Information Sciences 310 (2015) 52–68

Contents lists available at ScienceDirect

Information Sciences

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

Fast optimal aggregate point search for a merged set on road networks ☆

Weiwei Sun [a], Chong Chen [a,*], Baihua Zheng [b], Chunan Chen [a], Liang Zhu [a], Weimo Liu [a], Yan Huang [c]

[a] Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University, Shanghai, China
[b] School of Information Systems, Singapore Management University, Singapore
[c] Computer Science and Engineering Department, University of North Texas, USA

ARTICLE INFO

Article history:
Received 3 September 2013
Received in revised form 15 February 2015
Accepted 11 March 2015
Available online 19 March 2015

Keywords:
Query processing
Aggregate nearest neighbor
Road networks
Spatial databases

ABSTRACT

Aggregate nearest neighbor query, which returns an optimal target point that minimizes the aggregate distance for a given query point set, is one of the most important operations in spatial databases and their application domains. This paper addresses the problem of finding the aggregate nearest neighbor for a merged set that consists of the given query point set and multiple points needed to be selected from a candidate set, which we name as merged aggregate nearest neighbor(MANN) query. This paper proposes two algorithms to process MANN query on road networks when aggregate function is max. Then, we extend the algorithms to support other aggregate functions (e.g., sum). Extensive experiments are conducted to examine the behaviors of the solutions in terms of five parameters affecting the performance. The overall experiments show that our strategies to minimize the response time are effective.

1. Introduction

Location-based services (LBSs) become more and more important in our everyday life. Worldwide revenues from LBSs are expected to go beyond $6 Billion by 2017, according to ABI Research. This huge and ever-growing market has attracted lots of attentions from both academy and industry. In this paper, we study a new location-based query, namely *Merged Aggregate Nearest Neighbor (MANN)*, on a spatial road network.

Formally, given a target set P, a query set Q, a candidate set C and an integer n, an MANN query returns an optimal target point $p \in P$ and a set of n candidates C_s from C ($C_s \subseteq C$), such that the aggregate distance from target point p to all the points in Q and C_s is minimized, i.e., $q(P,Q,n,C) = \{\langle p, C_s\rangle | p \in P \wedge C_s \in \Gamma(C,n) \wedge \forall p' \in P, \forall C' \in \Gamma(C,n), f(p, C_s \cup Q) \leqslant f(p', C' \cup Q)\}$.

☆ A preliminary version of this work was published in the Proceedings of the 22nd International Conference on Information and Knowledge Management (CIKM 2013). Substantial new technical materials have been added to this journal submission. Specifically, the paper extends the CIKM 2013 paper by contributing (i) two new pruning strategies and a new searching strategy presented in Section 3, (ii) two new algorithms based on the new and old strategies presented in Section 3, and (iii) enhanced experimental evaluation that incorporates more parameters and metrics as presented in Section 4. **Notes:** (i) This manuscript is the authors' original work and has not been published nor has it been submitted simultaneously elsewhere, except for the preliminary version (i.e., [21]) mentioned previously. (ii) The main differences between the conference version and this submission are stated above. (iii) All authors have checked the manuscript and have agreed to the submission.

* Corresponding author.

http://dx.doi.org/10.1016/j.ins.2015.03.028
0020-0255/

Here, $f(p,S)$ is the aggregate distance function and it could be max or sum or others based on application needs. Take max as an example, $f(p,S) = \max_{\forall s \in S} \|p,s\|$ with $\|p,s\|$ returning the network distance between p and s on a given road network; and function $\Gamma(C,n)$ is a function which returns all the subsets C's that are formed by n candidate points of C, i.e., $\forall C' \in \Gamma(C,n), C' \subseteq C$ and $|C'| = n$. As we consider the merged set of Q and C_s and the aggregated distance, we simply name the new query as *Merged Aggregate Nearest Neighbor (MANN)*.

MANN can be fit into many real life applications. For example, three friends want to play basketball. They need to find a basketball court and meanwhile invite seven of their friends to play basketball. Here, the target set P is the set of basketball courts available, the candidate set C is a set of friends, and n is 7. MANN can help to select 7 friends and meanwhile locate a basketball court such that the maximum distance from 10 participants' locations to the court is minimum. Our second example could be doctors offering free health consultations. Assume a group of four doctors decide to offer voluntary health consultation to public. They need to find a place that can reach the public easily, and invite a few nurses (e.g., 10) to facilitate the service. Here, the target set P is the set of public areas that can host the voluntary health consultation, such as parks and subway exits, the candidate set C is the set of nurses that have good working relationship with at least one of those four doctors, and n is 10. MANN can help to select 10 nurses and meanwhile locate a public area such that the total distance from the 4 doctors and 10 nurses to the public area is minimum. Another example could be supermarket chain planning expansion. Assume a supermarket chain currently operates one branch in Shanghai and it plans to open another 3 branches in the coming year. To cut down its operating cost, it also plans to have its own warehouse somewhere in Shanghai. Here, the target set P is the set of available locations for warehouses, the candidate set C is the set of available locations for new branches, and n is 3. MANN can help to select 3 branch locations and meanwhile locate one warehouse such that the maximum distance from the warehouse to any branch is minimized.

Fig. 1 shows a simple example of the MANN query that will be used as the running example throughout this paper. The star points form the target set P, the circle points form the candidate set C, the square points form the query set Q, and the integers next to the edges represent the edges' length. Suppose n is 2 and the aggregate distance function considered is max, we list in Table 1 some potential answers and MANN will return p_1 as the answer target point, and c_1 and c_4 as the corresponding answer candidate points.

As shown in the above example, MANN is complex. It considers the distance from the target point to points in Q and the distance from the same target point to n candidate points, while both the target point and the n candidate points are unknown. To the best of our knowledge, MANN query has not been studied in the literature and the most close one is ANN query [17]. However, ANN only considers the distance from a target point to the query points that are fixed and there is no need to locate n candidate points. Given the definition of MANN, there are two naive solutions to process MANN query, namely *p-oriented algorithm* and *c-oriented algorithm*. p-oriented algorithm considers target point $p \in P$ first and it locates, for each target point $p \in P$, the n points from C that are nearest to p as the candidate points. The one that has the minimum aggregated distance to all the query points and the n candidate points is the answer. On the other hand, c-oriented algorithm considers candidate points first. It enumerates all the potential candidate set C_s and there are in total $\binom{|C|}{n}$ potential C_ss. For each potential C_s, an ANN query is issued with $Q \cup C_s$ as the input, and the one with minimal aggregated distance forms the answer to MANN. Obviously, both approaches are inefficient as they blindly scan either all the points in P or all those $\binom{|C|}{n}$ potential C_ss.

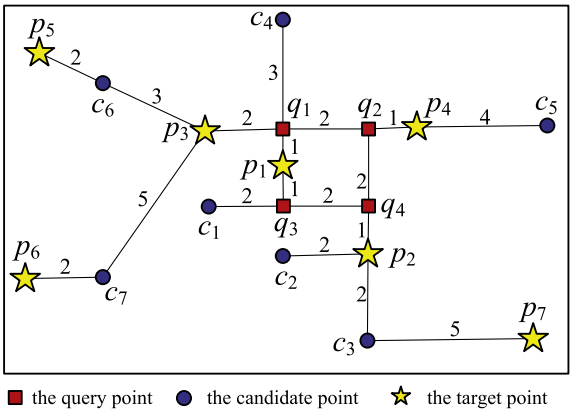


Fig. 1. Example of a MANN query.

Table 1
Distance from p to points in $C_s \cup Q$.

p	$\max_{\forall q \in Q} \|p, q\|$	C_s	$\max_{\forall c \in C_s} \|p, c\|$
p_1	$\max\{1,3,1,3\} = 3$	$\{c_1, c_4\}$	$\max\{3,4\} = 4$
p_2	$\max\{5,3,3,1\} = 5$	$\{c_2, c_3\}$	$\max\{2,2\} = 2$
p_3	$\max\{2,4,4,6\} = 6$	$\{c_6, c_7\}$	$\max\{3,5\} = 5$
p_4	$\max\{3,1,5,3\} = 5$	$\{c_5, c_4\}$	$\max\{4,6\} = 6$

Motivated by the fact that existing algorithms cannot support MANN queries efficiently, we develop two novel algorithms to process MANN queries when aggregate function is max. Then, we extend the algorithms to support other aggregate functions (e.g., sum). In brief, we mainly made four main contributions in this paper.

- We formalize the MANN query on road networks.
- We propose two efficient algorithms to support MANN queries when aggregate function is max, and perform a theoretical analysis on the time efficiency of the algorithms.
- We extend the algorithm to support other aggregate functions (e.g., sum).
- We conduct comprehensive simulation study based on the real and synthetic datasets to evaluate our algorithms. The experimental results demonstrate that our algorithms achieve excellent search performance and also have great scalability.

The rest of the paper is organized as follows. First, we present the preliminaries and problem statement in Section 2. Next, we develop the Pruning-Oriented Algorithm and Searching-Oriented Algorithm to support function max in Section 3, together with an analytical model for performance evaluation and the extended algorithms to support other function (e.g., sum). Then, we study the performance of our proposals on the real and synthetic datasets in Section 4. Finally, we discuss related work in Section 5 and conclude our paper in Section 6.

2. Preliminary

In this section, we first present the formal definitions of road networks and MANN, and then we define the candidate result, which will be used in our algorithm. Table 2 defines the common symbols used in this paper.

We model a road network G as a weighted graph that consists of a set of nodes N and a set of edges E, i.e., $G = (N, E)$. A node $n \in N$ represents a road intersection and an edge $(n, n') \in E$ represents a road segment connecting nodes n and n'. $w(n, n')$ denotes the edge weight, which can represent the travel distance or trip time, and we assume all distances are positive. For simplicity, we use distance hereafter. A path $P(u, v)$ stands for a set of edges connecting nodes u and v and its distance $|P(u, v)| = \sum_{(n,n') \in P(u,v)} w(n, n')$. Among all paths connecting node u and node v, the one with the shortest distance is referred to as the *shortest path*, denoted by $SP(u, v)$. The network distance $\|u, v\|$ between u and v is the distance of their shortest path $SP(u, v)$, i.e., $\|u, v\| = |SP(u, v)|$. On road network, MANN query is introduced, as defined in Definition 1.

Definition 1 (*MANN Query*). Given a set of target points P and a road network $G(N, E)$, an MANN query $q(P, Q, n, C)$ specifies a query set Q that contains one or multiple query points, an integer n, and a candidate set C that contains at least n candidate points. It returns a target point $p \in P$ and a set of n candidate points from candidate set C (denoted as C_s) such that the aggregated distance from p to all the points in $Q \cup C_s$ is minimum, i.e., $q(P, Q, n, C) = \{\langle p, C_s \rangle | p \in P \wedge C_s \in \Gamma(C, n) \wedge \forall p' \in P, \forall C' \in \Gamma(C, n), f(p, C_s \cup Q) \leqslant f(p', C' \cup Q)\}$.

Here, $\Gamma(C, n)$ is a function to return all the subsets of C that contain n points of C, i.e., $\Gamma(C, n) = \{C' | C' \subseteq C \wedge |C'| = n\}$; and $f(p, S)$ is an aggregate distance function and it can be max or sum or other operations based on application needs. If max is considered, $f(p, S) = \|p, S\|_{max}$; if sum is considered, $f(p, S) = \|p, S\|_{sum}$.

To facilitate our discussion, we also introduce a concept, namely *Candidate Result*, as defined in Definition 2. If function f considers max, the candidate result $p.CR$ of a target point p actually contains top-n nearest points of p. Take the running

Table 2
Frequently used symbols.

Symbol	Description
$d_e(p, q)$	The Euclidean distance between p and q
$\|p, q\|$	The minimum network distance from p to q
$\|p, S\|_{max}$	The maximum network distance from p to a point of S, i.e., $\forall s \in S, \|p, s\| \leqslant \|p, S\|_{max}$ and $\exists s' \in S$ such that $\|p, s'\| = \|p, S\|_{max}$
$\|p, S\|_{sum}$	The aggregate network distance from point p and all the points in set S, i.e., $\|p, S\|_{sum} = \sum_{\forall s \in S} \|p, s\|$

example depicted in Fig. 1 as the example. $p_1.CR = \{c_1, c_4\}$, $p_2.CR = \{c_2, c_3\}$, and $p_3.CR = \{c_6, c_7\}$. Given two target points p_1 and p_2, we say p_1 is better than p_2 for a given MANN query $q(P, Q, n, C)$ if $f(p_1, p_1.CR \cup Q) \leqslant f(p_2, p_2.CR \cup Q)\}$. Back to our running example. Target points p_1 is better than p_2 as $f(p_1, p_1.CR \cup Q) = \|p_1, \{c_1, c_4, q_1, q_2, q_3, q_4\}\|_{max} = 4$, and $f(p_2, p_2.CR \cup Q) = \|p_2, \{c_2, c_3, q_1, q_2, q_3, q_4\}\|_{max} = 5$.

Definition 2 (*Candidate Result*). Given a target point $p \in P$ and an MANN query $q(P, Q, n, C)$ on a road network $G(N, E)$, the candidate result of p is defined as the set of n candidate points, denoted as $p.CR$, that can minimize the aggregated distance from p to n candidate points, i.e., $\forall C' \in \Gamma(C, n), f(p, p.CR) \leqslant f(p, C')$. In other words, if p is returned by $q(P, Q, n, C)$, $p.CR$ must be the corresponding candidate points returned.

3. Algorithm

As we explain in Section 1, the baseline p-oriented algorithm and c-oriented algorithm do not perform well because they need to blindly scan all the target points $p \in P$ or all the potential candidate point sets C_s formed by n candidate points of C. In this section, we focus on aggregate function max and develop the *Pruning-Oriented (PO) Algorithm* to improve the performance of p-oriented algorithm via pruning away certain target points, and *Searching-Oriented (SO) Algorithm* to reduce repeated search. Then, we perform a theoretical analysis on the performance of the algorithms and extend algorithm SO to support aggregate function sum.

3.1. Pruning-Oriented Algorithm

In the following, we first present how to find the candidate result, based on which our pruning strategy is developed, and then we present the *Pruning-Oriented (PO) Algorithm*.

3.1.1. Candidate result

First, we introduce the algorithm used to locate the candidate result $p.CR$ for a given target point p in Algorithm 1, which extends the Dijkstra algorithm. It expands the road network from the target point p, and always explores nodes that have the shortest distances to p. Although the candidate result $p.CR$ contains only n candidate points, our network expansion stops when the set $p.CR$ contains n candidate points and meanwhile all the query points in Q have been visited. This is because the pruning rule used by PO is based on $f(p, Q \cup p.CR)$ and Algorithm 1 returns not only $p.CR$ but also $f(p, Q \cup p.CR)$. It is worth noting that finding the candidate result of a target point can be costly, so reducing the times of invoking Algorithm 1 is important, which is the main idea of our work.

Algorithm 1. Find Candidate Result for *max* function

Input: $p, Q, C, n, G(N, E)$
Output: $p.CR, \|p, Q \cup p.CR\|_{max}$
1: **if** $|C| < n$ **then**
2: return $p.CR = \emptyset, \|p, Q \cup p.CR\|_{max} = \infty$;
3: $max\_dist \leftarrow 0; C_p \leftarrow \emptyset$;
4: $H \leftarrow$ new min-heap;
5: insert $H(p, \|p, p\| = 0)$;
6: **while** $(H \neq \emptyset)$ **and** $(|Q| > 0$ **or** $|p.CR| < n)$ **do**
7: $(u, \|u, p\|) \leftarrow H.deheap()$;
8: **if** $u \in Q$ **then**
9: update max_dist by $\|u, p\|$;
10: remove u from Q;
11: **else if** $u \in C$ **and** $|p.CR| < n$ **then**
12: update max_dist by $\|u, p\|$;
13: insert u to $p.CR$;
14: explore u's neighbor nodes and put them into H;
15: return $p.CR, \|p, Q \cup p.CR\|_{max} = max\_dist$;

3.1.2. Pruning rules for target points

Based on a given target point p and its corresponding candidate result, we develop two pruning strategies for aggregate distance function max to prune away certain target points that definitely will not produce results better than p, as stated in Theorems 1 and 2. The statement "point p is better than point p'" means that point p produces a result better than p'.

Theorem 1. *Given an MANN query $q(P,Q,n,C)$, a road network $G(N,E)$, and an aggregate function* max, $\forall p_1,p_2 \in P$, *if* $\max_{q_i\in Q}(d_e(p_1,q_i)) \geqslant \|p_2, Q \cup p_2.CR\|_{max}$, *it is certain that p_1 cannot be better than p_2.*

Proof. We all understand that

$$\max_{q_i\in Q}\|p_1,q_i\| \geqslant \max_{q_i\in Q} d_e(p_1,q_i) \tag{1}$$

Based on our assumption, we have

$$\|p_2,Q\cup p_2.CR\|_{max} \leqslant \max_{q_i\in Q} d_e(p_1,q_i) \leqslant \max_{q_i\in Q}\|p_1,q_i\| \leqslant \|p_1,Q\cup p_1.CR\|_{max} \tag{2}$$

According to Eqs. (1) and (2), it is obvious that p_1 cannot be better than p_2 and our proof completes. □

Theorem 2. *Given an MANN query $q(P,Q,n,C)$, a road network $G(N,E)$ and an aggregate function* max, *let p_1 and p_2 be any two points of P and let $d = \|p_2,Q\cup p_2.CR\|_{max}$. If there are less than n points of set C located inside the circle centered at p_1 with d as the radius, denoted as $\|Circle(p_1,d,C)\| < n$, it is certain that p_1 cannot be better than p_2.*

Proof. We denote $p_1.ECR$ as n nearest candidate points from C according to Euclidean distance. Then $\|Circle(p_1,d,C)\| < n$ is equivalent to $\max_{c_i\in p_1.ECR} d_e(p_1,c_i) > d$.

When max is considered, $p_1.CR$ actually contains the n nearest candidate points according to the network distance. Then we have

$$\max_{c_i\in p_1.CR}\|p_1,c_i\| \geqslant \max_{c_i'\in p_1.ECR} d_e(p_1,c_i') \tag{3}$$

Based on our statement, we have

$$\|p_2,Q\cup p_2.CR\|_{max} < \max_{c_i'\in p_1.ECR} d_e(p_1,c_i') \leqslant \max_{c_i\in p_1.CR}\|p_1,c_i\| \leqslant \|p_1,Q\cup p_1.CR\|_{max} \tag{4}$$

Again according to Eqs. (3) and (4), it is obvious that p_1 cannot be better than p_2 and our proof completes. □

3.1.3. PO algorithm

Based on the pruning rules presented in Theorems 1 and 2, we present our algorithm PO with its pseudo-code listed in Algorithm 2. Since the algorithm needs to process kNN query according to Euclidean distance on set C which cannot be known beforehand, we index the network by a grid index. We prefer grid index instead of others (e.g, R-tree) because the grid index partitions the network into equal-sized rectangular regions. It can support kNN query and range query effectively, and it is very efficient to map the point set into corresponding region. We regard the grid index as an input of the algorithm. Note the grid index can be replaced by any similar index, and the selection of the index is orthogonal to our work.

The algorithm follows filtering-refining framework. It first completes the initialization for the pruning distance d, which captures the maximum distance corresponding to current result maintained by S (lines 1–5). It then prunes away the target points based on Theorems 1 and 2 (lines 6–12). It finally evaluates the remaining target points. The three steps are detailed in the following.

In our first step, we find the geometry center q_m for the input query set Q, and then retrieve the nearest target point $q_m.NN$ of q_m by the index *grid*. We then retrieve the candidate result $(q_m.NN).CR$ corresponding to $q_m.NN$ based on Algorithm 1. The intuition behind is that in most cases, a target point closer to the center of Q usually has a smaller maximum distance to Q and hence initializing the pruning distance d based on $q_m.NN$ and its corresponding candidate result is a good choice. The experimental results to be presented in Section 4 will further justify our selection.

Next, we start our second step to prune away all the target points that cannot be better than $q_m.NN$. The function $Circle(q,d,P)$ in Line 8 refers to a range query that retrieves all the points $p \in P$ with $d_e(q,p) \leqslant d$. For any target point $p \notin S_p$, there is a query point $q_i \in Q$ satisfying $d_e(q_i,p) > d$. From Theorem 1 we can know, these target points can be pruned away. Then we continue to prune away unwanted target points in S_p based on Theorem 2 (lines 9–12). The function $nNN(p,C,n)$ in Line 11 retrieves n nearest points from C according to Euclidean distance.

Finally, we perform the refinement step. In order to enable an early termination of this step, we strategically visit target points based on ascending order of d_p, which is the maximum Euclidean distance to Q and their n-th nearest candidate point. Once the distance d_p of the top entry is larger than d, it is guaranteed that all the remaining entries in H have their d_p larger than d and hence cannot be better than the target point maintained by S. The correctness of PO algorithm is guaranteed by Lemma 1.

Algorithm 2. Pruning-Oriented Algorithm for *max* function

Input: $P, Q, C, n, G(N, E), grid$
Output: $q(P, Q, n, C)$
1: get the geometry center of set Q, denoted as q_m;
2: get q_m's NN in Euclidean Space, denoted as $q_m.NN$;
3: retrieve $q_m.NN$'s candidate result $(q_m.NN).CR$ based on Algorithm 1;
4: $d \leftarrow \|q_m.NN, Q \cup (q_m.NN).CR\|_{max}$;
5: $S \leftarrow \langle q_m.NN, q_m.NN.CR\rangle$;
6: $S_p \leftarrow P, H \leftarrow$ new min-heap;
7: **for** each $q \in Q$ **do**
8: $S_p = S_p \cap Circle(q, d, P)$;
9: **for** each $p \in S_p$ **do**
10: **if** $\|Circle(p, d, C)\| \geqslant n$ **then**
11: $d_p \leftarrow \max_{p' \in Q \cup nNN(p,C,n)}(d_e(p, p'))$;
12: $H.enqueue(p, d_p)$;
13: **while** H is not empty
14: $\langle p, d_p\rangle \leftarrow H.deheap()$;
15: **if** $d_p \geqslant d$ **then**
16: return S;
17: retrieve p's candidate result $p.CR$ based on Algorithm 1;
18: **if** $\|p, Q \cup p.CR\|_{max} < d$ **then**
19: $d \leftarrow \|p, Q \cup p.CR\|_{max}; S \leftarrow \langle p, p.CR\rangle$;

Lemma 1. *The result identified by Algorithm 2 must be the real result for an MANN query.*

Proof. Assume the above statement is not valid, and the real result $q(P, Q, n, C)(= \langle p', p'.CR\rangle)$ is different from the one $\langle p, p.CR\rangle$ returned by Algorithm 2. From Theorems 1 and 2 we can know that only the target points maintained by the min-heap H can be the result. Given the fact that the target point p' is not returned by Algorithm 2, it must still be maintained by H. In other words, $d_{p'} \geqslant \|p.Q \cup p.CR\|_{max}$. It is easy to prove that $\|p', Q \cup p'.CR\|_{max} \geqslant d_{p'}$, then we have $\|p', Q \cup p'.CR\|_{max} \geqslant \|p, Q \cup p.CR\|_{max}$. That is to say p' cannot be better than p. Consequently, our assumption is invalid and the proof completes. □

Notice that we invoke Algorithm 1 to look for the candidate result corresponding to each examined target point p (Line 17). However, it is not always necessary to locate the candidate result if we know the maximum distance generated by p and its candidate result will not be shorter than that of the current best candidate maintained by S of Algorithm 2 (i.e., d). To enable this early termination of Algorithm 1, we can add following code right before Line 14 of Algorithm 1.

if $(max\_dist \geqslant \|S.p, Q \cup S.C_p\|_{max})$ **then** return $S = \emptyset, max\_dist = \infty$;

In the following, we use an example, as shown in Fig. 1, to illustrate how PO works. Here, $Q = \{q_1, q_2, q_3, q_4\}$, $P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7\}$, $n = 2$, and $C = \{c_1, c_2, c_3, c_4, c_5, c_6, c_7\}$. First, we derive the geometry center q_m of all the query points and locate its nearest target point p_1 (i.e., $q_m.NN = p_1$). Thereafter, we invoke Algorithm 1 to find the candidate result $p_1.CR$ of p_1, that is $\{c_1, c_4\}$. We initialize the result set S as $\langle p_1, p_1.CR\rangle$, and set the pruning distance d to $\|p_1, Q \cup p_1.CR\|_{max} = 4$. Next, we process range queries $Circle(q_i, d, P)$ for all the points in Q. The results for q_1, q_2 and q_4 are all $\{p_1, p_2, p_3, p_4\}$ while the result for q_3 is $\{p_1, p_2, p_4\}$, so S_p is changed to $\{p_1, p_2, p_4\}$. Then, for any target point in S_p, the algorithm checks whether there are enough candidate points located in the nearby area. For example, as the circle centered at point p_4 and having d as its radius only bounds one point, it can be pruned away. Finally, only two target points p_1 and p_2 are maintained by H with their d_p being $\sqrt{5}$ and $\sqrt{13}$ respectively. The algorithm returns the result $\langle p_1, p_1.CR(= \{c_1, c_4\})\rangle$ to end its process.

3.2. Searching-Oriented Algorithm

The PO algorithm aims to cut down the number of target points visited, while repeatedly invoking Algorithm 1 to find the candidate result of each unpruned target point which is costly. Motivated by this disadvantage, we propose a new pruning strategy and develop a new algorithm, namely *Searching-Oriented (SO) Algorithm*, accordingly.

3.2.1. New strategy for candidate result

We notice that all the target points of a given query share a common set of destination points when they look for their corresponding candidate results (i.e., the query points Q). For a given target point, algorithm PO invokes Algorithm 1 to locate the candidate result via incrementally visiting the target point's nearest nodes. In other words, the edges located in the nearby area of the query set may be visited again and again which increases the cost of a query.

In order to tackle this issue, we attempt to reuse all the distances computed by previous searches. In particular, we maintain $|Q|$ searches originated from each query point. For each query point q_i, a heap $q_i.H$ is used to record the search state and to continue the search from q_i if needed. In addition, we maintain a hash table $q_i.T$ for each q_i to record the distance between q_i and the nodes it has visited. Then for a given target point p, if p is in $q_i.T$, we can directly use the distance stored in $q_i.T$. Otherwise, we can continue the search from q_i by $q_i.H$.

Algorithm 3. Find Candidate Result II for *max* function

Input: $p, Q, C, n, G(N, E)$
Output: $p.CR, \|p, Q \cup p.CR\|_{max}$
1: **if** $|C| < n$ **then**
2: return $p.CR = \emptyset, \|p, Q \cup p.CR\|_{max} = \infty$;
3: $max\_dist \leftarrow 0; C_p \leftarrow \emptyset$;
4: **for** each q_i in Q **do**
5: **if** p is in $q_i.T$ **then**
6: get $\|q_i, p\|$ from $q_i.T$;
7: **else**
8: continue the search from q_i by $q_i.H$ until p is visited;
9: update max_dist by $\|q_i, p\|$;
10: retrieve p's candidate result $p.CR$ based on Algorithm 1;
11: update max_dist by $\|p, p.CR\|_{max}$;
12: return $p.CR, \|p, Q \cup p.CR\|_{max} = max\_dist$;

The pseudo-code of locating candidate results based on reuse technology is listed in Algorithm 3. The algorithm consists of two steps. The first step is to compute the maximum distance between the given target point p and the query point set (lines 4–9). If the target point has been visited, we can directly use the distance $\|q_i, p\|$ recorded (line 6). Otherwise, we utilize $q_i.H$ to continue the search from q_i (line 9). Whenever an unvisited node is visited, we record its distance from q_i in $q_i.T$. The second step is to find the candidate result corresponding to the target point and compute the maximum distance between them. Note that, Algorithm 1 used here has been changed (line 10), as it does not need to compute $\|p, Q\|_{max}$.

Algorithm 3 can be terminated earlier. Whenever the distance between a query point and the target point is derived (line 6 and line 8), we compare the updated current maximum distance with the best candidate that has been found so far. If the current one has exceeded the best candidate found so far, Algorithm 3 can be terminated. The implementation of this early termination condition is straightforward. Similar as the modification of Algorithm 1 stated in Section 3.1.3, we can update the current aggregate distance and compare it to the best candidate whenever a new candidate point is visited.

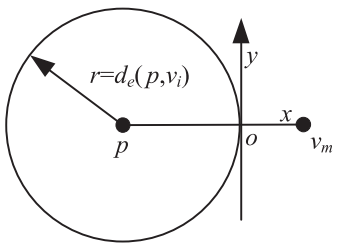


Fig. 2. An invalid assumption of Lemma 2.

3.2.2. Pruning rules for target points

The pruning rules presented in Section 3.1.2 consider the distribution of both Q and C and use the Euclidean distance to evaluate the maximum network distance. While SO algorithm focuses on reducing the repeated search, a faster pruning strategy is needed. In the following, we develop a new pruning strategy only based on the Euclidean distance between the target point and the geometry center of Q to reduce the cost of pruning step. The theoretical analysis and experimental results in latter sections will further justify our decision.

Before we present the pruning strategy, we first present a geometry property of the distance between a point and a set of points, as stated in Lemma 2. Thereafter, our new pruning strategy is presented in Theorem 3.

Lemma 2. *Let V be a point set in the Euclidean Space and point v_m be the geometry center of V. For any point p in the space, we have*

$$d_e(p, v_m) \leqslant \max_{v_i \in V} d_e(p, v_i) \tag{5}$$

Proof. Assume the above statement is not valid, i.e., if $d_e(p, v_i) = \max_{v_i \in V} d_e(p, v_i)$, we have $d_e(p, v_m) > d_e(p, v_i)$. We then can draw a circle cir_p centered at p with $d_e(p, v_i)$ as the radius. Since $d_e(p, v_i) = \max_{v_i \in V} d_e(p, v_i)$ and $d_e(p, v_m) > d_e(p, v_i)$, all the points of V must be located inside the circle cir_p. But the point v_m is located outside of the circle cir_p, as shown in Fig. 2. If we set the line $l(p, v_m)$ that passes point p and v_m as the x-axis and set the intersection point o of $l(p, v_m)$ and the circumference of circle cir_p as the origin, we can find that all the points of V are located at one side of the y-axis while v_m is on the other side. This contradicts our statement that v_m is the geometry center of points in V and hence our assumption is invalid. The proof completes. □

Theorem 3. *Given an MANN query $q(P, Q, n, C)$ and a road network $G(N, E)$, let q_m be the geometry center of Q. Assume* max *is considered. $\forall p_1, p_2 \in P$, if $d_e(p_1, q_m) \geqslant \|p_2, Q \cup p_2.CR\|_{max}$, it is certain that p_1 cannot be better than p_2.*

Proof. Based on Lemma 2, we have

$$\max_{q_i \in Q} d_e(p_1, q_i) \geqslant d_e(p_1, q_m) \tag{6}$$

As we all understand that the Euclidean distance is a lower bound of the network distance, we have

$$\max_{q_i \in Q} \|p_1, q_i\| \geqslant \max_{q_i \in Q} d_e(p_1, q_i) \tag{7}$$

Based on our assumption, we have $d_e(p_1, q_m) \geqslant \|p_2, Q \cup C_{p_2}\|_{max}$ and hence we have

$$\|p_2, Q \cup C_{p_2}\|_{max} \leqslant \max_{q_i \in Q} d_e(p_1, q_i) \leqslant \max_{q_i \in Q} \|p_1, q_i\| \leqslant \|p_1, Q \cup p_1.CR\|_{max} \tag{8}$$

It is obvious that p_1 cannot be better than p_2 and our proof completes. □

3.2.3. SO Algorithm

After presenting the new algorithm for locating the candidate result based on re-use technique (i.e., Algorithm 3) and the new pruning rule (i.e., Theorem 3), we are ready to present the new search algorithm namely *Searching-Oriented (SO) Algorithm*. Since the new pruning rule is based on Euclidean distance, we assume all the target points are indexed by an R-tree with its root node *root* being an input for Algorithm 4. The algorithm mainly contains two steps. The first step is to initialize the pruning distance d, which captures the maximum distance corresponding to current result maintained by S (lines 1–7). The second step is to prune away the target points based on Theorem 3 and update the pruning distance d if necessary. They are detailed in the following.

In our first step, similar as Algorithm 2, we find the geometry center q_m for the input query set Q, and then retrieve the nearest target point $q_m.NN$ of q_m. We then retrieve the candidate result $(q_m.NN).CR$ corresponding to $q_m.NN$ based on Algorithm 3, and initialize the pruning distance d as $\|q_m.NN, Q \cup (q_m.NN).CR\|_{max}$.

Next, we start our second step to evaluate the target points. In order to enable an early termination of this step, we strategically visit target points based on ascending order of their mindist to q_m. To be more specific, we visit the nodes of R-tree that indexes all the target points based on best-first order, with the help of the min-heap H. Initially, H has only one node, that is the root of the R-tree. Thereafter, we de-heap the top entry $\langle e, d_e \rangle$ of H for evaluation. Note that e is the entry within H that has the smallest mindist to q_m. If e's mindist to q_m (i.e., d_e) is already larger than d, it is guaranteed that all the remaining entries in H and all the unvisited objects will have their mindist to q_m larger than d and can be pruned away based on Theorem 3. Otherwise, we need to evaluate e. If e is a non-leaf node, all its child nodes are en-heaped to H. Otherwise, e must be an object. We then check whether e is better than the current result and update the pruning distance d and result S if necessary. The correctness of our SO algorithm is guaranteed by Lemma 3.

Algorithm 4. Searching-Oriented Algorithm for *max* function

Input: $P, Q, C, n, G(N, E), root$
Output: $q(P, Q, n, C)$
1: $H \leftarrow$ new min-heap;
2: get the geometry center of set Q, denoted as q_m;
3: $H.enheap(\langle root, mindist(q_m, root)\rangle)$;
4: get q_m's NN in Euclidean Space, denoted as $q_m.NN$, based on best-first order and maintain all the unexamined nodes in H based on ascending order of their *mindist* to q_m;
5: retrieve $q_m.NN$'s candidate result $(q_m.NN).CR$ based on Algorithm 3;
6: $d \leftarrow \|q_m.NN, Q \cup (q_m.NN).CR\|_{max}$;
7: initialize result $S \leftarrow \langle q_m.NN, q_m.NN.CR\rangle$;
8: **while** H is not empty **do**
9: $\langle e, d_e\rangle \leftarrow H.deheap()$;
10: **if** $d_e \geqslant d$ **then**
11: breaks;
12: **else if** e is not an object
13: **for** each child $e.c$ of e **do**
14: $H.enheap(\langle e.c, mindist(q_m, e.c)\rangle)$;
15: **else**
16: retrieve e's candidate result $e.CR$ based on Algorithm 3;
17: **if** $\|e, Q \cup e.CR\|_{max} \leqslant d$ **then**
18: $d \leftarrow \|e, Q \cup e.CR\|_{max}; S \leftarrow \langle e, e.CR\rangle$;
19: return S;

Lemma 3. *The result identified by Algorithm 4 must be the real result for a MANN query.*

Proof. Assume the above statement is not valid, and the real result $q(P, Q, n, C)$ $(=\langle p, p.CR\rangle)$ is different from the one $\langle e, e.CR\rangle$ returned by Algorithm 4. Given the fact that the target point p is not returned by Algorithm 4, it must be enclosed by a node N such that $mindist(N, q_m) \geqslant \|e, Q \cup e.CR\|_{max}$. In other words, $d_e(p, q_m) \geqslant mindist(N, q_m) \geqslant \|e, Q \cup e.CR\|_{max}$. Based on Theorem 3, we understand that p cannot be better than e. Consequently, our assumption is invalid and the proof completes. □

In the following, we also use an example, as shown in Fig. 1, to illustrate how SO works. First, like PO, we derive the geometry center q_m of all the query points, denoted as the triangle in Fig. 3, and locate its nearest target point p_1 (i.e., $q_m.NN = p_1$). Notice that we adopt the best-first order for NN search here, and we maintain all the unexamined nodes in H for later exploration. Thereafter, we invoke Algorithm 3 to find the candidate result $p_1.CR$ of p_1, then initialize the result set S as $\langle p_1, p_1.CR\rangle$ and the pruning distance d as $\|p_1, Q \cup p_1.CR\|_{max} = 4$. During this step, $\|q_2, p_4\|$, $\|q_2, p_2\|$, $\|q_4, p_2\|$ and $\|q_4, p_4\|$ have been recorded. Next, we evaluate the nodes maintained in H. This is to continue previous NN search, and try to locate the next NN objects of q_m. As we do not depict the R-tree structure of target points, we use target points directly. Since pruning distance d is set to 4, we only need to explore the target points with their Euclidean distance to q_m not exceeding 4, i.e., the target points located inside the shaded circle centered at q_m as shown in Fig. 3[1]. Then, we retrieve p_2 as the next NN of q_m. Note Algorithm 3 only needs to continue the search from q_1 and q_3 to compute $\|p_2, Q\|_{max}$. Because $\|p_2, Q\|_{max} = 4$, which has exceeded d, $p_2.CR$ will not be retrieved. The same thing happens to p_3 and p_4. After the evaluation of p_4, the algorithm can terminate as the rest objects have their mindist to q_m larger than d (i.e., 4). The algorithm ends and the final result is $\langle p_1, p_1.CR(=\{c_1, c_4\})\rangle$.

3.2.4. Discussion

The naive *p-oriented* algorithm tries to retrieve the candidate result for all the target points and return the one with the minimum distance. As Algorithm 1 extends the Dijkstra algorithm. For a network has $|N|$ nodes and $|E|$ edges, its time complexity is $O((|V| + |E|) * log|V|)$. As in road networks, $|E| = O(|V|)$, the complexity can be simplified to $O((|V|) * log|V|)$. In other words, the time complexity of the naive algorithm is $|P| * O(|V| * log|V|)$.

PO algorithm focuses on pruning away certain target points. The number of target points visited can be reduced to $\alpha * |P|(\alpha \in (0, 1])$, where α represents the visit ratio of PO, which is the ratio of the number of the visited target points to the total target points in the network. With the restriction of the pruning distance d, Algorithm 1 only needs to expand a

[1] Note that the shaded circle may keep shrinking as better results are retrieved and pruning distance d gets reduced.

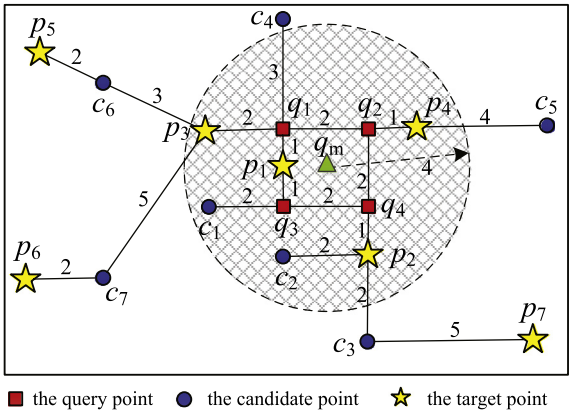


Fig. 3. Process of the running example.

small part of the network. The time complexity of retrieving the candidate result is $O((|V|) * log|V|) * \beta(\beta \in (0,1])$, and the time complexity of PO is $\alpha * |P| * O((|V|) * log|V|) * \beta$. Compared with the naive algorithm, PO performs much better in most cases with its worst case performance being exactly same as the naive algorithm.

SO maintains $|Q|$ searches originated from each query point. For each query point q_i, it only explores the nodes n with $\|q_i, n\| \leqslant d$ with the time complexity of this step being $|Q| * O((|V|) * log|V|) * \beta$. Then for a target point p, the time complexity of computing $\|p, Q\|_{max}$ is only $O(|Q|)$. We use $\alpha'(\alpha\prime \in (0,1])$ here to represent the visit ratio of SO, then the time complexity of computing the maximum distance to Q for all the visited target points is $\alpha' * |P| * O(|Q|)$. In addition, the time complexity of retrieving candidate points in SO is $\gamma * |P| * O((|V|) * log|V| * \beta), \gamma \in [0, \alpha']$. Overall, the time complexity of SO is $|Q| * O((|V|) * log|V|) * \beta + \alpha' * |P| * O(|Q|) + \gamma * |P| * O((|V|) * log|V|) * \beta$.

As $|N|$ is usually much larger than $|P|$ and $|Q|$, the efficiency of these two algorithms mainly depends on the times of the process to retrieve the candidate result of a target point. From the running examples of PO and SO, we find that SO needs to process four target points, p_1, p_2, p_3 and p_4. On the other hand, PO only needs to process two of them, p_1 and p_2. This is because Theorem 1 helps to prune away p_3 and Theorem 2 helps to discard another target point p_4.

Theorem 4. *With the same pruning distance d,* Theorem 3 *cannot discard more target points than* Theorem 1.

Proof. For any target point p discarded by Theorem 3, we have

$$d_e(p, q_m) \geqslant d \tag{9}$$

From Lemma 2, we know

$$d_e(p, q_m) \leqslant \max_{q_i \in Q} d_e(p, q_i) \tag{10}$$

Then we have

$$\max_{q_i \in Q} d_e(p, q_i) \geqslant d \tag{11}$$

From Theorem 1, we know p cannot be the result. That is to say, for any target point, if it is discarded by Theorem 3, it will for sure be discarded by Theorem 1 as well. Therefore, the visit ratio of Theorem 3 is always greater than Theorem 1. □

As algorithm PO and algorithm SO both are better than the naive algorithm and they use different methods to improve the performance, it seems to be a good choice to combine them. To be more specific, we can use the pruning rules of PO to prune away target points and use Algorithm 3 to retrieve candidate result, with the new algorithm named *Combination Algorithm (CA)*. Note the main difference between CA and PO is that CA uses Algorithm 3 (instead of Algorithm 1) to retrieve candidate result; the main difference between CA and SO is that CA uses Theorems 1 and 2 to discard target points while SO only uses Theorem 3.

3.2.5. SO Algorithm for sum function

In the above discussion, we only focus on aggregate function max. However, our algorithms developed are flexible and they can be extended to support other functions as well. In the following, we explain how to extend algorithm SO to support aggregate function sum.

SO algorithm mainly consists pf two parts, i.e., using Algorithm 3 to retrieve candidate result and invoking a pruning strategy to prune away certain target points. In order to support sum, Algorithm 3 needs only one modification. Instead of

maintaining the maximum distance, it needs to maintain the summation of distances. It then can locate the candidate result for a given target point for aggregate function sum. For the pruning rule, we develop a new pruning rule to support function sum, as stated in Theorem 5. Note, Lemma 4 presents a geometry property used by the new pruning rule.

Lemma 4. *Let V be a point set in a Euclidean Space with $V = \{v_1, v_2, \cdots, v_n\}$ and v_m be the geometry center of V. Then, for any point p in the space, we have*

$$n \times d_e(p, v_m) \leqslant \sum_{v_i \in V} d_e(p, v_i) \tag{12}$$

Proof. Given four real numbers a_1, a_2, b_1, b_2, we have

$$\sqrt{a_1^2 + b_1^2} + \sqrt{a_2^2 + b_2^2} \geqslant \sqrt{(a_1 + a_2)^2 + (b_1 + b_2)^2}$$

By the mathematical induction, we can extend this formula to any number of real numbers, that is

$$\sum_{i=1}^{n} \sqrt{a_i^2 + b_i^2} \geqslant \sqrt{\left(\sum_{i=1}^{n} a_i\right)^2 + \left(\sum_{i=1}^{n} b_i\right)^2} \tag{13}$$

Assume point p is located at $\langle x, y\rangle$, and point p_i is located at $\langle x_i, y_i\rangle$. Then, the geometry center v_m is located at $\langle \frac{\sum_{i=1}^{n} x_i}{n}, \frac{\sum_{i=1}^{n} y_i}{n}\rangle$. If we replace a_i and b_i by $(x - x_i)$ and $(y - y_i)$ respectively, then we have

$$\sum_{i=1}^{n} \sqrt{(x-x_i)^2 + (y-y_i)^2} \geqslant \sqrt{\left(\sum_{i=1}^{n} x - x_i\right)^2 + \left(\sum_{i=1}^{n} y - y_i\right)^2} \geqslant n \times \sqrt{\left(x - \frac{\sum_{i=1}^{n} x_i}{n}\right)^2 + \left(y - \frac{\sum_{i=1}^{n} y_i}{n}\right)^2} = n \times d_e(p, v_m)$$

The proof completes. □

Theorem 5. *Given an MANN query $q(P, Q, n, C)$ and a road network $G(N, E)$, let q_m be the geometry center of Q. Assume* sum *is considered. $\forall p_1, p_2 \in P$, it is certain that p_1 cannot be better than p_2 if $d_e(p_1, q_m) \geqslant \frac{\|p_2.Q \cup p_2.CR\|_{sum}}{|Q|}$.*

Proof. Based on Lemma 4, we have

$$|Q| \times d_e(p_1, q_m) \leqslant \sum_{q_i \in Q} d_e(p_1, q_i). \tag{14}$$

As the Euclidean distance between two objects does not exceed the network distance, we have

$$\sum_{q_i \in Q} d_e(p_1, q_i) \leqslant \sum_{q_i \in Q} \|p_1, q_i\| \tag{15}$$

If $d_e(p_1, q_m) \geqslant \frac{\|p_2.Q \cup p_2.CR\|_{sum}}{|Q|}$, then we have

$$\|p_2, Q \cup p_2.CR\|_{sum} \leqslant |Q| \times d_e(p_1, q_m) \leqslant \sum_{q_i \in Q} \|p_1, q_i\| \leqslant \|p_1, Q \cup p_1.CR\| \tag{16}$$

Here, $p_i.CR$ is the candidate result corresponding to p_i. It is obvious that p_1 cannot be better than p_2. Our proof completes. □

When sum is considered, all the maximum distance maintained in Algorithm 4 need to be replaced by the summation of distances. In addition, we need to replace the termination condition listed in Lines 10–11 of Algorithm 4 with the following code, according to Theorem 5. With these two changes, algorithm SO can support the aggregate function sum.

```
if (d_e ⩾ d/|Q|) then
    break;
```

4. Experimental study

In this section, we conduct extensive experiments with the real and synthetic datasets to evaluate the performance of the proposed algorithms for supporting MANN queries. The performance metrics considered include the total execution time, edges expanded (i.e., the number of the edges expanded by the query processing), the number of the unpruned target points

and the extent of the search area. The last metric, i.e., the extent of the search area, refers to the extent of the sub-network formed by all the edges visited by a query. All the algorithms and the underlying data structures (e.g., R-Tree and Grid) were implemented in C++. The experiments were conducted on a machine with an Intel Core i7-3770 CPU @ 3.40 GHz and 32 GB RAM.

We use the real road network London [19] in our experiments. The network has 236,456 nodes, 300,702 edges and 34,341 target points. Based on the road network, we generate the query points and candidate points uniformly.

In our experiments, we study five parameters. They are (i) the number of target points $|P|$, (ii) the number of query points $|Q|$, (iii) the Euclidean extent of Q, (iv) the value of n, and (v) the number of candidate points $|C|$. Table 3 lists the settings for each parameter, with bold values representing the default settings. In each set of experiments, we only change the value of one parameter while others are fixed at their default values.

Notice that the value of $|P|$ is set based on the total number of nodes on the road network. For example, when $|N|$ = 236,456, and $|P|$ = 2%, there are in total $236,456 \times 2\%$ (i.e., 4730) target points. Note we set $|P|$ to 2% as it is the largest individual real group. In the London dataset, each target point has several keywords associated with it. We group the target points with the same keyword into the same cluster. As the largest cluster has only 4933 target points, we combine several groups together to generate larger groups. The Euclidean extent of Q is defined as the ratio of the area of the minimum bounding rectangle (MBR) of Q to the overall area of the MBR corresponding to the road network, i.e., $\frac{MBR(Q).area}{MBR(G).area}$. For example, assume the MBR of $G(N, E)$ is a 4000×5000 rectangle and the Euclidean extent of Q is 4%. The MBR of Q could be any rectangle whose area is $4000 \times 5000 \times 4\% = 800,000$ such as a 800×1000 rectangle or a 1600×500 rectangle. This parameter decides how close the query points are located. In each set of experiments, 100 queries are generated randomly, and the average performance is reported in this paper.

In total we implement four algorithms, namely PO, SO, CA and BASE. Here, BASE is identical to FP[21] and it refers to a baseline algorithm that is very similar as SO. The only difference is that BASE uses Algorithm 1 to retrieve candidate result instead of Algorithm 3. In the following, we first report the performance of all the algorithms under max and then report their performance under sum.

4.1. Impact of P under max

Our first set of experiments is to evaluate the impact of $|P|$ on the search performance for max, with the results shown in Fig. 4. The vertical axis of Fig. 4(b) displays the ratio of the number of the edges expanded to the total number of the edges on the network. Note that as we only develop two pruning strategies in this work, we only display the result of PO and SO in Fig. 4(c) to show the efficiency of our pruning strategies. The efficiency is measured by the ratio of the number of the visited target points to the total number of target points on the network. We also depict the search area by the ratio of the number of visited edges to the total number of the edges on the network in Fig. 4(d).

It is obvious that all the three algorithms perform much better than the baseline algorithm. For example, compared with the baseline algorithm, when $|P| = 2\%$, PO takes about 34% execution time while SO and CA only take about 5% execution time. Based on the visit ratio, we can see that Theorem 3 employed by SO discards about 85% target points and Theorems 2 and 1 employed by PO prune away more than 95% target points. The superior pruning power explains why PO outperforms the baseline algorithm.

Our second observation is that as $|P|$ increases, both the execution time and the number of edges expanded increase, which is consistent with our expectation. That is because when the number of target points becomes larger, more target points are located within a given sub-network. Most likely, all the algorithms have to explore more target points. However, parameter $|P|$ has very limited impact on SO and CA. From Fig. 4(d) we can see that, even more target points need to be explored, the search area of all the algorithms stay stable, because the pruning distance will not enlarge with the increase of $|P|$. It demonstrates that only the target points located in the nearby area of the query set will be processed and the search area of these target points overlap a lot. Refer to SO and CA, the distances between these target points and the query points have already been recorded in the hash tables of the query points and can be reused, finding the candidate result of more target points does not bring much more cost. From this we can see that, the effect of Algorithm 3 will be more notable when there are more target points on the network.

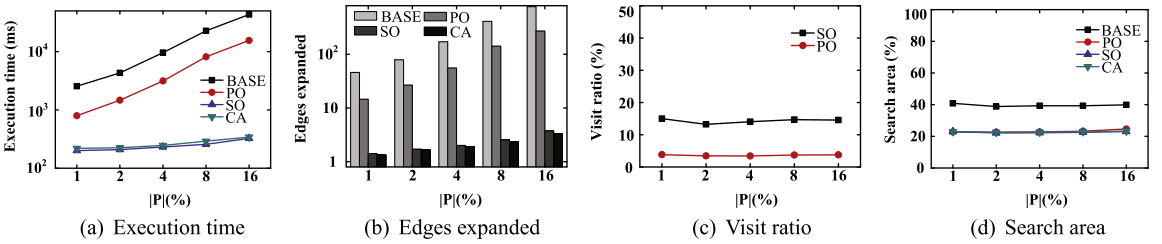
Another observation we make is that the differences between SO and CA are almost negligible. From our statistics, the baseline algorithm needs to visit more than 14 candidate points in C on average to complete the process of finding candidate result, while both SO and CA only need to visit at most one candidate point. In other words, when Algorithm 3 is used to retrieve candidate result, most target points can be pruned away once their maximum distance to Q are derived. It demonstrates that the parameter γ in the time complexity of SO (Section 3.2.4) is very small, and it is the reason why SO outperforms PO significantly while CA performs almost the same as SO.

4.2. Impact of Q under max

Our second set of experiments is to evaluate the impact of set Q on the search performance, including $|Q|$ and the Euclidean extent of Q. The experimental results are shown in Figs. 5 and 6. As we explain before, Q's Euclidean extent is defined as the ratio of the area of the MBR that bounds all the query points to the area of the MBR bounding the road network.

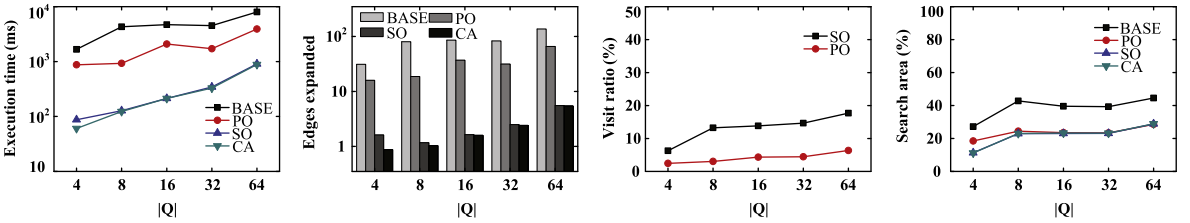
Table 3
Parameter settings.

Parameter	Value
Size of P	1%, **2%**, 4%, 8%, 16% of $\|N\|$
Size of Q	4, 8, **16**, 32, 64
Q's Euclidean extent	1%, 2%, **4%**, 8%, 16%
Value of n	4, 8, **16**, 32, 64
Size of C	100, 200, **500**, 1000, 2000



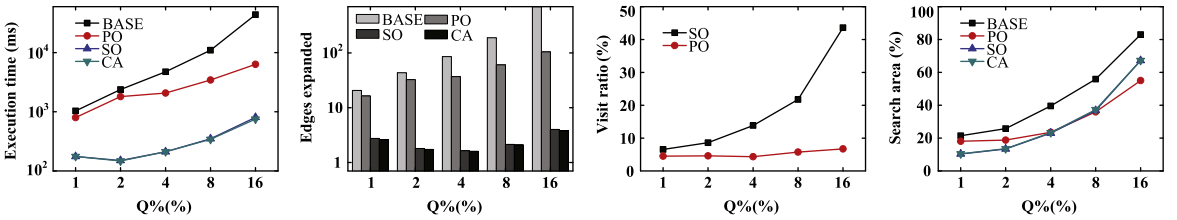
(a) Execution time (b) Edges expanded (c) Visit ratio (d) Search area

Fig. 4. The impact of $|P|$ for function max.



(a) Execution time (b) Edges expanded (c) Visit ratio (d) Search area

Fig. 5. The impact of $|Q|$ for function max.



(a) Execution time (b) Edges expanded (c) Visit ratio (d) Search area

Fig. 6. The impact of the Euclidean extent of Q for function max.

We observe that when these two parameters increase, the execution time of all four algorithms increases. As $|Q|$ and the Euclidean extent of Q increase, it is very likely the maximum distance from a target point to the query set enlarges as well and hence less target points are pruned. From Fig. 5(c) we can see that, for the two pruning strategies, the number of the visited target points is almost tripled when $|Q|$ increases from 4 to 64, and the impact of the Euclidean extent of Q on the visit ratio of SO is more significant, as shown in Fig. 6(c). In addition, the increase of the pruning distance means the restriction to Algorithms 1 and 3 reduces. In other words, the parameter β in the time complexity becomes larger, which leads to the increase of the cost of all these four algorithms.

We also observe that the impact of $|Q|$ on SO and CA is more significant than it on BASE and PO. According to the performance analysis conducted in Section 3.2.4, we understand that the efficiency of SO depends on $|Q|$, while the efficiency of PO only depends on the number of unvisited target points. Overall, SO and CA still have the best performance.

Our third observation is that the search area of all the algorithms expands when Q increases. Unlike the increase of $|P|$, when there are more target points to explore, these target points are always located faraway from Q, and hence the search area expands. Note that Q has a more significant impact on the search area. It is easy to see that the search area is around the Euclidean extent of Q, so the expand of Q does have a direct impact on the search area.

4.3. Impact of candidate result under max

Our third set of experiments is to investigate the impact of the candidate result on the overall performance, including n and $|C|$, with the results shown in Figs. 7 and 8 respectively.

We observe that when n increases, the cost of all the four algorithms increases. As more candidate points are requested, the maximum distance between a target point and its candidate result usually enlarges, which means more target points need to be processed. It is worth noting that, initially as n increases from 4 to 8 to 16, the cost of SO and CA remains almost same and the cost increase is negligible; however when n changes from 32 to 64, the cost change of SO and CA becomes obvious. There are a few reasons. First, $\|p, Q \cup p.CR\|_{max}$ may depend on $\|p, p.CR\|_{max}$ as there are much more candidate points in the candidate result of a target point. That means for a considerable part of the target points, after their maximum distances to Q are derived, they cannot be pruned away by Algorithm 3 based purely on the distance. Algorithm 3 still needs to retrieve their candidate result. The second reason is that for an unpruned target point, it needs to visit more candidate points to construct its candidate result. From our statistics, SO and CA need to visit about 3.8 and 7.5 candidate points respectively on average to complete the process of retrieving candidate result when n is 64. However, the number decreases to less than 0.2 when n is 4.

We also observe that when there are more candidate points on the network, the performance of all the algorithms is improved. As more candidate points are available, it is very likely that for a given target point, its distance to its nth nearest neighbor becomes smaller. This contributes to the shortening of the pruning distance which in turn prunes away more target points as Fig. 8(c) shows. In addition, in contrast to the case when n increases, for Algorithm 3, more target points can be discarded just after their maximum distances to Q are computed. That is why the search performance of our algorithms improves.

As mentioned above, the enlargement of n ($|C|$) will lead to the increase (decrease) of the pruning distance, which in turn causes the expansion (shrink) of the search area. To make it simple, n and $|C|$ always have the opposite impacts on the algorithm performance.

4.4. Experiments for sum function

In our last set of experiments, we evaluate the perfomance of SO algorithm for supporting sum function. In order to demonstrate the efficiency of SO algorithm, we also implement a baseline algorithm. Our baseline algorithm is very similar as SO. The only difference is that Algorithm 3 used in SO is replaced by Algorithm 1. As algorithm SO and the baseline algorithm implement the same pruning rule, we only report the visit ratio once in all the figures related to visit ratio. Similar as the experiments conducted previously, we also investigate the impact of five parameters, and they are $|P|$, $|Q|$, the Euclidean extent of Q, n, and $|C|$, with the results shown in Figs. 9–13 respectively.

Most of the results obtained under sum are similar as those obtained under the function max. We observe that SO outperforms the baseline algorithm in most cases although the search area of SO is always larger than that under the baseline algorithm. According to the number of the edges expanded, we can find that re-using of the distance really helps to improve the efficiency of the algorithm.

Our second observation is that SO does not perform that well under sum, as compared with it under max. When the function is sum, Algorithm 3 first computes $\|p, Q\|_{sum}$ for a target point p. Unlike $\|p, Q\|_{max}$ that usually equals to $\|p, Q \cup p.CR\|_{max}$ in our experiments, $\|p, Q\|_{sum}$ is absolutely smaller than $\|p, Q \cup p.CR\|_{sum}$. Consequently, Algorithm 3 needs to retrieve the candidate result for more target points which results in the increase of the cost.

Another observation is that the impact of $|Q|$ and n on SO under sum is different from that on SO under max. The baseline algorithm cuts down its execution time as $|Q|$ increases. This is because given a fixed n setting, when $|Q|$ enlarges, the pruning distance $\frac{\|q_m.NN, Q \cup (q_m.NN).CR\|}{|Q|}$ used by these two algorithms decreases and hence less target points are evaluated which helps to improve the search performance. From Fig. 10(c) we can see that, when $|Q|$ increases to 64, the algorithms only need to explore about 20% target points explored when $|Q|$ is 4. Even though, both the execution time and the number of the edges expanded of SO increase. As discussed before, the time complexity of SO is related to $|Q|$, and the experiment result shows that the benefit of the pruning of the target points cannot pay off the cost brought by the increase of $|Q|$. Fortunately, $|Q|$ is fixed for a given query. We can define a threshold θ for a given road network (e.g. 25 in this network) and simply combine the two algorithms, when $|Q|$ is smaller than θ, SO can be used to process the query, otherwise BASE is used. Thus we can always get the best performance.

The impact of n is just contrary to that of $|Q|$. Given a fixed $|Q|$, when n enlarges, the pruning distance $\frac{\|q_m.NN, Q \cup (q_m.NN).CR\|}{|Q|}$ increases and hence more target points are evaluated, as shown in Fig. 12(c)).

5. Related work

In this section, we review existing works that are related to MANN, including the nearest neighbor (NN) search, the aggregate nearest neighbor (ANN) search, and some relevant group queries on road networks.

NN search on road networks has been well studied. Work presented in [2] introduces a storage scheme for object search on road networks and proposes two algorithms, namely IER and INE. IER utilizes the Euclidean distance as the lower bound

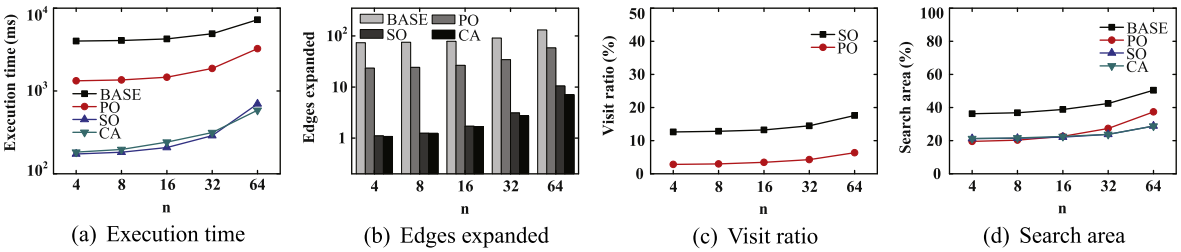


Fig. 7. The impact of n for function max.

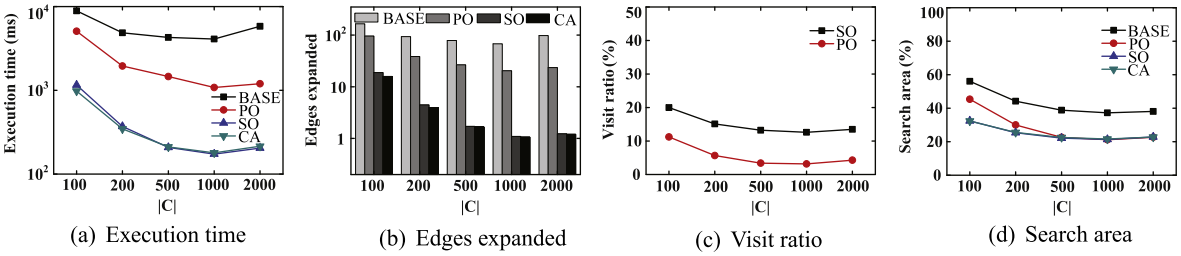


Fig. 8. The impact of $|C|$ for function max.

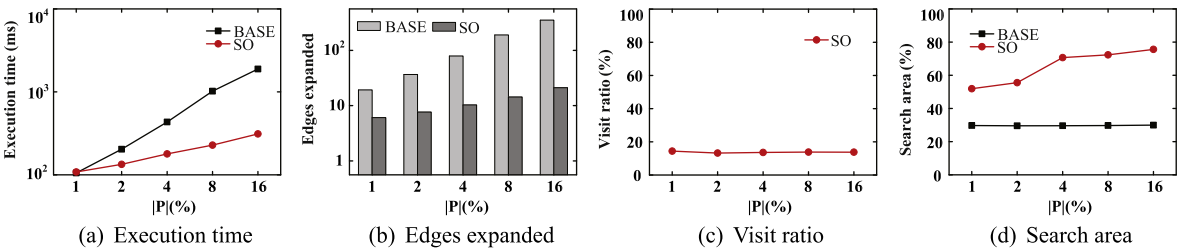


Fig. 9. The impact of $|P|$ for function sum.

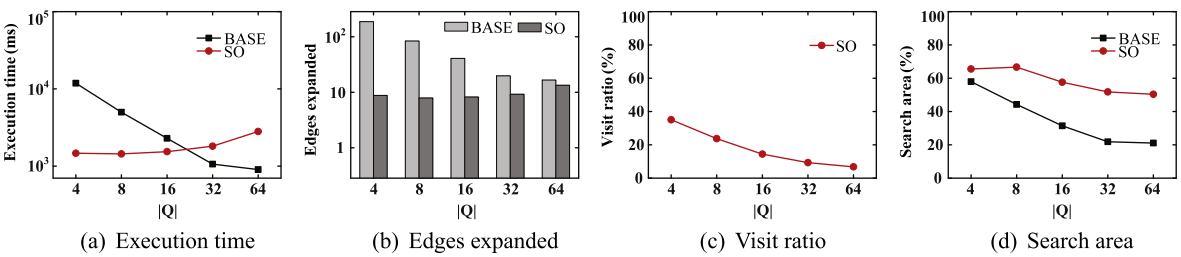


Fig. 10. The impact of $|Q|$ for function sum.

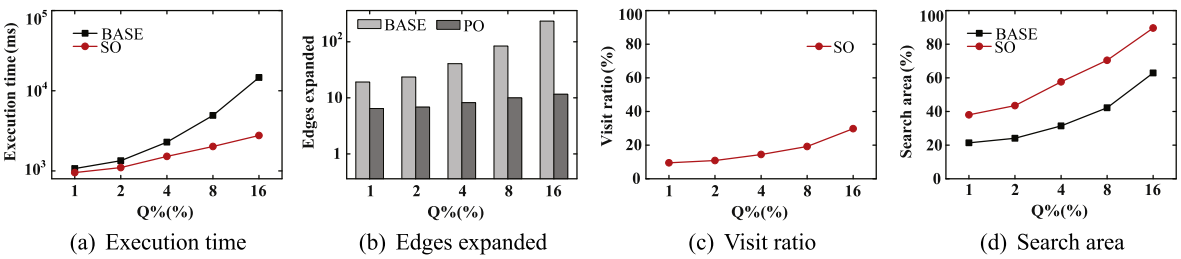


Fig. 11. The impact of the Euclidean extent of Q for function sum.

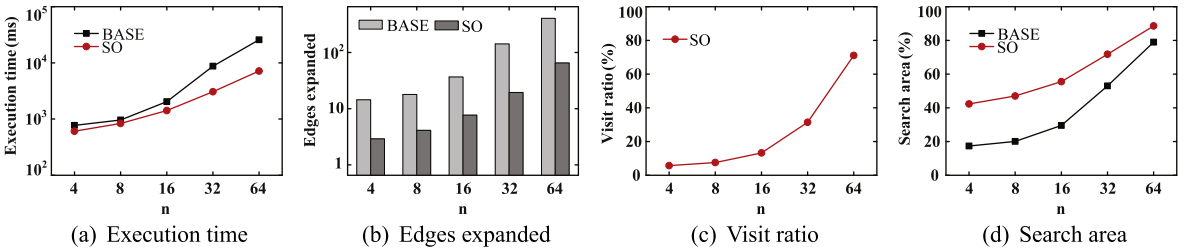


Fig. 12. The impact of n for function sum.

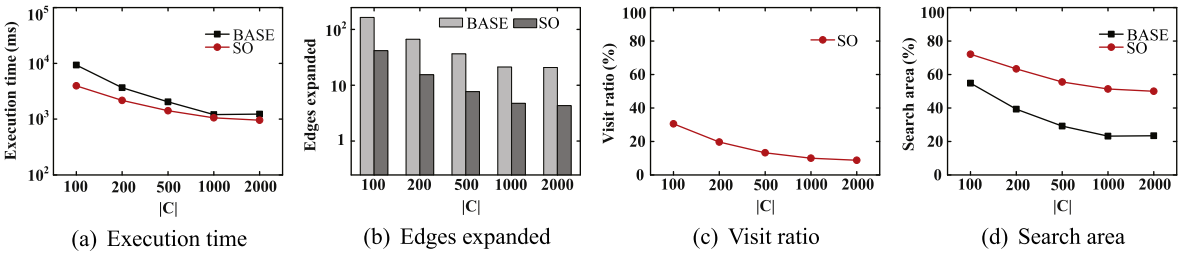


Fig. 13. The impact of $|C|$ for function sum.

of the network distance for object pruning, while INE incrementally expands the network and searches the objects from the query location. The Network Voronoi Diagram (NVD) storage scheme is introduced in [15], together with a corresponding *k*NN search algorithm. A network graph embedding approach to *k*NN queries on road networks is proposed in [7]. [5] proposes a distance signature based index for *k*NN search on road networks. A best-first *k*NN search algorithm on shortest path quadtree is proposed in [8]. Ref. [11] presents a framework for fast object search on hierarchical road networks, and proposes an *k*NN search algorithm which is based on pre-computing the object abstract and the shortcut of each region on the road network. Some other variants of nearest neighbor queries on road networks have also been studied in the literature, e.g., reverse nearest neighbor (RNN) queries [16], continuous nearest neighbor (CNN) queries [13,6], in route nearest neighbor queries [9], and path nearest neighbor (PNN) queries [22]. All the existing works mentioned above consider only one single query point and hence they are different from the MANN query studied in this paper.

ANN query on road networks [17,14] share some similarities with MANN query proposed in this paper, but they are different. The main difference is that, ANN query assumes that the query points are given, while in MANN queries, the query points consist of the given query points and multiple points that are needed to be selected from a candidate set. MANN queries will find the optimal aggregate point as well as the undetermined query points. Ref. [17] proposes three algorithms, namely Incremental Euclidean Restriction (IER), Threshold algorithm (TA), and Concurrent Expansion (CE). Ref. [14] presents a pruning technique for ANN queries based on the network Voronoi diagram. The optimal meeting point problem [3], spatial skyline queries [18], and multi-source skyline queries [12] are also relevant to multiple-query-point problem on road networks. However, these problems also assume query points are available so they are different to MANN queries in this paper. Ref. [21] is a previous work of us on MANN and will be compared in this paper.

The Social-Temporal Group Query (STGQ) problem [1] is also related to our work. Given the query point, STGQ is to find a group of members such that the social relation between the members is tight. However, STGQ considers only social distance and ignores the location of the members, so it is different from MANN queries. The Circle of Friend Queries (CoFQ) [20] extends the STGQ problem into the geo-social networks and considers both spatial and social proximity. However, CoFQ measures the closeness of a group using the diameter, but not the sum or maximum of the distance from the members to the aggregate location. Another similar work is the Aggregate Keyword Routing (AKR) problem [10], which is to find the aggregate point considering both spatial proximity and textual relevance between the spatial-textual objects [4] and the query points. AKR problem also assumes that the query points are given, so it is different from MANN queries.

6. Conclusions and future work

This paper studies the merged aggregate nearest neighbor (MANN) query. We develop two algorithms for processing this query when the aggregate function is max. The Pruning-Oriented (PO) algorithm considers the distribution of both the query set and the candidate set and tries to prune away as many target points as possible. The experimental results show that it can discard a considerable part of target points which in turn save the execution time. In addition, we also develop the Searching-Oriented (SO) algorithm to accelerate the distance computation to the query set, and the experiment results show that SO always keeps the good performance. We provide a theoretical analysis on these two algorithms and extend SO to support sum

function. In our future work, we plan to develop new algorithms for aggregate function sum to get better performance and apply our approach into social networks and explore the possibility of using MANN queries to support real-life social network applications.

Acknowledgements

This research is supported in part by the National Natural Science Foundation of China (NSFC) under Grant 61073001 and Natural Science Foundation of Shanghai under Grant 14ZR1403100.

References

[1] D.N. Yang, Y.L. Chen, W.C. Lee, M.S. Chen, On social–temporal group query with acquaintance constraint, Proc. VLDB Endow. 4 (6) (2011) 397–408.
[2] D. Papadias, J. Zhang, N. Mamoulis, Y. Tao, Query processing in spatial network databases, in: Proceedings of the 29th International Conference on Very Large Data Bases (VLDB), vol. 29, 2003, pp. 802–813.
[3] D. Yan, Z. Zhao, W. Ng, Efficient algorithms for finding optimal meeting point on road networks, Proc. VLDB Endow. 4 (11) (2011).
[4] G. Cong, C.S. Jensen, D. Wu, Efficient retrieval of the top-k most relevant spatial web objects, Proc. VLDB Endow. 2 (1) (2009) 337–348.
[5] H. Hu, D.L. Lee, V. Lee, Distance indexing on road networks, in: Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB), 2006, pp. 894–905.
[6] H.J. Cho, C.W. Chung, An efficient and scalable approach to cnn queries in a road network, in: Proceedings of the 31st International Conference on Very Large Data Bases (VLDB), 2005, pp. 865–876.
[7] H. Kriegel, P. Kröger, P. Kunath, M. Renz, T. Schmidt, Proximity queries in large traffic networks, in: Proceedings of the 15th Annual ACM International Symposium on Advances in Geographic Information systems (GIS), 2007, p. 21.
[8] H. Samet, J. Sankaranarayanan, H. Alborzi, Scalable network distance browsing in spatial databases, in: Proceedings of ACM International Conference on Management of Data (SIGMOD), 2008, pp. 43–54.
[9] J.S. Yoo, S. Shekhar, In-route nearest neighbor queries, GeoInformatica 9 (2) (2005) 117–137.
[10] K. Chen, W. Sun, C. Tu, C. Chen, Y. Huang, Aggregate keyword routing in spatial database, in: Proceedings of the 20th International Conference on Advances in Geographic Information Systems (GIS), 2012, pp. 430–433.
[11] K.C. Lee, W.C. Lee, B. Zheng, Y. Tian, Road: a new spatial object search framework for road networks, IEEE Trans. Knowl. Data Eng. (TKDE) 24 (3) (2012) 547–560.
[12] K. Deng, X. Zhou, H.T. Shen, Multi-source skyline query processing in road networks, in: Proceedings of the 23rd International Conference on Data Engineering (ICDE), 2007, pp. 796–805.
[13] K. Mouratidis, M.L. Yiu, D. Papadias, N. Mamoulis, Continuous nearest neighbor monitoring in road networks, in: Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB), 2006, pp. 43–54.
[14] L. Zhu, Y. Jing, W. Sun, D. Mao, P. Liu, Voronoi-based aggregate nearest neighbor query processing in road networks, in: Proceedings of the 185th Annual ACM International Symposium on Advances in Geographic Information Systems (GIS), 2010, pp. 518–521.
[15] M. Kolahdouzan, C. Shahabi, Voronoi-based k nearest neighbor search for spatial network databases, in: Proceedings of the 30th International Conference on Very Large Data Bases (VLDB), vol. 30, 2004, pp. 840–851.
[16] M.L. Yiu, D. Papadias, N. Mamoulis, Y. Tao, Reverse nearest neighbors in large graphs, IEEE Trans. Knowl. Data Eng. (TKDE) 18 (4) (2006) 540–553.
[17] M.L. Yiu, N. Mamoulis, D. Papadias, Aggregate nearest neighbor queries in road networks, IEEE Trans. Knowl. Data Eng. (TKDE) 17 (6) (2005) 820–833.
[18] M. Sharifzadeh, C. Shahabi, L. Kazemi, Processing spatial skyline queries in both vector spaces and spatial network databases, ACM Trans. Database Syst. (TODS) 34 (3) (2009) 14.
[19] João B. Rocha-Junior, K. Nørvåg, Top-k spatial keyword queries on road networks, in: Proceedings of the 15th International Conference on Extending Database Technology (EDBT), 2012, pp. 168–179.
[20] W. Liu, W. Sun, C. Chen, Y. Huang, Y. Jing, K. Chen, Circle of friend query in geo-social networks, in: Database Systems for Advanced Applications (DASFFA), 2012, pp. 126–137.
[21] W. Sun, C. Chen, B. Zheng, C. Chen, L. Zhu, W. Liu, Y. Huang, Merged aggregate nearest neighbor query processing in road networks, in: Proceedings of the 22nd International Conference on Information and Knowledge Management (CIKM), 2013, pp. 2243–2248.
[22] Z. Chen, H.T. Shen, X. Zhou, J.X. Yu, Monitoring path nearest neighbor in road networks, in: Proceedings of ACM International Conference on Management of Data (SIGMOD), 2009, pp. 591–602.