Merged Aggregate Nearest Neighbor Query Processing in Road Networks

Weiwei Sun
Fudan University
wwsun@fudan.edu.cn

Chong Chen
Fudan University
chenchong@fudan.edu.cn

Baihua Zheng
Singapore Management University
bhzheng@smu.edu.sg

Chunan Chen
Fudan University
chenchunan@fudan.edu.cn

Liang Zhu
Fudan University
zhuliang@fudan.edu.cn

Weimo Liu
Fudan University
liuweimo@fudan.edu.cn

ABSTRACT

Aggregate nearest neighbor query, which returns a common interesting 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 an effective algorithm to process MANN query in road networks based on our pruning strategies. Extensive experiments are conducted to examine the behaviors of the solutions and the overall experiments show that our strategies to minimize the response time are effective and achieve several orders of magnitude speedup compared with the baseline methods.

Categories and Subject Descriptors

H.2.8 [**Database Application**]: Spatial databases

Keywords

Query Processing; Nearest Neighbor; Road Networks

1. INTRODUCTION

Location-based services (LBSs) become more and more important in our everyday life. Worldwide revenues from LBSs are expected to break the US$4 billion mark in 2012, 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, a 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., $q(P, Q, n, C) = \{\langle p, C_s\rangle | p \in P \wedge C_s \in \Gamma(C, n) \wedge \forall p' \in P, \forall C' \in \Gamma(C, n), f(p, C_s \cup Q) \leq f(p', C' \cup Q)\}$. Here, $f(p, S)$ is the aggregate distance function and it could be sum or max or others based on application needs. Take sum as an example, $f(p, S) = \sum_{\forall s \in S} ||p, s||$ with $||p, s||$ returning the network distance between p and s in 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 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, target set P is the set of basketball courts available, 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 total distance from 10 participants' locations to the court is minimum.

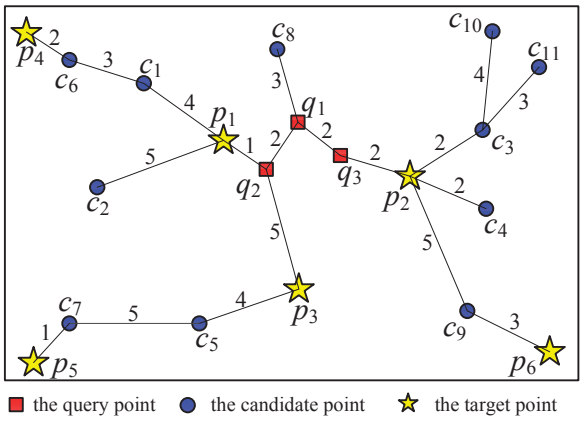


Figure 1: Example of a MANN query

Figure 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 sum, we list in Table 1 some potential answers and MANN will return p_2 as the answer target point, and c_3 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

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CIKM'13, Oct. 27–Nov. 1, 2013, San Francisco, CA, USA.

Copyright 2013 ACM 978-1-4503-2263-8/13/10 ...$15.00.

http://dx.doi.org/10.1145/2505515.2505738.

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

p	$\sum_{\forall q \in Q} \|\|p, q\|\|$	C_s	$\sum_{\forall c \in C_s} \|\|p, c\|\|$
p_1	3+1+5=9	$\{c_1, c_2\}$	4+5=9
p_2	4+6+2=12	$\{c_3, c_4\}$	2+2=4
p_3	7+5+9=21	$\{c_5, c_7\}$	4+9=13
p_4	12+10+14=36	$\{c_1, c_6\}$	2+5=7

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 [11]. 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* and *c-oriented*. The 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, the 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.

Motivated by the fact that existing algorithms cannot support MANN queries efficiently, we develop a few pruning strategies in this paper to prune away unnecessary target points and develop a search algorithm to process MANN queries efficiently.

Our main contribution presented in this paper is threefold. (1) We formalize the MANN query in road networks. (2) We propose an efficient algorithm to support MANN queries based on our pruning strategies. (3) We perform comprehensive simulation study based on the real dataset to evaluate our algorithm and the experimental results demonstrate that our algorithm achieves excellent search performance and also has great scalability.

2. PRELIMINARY

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

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
$\|\|p, S\|\|_{sum}$	the aggregate network distance from point p and all the points in set S, i.e., $\|\|p, S\|\|_{sum} = \sum_{\forall s \in S} \|\|p, s\|\|$
$\|\|p, S\|\|_{max}$	the maximum network distance from p to a point of S, i.e., $\forall s \in S$, $\|\|p, s\|\| \leq \|\|p, S\|\|_{max}$ and $\exists s' \in S$ such that $\|\|p, s'\|\| = \|\|p, S\|\|_{max}$

We model a road network G as a weighted graph that consists of a set of nodes N and 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 smallest 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 $G(N, E)$, a 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 and C_s is minimum, i.e., $q(P, Q, n, C) = \{\langle p, C_s \rangle | p \in P \wedge C_s \in \Gamma(C, n) \wedge \forall p' \in P, \forall C' \in \Gamma(C, n), f(p, C_s \cup Q) \leq f(p', C' \cup Q)\}$* □

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 sum or max or other operations based on application needs. If sum is considered, $f(p, S) = ||p, S||_{sum}$; if max is considered, $f(p, S) = ||p, S||_{max}$. To simplify our discussion, we use sum as the default setting for $f(p, S)$.

To facilitate our discussion, we also introduce a concept, namely *Candidate Result*, as defined in Definition 2. If function f considers sum, the candidate result $p.CR$ of a target point p actually contains the n nearest candidate points. Take the running example depicted in Figure 1 as the example. $p_1.CR = \{c_1, c_2\}$, $p_2.CR = \{c_3, c_4\}$, and $p_3.CR = \{c_5, c_7\}$. 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 $f(p_1, p_1.CR \cup Q) \leq f(p_2, p_2.CR \cup Q)\}$. Back to our running example. Target points p_2 is better than p_1 as $f(p_1, p_1.CR \cup Q) = ||p_1, \{c_1, c_2, q_1, q_2, q_3\}||_{sum} = 18$, and $f(p_2, p_2.CR \cup Q) = ||p_2, \{c_3, c_4, q_1, q_2, q_3\}||_{sum} = 16$.

Definition 2. ***Candidate Result.*** *Given a target point $p \in P$ and a 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., $\forall C' \in \Gamma(C, n)$, $f(p, p.CR) \leq f(p, C')$. In other words, if p is returned by $q(P, Q, n, C)$, $p.CR$ must be the corresponding candidate points returned.*

3. FAST-PRUNING 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 set C_s formed by n candidate points of C. In this section, we develop a pruning strategy to improve the performance of p-oriented algorithm via pruning away certain target points. In the following, we first present how to find the candidate result, based on which pruning strategy is developed, finally we present the *Fast-Pruning (FP) Algorithm*.

3.1 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 explores 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 FP is based on $f(p, Q \cup p.CR)$ and Algorithm 1 returns not only $p.CR$ but also $f(p, Q \cup p.CR)$. If we consider max, instead of maintaining the aggregated distance, it needs to maintain the maximum distance. As the adjustment is straightforward, we omit the details to save space. It's worth noting that finding the candidate result of a target point can be costly, so reducing the times of invoking Algorithm 1 is quite important, which is the main point of our work.

Algorithm 1 Find Candidate Result

Input: $p, Q, C, n, G(N, E)$
Output: $p.CR, ||p, Q \cup p.CR||_{sum}$

```
1: if |C| < n then
2:    return p.CR = ∅, ||p, Q ∪ p.CR||_sum = ∞;
3: aggregate_dist ← 0; C_p ← ∅;
4: H ← new min-heap;
5: insert H(p, ||p, p|| = 0);
6: while (H ≠ ∅) and (|Q| > 0 or |p.CR| < n) do
7:    (u, ||u, p||) ← H.deheap();
8:    if u ∈ Q then
9:       update aggregate_dist by ||u, p||;
10:      remove u from Q;
11:   else if u ∈ C and |p.CR| < n then
12:      update aggregate_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 ∪ p.CR||_sum = aggregate_dist;
```

3.2 Pruning Rules for Target Points

Based on a given target point p and its corresponding candidate result, we develop two pruning strategies for different aggregate distance functions to prune away certain target points that definitely will not produce results better than p. In the following, we first present the corresponding lemma that builds the theory foundation of our pruning strategy and then present our pruning rules.

Lemma 1. *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*

$$n \times d_e(p, v_m) \leq \sum_{v_i \in V} d_e(p, v_i) \tag{1}$$

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

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

By the mathematical induction, we have

$$\sum_{i=1}^{n} \sqrt{{a_i}^2 + {b_i}^2} \geq \sqrt{(\sum_{i=1}^{n} a_i)^2 + (\sum_{i=1}^{n} b_i)^2} \tag{3}$$

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

$$\begin{aligned} \textstyle\sum_{i=1}^{n} \sqrt{(x-x_i)^2+(y-y_i)^2} &\geq \sqrt{(\sum_{i=1}^{n} x-x_i)^2+(\sum_{i=1}^{n} y-y_i)^2} \\ &\geq n \times \sqrt{(x-\frac{\sum_{i=1}^{n} x_i}{n})^2+(y-\frac{\sum_{i=1}^{n} y_i}{n})^2} \\ &= n \times d_e(p, v_m) \end{aligned}$$

The proof completes. ■

Theorem 1. *Given a 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. $\forall p_1, p_2 \in P$, if $d_e(p_1, q_m) \geq \frac{||p_2, Q \cup p_2.CR||_{sum}}{|Q|}$, it is certain that p_1 can't be better than p_2.*

proof Based on Lemma 1, we have

$$|Q| \times d_e(p_1, q_m) \leq \sum_{q_i \in Q} d_e(p_1, q_i). \tag{4}$$

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

$$\sum_{q_i \in Q} d_e(p_1, q_i) \leq \sum_{q_i \in Q} ||p_1, q_i|| \tag{5}$$

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

$$\begin{aligned} ||p_2, Q \cup p_2.CR||_{sum} &\leq |Q| \times d_e(p_1, q_m) \\ &\leq \sum_{q_i \in Q} ||p_1, q_i|| \\ &\leq ||p_1, Q \cup p_1.CR|| \end{aligned} \tag{6}$$

Here, $p_i.CR$ is the candidate result corresponding to p_i. It is obvious that p_1 can't be better than p_2. Our proof completes. ■

The above pruning rule is to prune away certain target points based on sum function. We also develop following pruning rules for max function.

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

$$d_e(p, v_m) \leq \max_{v_i \in V} d_e(p, v_i) \tag{7}$$

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 Figure 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. ■

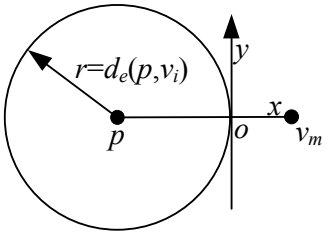


Figure 2: An invalid assumption of Lemma 2

Theorem 2. *Given a 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 $d_e(p_1, q_m) \geq ||p_2, Q \cup p_2.CR||_{max}$, it is certain that p_1 can't be better than p_2.*

Proof. Based on Lemma 2, we have

$$\max_{q_i \in Q} d_e(p_1, q_i) \geq d_e(p_1, q_m) \tag{8}$$

We all understand that

$$max_{q_i \in Q}||p_1, q_i|| \geq \max_{q_i \in Q} d_e(p_1, q_i) \quad (9)$$

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

$$\begin{aligned} ||p_2, Q \cup C_{p_2}||_{max} &\leq \max_{q_i \in Q} d_e(p_1, q_i) \quad (10) \\ &\leq max_{q_i \in Q}||p_1, q_i|| \\ &\leq ||p_1, Q \cup p_1.CR||_{max} \end{aligned}$$

It is obvious that p_1 can't be better than p_2 and our proof completes. ■

3.3 FP Algorithm

Based on the pruning rules we propose, the FP algorithm is developed. Its pseudo-code is listed in Algorithm 2. Since our pruning rule (i.e., Theorem 1) is based on Euclidean distance, we assume all the target points are indexed by a R-tree with its root node $root$ being an input for Algorithm 2. The algorithm mainly contains two steps. The first step is to complete the initialization for the pruning distance d, which captures the aggregate distance corresponding to current result maintained by S (lines 1-7). The second step is to prune away the target points based on Theorem 1 and update the pruning distance d if necessary. They 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 based on best-first NN search [1]. We then retrieve the candidate result $(q_m.NN).CR$ corresponding to $q_m.NN$ based on Algorithm 1, and initialize the pruning distance d as $||q_m.NN, Q \cup (q_m.NN).CR||_{sum}$. In most cases, the target point closer to the center of Q usually has a smaller aggregate 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 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's 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 the $d/|Q|$, it is guaranteed that all the remaining entries in H and all the unvisited objects will have their mindist to q_m larger than $d/|Q|$ and can be pruned away based on Theorem 1. Otherwise, we need evaluate e. If e is a non-leaf node, all its child nodes are en-heap 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.

Lemma 3. *The result identified by Algorithm 2 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 2. Given the fact that the target point p is not returned by Algorithm 2, it must be enclosed by a node N such that $mindist(N, q_m) \geq \frac{||e, Q \cup e.CR||_{sum}}{|Q|}$. In other words, $d_e(p, q_m) \geq mindist(N, q_m) \geq \frac{||e, Q \cup e.CR||_{sum}}{|Q|}$. Based on Theorem 1, we understand that p cannot be better than e. 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 object e (Line 16). However, it is not always necessary to locate the candidate result if we know the aggregated distance generated by e 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. In addition, Algorithm 2 can be easily adjusted to support max function based on Theorem 2. We skip the details for space saving.

if $(aggregate\_dist \geq ||S.e, Q \cup S.C_e||_{sum})$ **then**
  return $S = \emptyset, aggregate\_dist = \infty$;

Algorithm 2 FP Algorithm for sum 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 Algorithm1;
6: $d \leftarrow ||q_m.NN, Q \cup (q_m.NN).CR||_{sum}$;
7: initialize result $S \leftarrow \langle q_m.NN, q_m.NN.CR \rangle$;
8: **while** H is not empty **do**
9: $\quad \langle e, d_e \rangle \leftarrow H.deheap()$;
10: $\quad$ **if** $d_e \geq d/|Q|$ **then**
11: $\quad\quad$ breaks;
12: $\quad$ **else if** e is not an object **then**
13: $\quad\quad$ **for** each child $e.c$ of e **do**
14: $\quad\quad\quad H.enheap(\langle e.c, mindist(q_m, e.c) \rangle)$;
15: $\quad$ **else**
16: $\quad\quad$ retrieve e's candidate result $e.CR$ based on Algorithm1;
17: $\quad\quad$ **if** $||e, Q \cup e.CR||_{sum} \leq d$ **then**
18: $\quad\quad\quad d \leftarrow ||e, Q \cup e.CR||_{sum}$; $S \leftarrow \langle e, e.CR \rangle$;
19: return S;

In the following, we use an example, as shown in Figure 1, to illustrate how FP algorithm works. Here, $Q = \{q_1, q_2, q_3\}$, $P = \{p_1, p_2, p_3, p_4, p_5, p_6\}$, $n = 2$, and C=$\{c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8, c_9, c_{10}, c_{11}\}$. First, we derive the geometry center q_m of all the query points, denoted as the green triangle in Figure 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, and we maintain all the unexamined nodes in H for later exploration. Thereafter, we invoke Algorithm 1 to find the candidate result $p_1.CR$ of p_1, that is $\{c_1, c_2\}$. We initialize the result set S as $\langle p_1, p_1.CR \rangle$, and the pruning distance d is set to $||p_1, Q \cup p_1.CR||_{sum} = 18$. 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 $\frac{||p_1, Q \cup p_1.CR||_{sum}}{|Q|} = 6$, we only need to explore the target points whose Euclidean distance to q_m do not exceed 6, i.e., the target points located inside the shaded circle centered at q_m as shown in Figure 3 [1]. Then, we retrieve p_2 as the next NN of q_m and its corresponding candidate result and update the pruning

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

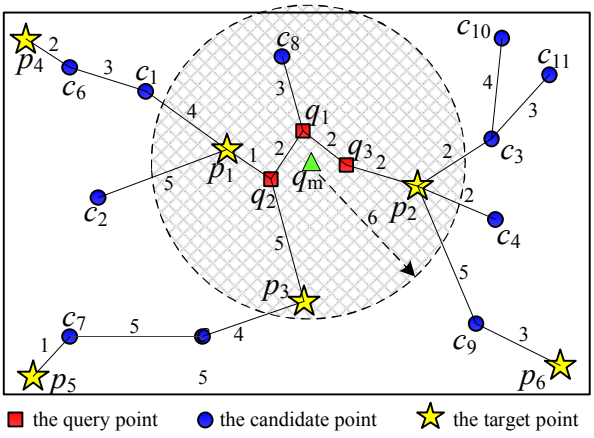
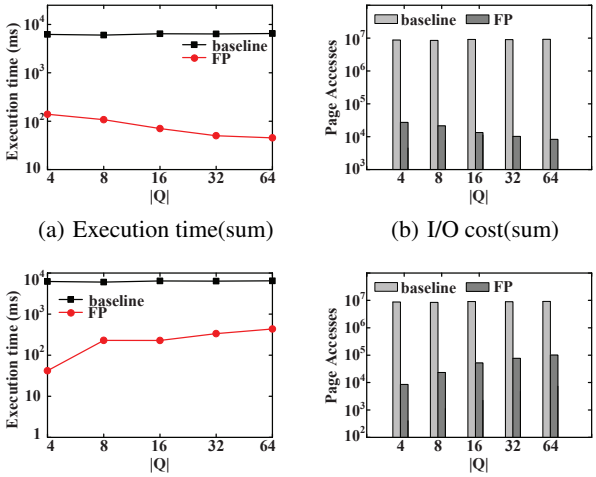


Figure 3: Process of the running example



(a) Execution time(sum) (b) I/O cost(sum)

(c) Execution time(max) (d) I/O cost(max)

Figure 4: The impact of $|Q|$

distance to a smaller value 16. Next, we explore target point p_3 and retrieve its candidate result $\{c_5, c_7\}$. After the evaluation of p_3, the algorithm can terminate as the rest objects have their mindist to q_m larger than $d/|Q|$ (i.e., $\frac{16}{3}$). The algorithm ends and the final result is $\langle p_2, p_2.CR(=\{c_3, c_4\})\rangle$.

4. EXPERIMENTAL STUDY

In this section, we conduct several experiments to evaluate the performance of proposed FP algorithm in supporting MANN queries. In addition, we also implement baseline algorithms, i.e., the *p-oriented* algorithm explained in Section 1 as the competitors. The performance metrics considered include the total execution time and I/O cost. All the algorithms were implemented in C++. The experiments are conducted on a machine with an Intel Core i7-3770 CPU @ 3.40GHz and 16GB RAM. We assume the buffer size is 1Mb, managed by LRU, and the page size is 4Kb.

We use the real road network California Road Network[6]. Based on the road network, we generate the target points and candidate points uniformly. Three parameters are studied, and they are i) the number of query points $|Q|$, ii) the Euclidean extent of Q, iii) the value of n, . In each set of experiments, we only change the value of one parameter while fixing others at their default values. Table 3 lists the settings for each parameter, and the bold one represents the default setting. Notice that 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. In each set of experiments, 10 queries are generated randomly, and the average performance is reported in the following.

Table 3: Parameter settings

Parameter	Value
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

4.1 Impact of $|Q|$

Our first set of experiments is to evaluate the impact of $|Q|$ on the search performance, with the results shown in Figure 4. It is obvious that FP performs much better than the baseline algorithm. For example, compared with the baseline algorithm, when $|Q| =$ 16 and sum is the function considered, FP only takes about 1% execution time.

Our second observation is that the impact of $|Q|$ is quite different between different functions. When sum is considered, FP 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 FP algorithm decreases and hence less target points are evaluated which helps to improve the search performance. From our statistics, when $|Q|$ increases to 64, FP only needs to explore about 26% target points of that when $|Q|$ is 4.

On the other hand, when max is considered, the cost of FP increases. As $|Q|$ increases, it is very likely the maximum distance from a target point to a query point enlarges as well and hence less target points are pruned.

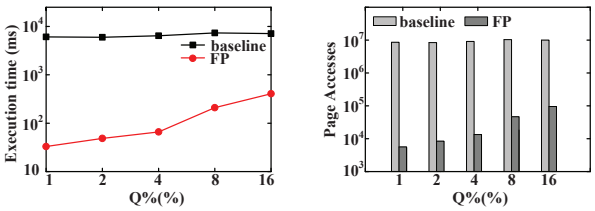
4.2 Impact of Q's Euclidean extent

Our second set of experiments is to evaluate the impact of the Euclidean extent of Q on the search performance, shown in Figure 5.

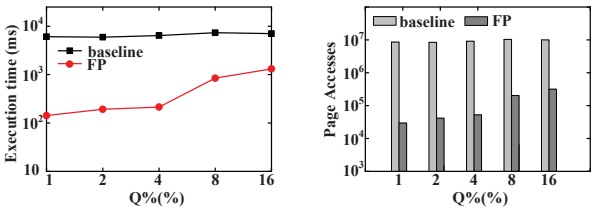
As the Euclidean extent of Q enlarges, the pruning distance used in our algorithms always gets larger, and we compare the Euclidean distance $d_e(p, q_m)$ between the evaluated target point p and Q's geometry center q_m with the pruning distance. As the Euclidean extent of Q does not affect $d_e(p, q_m)$, more target points need to be visited as the Euclidean extent of Q increases. Nevertheless, FP is still 7 times faster than the baseline algorithm in the worst case, i.e., $Q\%$ is 16 and max is considered.

4.3 Impact of n

Our last set of experiments is to investigate the impact of n on the overall performance, and the result is depicted in Figure 6. We observe that the impact of n is dependent on the function considered. when sum is considered, both the execution time and I/O cost increase as n increases. The reason is just contrary to that of $|Q|$, given a fixed $|Q|$ setting, when n enlarges, the pruning distance $\frac{||q_m.NN, Q \cup (q_m.NN).CR||}{|Q|}$ used by our algorithms increases and hence more target points are evaluated which leads to the increase of the cost.

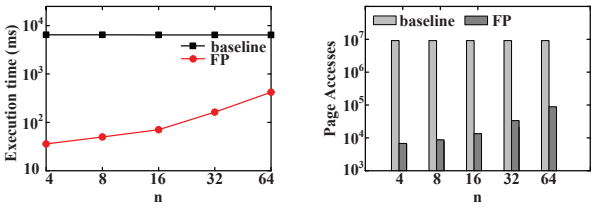


(a) Execution time(sum) (b) I/O cost(sum)

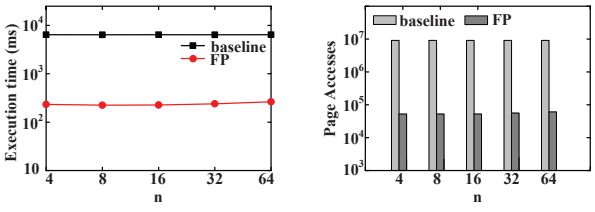


(c) Execution time(max) (d) I/O cost(max)

Figure 5: The impact of the Euclidean extent of Q



(a) Execution time(sum) (b) I/O cost(sum)



(c) Execution time(max) (d) I/O cost(max)

Figure 6: The impact of n

On the other hand, when max is considered, the effect of n is much less. But notice that, the cost starts to increase a little notable when n increases to 64, which is four times of $|Q|$. It means $||p, Q \cup p.CR||_{max}$ may be affected by $||p, p.CR||_{max}$ as there are much more candidate points in the candidate result of a target point.

5. RELATED WORK

Nearest Neighbor(NN) query and its variants in road networks have been well studied. [8] introduces a storage scheme for object search in road networks and proposes two algorithms. Hereafter many efficient algorithms have been proposed [3, 4, 2, 9, 5].

The most related studies to our problem are group queries [11, 12, 7, 10]. [11] is the first work to process ANN query in road networks. [12] presents a pruning technique based on the network Voronoi diagram to accelerate ANN query processing. [7, 10] can be regarded as two special cases of ANN query. While they all assume that the query points are given, while in MANN queries, the query points are consisted of the given query points and multiple points needed to be selected from a candidate set. MANN queries will find the optimal aggregate point as well as the undetermined query points.

6. CONCLUSIONS

This paper studies the merged aggregate nearest neighbor (MANN) query. We develop an algorithm for processing this query, the Fast-Pruning algorithm. It uses the Euclidean aggregate distance between a target point and the query set as the pruning distance to prune away unnecessary target points, the experiment results show that it can discard a considerable part of target points which in turn save the execution time and I/O cost.

7. ACKNOWLEDGEMENTS

This research is supported in part by the National Natural Science Foundation of China (NSFC) under grant 61073001.

8. ADDITIONAL AUTHORS

Additional author: Yan Huang (University of North Texas, e-mail: huangyan@ntu.edu)

9. REFERENCES

[1] G. R. Hjaltason and H. Samet. Distance browsing in spatial databases. *TODS*, 24(2):265–318, 1999.

[2] H. Hu, D. L. Lee, and V. Lee. Distance indexing on road networks. In *PVLDB*, pages 894–905, 2006.

[3] M. Kolahdouzan and C. Shahabi. Voronoi-based k nearest neighbor search for spatial network databases. In *VLDB*, pages 840–851, 2004.

[4] H. Kriegel, P. Kröger, P. Kunath, M. Renz, and T. Schmidt. Proximity queries in large traffic networks. In *GIS*, page 21, 2007.

[5] K. C. Lee, W.-C. Lee, B. Zheng, and Y. Tian. Road: a new spatial object search framework for road networks. *TKDE*, 24(3):547–560, 2012.

[6] F. Li, D. Cheng, M. Hadjieleftheriou, G. Kollios, and S. Teng. On trip planning queries in spatial databases. *SSTD*, pages 923–923, 2005.

[7] D. Papadias, Q. Shen, Y. Tao, and K. Mouratidis. Group nearest neighbor queries. In *ICDE*, pages 301–312, 2004.

[8] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao. Query processing in spatial network databases. In *VLDB*, pages 802–813, 2003.

[9] H. Samet, J. Sankaranarayanan, and H. Alborzi. Scalable network distance browsing in spatial databases. In *SIGMOD*, pages 43–54, 2008.

[10] D. Yan, Z. Zhao, and W. Ng. Efficient algorithms for finding optimal meeting point on road networks. *PVLDB*, 4(11), 2011.

[11] M. Yiu, N. Mamoulis, and D. Papadias. Aggregate nearest neighbor queries in road networks. *TKDE*, 17(6):820–833, 2005.

[12] L. Zhu, Y. Jing, W. Sun, D. Mao, and P. Liu. Voronoi-based aggregate nearest neighbor query processing in road networks. In *GIS*, pages 518–521, 2010.