}$, if $offset > 0$, which means a critical location and $r\overrightarrow{v_jv_i}$ exist, we insert the reverse edge into LNT. Thirdly, iteratively scan all reference locations following the two steps above. Finally, we accumulate $\Delta^+(v_iv_j)$s of each v_iv_j with corresponding reference locations.

Utilizing LNT, as shown in Algorithm 3, it does not need any network traversal when querying the optima from given candidates C. We first initialize a Max Heap *LNH* from LNT ordered by $\Delta^+(v_iv_j)$ of edge v_iv_j on which a candidate locates.

Algorithm 3: Local Network Based (LNB).

Input: *LNT*, candidate locations C
Output: optimal location $o\ell$

```
1  initialize LNH based on LNT and C ordered by Δ+(v_i v_j);
2  while LNH ≠ ∅ do
3      ⟨c, Δ+(v_i v_j), {v_i v_j(, v_j v_i)}⟩ := Pop(LNH);
4      if Δ(oℓ) > Δ+(v_i v_j) then
5          return oℓ;
6      else
7          Δ(c) := 0;
8          foreach r associated with v_i v_j in LNT do
9              if offset > d(c, v_j) then
10                 compute δ_r(c) based on Lemma 5;
11                 Δ(c) := Δ(c) + δ_r(c);
12                 continue;
13             d := σ−(r v_i→ v_j→) + d(c, v_j);
14             if 0 < d ≤ σ+(r v_i→ v_j→) then
15                 Δ(c) := Δ(c) + d × Pr[l(u) = r];
16             else if d > σ+(r v_i→ v_j→) then
17                 if v_j v_i exists then
18                     d := σ−(r v_j→ v_i→) + d(c, v_i);
19                     if 0 < d then
20                         Δ(c) := Δ(c) + d × Pr[l(u) = r];
21         if Δ(c) > Δ(oℓ) then
22             update oℓ := c and Δ(oℓ) := Δ(c);
23 return oℓ;
```

Each entry of *LNH* is in the form of $\langle c, \Delta^+(v_iv_j), \{v_iv_j(, v_jv_i)\}\rangle$, where in parenthesis is the reverse directed edge, if it exists. If a candidate is correlated with a bidirectional directed edge v_iv_j, we use $\Delta^+(v_iv_j) + \Delta^+(v_jv_i)$ for ordering (line 1). Then we iteratively check the top candidate c in *LNH* (lines 2 and 3). If $\Delta(o\ell)$ of current optimal candidate $o\ell$ is greater than *ERD* upper bound of the popped edge v_iv_j, the validation is finished (lines 4 and 5). Otherwise, we evaluate $\Delta(c)$ based on σ^+, σ^- and *offset* of the related edge(s). $\Delta(c)$ is initialized to 0 (line 7). For every reference location whose local network covers v_iv_j, we first check whether *offset* $< d(c, v_j)$. If it holds, we compute $\delta_r(c)$ following Lemma 5 and continue to probe the next reference location (lines 8–12). Normally, we compute $\sigma^-(r\overrightarrow{v_iv_j}) + d(c, v_j)$, which is denoted by a temporary variable d in Algorithm 3, and check its value. If the value is valid, we accumulate it to $\Delta(c)$ (lines 13–15). Otherwise, if c is associated with a bidirectional edge, we compute the reduction of distance in the opposite direction (lines 16–20). Notably, in line 19 σ^+ is not checked, as it is impossible that $\delta_r(c)$ is greater than $\sigma^+ \times Pr[l(u) = r]$ in both directions by contradiction. If a better candidate is found, we update $o\ell$ (lines 21 and 22). Afterwards, we probe the unchecked candidates until the optima is finally obtained.

5.3. Extended network method

In both of the methods proposed above, the information of existing facilities and movements of users is assumed to be known in advance, and we can precompute and index them offline, such that an arbitrary query with a group candidates of MILE-RUN problem can be answered efficiently. Nevertheless, in some cases we may not have any idea about the existing facilities or users in advance, which prevents the aforementioned methods being applied directly. *Is it possible to address* MILE-RUN *in an online manner?* In order to solve the problem, in this part we propose a method based on *network extension* such that MILE-RUN can be answered from scratch even if we do not have any index of existing facilities or users. The main idea is to perform the network extension for every reference location to find its nearest facility, and in the process, we progressively accumulate $\delta_r(c)$ for each candidate that is encountered before the nearest facility of every reference location.

We employ a *Candidate Accumulative Table* (CAT) to store the accumulative $\delta_r(c)$ of every candidate. Each entry of CAT is in the form of $\langle c, \sum_{r\in R'} \delta_r(c)\rangle$, where R' denotes the reference locations that encounter c before finding the nearest facilities (*i.e.*, c is nearer to r). For instance, Fig. 8 shows CAT of Fig. 3(a). The method consists of four steps and is outlined in Algorithm 4 . Firstly, we initialize CAT for all candidates. Similar to the augmentation in Algorithm 1, road network $G(V, E)$

Candidate	Accumulation
c_1	$\delta_{r_{21}}(c_1)+\delta_{r_{31}}(c_1)=6$
c_2	$\delta_{r_{32}}(c_2)=2$

Fig. 8. Candidate accumulative table (CAT).

Algorithm 4: Extended Network (EN).

Input: network graph $G(V, E)$, existing facilities F, users U, candidate locations C
Output: optimal location $o\ell$

1 initialize CAT.$c := 0$ ($\forall c \in C$);
2 augment $G(V, E)$ with F and C as extra vertices, *i.e.,* $G'(V', E')$ where $V' = V \cup F \cup C$;
3 **foreach** $u_i \in U$ **do**
4 **foreach** $r_j \in L(u_i)$ **do**
5 initialize $V_2 := \emptyset$;
6 **while do**
7 traverse G' from r_i (by *Dijkstra algorithm*) until a candidate or facility vertex $v \in V'$ is found;
8 **if** v *is a candidate vertex* **then**
9 record $\langle v, d(r_j, v)\rangle$ in V_2;
10 **else if** v *is a facility vertex* **then**
11 **foreach** $\langle c_k, d(r_j, c_k)\rangle \in V_2$ **do**
12 CAT.c_k := CAT.$c_k + \delta_{r_j}(c_k)$; // $\delta_{r_j}(c_k) = (d(r_j, v) - d(r_j, c_k)) \times Pr[l(u_i) = r_j]$
13 release V_2;
14 break;
15 return $o\ell$ with greatest $\Delta(c)$;

is augmented by adding existing facilities F and candidates C as extra vertices (lines 1 and 2). Secondly, for each reference location of users, we explore its nearest facility $f \in F$ by Dijkstra algorithm. During the process, if a candidate vertex $c \in C$ is encountered before f is found, which means $d(r, c) \leq nnd(F, r)$, we temporarily record c and the distance $d(r, c)$ in a two-dimensional Vector V_2 (lines 8 and 9). Thirdly, we traverse the augmented network until the nearest facility f is found. At this point, all the candidates that can affect the reference location have been in V_2. We compute $\delta_r(c)$ for each candidate c in V_2 and accumulate it to the corresponding CAT entry. Afterwards, the temporary V_2 is released and we proceed to the next reference location (lines 10–14). Finally, after the exploration of all users, all the reference locations that a certain candidate c can affect are known, that is $\sum_{r \in R'} \delta_r(c) = \Delta(c)$, and the answer with greatest $\Delta(c)$ can be obtained.

6. Cost analysis

In this section, we conduct theoretical study over all proposed methods (*i.e.,* Straightforward, NSJ, LNB and EN). Observe that the time complexity for finding the nearest facility of a reference location is $O(|V|\log|V|)$, as the network traversal terminates once a facility vertex is found, which means only $|V| + 1$ vertices (1 means the facility vertex) are involved in the exploration.

The straightforward method described in the very beginning of Section 5 iteratively checks candidates based on Theorem 1, without precomputation and index. Obviously, its complexity is $O(|C| \cdot |R| \cdot |V|\log|V|)$, where $|R|$ is the number of reference locations of all users.

NSJ method needs to precompute $nnd(F, r)$s for reference locations, and the time cost is $O(|R| \cdot |V|\log|V|)$. NSJ utilizes an R-tree RT_r to index NNFCs. The height and maximum number of nodes of RT_r can be denoted as $h_r = \lceil \log_{|e|} |R| \rceil$ and $|N| = \lceil \frac{|R|}{|e|} \rceil + \lceil \frac{|R|}{|e|^2} \rceil + \ldots + 1$, respectively, where $|e|$ denotes the average number of entries in a node of RT_r. Then the space complexity of RT_r is $O(|R| + |N|)$. As the insert operation of R-tree runs in $O((h_r - 1) + |e|^k)$ time, where $k = 1$ and $k = 2$ are for the linear and quadratic splitting algorithms, respectively [8], the time complexity of index is $O(|R| \cdot ((h_r - 1) + |e|^k))$.

The query cost of NSJ depends on both the spatial join and max heap procedures. In spatial join process, RT_c is explored in a depth-first order. At most, for each RT_c node, $(h_r - 1) \cdot |e|$ nodes in RT_r is required, that is $|C| \cdot (h_r - 1) \cdot |e|$ for candidates. For querying the max heap CH, in the best case, the top candidate is the optimal one, thus only one R_{c_t} set of reference locations need network traversals. In the worst case, all candidates need to be scanned. Normally, as demonstrated in Section 7, $|C'|$ candidates are explored, where $|C'| \ll |C|$. On the other hand, the number $|R'|$ of reference locations affected by each candidate is very limited compared to $|R|$, namely $|R'| \ll |R|$. Hence, the query complexity is $O(|C| \cdot (h_r - 1) \cdot |e| + |C'| \cdot |R'| \cdot |V| \log |V|)$.

Table 2
Real facilities of CA and BJ.

California(CA)	$\|F\|$	Beijing City(BJ)	$\|F\|$
Bars(BA)	200	*Cafes(CF)*	1500
Hospitals(HO)	800	*Gas Stations(GS)*	1500
Post Offices(PO)	**800**	***Logistics(LO)***	**2500**
Parks(PA)	3200	*Banks(BK)*	2500
Schools(SC)	3200	*Schools(SC)*	2500

For LNB, it also precomputes $nnd(F, r)$ for reference locations, whose time complexity is $O(|R| \cdot |V| \log |V|)$, and LNT is constructed during the process. The best case occurs when each reference location has its nearest facility on the same edge, then LNT contains $|R|$ records and costs $O(|R|)$ time, where the insertion of each record can be run in $O(1)$ time. In the worst case, if there is only one facility, each reference location traverses massive edges to get there, then LNT requires $O(|E| \cdot |R|)$ space and $O(|E| \cdot |R|)$ time.

The query cost of LNB only depends on querying max heap LNH, as it does not need any network traversal when querying candidates. The best (*resp.*, worst) case occurs when only the top entry (*resp.*, all entries) is (*resp.*, are) explored. Experiments in Section 7 show that a very tiny proportion of directed edges $|E'|$ ($|E'| \ll |E|$) are checked. Despite many reference locations are associated with each directed edge, the computation of the exact $\Delta(c)$ is a simple floating-point operation, and its cost can be viewed as $O(1)$. Hence, the query complexity of LNB is $O(|E'|)$.

For EN method, the Candidate Accumulative Table (CAT) consists of $|C|$ entries. As the network is augmented by candidate locations as extra vertices, the time complexity of EN is $O(|R| \cdot (|V| + |C|) \log(|V| + |C|))$.

In summary, in the aspect of query efficiency, we have LNB $\gg$ NSJ $>$ EN $\gg$ Straightforward. Our experimental study will also validate these analyses.

7. Performance study

In this section, we investigate the performance of our proposed MILE-RUN solutions from a variety of aspects.

7.1. Experimental setup

Datasets. We use two real spatial networks, California (CA) [10] and Beijing City (BJ),[2] in the experiments. CA contains 21,693 bidirectional edges and 21,047 vertices. BJ consists of 433,391 unidirectional edges and 171,504 vertices. Comparing CA with BJ, the former is relatively sparse with respect to the area it covers. Table 2 describes the real datasets of existing facilities for CA and BJ. Facilities in each group are randomly chosen from respective datasets, and with different purposes. For CA, facilities are mainly for comparing the impact of cardinality. For BJ, we focus on the effect on different geographical distributions. PO and LO are default groups for CA and BJ, respectively.

We conduct experiments on both real and synthetic datasets of reference locations of users. The real data for BJ, which is available from [31], collected moving users in Beijing, and there are 136,686 sample points for each user on average. For CA, we use the discrete check-in data[3] for representing users. Excluding the users with less than 20 check-ins, the average number of check-ins for a user is about 125. For each user, the average differences between the maximum and minimum present probabilities are 0.278(CA) and 0.367(BJ), respectively, which means the distributions are skewed. Accordingly, we synthesize 1 to 6 reference locations with proportions 5%, 20%, 35%, 25%, 10% and 5% for each of users, the number of which ranges from 20 k to 100 k. Furthermore, we use two schemes, Random and Uniform, to generate present probabilities of reference locations. For the former, probabilities are distributed purely randomly. For Uniform, if a user has n reference locations, the present probability at each location is uniformly set as $1/n$. The geographical location of each reference location is uniformly distributed. For candidate locations, we choose 50, 100, 200, 400 and 800 locations within the geographical domain at random[4] Unless specified otherwise, the default values of $|C|$, m and the distribution of present probability are 200, 60 k and Random, respectively.

Algorithms. Following algorithms are evaluated in the experiments. They are implemented in C++ and tested on a 3.2 GHz quad-core machine with 32GB RAM, running Windows 10 (64 bit).[5]

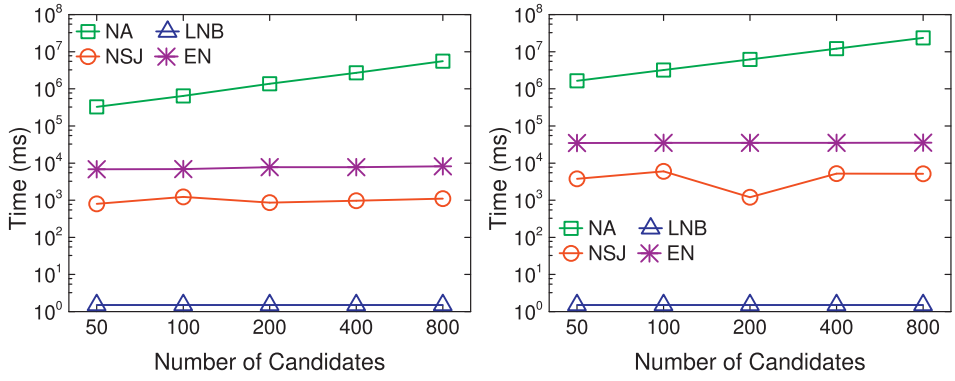
- NA: A baseline solution that exhaustively computes the expected reduction of total distance for each candidate, based on which we retrieve the optimal one.

[2] The road network and facilities datasets are available from http://www.datatang.com/datares/go.aspx?dataid=619746 and http://www.datatang.com/datares/go.aspx?dataid=617882.

[3] Check-ins data in California are obtained and integrated from http://snap.stanford.edu/data/.

[4] For the synthetic objects that are not exactly located on roads, we shift them to the closest point of on the road network.

[5] https://github.com/lihuixidian/MILE-RUN



(a) CA. (b) BJ.

Fig. 9. Effect of |C|.

- NSJ: NNFC Spatial Join method described in Algorithms 1 and 2. The R-trees are loaded in main memory. The maximum number of elements in R-tree nodes is 16.[6]
- LNB: Local Network Based method described in Algorithm 3. The LNT is loaded in main memory.
- EN: Extended Network method that is described in Algorithm 4.

7.2. Experimental results

Effect of |C|. We first investigate the performances when querying with different algorithms. Fig. 9 reports the running time varying the number of candidates. As NA iteratively computes for each candidate, its cost is the worst and increases linearly with |C|. Compared to NA, the other methods significantly reduce the time by several orders of magnitude. LNB shows the best scalability, followed by NSJ and EN. The query time of LNB is stable within milliseconds regardless of |C| and road networks. The reason is twofold. Firstly, the queries are only relevant to the lookup process of LNH without network retrieval. Secondly, the max heap prunes quite a few inferior candidates. The time of EN changes very little as |C| is much smaller than |V| such that it has limited impact on network traversal. The situation of NSJ is more complicated as the efficiency varies irregularly with the number of candidates. There are two factors that determine the running time. One factor is the number of candidates to be checked in max heap CH. The other factor is the number of reference locations correlated with each candidate to be checked. The latter factor is dominant due to the cost of network traversal. The proportions of candidates pruned by max heaps in NSJ and LNB are both high (*e.g.,* $\geq 95\%$). Despite the numbers of checked candidates are very close (*e.g.,* the maximum difference between NSJ and LNB is only 7 candidates), LNB is significantly better than NSJ, which is because network traversal is more expensive than floating-point operation.

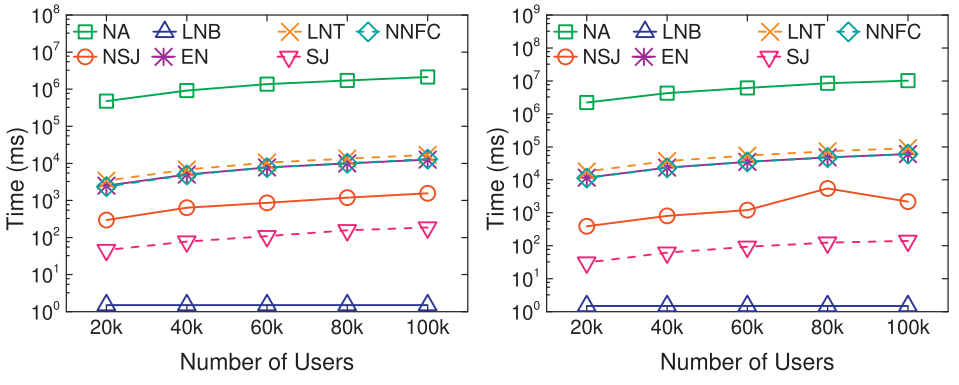
Effect of |U|. In this part, we first test the scalability of the algorithms with respect to the cardinality of users. As illustrated in Fig. 10, the cost mostly increases when the number of users (*i.e.,* reference locations) grows. Obviously, the results are qualitatively similar to Fig. 9. LNB exhibits the best performance, followed by NSJ, EN and NA. LNB is still stable and within milliseconds.

Next, we study the precomputation costs of NSJ and LNB. In line with the theoretical analysis, the complexity of the construction of NNFCs is the same as EN, and the time of constructing LNT is also very close. On the other hand, we also study the cost of spatial join using R-trees (referred to as SJ in Fig. 10). It costs much less compared to network traversal in NSJ, which means Euclidean spatial locality takes effect in pruning. Moreover, the irregular case of 80 k in Fig. 10(b) shows again that the number of reference locations correlated with candidates to be checked is dominant in NSJ.

Effect of *F*. In this part, we study the algorithms on different existing facility groups. Apparently, the efficiencies are qualitatively similar to aforementioned results, where LNB is the best, followed by NSJ, EN and NA. Except LNB, the running cost mostly drops with the increase of |F| for the other algorithms. This is because LNB only depends on float-point computation, while the others require network traversal. Obviously, more facilities lead to shorter network paths on average. Moreover, compared to EN, NSJ is more sensitive to |F|. For instance, in CA, the number of parks is 16 times as many as that of bars. Compared to bars, NSJ and EN reduce the time of parks by two and one order(s) of magnitude, respectively. The reason is the NNFC policy. $nnd(F, r)$ is nearer with shorter network paths, then NNFCs are smaller, which means more candidates can be pruned and less network traversals are needed.

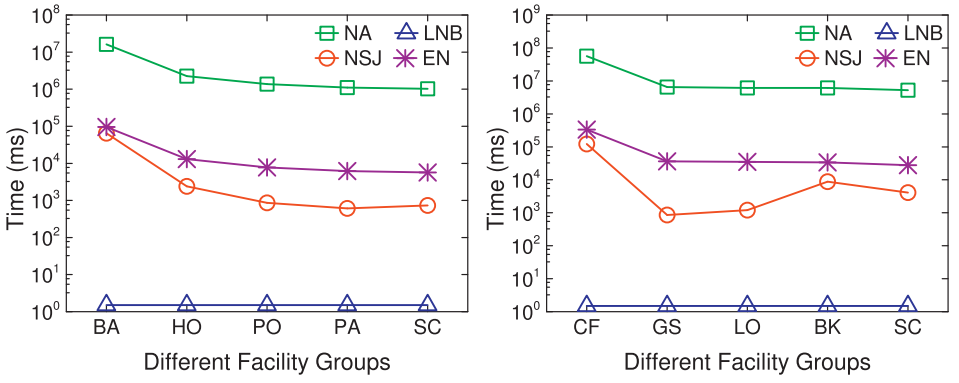
We also explore the effect of geographical distribution of facilities. As shown in Fig. 12, the two facility groups have the same cardinality, while their distribution is completely distinct: Gas stations are relatively uniform, while cafes are skewed.

[6] According to the following experiments, the cost of spatial query using R-tree is small. Hence, the number of elements has limited effect. Also, other variations of R-tree and hierarchical spatial data structures can also be applied.



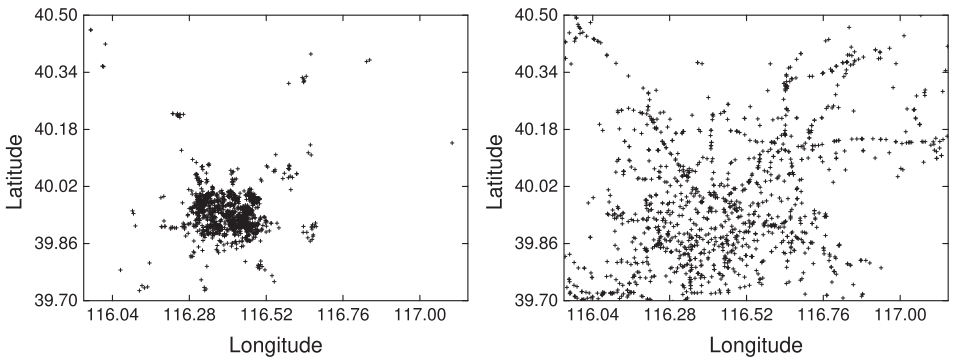
(a) CA (b) BJ

Fig. 10. Effect of $|U|$.



(a) CA (b) BJ

Fig. 11. Effect of $|F|$.



(a) Cafes (BJ) (b) Gas stations (BJ)

Fig. 12. Distribution of facilities.

This results in cost differences of NA, EN and NSJ, where NSJ is still the most sensitive algorithm. For LO, BK and SC in Fig. 11(b), EN shows they are with similar quantity of network traversal. However, due to geographical distribution, the NNFC policy has different pruning powers. Additionally, as BA (*resp.*, CF) shown in Fig. 11(a) (*resp.*, Fig. 11(b)), if facilities are both skewed distributed and with small cardinality, the performance of NSJ will degrade.

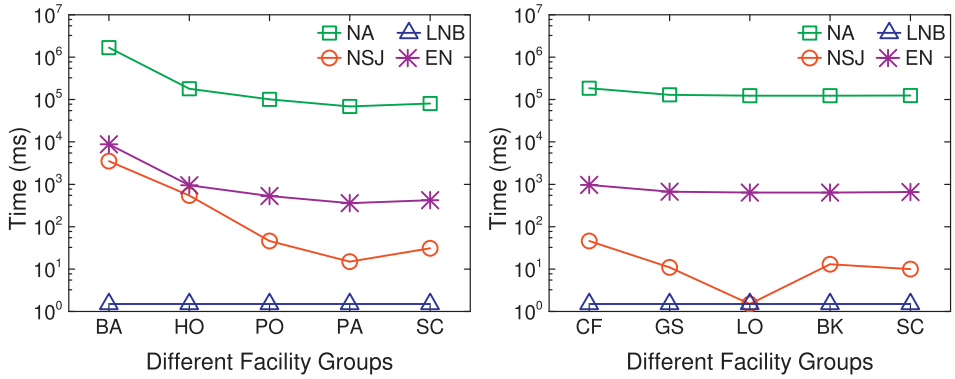
Effect of distributions on results. Below, we study the effect on resulting candidates in terms of two kinds of distributions: the geographical distribution of facilities and the probability distribution that a user appears at her reference locations.

Table 3
Average precision (AP) comparison.

# of Users.	1 k (CF)	100 k (CF)	1 k (GS)	100 k (GS)
AP@50	0.896	0.959	0.756	0.960

Table 4
Expected reduction of total distance (ERD) (*km*).

	CA	BJ
avg. *ERD* on RANDOM	48,527	3133
avg. *ERD* on UNIFORM	54,581	3534
avg. *ERD* difference (%)	12.5	12.8



(a) CA (b) BJ

Fig. 13. Real data.

For the former, we still use CF and GS in Fig. 12, due to the apparent distinction. For the latter, we use two groups, 1 k and 100 k, of the same users with Uniform and Random present probabilities, respectively. We record the top-50 of 800 candidates for comparison. To reflect the ranking factor of the candidates, we use Average Precision (AP) as the measure to compare the results based on Uniform to those on Random. Table 3 reports these AP@50 values. The difference of results is more significant for less users. The reason is that, if the number of users, whose reference locations are uniformly distributed, is large enough, all present probabilities of reference locations approach to be uniform in the space domain. *That is, the more users there are, the more important for the reference locations comparing to the present probabilities. It implies reference locations, nearby which users conduct their daily activities, are indeed closely related for choosing an optimal candidate*. This observation is exactly in line with our original intention that movement should be taken into account for location selection. On the other hand, the more uniform geographical distribution of facilities, the bigger difference of results, as less users will be affected by a new candidate.

We also compare the ERD differences between Uniform and Random distributions on average, where ERDs are associated with the optimal candidates selected on Random distribution. Table 4 shows that candidates in CA gain larger ERDs than in BJ, due to the larger geographical area of CA. In addition, the proportion difference of ERDs, computed as $\frac{\text{ERD on Uniform}-\text{ERD on Random}}{\text{ERD on Random}} \times 100\%$, are around 12.5% (CA) and 12.8% (BJ). This further shows the movements of users, namely the skewed present probabilities at reference locations, influence the result.

Effect on network. In this part, we briefly report the effect on network scale. As demonstrated in the experiments above, except LNB, the running times in BJ are an order of magnitude longer than the counterparts in CA. As the scale of BJ has a larger magnitude than that of CA, the scalability is exactly in line with the theoretical analysis. Note that, if the geographical distribution of facilities is quite different, *e.g.,* the skewed CF versus the uniform GS in BJ, the time cost might deviate from the network scale.

Experiments on real users. The experimental results on real users are shown in Fig. 13. The comparative performance of the methods is similar to that of experiments conducted for the synthetic datasets. LNB still outperforms the other methods. In CA, the running time of NSJ and EN for BA and HO are close, due to the skewed geographical distributions. For LO in BJ, the cost of NSJ is similar to that of LNB, which is because the number of users is relatively small and thus the efficiency of NSJ improves.

Comparison with the state-of-the-art. Finally, we compare our algorithms with the solutions proposed in paper [26]. To the best of our knowledge, they were the only efforts for solving the LS problem on road network with Min-dist objective

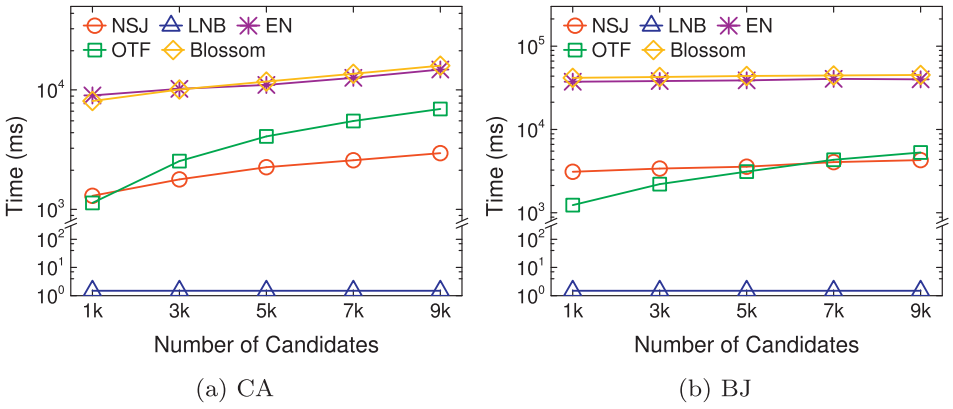


Fig. 14. Comparison of $|C|$.

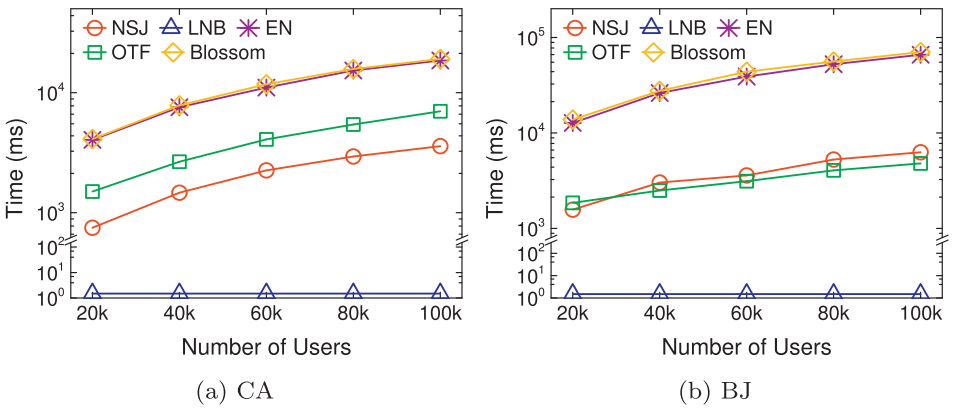


Fig. 15. Comparison of $|U|$.

function. For addressing our problem, we need to adapt the solutions from continuous setting to candidates. In order to avoid evaluating all locations on edges, the authors used network partition strategy, namely attraction sets of sub-networks, each of which is the union of attraction sets of edge vertices in the sub-network. Obviously, this strategy is redundant for specific candidates. Consequently, we directly compare the proposed algorithms, Blossom and OTF,[7] with ours. The query efficiencies of Blossom and OTF are mainly associated with the number of users and candidates, respectively. Hence, we focus on the query effects of $|C|$ and $|U|$, where $|C|$ varies from 1 k to 9 k, and $|U|$ is from 20 k to 100 k with 5 k candidates as default.

As illustrated in Figs. 14 and 15, the LNB algorithm has the best efficiency both for the scalability of $|C|$ and $|U|$. The query cost of Blossom is very close to that of EN as they both utilize Dijkstra algorithm for each user (*i.e.,* reference locations). The case between NSJ and OTF is complicated. For the relatively small road network CA, NSJ is mostly better than OTF, especially when the number of candidates grows larger, as illustrated in Fig. 14(a). While in the relatively larger network BJ, OTF is slightly better than NSJ. Nevertheless, this trend reverses if the number of candidates grows or the number of users drops. As a result, in most cases, our proposed algorithms are better than the solutions designed in [26].

8. Conclusions

In this paper, we introduce a novel location selection problem, Min-dIst Location SElection oveR User MovemeNt (MILE-RUN), which takes into account the movements of objects. Based on reference locations, which can model moving users and be captured from movement data, we present two groups of algorithms, index-based and index-free ones for solving MILE-RUN. The first group, including NSJ and LNB, answers MILE-RUN efficiently with carefully designed index structures, and fits the case where facilities and users are known apriori. The second group, namely EN, solves the problem from scratch. Extensive experiments on both real and synthetic datasets demonstrate the superiority of our proposed approaches compared to the baseline method by orders of magnitude.

[7] The source code of Blossom and OTF algorithms is obtained from the authors (http://cs.sjtu.edu.cn/~yaobin/olq).

Acknowledgments

The authors would like to thank the editor and anonymous reviewers for their constructive comments on this paper, and the authors [26] for sharing their codes. This work was supported by the National Natural Science Foundation of China (Nos. 61672408 and 61472298), CCF-VenustechRP2017005 and China 111 Project (No. B16037).

References

[1] S. Bespamyatnikh, B.K. Bhattacharya, D.G. Kirkpatrick, M. Segal, Mobile facility location, in: DIAL-M, 2000, pp. 46–53.
[2] M.A. Cheema, X. Lin, W. Wang, W. Zhang, J. Pei, Probabilistic reverse nearest neighbor queries on uncertain data, TKDE 22 (4) (2010) 550–564.
[3] Z. Chen, Y. Liu, R.C. Wong, J. Xiong, G. Mai, C. Long, Efficient algorithms for optimal location queries in road networks, in: SIGMOD, 2014, pp. 123–134.
[4] E.W. Dijkstra, A note on two problems in connection with graphs, Numer. Math. 1 (1) (1959) 269–271.
[5] Y. Du, D. Zhang, T. Xia, The optimal-location query, in: SSTD, 2005, pp. 163–180.
[6] Y. Gao, S. Qi, L. Chen, B. Zheng, X. Li, On efficient k-optimal-location-selection query processing in metric spaces, Inf. Sci. 298 (2015) 98–117.
[7] P. Ghaemi, K. Shahabi, J.P. Wilson, F.B. Kashani, Optimal network location queries, in: SIGSPATIAL/GIS, 2010, pp. 478–481.
[8] A. Guttman, R-trees: a dynamic index structure for spatial searching, in: SIGMOD, 1984, pp. 47–57.
[9] J. Huang, Z. Wen, J. Qi, R. Zhang, J. Chen, Z. He, Top-k most influential locations selection, in: CIKM, 4, 2011, pp. 2377–2380.
[10] F. Li, D. Cheng, M. Hadjieleftheriou, G. Kollios, S. Teng, On trip planning queries in spatial databases, in: SSTD, 2005, pp. 273–290.
[11] Z. Li, B. Ding, J. Han, R. Kays, P. Nye, Mining periodic behaviors for moving objects, in: KDD, 2010, pp. 1099–1108.
[12] D. Papadias, Y. Tao, K. Mouratidis, C.K. Hui, Aggregate nearest neighbor queries in spatial databases, TODS 30 (2) (2005) 529–576.
[13] D. Papadias, J. Zhang, N. Mamoulis, Y. Tao, Query processing in spatial network databases, in: VLDB, 2003, pp. 802–813.
[14] J. Qi, R. Zhang, L. Kulik, D. Lin, Y. Xue, The min-dist location selection query, in: ICDE, 2012, pp. 366–377.
[15] J. Qi, R. Zhang, Y. Wang, A.Y. Xue, G. Yu, L. Kulik, The min-dist location selection and facility replacement queries, World Wide Web 17 (6) (2014) 1261–1293.
[16] S. Shang, B. Yuan, K. Deng, K. Xie, X. Zhou, Finding the most accessible locations: reverse path nearest neighbor query in road networks, in: SIGSPATIAL, 2011, pp. 181–190.
[17] W. Sun, C. Chen, B. Zheng, C. Chen, L. Zhu, W. Liu, Y. Huang, Merged aggregate nearest neighbor query processing in road networks, in: CIKM, 2013, pp. 2243–2248.
[18] L.A. Tang, Y. Zheng, X. Xie, J. Yuan, X. Yu, J. Han, Retrieving k-nearest neighboring trajectories by a set of point locations, in: SSTD, 2011, pp. 223–241.
[19] B. Thierry, B. Chaix, Y. Kestens, Detecting activity locations from raw GPS data: a novel kernel-based algorithm, Int. J. Health Geograph. 12 (11) (2013) 213.
[20] W.R. Tobler, A computer movie simulating urban growth in the detroit region, Econ. Geogr. 46 (1970) 234–240.
[21] M. Wang, H. Li, J. Cui, K. Deng, S.S. Bhowmick, Z. Dong, PINOCCHIO: probabilistic influence-based location selection over moving objects, TKDE 28 (11) (2016) 3068–3082.
[22] R.C. Wong, M.T. Özsu, A.W. Fu, P.S. Yu, L. Liu, Y. Liu, Maximizing bichromatic reverse nearest neighbor for L p-norm in two- and three-dimensional spaces, VLDB J. 20 (6) (2011) 893–919.
[23] R.C.-W. Wong, M.T. Özsu, P.S. Yu, A.W.-C. Fu, L. Liu, Efficient method for maximizing bichromatic reverse nearest neighbor, PVLDB 2 (1) (2009) 1126–1137.
[24] B.J. Worton, Kernel methods for estimating the utilization distribution in home-range studies, Ecology 70 (1) (1989) 164–168.
[25] T. Xia, D. Zhang, E. Kanoulas, Y. Du, On computing top-t most influential spatial sites, in: VLDB, 2005, pp. 946–957.
[26] X. Xiao, B. Yao, F. Li, Optimal location queries in road network databases, in: ICDE, 2011, pp. 804–815.
[27] D. Yan, Z. Zhao, W. Ng, Efficient algorithms for finding optimal meeting point on road networks, PVLDB 4 (11) (2011) 968–979.
[28] M. Ye, P. Yin, W.-C. Lee, D.L. Lee, Exploiting geographical influence for collaborative point-of-interest recommendation, in: SIGIR, 2011, pp. 325–334.
[29] M.L. Yiu, N. Mamoulis, D. Papadias, Aggregate nearest neighbor queries in road networks, TKDE 17 (6) (2005) 820–833.
[30] D. Zhang, Y. Du, T. Xia, Y. Tao, Progressive computation of the min-dist optimal-location query, in: VLDB, 2006, pp. 643–654.
[31] Y. Zheng, L. Zhang, X. Xie, W. Ma, Mining interesting locations and travel sequences from GPS trajectories, in: WWW, 2009, pp. 791–800.
[32] L. Zhu, Y. Jing, W. Sun, D. Mao, P. Liu, Voronoi-based aggregate nearest neighbor query processing in road networks, in: SIGSPATIAL, 2010, pp. 518–521.