Mobile Information Systems 9 (2013) 37–53
DOI 10.3233/MIS-2012-0146
IOS Press

A safe exit algorithm for continuous nearest neighbor monitoring in road networks

Hyung-Ju Cho*, Se Jin Kwon and Tae-Sun Chung
Department of Information & Computer Engineering, Ajou University Woncheon-dong, Suwon Si Yeongtong-gu, Gyeonggi-Do, South Korea

Abstract. Query processing in road networks has been studied extensively in recent years. However, the processing of moving queries in road networks has received little attention. In this paper, we introduce a new algorithm called the Safe Exit Algorithm (SEA), which can efficiently compute the safe exit points of a moving nearest neighbor (NN) query on road networks. The safe region of a query is an area where the query result remains unchanged, provided that the query remains inside the safe region At each safe exit point, the safe region of a query and its non-safe region meet so that a set of safe exit points represents the border of the safe region. Before reaching a safe exit point, the client (query object) does not have to request the server to re-evaluate the query This significantly reduces the server processing costs and the communication costs between the server and moving clients. Extensive experimental results show that SEA outperforms a conventional algorithm by up to two orders of magnitude in terms of communication costs and computation costs

Keywords: Continuous monitoring, nearest neighbor query, safe exit algorithm, road network

1. Introduction

The points of interest (POIs; e.g., accommodation, restaurants, and gas stations) marked on web mapping services such as Google Maps, Bing Maps, and Yahoo Maps are located in a road network and their proximity is measured as their shortest path distances [7,8,13,14]. A k nearest neighbor (kNN) query is one of the most important query types in location-based services [2,16], where a user submits a kNN query to the service provider for the k-nearest objects (i.e., POIs) to his/her current location. With the advances in map software, location detection devices, wireless communication, and database systems, the kNN query has been extended from the Euclidean space to the road network environment [2, 5,6,15,16,18,29], where the user can submit the kNN query to request his/her k-nearest objects on the basis of the network distance. In many cases, the kNN query result based on the network distance is more relevant to the user than that based on the Euclidean distance because the user's movement is typically restricted by an underlying road network.

Recently, the kNN query has been further extended via continuous monitoring. However, continuous monitoring algorithms for moving queries related to POIs in road networks have received little attention [26]. We can consider an example of a monitoring query, i.e., "find the three closest restaurants to my current location for the next 10 minutes". The main idea of continuous nearest neighbor monitoring

*Corresponding author: Hyung-Ju Cho, Department of Information & Computer Engineering, Ajou University Woncheon-dong, Suwon Si Yeongtong-gu, Gyeonggi-Do, 443-749, South Korea. Tel.: +82 31 219 2535; Fax: +82 31 219 1834; E-mail: hjcho@ajou.ac.kr.

is to find the k-nearest objects of interest to a user's current location while he/she moves freely. This study assumes that the path is not known in advance and that the user moves arbitrarily in road networks.

The key problem with continuous monitoring algorithms is maintaining the freshness of the query answer when the query point moves freely and arbitrarily. A simple approach involves the client q (i.e., query point) requesting the server to re-evaluate the query periodically (e.g., every second). However, such a periodic monitoring approach cannot solve the problem because the query answer may still become stale in between each call to the server. This approach also places an excessive computation burden on the server side as well as the high communication frequency burden imposed on the communication channel.

To overcome the problems (i.e., excessive computation and communication costs) of periodic monitoring, safe region-based algorithms have been introduced [4,9,26,27]. The safe region of a query is the region where a query answer remains unchanged, provided that the query point is within the safe region. The safe region approach allows a client to get fresh query answers without excessive overheads on the server side or communication channel. However, the network bandwidth required to provide the client with a safe region (which may consist of complex road segments) is more than that required to provide a set of safe exit points [26] that represent the boundary of the corresponding safe region.

We address this issue by proposing a new Safe Exit Algorithm (SEA), which efficiently computes the safe exit points for moving NN queries in road networks. A safe exit point denotes a point where the safe region of q and its non-safe region meet. Safe exit points satisfy the following four constraints which are introduced in [1] for the design of the safe region in the Euclidean space: (1) lightweight construction, (2) compact representation, (3) fast containment check and (4) device heterogeneity. In particular, in road networks, the set of safe exit points is more concise than the safe region (which may consist of complex road segments); hence, it incurs a low communication cost between clients and the server. Until a client q reaches a safe exit point, he/she is guaranteed to remain in the safe region and thus the query answer is valid. Upon traveling beyond the safe exit point, the client requests the server to evaluate the query in order to refresh the query answer and its safe exit points.

The following are the distinguishing features of our study: (1) SEA focuses on the computation of safe exit points for moving NN queries in road networks, whereas Yung et al. [26] focus on the computation of safe exit points for moving range queries. (2) SEA does not require the path for the computation of safe exit points, whereas existing works (e.g. [6,18]) focus solely on the computation of split points for a given path. The split points indicate the locations in the path, at which the k NNs of a moving query object change. The contributions of this paper are summarized as follows:

- We propose a Safe Exit Algorithm (SEA) for the continuous monitoring of moving NN queries over static objects by assuming that the path is not known in advance and that query points move arbitrarily in road networks
- We present both lemmas and mathematical analyses that make it possible to effectively determine the safe exit points of a moving NN query.
- A thorough experimental study confirms that SEA clearly outperforms a traditional approach that evaluates queries periodically, in terms of both communication and computation costs.

The remainder of the paper is structured as follows: Section 2 reviews existing work on spatial queries in road networks. Section 3 presents the specific background and formulates the preliminaries of the problem. Section 4 elaborates on the proposed SEA for computing the safe exit points of moving NN queries in road networks. We present the performance analysis in Section 5, followed by the conclusion in Section 6.

2. Related work

Significant research attention has been given to developing techniques for spatial queries in road networks. NN queries [5,15,16,20,28–30] and range queries [4,11,16,19,20,22–26] are among the most studied spatial queries in road networks.

The related research can be classified into the following four categories, according to the mobility of the queries and the objects: (1) static queries for static objects, (2) static queries for moving objects, (3) moving queries for static objects, and (4) moving queries for moving objects. Section 2.1 reviews previous studies that have dealt with static queries for static objects or moving objects. Sections 2.2 and 2.3 review previous studies of moving queries for static objects and moving queries for moving objects, respectively.

2.1. Static queries for static/moving objects

Papadias et al. [16] proposed a framework to support nearest neighbor queries, closest pairs queries, range queries and distance joins in a road network. However, they assumed that queries and objects have fixed positions in a spatial network. Wang et al. [20] proposed an infrastructure known as MOVNET that utilizes dual index structures for location-based services with moving objects in road networks. Based on the infrastructure, they presented two algorithms for processing snapshot range queries and kNN queries. Pesti et al. [17] proposed a road network-based query-aware location update framework known as RoadTrack to reduce the communication costs of moving objects. RoadTrack partitions the road network into precincts and identifies the relevant range queries for each precinct. A moving object reports its latest location to the server only when it enters/leaves a precinct or the range of some query.

2.2. Moving queries for static objects

Our work belongs to this category, in which queries move freely while data objects are static. Chen et al. [5] studied path k-NN queries that return kNNs with respect to the shortest path that connects the destination to the user's current location. Bao et al. [2] introduced a new type of query known as a k-range nearest neighbor (kRNN) query to find the k closest objects for every point on road segments inside a given query region on the basis of the network distance. Cheema et al. [4] proposed a safe region-based approach to the continuous monitoring of range queries for static objects in Euclidean space and in road networks. They devised pruning rules and a unique access order in order to efficiently compute the safe region. Xuan et al. proposed several Voronoi-based algorithms for continuous range queries [23] and continuous kNN queries [29] in road networks. Wang et al. extended the functionality of MOVNET [20] to support continuous range query processing [21]. Yung et al. [26] proposed an algorithm to compute the safe exit points of a moving range query for static objects in road networks. Safar et al. [18] and Cho et al. [6] proposed eDAR algorithm and UNICONS algorithm, respectively, both of which effectively compute split points within a given path in order to support continuous NN queries in road networks.

SEA is characterized by the following features First, unlike the previous work [11], SEA makes very few assumptions on moving clients. These assumptions are as follows: (1) No computational capabilities, (2) no storage capabilities, and (3) no velocity assumptions. Second, existing algorithms (e.g. [6,18]) compute split points for a given path, whereas SEA computes safe exit points for a given query point. In other words, SEA considers a more practical situation where the path is not known apriori as well as the moving speed and direction of the client are arbitrary.

Finally, as in the case of most continuous monitoring algorithms [4,15,26], this study assumes (1) a client server model and (2) main-memory query evaluation on the server side. In other words, the client (i.e., a moving query) can communicate with the server via a wireless communication infrastructure (e.g., cellular services and Wi-Fi) and the server's main memory (e.g., 4 GB memory) is, in general, sufficiently large to accommodate the entire dataset.

2.3. Moving queries for moving objects

Mouratidis et al. [15] addressed the issue of processing continuous k-NN queries by proposing two algorithms (i.e., IMA and GMA) that handle arbitrary object and query movement patterns on road networks. Liu et al. [11] developed a distributed processing technique to solve the moving range queries for moving objects. Their strategy leverages the computational power of the moving objects and each moving object reports to the server when it affects the results of one or more queries. Kriegel et al. [10] studied the problem of proximity monitoring in road networks. Given a proximity threshold and a set of moving objects, a server responsible for proximity monitoring continuously reports the pairs of objects that are within a specific distance of each other. Stojanovic et al. [19] proposed a technique for the continuous monitoring of range queries for moving objects where the mobility pattern (e.g., speed, direction, and route) of the moving objects is utilized. However, the assumption of being able to predict the trajectories of moving objects is not always acceptable. This approach cannot be used if the prediction of an object's movements fails (e.g., a shopper randomly strolling in a shopping mall).

3. Preliminaries

Section 3.1 defines terms and notations used in the paper, while Section 3.2 formulates our continuous monitoring problem using an example of a road network.

3.1. Definition of terms and notations

Road network A road network is represented by a weighted undirected graph $G = (NEW)$ where $N = \{n_1, n_2, \ldots, n_{|N|}\}$ is a set of nodes, $E = \{e_1, e_2, \ldots, e_{|E|}\}$ is a set of edges (i.e., road segments) each of which connects two distinct nodes, and $W(e)$ is a function that returns a weight for an edge e in G. Each edge is given the length of its corresponding road segment as a weight. Note that $|S|$ denotes the cardinality of a set S.

Sequence A sequence $SQ_{(n_s, n_{s+1}, \ldots, n_e)}$ is a path between two nodes, n_s and n_e, such that the degrees of n_s and n_e are not equal to 2 and all intermediate nodes $n_{s+1}, \ldots, n_{e-1}$ on the path have a degree of 2. In particular, the sequence where a query q remains is called the *active sequence* of q. The two end nodes, n_s and n_e, of the sequence are referred to as the boundary nodes. The sequence length is the total weight of the edges in the sequence. A boundary node with a smaller node id is referred to as the base node of the sequence. This study assumes that $n_s \leqslant n_e$. Hence, n_s is the base node.

To simplify the presentation, Table 1 summarizes the notations used in this paper.

3.2. Problem formulation

To provide a clear explanation, we use the example road network shown in Fig. 1 where there are five POIs (i.e., a, b, c, d, and e) and a NN query q that requests three NNs. The number on each edge

Table 1
Notations used in the paper

Notation	Definition
$G = (N, E, W)$	The graph model of road network
$A_q = \{c, d, e\}$	The length of the shortest path between objects a and b
n_i	A node in the road network
$e_i = (n_j, n_k)$	An edge in the edge set E
$W(e_i) = d(n_j, n_k)$	The weight of the edge $e_i = (n_j, n_k)$
$SQ_{(n_s, n_{s+1}, \ldots, n_e)}$	A sequence where n_s (or n_e) is the start (or end) boundary node and the others $A_q = \{c, d, e\}$ are intermediate nodes. Note that $O_{(n_2,q)} = \{c\}$, $SQ_{(n_2,n_5)}$, and $SQ_{(n_2,n_5)}$ where $A_{n_5} = \{c, d, e\}$ indicates the degree of a node n_i, which is the number of edges incident to n_i
q	A query point in the road network
k	The number of requested NNs
Ap	The set of k NNs at a point p
$O_{(n_s, n_{s+1}, \ldots, n_e)}$	The set of objects on the sequence $SQ_{(n_5,n_6)}$
p_{se}	A safe exit point at which the safe region of q and its non-safe region meet
p_{anchor}	Anchor point which becomes the start point of expansion in the sequence such that $A_{n_6} = \{c, d, e\}$
$o^-_{nearest}$	The nearest non-answer object to a point $p \in G$ such that $d(p, o^-_{nearest}) = MIN(d(p, o^-_1), d(p, o^-_2), \ldots, d(p, o^-_{\lvert O^- \rvert}))$ where $O^- = \{o^-_1, o^-_2, \ldots, o^-_{\lvert O^- \rvert}\}$ indicates the set of non-answer objects
$o^+_{farthest}$	The farthest answer object to a point $p \in G$ such that $d(p, o^+_{farthest}) = MAX(d(p, o^+_1), d(p, o^+_2), \ldots, d(p, o^+_{\lvert O^+ \rvert}))$ where $O^+ = \{o^+_1, o^+_2, \ldots, o^+_{\lvert O^+ \rvert}\}$ indicates the set of answer objects

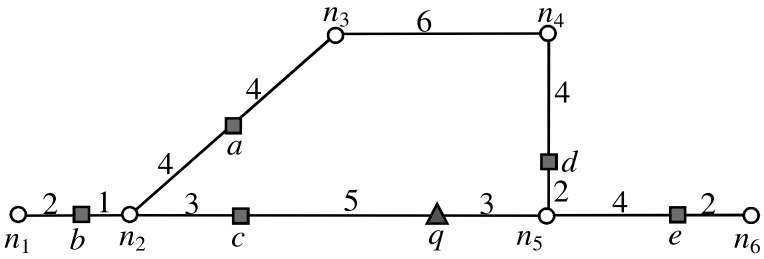


Fig. 1. Example of a road network and a 3-NN query q.

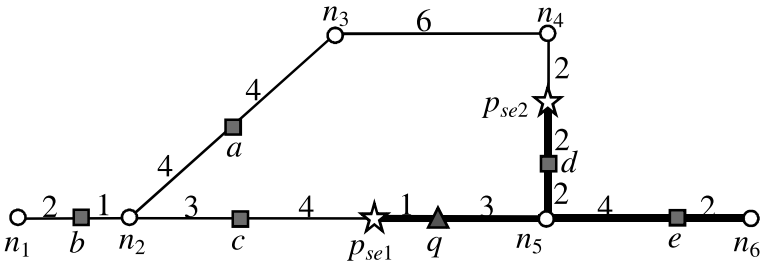


Fig. 2. Safe region of q and its safe exit points p_{se1} and p_{se2}.

indicates the network distance between two adjacent objects. For instance, $d(n_1, b) = 2$, $d(a, n_2) = 4$, etc. For simplicity, we assume that there is only one query q in the system. The active sequence of q is $SQ_{(n_2,n_5)}$ and its length is 11.

Figure 2 shows the safe region of q and its safe exit points of the example in Fig. 1. Note that the example has four sequences, i.e., $SQ_{(n_1,n_2)}$, $SQ_{(n_2,n_5)}$, $SQ_{(n_2,n_3,n_4,n_5)}$, and $SQ_{(n_5,n_6)}$. The query

answer A_q is $A_q = \{c, d, e\}$ since $d(q, a) = 12$, $d(q, b) = 9$, $d(q, c) = 5$, $d(q, d) = 5$, and $d(q, e) = 7$. The safe region and safe exit points are labeled with bold lines and five-pointed stars, respectively. The safe region is the union of the sub-safe region for each sequence. The sub-safe region of a sequence can be expressed as a sequence id and the segment within the sequence. Each end point of the segment can be represented as the distance from the base node to the point.

In the example road network, the safe region (i.e., *q.SR*) of q can be encoded as follows: $q.SR = \{(SQ_{(n_2,n_5)}, [7, 11]), (SQ_{(n_2,n_3,n_4,n_5)}, [16, 20]), (SQ_{(n_5,n_6)}, [0, 6])\}$. Similarly, the set of safe exit points (i.e., P_{SE}) of q can be encoded as follows: $P_{SE} = \{(SQ_{(n_2,n_5)}, 7), (SQ_{(n_2,n_3,n_4,n_5)}, 16)\}$ where $(SQ_{(n_2,n_5)}, 7)$ and $(SQ_{(n_2,n_3,n_4,n_5)}, 16)$ correspond to safe exit points p_{se1} and p_{se2}, respectively. As shown in Fig. 2, until q reaches either p_{se1} or p_{se2}, the three NNs of q are $\{c, d, e\}$. When q passes through either p_{se1} or p_{se2}, the query is re-evaluated based on the updated location of q in order to refresh the query answer and safe exit points.

4. Safe exit algorithm for moving NN query

Section 4.1 elaborates on SEA, which determines the safe exit points of a moving NN query in road networks. Section 4.2 then discusses the computation of the safe exit points of a query in the example road network.

4.1. Safe exit algorithm

First, we formally define a set of safe exit points for a moving NN query in the road network. Let P_{SE} be the set of safe exit points for a k-NN query point q and $O = \{o_1, o_2, \ldots, o_{|O|}\}$ be the set of objects of interest to q. Assume that the answer set (i.e., A_q^+) of q and its non-answer set (i.e., $A_q^- = O - A_q^+$) are $A_q^+ = \{o_1^+, o_2^+, \ldots, o_k^+\}$ and $A_q^- = \{o_{k+1}^-, o_{k+2}^-, \ldots, o_{|O|}^-\}$, respectively. Then, it holds that $d(q, o^+) \leqslant d(q, o^-)$ for an answer object $o^+ \in A_q^+$ and a non-answer object $o^- \in A_q^-$. In addition, $A_q^+ \cap A_q^- = \phi$ and $A_q^+ \cup A_q^- = O$. Finally, P_{SE} is defined as follows:

$$\begin{aligned} P_{SE} &= \{p_{se} \in G | MAX(d(p_{se}, o_1^+), d(p_{se}, o_2^+), \ldots, d(p_{se}, o_k^+)) \\ &= MIN(d(p_{se}, o_{k+1}^-), d(p_{se}, o_{k+2}^-), \ldots, d(p_{se}, o_{|O|}^-))\} \end{aligned}$$

where *MIN*() and *MAX*() return the minimum and maximum values of the input array, respectively. In other words, a safe exit point p_{se} is the midpoint (i.e., $MAX(d(p_{se}, o_1^+), \ldots, d(p_{se}, o_k^+)) = MIN(d(p_{se}, o_{k+1}^-), \ldots, d(p_{se}, o_{|O|}^-))$) between the farthest answer object and the nearest non-answer object.

Algorithm 1 depicts the skeleton for SEA, which identifies the safe exit points for a moving NN query. SEA begins with the exploration of the active sequence The traversal of sequences is terminated if no more sequences can be explored in the queue Each entry in the queue takes the form (*sequence*, *anchor point*) where the anchor point corresponds to a point in the sequence. More precisely, if the sequence to be explored is the active sequence, a query point q becomes the anchor point. Otherwise, either of end nodes (i.e., n_s and n_e) of the sequence becomes the anchor point. Thus, $p_{anchor} \in \{n_s, n_e, q\}$ for a sequence $SQ_{(n_s,n_{s+1},\ldots,n_e)}$. Unless there is a safe exit point in the segment between n_s (or n_e) and the anchor point, the adjacent sequences of n_s (or n_e) are explored.

Finally, the evaluate_query(p, k) function retrieves k NNs from a given point p in the graph. According to our performance evaluation of SEA, the evaluate_query function typically accounts for most of the CPU

Algorithm 1: find_safe_exit_points (qk)

Input: q: query point, k: the number of requested NNs
Output: A_q: a set of k NNs of q, P_{SE}: a set of safe exit points of q

```
1:  queue ← φ                                   /* queue is a FIFO queue */
2:  P_SE ← φ                                    /* P_SE is initialized to the empty set */
3:  A_q ← evaluate_query (q,k) /* A_q is the set of k NNs of q */
4:  queue.push(seq_active,q)   /* seq_active is the active sequence of q*/
5:  while queue is not empty do
6:      (seq, p_anchor) ← queue.pop()
7:      if seq has not been explored before then
8:          Mark seq as explored
9:          P_SE_seq ← find_safe_exit_pt_in_sequence (seq, p_anchor)
10:         P_SE ← P_SE ∪ P_SE_seq
11:         if there is no safe exit point in [n_s, p_anchor] then
12:             queue.push(each adjacent sequence of n_s, n_s)
13:         if there is no safe exit point in [p_anchor, n_e] then
14:             queue.push(each adjacent sequence of n_e, n_e)
15: return (A_q, P_SE) /* query answer A_q and safe exit points P_SE are returned to the query issuer */
```

time particularly when the object density is very low. Therefore, to avoid redundant query evaluation, we use the shared execution paradigm [2,12]. In other words, the query condition (i.e., query location and the number of requested NNs) and its query answer (i.e., the set of NNs from the query location) are stored in the memory and re-cycled.

Algorithm 2: find_safe_exit_pt_in_sequence ($SQ_{(n_s,n_{s+1},...,n_e)}$, p_{anchor})

Input: $SQ_{(n_s,n_{s+1},...,n_e)}$: sequence to be examined, p_{anchor}: anchor point
Output: P_{SE_seq}: a set of safe exit points in $SQ_{(n_s,n_{s+1},...,n_e)}$

```
1:  P_SE_seq ← φ                                   /* P_SE_seq is initialized to the empty set. */
2:  A_p_anchor ← evaluate_query (p_anchor, k)      /* Recall that A_q = A_p_anchor. */
3:  if n_s ≠ p_anchor then  A_n_s ← evaluate_query (n_s, k)
4:  if n_e ≠ p_anchor then  A_n_e ← evaluate_query (n_e, k)
5:  O_(n_s,p_anchor) ← scan_sequence(n_s, p_anchor)/* Let O_(n_s,p_anchor) be the set of objects in segment between n_s
    and p_anchor */
6:  O_(p_anchor,n_e) ← scan_sequence(p_anchor, n_e)/* Let O_(p_anchor,n_e) be the set of objects in segment between
    p_anchor and n_e */
7:  if n_s ≠ p_anchor and A_n_s ∪ O_(n_s,p_anchor) ≠ A_p_anchor then   /* Refer to Lemma 3 */
8:      P_SE_seq ← P_SE_seq∪ find_safe_exit_pt(A_n_s, A_p_anchor, O_(n_s,p_anchor))
9:  if n_e ≠ p_anchor and A_n_e ∪ O_(n_e,p_anchor) ≠ A_p_anchor then   /* Refer to Lemma 3 */
10:     P_SE_seq ← P_SE_seq∪ find_safe_exit_pt (A_n_e, A_p_anchor, O_(p_anchor,n_e))
11: return P_SE_seq
```

Algorithm 2 retrieves the safe exit points in the sequence using A_{n_s}, A_{n_e}, and $O_{(n_s,n_{s+1},...,n_e)}$ Without loss of generality, lemmas 1 to 4 assume that $n_b \neq p_{anchor}$ where $n_b = \{n_s, n_e\}$ is a boundary node and $p_{anchor} = \{n_s, n_e, q\}$ is the anchor point. Naturally, if $n_b = p_{anchor}$, there is no safe exit point between n_b and p_{anchor} Lemmas 1 and 2 are introduced to prove Lemma 3.

Lemma 1. Let $A_{(n_s,n_{s+1},...,n_e)}$ be the union of a set of k NNs at every point in $SQ_{(n_s,n_{s+1},...,n_e)}$. Then, the following formula is satisfied.

$$A_{(n_s,n_{s+1},...,n_e)} = A_{n_s} \cup A_{n_e} \cup O_{(n_s,n_{s+1},...,n_e)}$$

Proof) Lemma 1 is well-known [2,6,18]; hence, the proof is omitted □

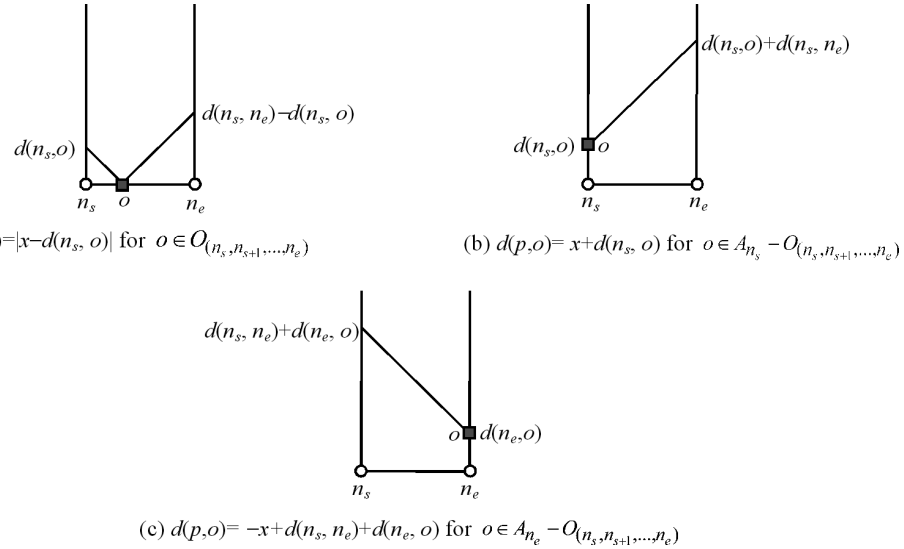


Fig. 3. Determination of $d(po)$ when $o \in O_{(n_s,n_{s+1},\ldots,n_e)} \cup A_{n_s} \cup A_{n_e}$, $p \in SQ_{(n_s,n_{s+1},\ldots,n_e)}$, and $x = d(n_s, p)$ (a) $d(po) = |x - d(n_s, o)|$ for $o \in O_{(n_s,n_{s+1},\ldots,n_e)}$; (b) $d(po) = x + d(n_s, o)$ for $o \in A_{n_s} - O_{(n_s,n_{s+1},\ldots,n_e)}$; (c) $d(po) = -x + d(n_s n_e) + d(n_e, o)$ for $o \in A_{n_e} - O_{(n_s,n_{s+1},\ldots,n_e)}$.

Lemma 2. Let $[n_b, p_{anchor}]$ be the segment between n_b and p_{anchor}, and $A_{(n_b,p_{anchor})}$ be the union of a set of k NNs at every point in $[n_b, p_{anchor}]$. Then, the following formula is satisfied.

$$A_{(n_b,p_{anchor})} = A_{n_b} \cup A_{p_{anchor}} \cup O_{(n_b,p_{anchor})}$$

Proof) Lemma 2 is easily extended from Lemma 1 and the proof is omitted. □

Lemma 3. If $A_{n_b} \cup O_{(n_b,p_{anchor})} \neq A_{p_{anchor}}$, there is a safe exit point p_{se} in the segment.

Proof) This lemma can be proved using the contradiction method. To do so, let us assume that there is no safe exit point in the segment. This means that for each point $p \in [n_b, p_{anchor}]$, $A_p = A_{p_{anchor}}$, which results in $A_{(n_b,p_{anchor})} = A_{p_{anchor}}$. According to Lemma 2, $A_{(n_b,p_{anchor})} = A_{n_b} \cup A_{p_{anchor}} \cup O_{(n_b,p_{anchor})}$. This means that $A_{n_b} = A_{p_{anchor}}$, $O_{(n_b,p_{anchor})} \subseteq A_{p_{anchor}}$, and $A_{n_b} \cup O_{(n_b,p_{anchor})} = A_{p_{anchor}}$. However, this leads to a contradiction to the given condition (i.e., $A_{n_b} \cup O_{(n_b,p_{anchor})} \neq A_{p_{anchor}}$). Therefore, there is a safe exit point p_{se} in the segment. □

Lemma 4. If $A_{n_b} \cup O_{(n_b,p_{anchor})} = A_{p_{anchor}}$, there is no safe exit point in the segment.

Proof) According to Lemma 2, $A_{(n_b,p_{anchor})} = A_{n_b} \cup A_{p_{anchor}} \cup O_{(n_b,p_{anchor})}$. Because $A_{n_b} \cup O_{(n_b,p_{anchor})} = A_{p_{anchor}}$ by the given condition, $A_{(n_b,p_{anchor})} = A_{p_{anchor}}$. This leads to $A_p = A_{p_{anchor}}$ for every point $p \in [n_b, p_{anchor}]$. Therefore, there is no safe exit point in the segment. □

If a sequence satisfies Lemma 3, SEA starts to compute the location of a safe exit point as shown in lines 7 and 9 of Algorithm 2. For this purpose, we introduce $o^-_{nearest}$ and $o^+_{farthest}$. For simplicity, let us assume that $A_{n_b} \cup O_{(n_b,p_{anchor})} - A_{p_{anchor}}$ corresponds to $O^- = \{o^-_1, o^-_2, \ldots, o^-_{|O^-|}\}$ and $A_{p_{anchor}}$ corresponds to $O^+ = \{o^+_1, o^+_2, \ldots, o^+_{|O^+|}\}$. Then, at a point $p \in [n_b, p_{anchor}]$, $o^-_{nearest}$ is referred to as the nearest non-answer object to p such that $d(p, o^-_{nearest}) = MIN(d(p, o^-_1), d(p, o^-_2), \ldots, d(p, o^-_{|O^-|}))$. Similarly, at a point $p \in [n_b, p_{anchor}]$, $o^+_{farthest}$ is referred to as the farthest answer object to p such

Table 2
Summary of $d(p,o)$ for $o \in O_{(n_s,n_{s+1},\ldots,n_e)} \cup A_{n_s} \cup A_{n_e}$ and $x = d(n_s,p)$

Condition	$d(p,o)$
$o \in O_{(n_s,n_{s+1},\ldots,n_e)} - (A_{n_s} \cup A_{n_e})$	$d(p,o) = \lvert x - d(n_s,o)\rvert$
$o \in A_{n_s} - (A_{n_e} \cup O_{(n_s,n_{s+1},\ldots,n_e)})$	$d(p,o) = x + d(n_s,o)$
$o \in A_{n_e} - (A_{n_s} \cup O_{(n_s,n_{s+1},\ldots,n_e)})$	$d(p,o) = -x + d(n_s,n_e) + d(n_e,o)$
$o \in A_{n_s} \cap A_{n_e} - O_{(n_s,n_{s+1},\ldots,n_e)}$	$d(p,o) = MIN(x + d(n_s,o), -x + d(n_s,n_e) + d(n_e,o))$
$o \in O_{(n_s,n_{s+1},\ldots,n_e)} \cap A_{n_s} - A_{n_e}$	$d(p,o) = MIN(\lvert x - d(n_s,o)\rvert, x + d(n_s,o))$
$o \in O_{(n_s,n_{s+1},\ldots,n_e)} \cap A_{n_e} - A_{n_s}$	$d(p,o) = MIN(\lvert x - d(n_s,o)\rvert, -x + d(n_s,n_e) + d(n_e,o))$
$o \in O_{(n_s,n_{s+1},\ldots,n_e)} \cap A_{n_s} \cap A_{n_e}$	$d(p,o) = MIN(\lvert x - d(n_s,o)\rvert, x + d(n_s,o), -x + d(n_s,n_e) + d(n_e,o))$

that $d(p,o^+_{farthest}) = MAX(d(p,o^+_1), d(p,o^+_2), \ldots, d(p,o^+_{|O^+|}))$. The midpoint between $o^-_{nearest}$ and $o^+_{farthest}$ becomes a safe exit point p_{se}. That is, $d(p_{se},o^-_{nearest}) = d(p_{se},o^+_{farthest})$. Next, we elaborate on how we determine $d(po)$ where $o \in A_{n_s} \cup O_{(n_s,n_{s+1},\ldots,n_e)} \cup A_e$ and $p \in SQ_{(n_s,n_{s+1},\ldots,n_e)}$.

Figure 3 shows the determination of $d(p,o)$ for an object $o \in O_{(n_s,n_{s+1},\ldots,n_e)} \cup A_{n_s} \cup A_{n_e}$ and a point $p \in SQ_{(n_s,n_{s+1},\ldots,n_e)}$ according to the location of object o (i.e., $o \in O_{(n_s,n_{s+1},\ldots,n_e)}$, $o \in A_{n_s} - O_{(n_s,n_{s+1},\ldots,n_e)}$ and $o \in A_{n_e} - O_{(n_s,n_{s+1},\ldots,n_e)}$). In the figure, the x-axis represents $d(n_s,p)$, while the y-axis represents $d(p,o)$ We note that $d(p,o)$ can be represented as a function of $x = d(n_s,p)$ for $0 \leqslant x \leqslant d(n_s,n_e)$ As shown in Fig. 3(a), if $o \in O_{(n_s,n_{s+1},\ldots,n_e)}$ $d(p,o) = |x - d(n_s,o)|$ As shown in Fig. 3(b), if $o \in A_{n_s} - O_{(n_s,n_{s+1},\ldots,n_e)}$, $d(p,o) = x + d(n_s,o)$. Finally, as shown in Fig. 3(c), if $o \in A_{n_e} - O_{(n_s,n_{s+1},\ldots,n_e)}$, $d(p,o) = -x + d(n_s,n_e) + d(n_e,o)$

Table 2 summarizes the determination of the $d(p,o)$ value for $p \in SQ_{(n_s,n_{s+1},\ldots,n_e)}$ and $o \in O_{(n_s,n_{s+1},\ldots,n_e)} \cup A_{n_s} \cup A_{n_e}$ If an object belongs to more than two sets (i.e., $o \in A_{n_s} \cap A_{n_e} - O_{(n_s,n_{s+1},\ldots,n_e)}$, $o \in O_{(n_s,n_{s+1},\ldots,n_e)} \cap A_{n_s} - A_{n_e}$, $o \in O_{(n_s,n_{s+1},\ldots,n_e)} \cap A_{n_e} - A_{n_s}$, and $o \in O_{(n_s,n_{s+1},\ldots,n_e)} \cap A_{n_s} \cap A_{n_e}$), the distance is the length of the shortest path connecting two objects. For example, given that $o \in A_{n_s} \cap A_{n_e} - O_{(n_s,n_{s+1},\ldots,n_e)}$, $d(n_s,o) = 3$, $d(n_e,o) = 5$, $d(n_s,n_e) = 20$, and $d(n_s,p) = 6$ $d(p,o) = MIN(x + d(n_s,o), -x + d(n_s,n_e) + d(n_e,o)) = MIN(9,19) = 9$ where $x = d(n_s,p) = 6$

Algorithm 3: find_safe_exit_pt($A_{n_b}, A_{p_{anchor}}, O_{(n_b,p_{anchor})}$)

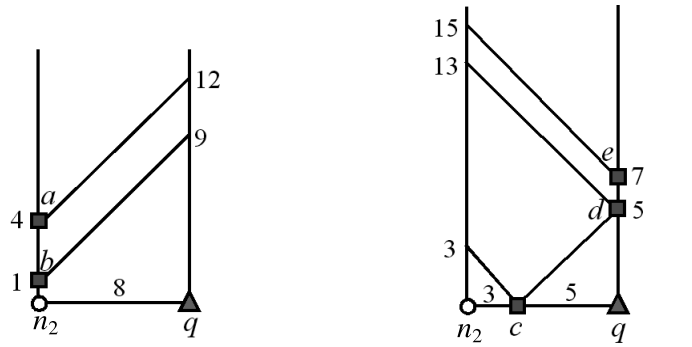
Input: Arguments have the same meanings as before
Output: p_{se}: a safe exit point in the segment $[n_b, p_{anchor}]$

1: $O^-_{nearest} \leftarrow \{\langle p, o^-_{nearest}\rangle |$ for each point $p \in [n_b, p_{anchor}]$, $o^-_{nearest}$ such that $d(p,o^-_{nearest})$ = MIN($d(p, o^-_1), \ldots, d(p,o^-_{|O^-|})$)$\}$
2: $O^+_{farthest} \leftarrow \{\langle p, o^+_{farthest}\rangle |$ for each point $p \in [n_b, p_{anchor}]$, $o^+_{farthest}$ such that $d(p,o^+_{farthest})$ = MAX($d(p, o^+_1), \ldots, d(p,o^+_{|O^+|})$)$\}$
3: /* Note that p_{se} is the midpoint between $o^-_{nearest}$ and $o^+_{farthest}$ */
4: Find the closest point p_{se} to p_{anchor}, such that $d(p_{se},o^-_{nearest}) = d(p_{se},o^+_{farthest})$ for $O^-_{nearest}$ and $O^+_{farthest}$
5: /* If more than two points satisfy $d(p_{se},o^-_{nearest}) = d(p_{se},o^+_{farthest})$, the closest point p_{se} to p_{anchor} is chosen. */
6: **return** p_{se}

Algorithm 3 finds a safe exit point that is located between n_b and p_{anchor}. Let $O^-_{nearest}$ be the set of objects, each of which becomes $o^-_{nearest}$ at a point $p \in [n_b, p_{anchor}]$ and $O^+_{farthest}$ be the set of objects, each of which becomes $o^+_{farthest}$ at the point p. First, we determine $O^-_{nearest}$ and $O^+_{farthest}$. Recall that $o^-_{nearest}$ is the nearest non-answer object to p, whereas $o^+_{farthest}$ is the farthest answer object to p. Since the safe exit point p_{se} is the midpoint between $o^-_{nearest}$ and $o^+_{farthest}$, p_{se} is the solution to the following

Table 3
Computation of the safe exit points for the example road network

Sequence (or segment)	p_{anchor}	A_{n_b}	$A_{p_{anchor}}$	$O_{(n_b,p_{anchor})}$	Safe exit point
$[n_2, q]$ in $SQ_{(n_2,n_5)}$	q	$A_{n_2} = \{a, b, c\}$	$A_q = \{c, d, e\}$	$O_{(n_2,q)} = \{c\}$	p_{se1}=($SQ_{(n_2,n_5)}$, 7)
$[q, n_5]$ in $SQ_{(n_2,n_5)}$	q	$A_{n_5} = \{c, d, e\}$	$A_q = \{c, d, e\}$	$O_{(q,n_5)} = \phi$	none
$SQ_{(n_5,n_6)}$	n_5	$A_{n_6} = \{c, d, e\}$	$A_{n_5} = \{c, d, e\}$	$O_{(n_5,n_6)} = \{e\}$	none
$SQ_{(n_2,n_3,n_4,n_5)}$	n_5	$A_{n_2} = \{a, b, c\}$	$A_{n_5} = \{c, d, e\}$	$O_{(n_2,n_3,n_4,n_5)} = \{a\}$	p_{se2}=($SQ_{(n_2,n_3,n_4,n_5)}$,16)



(a) $O^-_{nearest} = \{\langle[n_2,q],b\rangle\}$ (b) $O^+_{farthest} = \{\langle[n_2,q],e\rangle\}$

Fig. 4. Determination of $O^-_{nearest}$ and $O^+_{farthest}$ for $(SQ_{(n_2,n_5)}, [n_2, q])$. (a) $O^-_{nearest} = \{\langle[n_2, q], b\rangle\}$; (b) $O^+_{farthest} = \{\langle[n_2, q], e\rangle\}$.

equation: $d(p_{se}, o^-_{nearest}) = d(p_{se}, o^+_{farthest})$ If there are found more than two points in $[n_b, p_{anchor}]$, each of which satisfies $d(p_{se}, o^-_{nearest}) = d(p_{se}, o^+_{farthest})$, the closest point to the anchor point is chosen as the safe exit point.

4.2. Computation of the safe exit points for the example

We now discuss the computation of the safe exit points for the query q in the example road network shown in Fig. 1. Table 3 summarizes the computation of the safe exit points of query q for the example road network in Fig. 1. Note that the number (i.e., k) of NNs requested by q is $k = 3$ and the query answer is $A_q = \{c, d, e\}$

As shown in Algorithm 1, SEA first explores the active sequence $SQ_{(n_2,n_5)}$ where q remains. Since $SQ_{(n_2,n_5)}$ is the active sequence, the location of q is the anchor point. Each of the two segments $[n_2, q]$ and $[q, n_5]$ within $SQ_{(n_2,n_5)}$ is explored individually. For $[n_2, q]$, $p_{anchor} = q$, $A_q = \{c, d, e\}$, $A_{n_2} = \{a, b, c\}$, and $O_{(n_2,q)} = \{c\}$. By Lemma 3 (i.e., $A_{n_2} \cup O_{(n_2,q)} \neq A_q$), there exists a safe exit point in the segment $[n_2, q]$. For each point $p \in [n_2, q]$, $o^-_{nearest}$ is selected from the non-answer objects in $A_{n_2} \cup O_{(n_2,q)} - A_q = \{a, b\}$ while $o^+_{farthest}$ is selected from the answer objects in $A_q = \{c, d, e\}$. As shown in Fig. 4(a), $O^-_{nearest} = \{\langle[n_2, q], b\rangle\}$ because $d(p, b) \leqslant d(p, a)$ for every point $p \in [n_2, q]$. Similarly, as shown in Fig. 4(b), $O^+_{farthest} = \{\langle[n_2, q], e\rangle\}$ because $d(p, c) \leqslant d(p, d) \leqslant d(p, e)$ for every point $p \in [n_2, q]$

For $O^-_{nearest} = \{\langle[n_2, q], b\rangle\}$ and $O^+_{farthest} = \{\langle[n_2, q], e\rangle\}$, the midpoint (i.e., $d(o^-_{nearest}, p) = d(o^+_{farthest}, p)$) between $o^-_{nearest}$ and $o^+_{farthest}$ at a specific point p becomes a safe exit point. Thus, the safe exit point p_{se1} is determined as shown in Fig. 5. That is, $d(p_{se1}, b) = d(p_{se1}, e)$ where

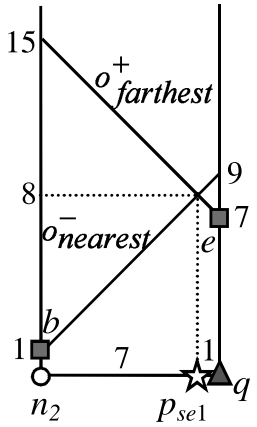


Fig. 5. Determination of location of safe exit point p_{se1}.

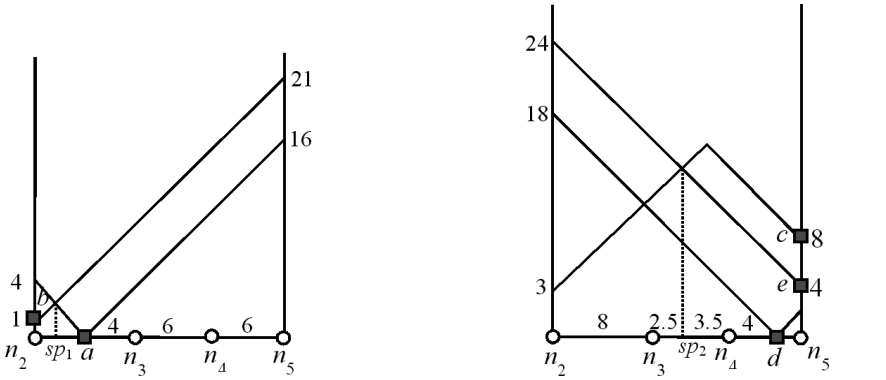


Fig. 6. Determination of $O^-_{nearest}$ and $O^+_{farthest}$ for $SQ_{(n_2,n_3,n_4,n_5)}$. (a) $O^-_{nearest} = \{\langle[n_2, sp_1], b\rangle, \langle[sp_1, n_5], a\rangle\}$; (b) $O^+_{farthest} = \{\langle[n_2, sp_2], e\rangle, \langle[sp_2, n_5], c\rangle\}$.

$d(p_{se1}, b) = x + 1$ and $d(p_{se1}, e) = -x + 15$ for $0 \leqslant x \leqslant 8$. Consequently, $x = 7$. This means that the distance from n_2 to p_{se1} is 7.

According to Lemma 4, there is no safe exit point within the segment $[q, n_5]$ because $A_{n_5} \cup O_{(q,n_5)} = A_q$, as shown in Table 3. Therefore, sequences (i.e., $SQ_{(n_2,n_3,n_4,n_5)}$ and $SQ_{(n_5,n_6)}$) adjacent to n_5 are explored using $p_{anchor} = n_5$. According to Lemma 4, there is also no safe exit point in $SQ_{(n_5,n_6)}$ because $A_{n_6} \cup O_{(n_5,n_6)} = A_{n_5}$, as shown in Table 3.

Finally, we determine a safe exit point in the sequence $SQ_{(n_2,n_3,n_4,n_5)}$. By Lemma 3 (i.e., $A_{n_2} \cup O_{(n_2,n_3,n_4,n_5)} \neq A_{n_5}$ as shown in Table 3), a safe exit point exists in the sequence. For each point $p \in SQ_{(n_2,n_3,n_4,n_5)}$, $o^-_{nearest}$ is selected from the non-answer objects in $A_{n_2} \cup O_{(n_2,n_3,n_4,n_5)} - A_{n_5} = \{a, b\}$ while $o^+_{farthest}$ is selected from the answer objects in $A_{n_5} = \{c, d, e\}$. As shown in Fig. 6(a), $O^-_{nearest} = \{\langle[n_2, sp_1], b\rangle, \langle[sp_1, n_5], a\rangle\}$ where a split point $sp_1 = (SQ_{(n_2,n_3,n_4,n_5)}, 1.5)$ is the cross point between $d(p, a) = |x - 4|$ and $d(p, b) = x + 1$ Similarly, as shown in Fig. 6(b) $O^+_{farthest} = \{\langle[n_2, sp_2], e\rangle, \langle[sp_2, n_5], c\rangle\}$ where a split point $sp_2 = (SQ_{(n_2,n_3,n_4,n_5)}, 10.5)$ is the cross point between $d(p, c) = MIN(x + 3, -x + 28)$ and $d(p, e) = -x + 24$ Recall that $p_{anchor} = n_5$ for $SQ_{(n_2,n_3,n_4,n_5)}$

Table 4
Experimental parameter settings

Parameter	Default value	Range
Number of POIs (N_{POI})	50k	1, 5, 50, 70, 100 (k)
Number of queries (N_{qry})	5k	1, 3, 5, 7, 10 (k)
Number of requested NNs (k)	32	8, 16, 32, 64, 128
Distribution of POIs (D_{POI})	uniform	(U)niform,(S)kewed
Distribution of queries (D_{qry})	skewed	(U)niform,(S)kewed
Query speed (V_{qry})	60 km/h	20, 40, 60, 80, 100 (km/h)

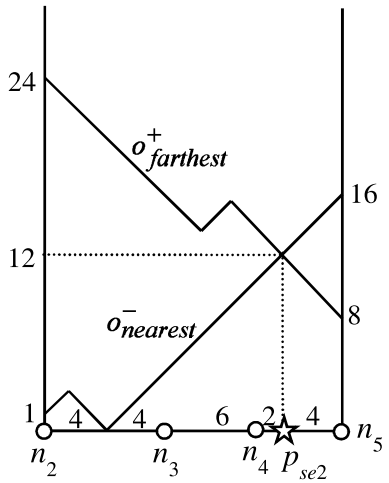


Fig. 7. Determination of location of safe exit point p_{se2}.

and that object c belongs to $A_{n_2} \cap A_{n_5} - O_{(n_2,n_3,n_4,n_5)}$ Hence, for $c \in A_{n_2}$, $d(p,c) = x + 3$, whereas for $c \in A_{n_5}$, $d(p,c) = -x + 28$ Thus, $d(p,c) = MIN(x + 3, -x + 28)$ for $p \in SQ_{(n_2,n_3,n_4,n_5)}$.

The safe exit point p_{se2} is determined as shown in Fig. 7. That is, $d(p_{se2}, a) = d(p_{se2}, c)$ where $d(p_{se2}, a) = |x - 4|$ and $d(p_{se2}, c) = -x + 28$. Consequently, $x = 16$. This means that the distance from n_2 to p_{se2} is 16. Note that object e (i.e., $d(p,e) = -x + 24$) is $o^+_{farthest}$ for $0 \leqslant x \leqslant 10.5$, whereas object c (i.e., $d(p,c) = MIN(x + 3, -x + 28)$) is $o^+_{farthest}$ for $10.5 \leqslant x \leqslant 20$. Similarly, object b (i.e., $d(p,b) = x + 1$) is $o^-_{nearest}$ for $0 \leqslant x \leqslant 1.5$ whereas object a (i.e., $d(p,a) = |x - 4|$) is $o^-_{nearest}$ for $1.5 \leqslant x \leqslant 20$.

5. Performance evaluation

In this section, we describe the performance evaluation of our proposed algorithm, SEA, by comparing it with a conventional method that evaluates each query at every timestamp. Section 5.1 describes our experimental settings while Section 5.2 presents our experimental results.

5.1. Experimental settings

In the experiments, we use a real-life road map that consists of the main roads of North America (175,813 nodes and 179179 edges) obtained from [31]. Note that the road map has been used for

performance evaluation in many papers (e.g. [4,26]). Table 4 summarizes the parameters that are investigated. In each experiment, we vary a single parameter in the range shown while fixing all other parameters to the default values shown in Table 4. A moving client performs a random walk in the network and covers a fixed distance of V_{qry}, where V_{qry} is the query speed. Whenever a moving client reaches a node, one of its adjacent nodes is selected randomly as a destination and the client continues traveling. We simulate the movement of each client using the network-based moving objects generator [3].

In the performance study, we evaluate the performance of our method using the following measures: (1) the total communication cost as the total number of points (including both POIs in the query answers and safe exit points) sent by the server per timestamp, (2) the total communication frequency as the number of messages sent from the client to the server per timestamp, and (3) the total server CPU time per timestamp. Battery power and bandwidth consumption of mobile devices typically increase with the size of data transferred between the server and clients [2,26]. Thus, the size of the transferred data is used as a metric for the communication cost. Note that the CPU time at each client is negligible. Hence, we measure only CPU time at the server.

For comparison, we also report the performance of the periodic querying approach (PERIOD), which issues a new NN query to the server at every timestamp. All queries are continuously monitored for 100 timestamps. Note that the figures show the measured values per timestamp. We implement all solutions in C++ and all the experiments are conducted on a Windows machine with a Pentium 2.8 GHz CPU and 4 GB memory.

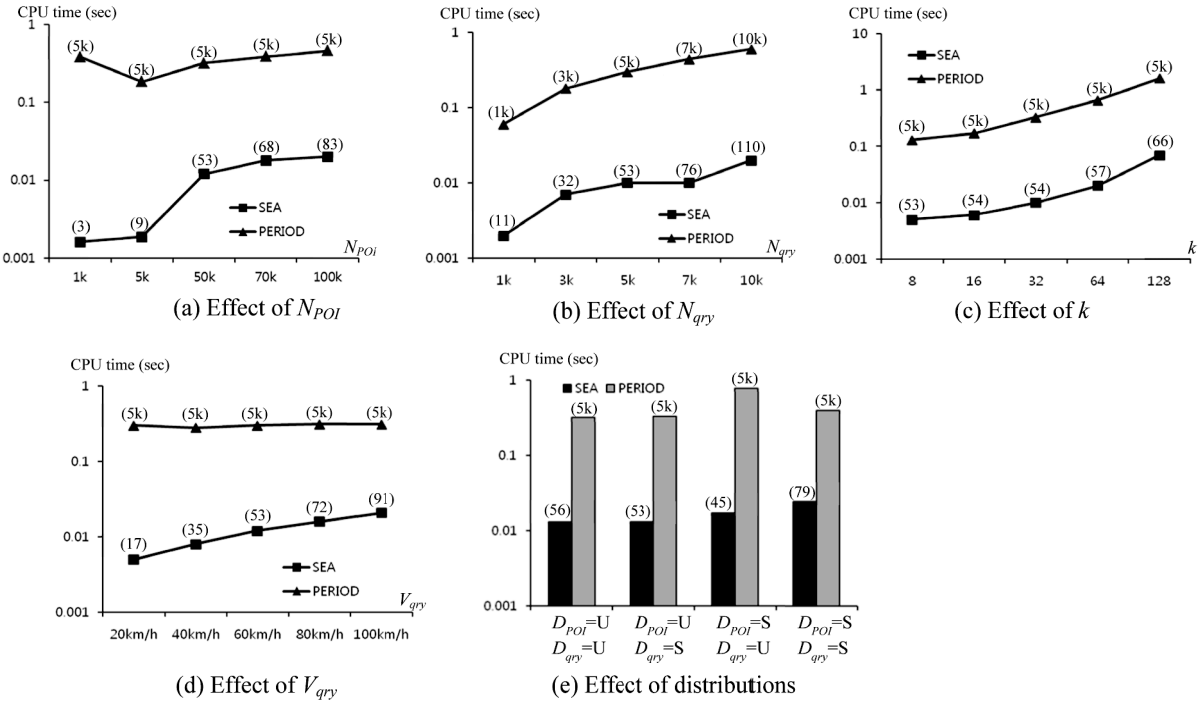
5.2. Experimental results

Figure 8 shows the comparison between SEA and PERIOD in terms of the CPU time at the server. The numbers shown in the parentheses in Figs 8 and 9 indicate the total communication frequencies, which are equal to the numbers of evaluated queries per timestamp. Note that the vertical axis in every chart in Fig. 8 is a log scale of the CPU time.

Figure 8(a) shows the effect of POI cardinality N_{POI} on the query processing time of SEA and PERIOD. Note that $N_{POI} = 1$ k corresponds to a low density of POIs while $N_{POI} = 100$ k corresponds to a high density. The query processing time of SEA is several orders of magnitude less than that of PERIOD. This is expected because about 1% of all moving clients in SEA ask the server to renew the query answers and safe exit points, whereas every client in PERIOD requests the server to refresh the query answer periodically. An interesting observation is that as opposed to PERIOD, SEA does not suffer from performance degradation when the density of POIs is very low (i.e., $N_{POI} = 1$ k). It benefits from the shared execution of multiple queries and a small number of clients that require the updated answers.

Figure 8(b) shows the performance of SEA and PERIOD as a function of query cardinality N_{qry} between 1 k and 10 k. For both SEA and PERIOD, the query processing time increases with the N_{qry} value. SEA achieves a much lower CPU time than PERIOD because most of the clients in SEA remain in their safe region, hence, they do not request query evaluations from the server. Figure 8(c) shows the query processing times of SEA and PERIOD as a function of the number k of NNs requested by clients. SEA is superior to PERIOD in all cases.

Figure 8(d) shows the effect of query speed V_{qry} on the query processing time of SEA and PERIOD. PERIOD incurs a constant processing cost because it issues queries to the server periodically, regardless of the client moving speed. However, PERIOD cannot guarantee the correctness of the result at any time;



(a) Effect of N_{POI} (b) Effect of N_{qry} (c) Effect of k

(d) Effect of V_{qry} (e) Effect of distributions

Fig. 8. SEA vs. PERIOD in terms of query processing costs.

hence, it suffers from stale query answers when the clients move at a high speed (e.g., 100 km/h). In SEA, the client reaches a safe exit point sooner when it moves faster and the communication frequency increases.

In all the previous experiments, the initial positions of the POIs follow a uniform distribution in the network while the clients follow a skewed distribution. In Fig. 8(e), we show the query processing time for different combinations of distributions of POIs and queries while setting the remaining parameters to the default values. The U and S values shown in the figure indicate the uniform and skewed distributions, respectively. SEA provides better performance than PERIOD in all cases.

Figure 9 shows a comparison between SEA and PERIOD in terms of communication costs given the same conditions as shown in Fig. 8. Note that the vertical axis of every chart in Fig. 9 is a log scale of the number of transferred points.

Figure 9(a) shows the effect of the N_{POI} value on the communication costs of SEA and PERIOD. We can see that PERIOD incurs constant communication costs and constant communication frequency, regardless of different densities of POIs. In contrast, the communication cost of SEA increases with the N_{POI} value. This is expected because the safe region becomes smaller as the density of POIs in the road map increases, which requires a high communication frequency. For a very low density of POIs (i.e., $N_{POI} = 1$ k and $N_{POI} = 5$ k), SEA markedly outperforms PERIOD because in SEA, a few clients ask the server to update their query results and safe exit points.

Figure 9(b) shows the communication costs of SEA and PERIOD with respect to the N_{qry} value. Again, SEA achieves much lower communication cost and communication frequency than PERIOD. For SEA and PERIOD, the communication costs increase in proportion to the N_{qry} value.

Figure 9(c) shows the effect of the number k of requested NNs on the communication costs of SEA and PERIOD. Even if PERIOD has a constant communication frequency, the number of result points

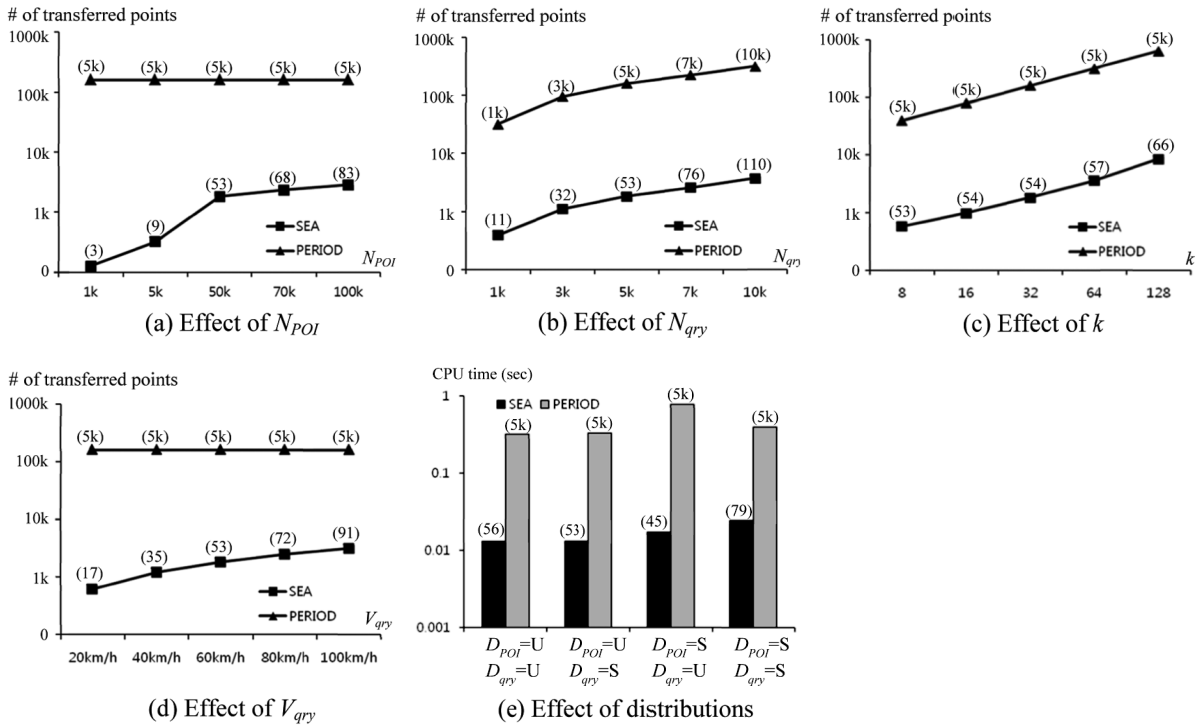


Fig. 9. SEA vs. PERIOD in terms of communication costs.

transferred from the server to clients increases with the k value; hence, the total communication costs also increase. In case of SEA, both the communication frequency and the number of POIs in the query answer increase with the k value. SEA significantly outperforms PERIOD, regardless of the k value. This is because only about 1% of all the clients in SEA request the updated query answers and safe exit points from the server.

Figure 9(d) shows the total communication costs and total communication frequency of SEA and PERIOD with respect to different client moving speeds (ranging from 20 km/h to 100 km/h). PERIOD incurs constant communication costs and frequency. This is because each client in PERIOD asks for periodically updated results (regardless of the client moving speed) for the precision desired by the client. However, PERIOD suffers from outdated results at high speed. For SEA, the client reaches a safe exit point sooner when the speed is increased; hence, both the communication costs and communication frequency increase. SEA outperforms PERIOD by an order of magnitude in terms of the communication costs and frequency. Figure 9(e) shows the effect of the initial distributions of POIs and moving clients on the communication costs. SEA outperforms PERIOD in all cases and achieves near-optimal communication costs.

6. Conclusion

We proposed a new algorithm called SEA which can efficiently compute the safe exit points for moving NN queries in road networks. The performance evaluation using a real-life road network shows that the communication costs of SEA are one or two orders of magnitude lower than those of a traditional

method. In addition, SEA outperforms the traditional method significantly in terms of CPU time at the server. Consequently, SEA could be highly beneficial in real-life scenarios where mobile devices have limited network bandwidth and where the server demands a high throughput.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper. This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2010-0013487).

References

[1] B. Bamba, L. Liu, A. Iyengar and P. Yu, Distributed processing of spatial alarms: A safe region-based approach, ICDCS, 2009, pp. 207–214.
[2] J. Bao, C. Chow, M. Mokbel and W. Ku, Efficient evaluation of k-Range nearest neighbor queries in road networks, Mobile Data Management, 2010, pp. 115–124.
[3] T. Brinkhoff, A framework for generating network-based moving objects, *GeoInformatica* **6**(2) (2002), 153–180.
[4] M. Cheema, L. Brankovic, X. Lin, W. Zhang and W. Wang, Continuous monitoring of distance-based range queries, *IEEE Trans Knowl Data Eng* **23**(8) (2011), 1182–1199.
[5] Z. Chen, H. Shen, X. Zhou and J. Yu, Monitoring path nearest neighbor in road networks, SIGMOD Conference, 2009, pp. 591–602.
[6] H. Cho and C. Chung, An efficient and scalable approach to cnn queries in a road network, VLDB, 2005, pp. 865–876.
[7] T. Delot, S. Ilarri, N. Cenerario and T. Hien, Event sharing in vehicular networks using geographic vectors and maps, *Mobile Information Systems* **7**(1) (2011), 21–44.
[8] N. Ghadiri, A. Dastjerdi, N. Aghaee and M. Nematbakhsh, Optimizing the performance and robustness of type-2 fuzzy group nearest-neighbor queries, *Mobile Information Systems* **7**(2) (2011), 123–145.
[9] H. Hu, J. Xu and D. Lee, PAM: An Efficient and privacy-aware monitoring framework for continuously moving objects, *IEEE Trans Knowl Data Eng* **22**(3) (2010), 404–419.
[10] H. Kriegel, P. Kröger and M. Renz, Continuous proximity monitoring in road networks, GIS, 2008, p. 12.
[11] F. Liu, T. Do and K. Hua, Dynamic range query in spatial network environments, DEXA, 2006, pp. 254–265.
[12] M. Mokbel, X. Xiong and W. Aref, SINA: Scalable incremental processing of continuous queries in spatio-temporal databases, SIGMOD Conference, 2004, pp. 623–634.
[13] D. Moon, B. Park, Y. Chung and J. Park, Recovery of flash memories for reliable mobile storages, *Mobile Information Systems* **6**(2) (2010), 177–191.
[14] F. Morvan and A. Hameurlain, A mobile relational algebra, *Mobile Information Systems* **7**(1) (2011), 1–20.
[15] K. Mouratidis, M. Yiu, D. Papadias and N. Mamoulis, Continuous Nearest Neighbor Monitoring on road Networks, VLDB, 2006, pp. 43–54.
[16] D. Papadias, J. Zhang, N. Mamoulis and Y. Tao, Query processing in spatial network databases, VLDB, 2003, pp. 802–813.
[17] P. Pesti, L. Liu, B. Bamba, A. Iyengar and M. Weber, RoadTrack: Scaling location updates for mobile clients on road networks with query awareness, *PVLDB* **3**(2) (2010), 1493–1504.
[18] M. Safar and D. Ebrahimi, eDAR algorithm for continuous knn queries based on pine, *IJITWE* **1**(4) (2006), 1–21.
[19] D. Stojanovic, A. Papadopoulos, B. Predic, S. Kajan and A. Nanopoulos, Continuous range monitoring of mobile objects in road networks, *Data Knowl Eng* **64**(1) (2008), 77–100.
[20] H. Wang and R. Zimmermann, A novel dual-index design to efficiently support snapshot location-based query processing in mobile environments, *IEEE Trans Mob Comput* **9**(9) (2010), 1280–1292.
[21] H. Wang and R. Zimmermann, Processing of continuous location-based range queries on moving objects in road networks, *IEEE Trans Knowl Data Eng* **23**(7) (2011), 1065–1078.
[22] K. Xuan, G. Zhao, D. Taniar and B. Srinivasan, Continuous range search query processing in mobile navigation, ICPADS, 2008, pp. 361–368.
[23] K. Xuan, G. Zhao, D. Taniar, J. Rahayu, M. Safar and B. Srinivasan, Voronoi-based range and continuous range query processing in mobile databases, *J Comput Syst Sci* **77**(4) (2011), 637–651.

[24] K. Xuan, G. Zhao, D. Taniar, M. Safar and B. Srinivasan, Constrained range search query processing on road networks, *Concurrency and Computation: Practice and Experience* **23**(5) (2011), 491–504.
[25] K. Xuan, G. Zhao, D. Taniar, M. Safar and B. Srinivasan, Voronoi-based multi-level range search in mobile navigation, *Multimedia Tools Appl* **53**(2) (2011), 459–479.
[26] D. Yung, M. Yiu and E. Lo, A safe-exit approach for efficient network-based moving range queries, *Data Knowl Eng Vol* **72** (2012), 126–147.
[27] J. Zhang, M. Zhu, D. Papadias, Y. Tao and D. Lee, Location-based Spatial Queries, SIGMOD Conference, 2003, pp. 443–454.
[28] G. Zhao, K. Xuan, D. Taniar and B. Srinivasan, LookAhead continuous KNN mobile query processing, *Comput Syst Sci Eng* **25**(3) (2010), 205–217.
[29] G. Zhao, K. Xuan, W. Rahayu, D. Taniar, M. Safar, M. Gavrilova and B. Srinivasan, Voronoi-Based continuous k Nearest NeighborSearch in mobile navigation, *IEEE Trans. on Industrial Electronics* **58**(6) (2011), 2247–2257.
[30] G. Zhao, K. Xuan and D. Taniar, Path kNN query processingin mobile systems, IEEE Trans. on Industrial Electronics, to appear (DOI: 10.1109/TIE.2011.2167113).
[31] Real Datasets for Spatial Databases, http://www.cs.fsu.edu/∼lifeifei/SpatialDataset.htm, 2009.

Hyung-Ju Cho received his B.S. and M.S. degrees in Computer Engineering from Seoul National University in February 1997 and February 1999, respectively, and his Ph.D. degree in Computer Science from KAIST in August 2005. He is currently a research assistant professor at the department of information & computer engineering, Ajou University, South Korea. His current research interests include moving object databases and query processing in mobile peer-to-peer networks.

Se Jin Kwon received the B.S. and M.S. degrees in Computer Engineering from Ajou University, Korea, in 2006 and in 2008, respectively. He is currently PH.D. candidate (expected in August 2012) in Computer Engineering from Ajou University, Korea. His current interests include flash memory and large database systems.

Tae-Sun Chung received his B.S. degree in Computer Science from KAIST in February 1995 and his M.S. and Ph.D. degrees in Computer Science from Seoul National University in February 1997 and August 2002, respectively. He is currently an associate professor at the School of Information and Computer Engineering, Ajou University. His current research interests include flash memory storage, XML databases, and database systems.