Research Article

Efficient Processing of Moving Top- k Spatial Keyword Queries in Directed and Dynamic Road Networks

Muhammad Attique ¹, Hyung-Ju Cho ², and Tae-Sun Chung ³

¹Department of Software, Sejong University, Republic of Korea

²Department of Software, Kyungpook National University, Republic of Korea

³Department of Software, Ajou University, Republic of Korea

Correspondence should be addressed to Tae-Sun Chung; tschung@ajou.ac.kr

Received 28 May 2018; Revised 15 August 2018; Accepted 18 September 2018; Published 1 November 2018

Academic Editor: Ke Guan

Copyright © 2018 Muhammad Attique et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

A top- k spatial keyword (TkSk) query ranks objects based on the distance to the query location and textual relevance to the query keywords. Several solutions have been proposed for top- k spatial keyword queries. However, most of the studies focus on Euclidean space or only investigate the snapshot queries where both the query and data object are static. A few algorithms study TkSk queries in undirected road networks where each edge is undirected and the distance between two points is the length of the shortest path connecting them. However, TkSk queries have not been thoroughly investigated in directed and dynamic spatial networks where each edge has a particular orientation and its weight changes according to the traffic conditions. Therefore, in this study, we address this problem by presenting a new method, called COSK, for processing continuous top- k spatial keyword queries for moving queries in directed and dynamic road networks. We first propose an efficient framework to process snapshot TkSk queries. Furthermore, we propose a safe-exit-based approach to monitor the validity of the results for moving TkSk queries. Our experimental results demonstrate that COSK significantly outperforms existing techniques in terms of query processing time and communication cost.

1. Introduction

With the popularization of geo-tagged data (e.g., geo-tagged photos, videos, check-ins, and text messages), many online location-based services such as Google Maps, Yahoo Maps, and Bing Maps have started providing useful information via location-based queries [1–4]. Moreover, textual descriptions of points of interest, e.g., hotels, shopping malls, and tourist attractions, are easily accessible on the Web. These developments demand techniques that efficiently process top- k spatial keyword queries that return a ranked list of the k best facilities based on their proximity to the query location and relevance to the query keywords. Several algorithms have been proposed for processing top- k spatial keyword queries in Euclidean space [5, 6]. Although few algorithms exist that study keyword queries in a road network, they all focus on undirected road networks. However, in real scenarios the urban road networks are directed and dynamic where each edge has a particular orientation and its weight changes

according to traffic conditions such as traffic congestion and reversible lanes. Therefore, in this study, we investigate moving top- k spatial keyword queries in directed and dynamic road networks.

Top- k keyword queries can be used for a wide range of applications in recommendation and decision support systems. For example, tourists may want to retrieve a sorted list of restaurants that serve Italian steak based on the shortest distance from their location and textual relevance to the query keywords. Tourists can issue a top- k spatial keyword query to the location-based services (LBS) to collect information about qualifying restaurants in their vicinity. However, through moving top- k spatial keyword queries if they does not like the results, they can simply keep moving, and the updated results will be provided until a desired restaurant is found. Typically, the query issuer follows the underlying road network to reach at the desired location. Therefore, TkSk algorithms based on Euclidean space does not work in road networks. A road network is generally

FIGURE 1: Illustration of directed road network.

modeled as a weighted directed graph, where each edge has some direction and its weight can vary according to the traffic conditions.

Given a set of data objects $D = \{d_1, d_2, \dots, d_{|D|}\}$, query location, and set of keywords, the TkSK query returns the best k data objects from D according to their combined textual and spatial relevance to the query. We use distance function $dist(q, d)$ to represent the shortest network distance from q to data object d . Figure 1 presents an example of a directed road network, where rectangles represent the data objects with a textual description, and the triangle represents the query location. The number label on each edge indicates the weight of that edge such as the amount of time required to travel along it, e.g., $dist(d_1, n_1) = 1$ and $dist(d_1, n_2) = 2$. Consider a scenario where a tourist is interested in finding an “Italian Restaurant.” If an undirected road network is considered, the top-1 “Italian Restaurant” is d_6 . However, in a directed road network, the shortest path from q to d_6 is $(q \rightarrow n_3 \rightarrow n_7 \rightarrow d_6)$. Therefore, for a directed road network, the top-1 result is d_3 because it is closer to the query location than d_6 . Now, consider that the tourist is looking for “Cafe Bakery.” The data object d_7 could score higher than data object d_1 because d_7 (“Cafe and bakery”) is more textually relevant to query keywords than d_2 (“Cafe”), and $dist(q, d_7)$ is only marginally greater than $dist(q, d_2)$.

Moving Top- k spatial keywords in directed and dynamic road networks are useful for many location-based applications. However, query processing is costly because movement of query object q may invalidate the query results. Therefore, the main challenge in moving TkSk is to maintain the freshness of the query results when the query objects are moving freely. A straightforward approach is to increase the update frequency of the query. However, this approach not only compromises the up-to-date query results but also increases the computation and communication overhead. Because whenever query object changes its location the query object has to report its location to server which increases the communication cost and server has to recompute the results again which increases the computation cost.

To address the aforementioned challenges, we first present an efficient processing technique of snapshot TkSK queries in directed road networks. Then, we present a safe-exit-based approach for processing and monitoring moving TkSk queries where query object q is freely moving in a directed spatial network. The safe exit point of query object q represents a boundary point between the safe region and non-safe region of q . A safe region of query points indicates that the query result remains valid if the query object lies within its respective safe region. Therefore, the query results will only be recomputed when q leaves its respective safe region which significantly reduces the computation and communication costs. To the best of our knowledge, this is the first attempt to study moving top- k spatial keyword queries in directed and dynamic road networks.

Below, we summarize our contributions:

- (i) We study the problem of continuous monitoring of moving top- k spatial keyword queries in a directed and dynamic road networks.
- (ii) We present an algorithm to monitor the moving TkSK queries which efficiently computes the safe exit points for query object q in a directed road network. The algorithm significantly minimizes the computation and communication costs for moving queries.
- (iii) We also propose a method that monitors the validity of query results and safe region when weight of road segments is updated due to traffic conditions.
- (iv) Finally, we conduct extensive experiments on real road network datasets and demonstrate the superiority of the proposed algorithm over the existing approach.

The remainder of this paper is structured as follows. Section 2 reviews the existing work on the processing of TkSk queries on Euclidean and road networks. Section 3 provides terminology definitions and describes the problem. Section 4 elaborates on the proposed query processing technique for

TkSK queries in directed road networks. In Section 5, we present our safe-exit-based technique to process moving TkSK queries. Section 7 presents a performance analysis of the proposed technique. Section 8 concludes this paper.

2. Related Work

In this section, we discuss some of the promising related studies of top- k spatial keyword queries. Our related work is divided into two sections: Section 2.1 reviews snapshot TkSK queries, and Section 2.2 presents the studies proposed to address moving TkSK queries.

2.1. Snapshot Top- k Spatial Keyword Queries. In recent years, spatial keyword queries have drawn the attention of many researchers. Several approaches have been proposed for ranking spatial data objects. Initially, Zhou *et al.* [7] worked on combining inverted indexes [8] and R-trees [9]. They proposed three different hybrid indexing structures. Their study demonstrated that building an inverted index on top of an R-tree provides superior performance. Hariharan *et al.* [10] proposed the indexing structure KR*-tree by capturing the joint distribution of keywords in space. Ian de Felipe *et al.* [11] proposed a data structure that combines an R-tree with text signatures. Each node of the R-tree exploits a signature to indicate the presence of keywords in the subtree of the node. However, both these approaches address only Boolean keyword queries in Euclidean space.

Top- k spatial keyword queries where data objects are ranked according to their combined textual and spatial relevance to keyword queries were first studied by Cong *et al.* [5] and Li *et al.* [6]. Both studies [6] integrate location indexing and text indexing to generate IR-trees. These studies process top- k spatial keyword queries only in Euclidean space and are not suitable for processing top- k spatial preference queries in road networks, where the distance between objects is determined by the shortest path connecting them. Later, Rocha *et al.* [12] proposed the indexing technique S2I, which maps each term in the vocabulary into a separate block or aR tree for efficient processing of top- k spatial keyword queries. Zhang *et al.* [13] proposed an m -closest keyword query that returns the closest object based on distance and which matches m query keywords.

Top- k spatial keyword queries in road networks were introduced by Rocha *et al.* [14]. In particular, they proposed three different indexing techniques (Basic Indexing, Enhanced Indexing, and Overlay Indexing) for processing spatial keyword queries in road networks.

2.2. Moving Top- k Spatial Keyword Queries. Recently, research focus has shifted to the continuous processing of spatial queries where query or data objects are arbitrarily moving in road networks, which is the most realistic scenario. Considerable research effort has been undertaken to process moving range, k nearest neighbor (kNN), and reverse k nearest neighbor queries (RkNN) [15–18]. However, there is a lack of efficient algorithms for moving top- k spatial keyword queries. Initially, Wu *et al.* [19] and Huang *et al.* [20]

TABLE 1: Comparisons with existing solutions.

Algorithm	Type	Space Domain	Orientation
Cong <i>et al.</i> [5]	Snapshot	Euclidean	No orientation
Rocha <i>et al.</i> [14]	Snapshot	Static Road	Undirected
Wu <i>et al.</i> [19]	moving	Euclidean	No orientation
Huang <i>et al.</i> [20]	moving	Euclidean	No orientation
Guo <i>et al.</i> [21]	moving	Static Road	Undirected
Li <i>et al.</i> [22]	moving	Static Road	Undirected
COSK	moving	Dynamic Road	Directed

proposed different methods for monitoring top- k spatial keyword queries in Euclidean space. Guo *et al.* [21] studied moving top- k spatial keyword queries on road networks. They presented two methods for monitoring moving queries in an continuous manner that reduces the traversing of network edges. Later, Li *et al.* [22] proposed TPR-tree-based indexing to monitor moving top- k spatial keyword queries. In contrast to [21, 22], in this study we consider moving top- k spatial keyword queries in directed and dynamic road networks where each road segment has a particular orientation and its weight changes due to according to traffic conditions.

Table 1 compares our problem scenario with related work in terms of query type, space domain, and orientation of road networks.

3. Preliminaries

Section 3.1 defines the terms and notations used in this paper. Section 3.2 formulates the problem using an example that illustrates the general results of top- k spatial keyword queries.

3.1. Definition of Terms and Notations

3.1.1. Road Network. A road network is represented by a weighted directed graph $G = (N, E, W)$ where N , E , and W denote the node set, edge set, and edge distance matrix, respectively. The network distance of an edge changes depending on the traffic conditions. Each edge is also assigned an orientation that is either undirected or directed. The undirected edge is represented by $e = (n_s, n_e)$ where n_s and n_e are the boundary nodes n_β of an edge, whereas the directed edge is represented by $e = \overrightarrow{(n_s, n_e)}$ or $e = \overleftarrow{(n_e, n_s)}$. Naturally, the arrow above the edge indicates the associated direction. We refer to n_s as the starting node and n_e as the ending node of an edge. For example, in Figure 1, n_6 is the starting node of edge $\overrightarrow{(n_6, n_2)}$, whereas it is the ending node for edge $\overleftarrow{(n_6, n_5)}$. The particular edge where a query object is located is called an active edge. It is important to note that, the distance between two points, p_1 and p_2 , is not symmetrical in directed road networks (i.e., $\text{dist}(p_1, p_2) \neq \text{dist}(p_2, p_1)$). For example in Figure 1, the $\text{dist}(d_3, d_4) = 3$, whereas the $\text{dist}(d_4, d_3) = 11$ because shortest path from d_4 to d_3 is $(d_4 \rightarrow n_6 \rightarrow n_2 \rightarrow n_3 \rightarrow d_3)$.

3.1.2. Segment. Segment $s = (p_1, p_2)$ is the part of an edge between two points, p_1 and p_2 , on the edge. An edge consists

of one or more segments. An edge is also considered a segment where the nodes are the end points of the edge. The weight of a segment (p_1, p_2) is denoted by $W(s)$.

3.2. Problem Formulation. Similar to previous studies [5, 14, 23], we assume each data object $d \in D$ has a point location $d.l$ in the road network and a text description $d.t$. Given a query location $q.l$, a set of keywords $q.t$, and k number of data objects to return, the top- k spatial keyword query Q_k is defined as $Q_k = (q.l, q.t, k)$, which takes three arguments and returns the best k data objects from D according to a score that considers spatial proximity and text relevance. The score $\psi(d)$ of a data object d is defined by the following equation:

$$\psi(d) = \frac{\mu(d.t, q.t)}{1 + \alpha \cdot \lambda(d.l, q.l)} \quad (1)$$

where $\lambda(d.l, q.l)$ is the spatial relevance between $d.l$ and $q.l$, $\mu(d.t, q.t)$ is the textual relevance between $d.t$ and $q.t$, and α is a positive real number that determines the importance of one measure over the other. For example, if only textual relevance is considered, then $\alpha = 0$. If more importance is given to spatial relevance, then $\alpha > 1$.

Spatial relevance (λ) is defined as the shortest distance between data objects d and q : $\lambda(d.l, q.l) = \text{dist}(d.l, q.l)$. Thus, $\text{dist}(d_i.l, q.l) < \text{dist}(d_j.l, q.l)$ indicates that data object d_i is more spatially relevant to q than data object d_j . The textual relevance (μ) can be computed using any popular information retrieval model, such as cosine similarity or the language model. In this study, we use the cosine similarity between $d.t$ and $q.t$. The textual relevance is defined as follows:

$$\mu(d.t, q.t) = \frac{\sum_{t \in q.t} w_{t(d.t)} \cdot w_{t(q.t)}}{\sqrt{\sum_{t \in d.t} [w_{t(d.t)}]^2 \cdot \sum_{t \in q.t} [w_{t(q.t)}]^2}} \quad (2)$$

The weight $w_{t(d.t)} = 1 + \ln(f_{t(d.t)})$, where $f_{t(d.t)}$ represents the frequency of term t in $d.t$. The weight $w_{t(q.t)} = \ln(1 + |D|/df_t)$, where $|D|$ is the number of objects in D , and df_t is the document frequency. A higher μ means a higher textual relevance to the query keywords. We used the variation of cosine similarity based on the significance factor $\theta_{t(n)}$ of term t in a document n , where n represents the description of data object $d.t$ or query keywords $q.t$. The significance $\theta_{t(n)} = w_{t(n)} / \sqrt{\sum_{t \in n} (w_{t(n)})^2}$ is the normalized weight of the term in the document by taking into account the length of the document [24, 25]. Hence, the textual relevance $\mu(d.t, q.t)$ can be rewritten as

$$\mu(d.t, q.t) = \sum_{t \in q.t} \theta_{t(d.t)} \cdot \theta_{t(q.t)} \quad (3)$$

4. Query Processing System

In this section, we present the proposed query processing system that indexes the data objects and prunes the irrelevant edges for efficient query processing. In Section 4.1, we discuss the indexing framework, and in Section 4.2, we present an efficient keyword query processing algorithm for snapshot queries.

4.1. Indexing Framework. In this study, our main work focuses on moving queries in a directed and dynamic road networks. We use a method similar to the enhanced technique presented in [12] as our basic framework for processing snapshot queries in directed and dynamic road networks. The indexing framework combines a road network framework [1] for storing spatial information and an inverted file for indexing data objects. For easy traversing of the network, we store the adjacent nodes of each given node by storing node id (n_{id}), edge id (e_{id}), the direction of the edge, and the weight of the edge. The indexing framework consists of two main components: a pruning component and an inverted file component. Figure 2 illustrates the main components of an indexing framework. The pruning component first prunes the edges that contain data objects irrelevant to the query keyword. To achieve this, we introduced the highest significance θ_t^+ of a given term t in the description of objects lying on the edge. The θ_t^+ on an edge is retrieved by a key composed of a pair of edge id and term id (e_{id}, t_{id}). The θ_t^+ represents an upper-bound significance of any object lying on an edge with term t in its description. The inverted list of a term t on an edge is accessed only if the upper-bound score composed by θ_t^+ and the minimum network distance between the starting node of the edge and query q may return a candidate data object. Naturally, the edges with upper-bound scores smaller than the score of the k -th object found so far are pruned.

We implement an inverted file for indexing data objects. The inverted file contains a vocabulary and inverted lists. The vocabulary keeps general information about each term (such as the frequency of the term), which is helpful in computing the textual relevance of the data objects. The inverted list stores the data objects located on the edge (n_s, n_e) that have a term t in their description. An inverted list is identified by a key composed of (e_{id}, t_{id}). Each inverted file is a set of inverted lists. A separate inverted list is used for each term in the object description. An inverted list stores two attributes for each data object: first, the distance between the data object and the starting node $\text{dist}(n_s, d_i)$; second, the significance factor $\theta(t_i, d_i)$ of the term t_i in the description of the data object. Note that the network distance between two points in a directed road network is not symmetrical (i.e., $\text{dist}(n_s, d_i) \neq \text{dist}(d_i, n_s)$). Recall that the starting node is chosen according to the orientation of the edge such that the direction of the edge is from the node toward the data object. In Figure 1, n_3 is the starting node for d_7 . For bidirectional edges, any of the adjacent nodes can act as a starting node.

The proposed indexing scheme has three main advantages. First, the object search relevant to query keywords is very efficient using the (e_{id}, t_{id}) pair. Second, inverted files also store the network distance between the starting node and the data object, which helps in accessing the data object in the directed road network. Finally, the pruning technique allows for faster query processing by exploring fewer edges.

Table 2 presents the notations used in this study.

4.2. Query Processing Algorithm. Our algorithm traverses the road network incrementally in a similar fashion to Dijkstra's

FIGURE 2: Indexing framework.

TABLE 2: Summary of notations used in this paper.

Notation	Definition
$G = (N, E, W)$	Graph model of road network
$dist(p_s, p_e)$	Length of shortest path from p_s to p_e , where p_s and p_e represent start and end points, respectively
$len(p_1, p_2)$	Length of segment connecting two points p_1 and p_2
n_i	Node in road network
$e = (n_s, n_e)$	Edge in edge set E , where n_s and n_e are start and end points of the edge
n_β	Boundary node corresponding to start (n_s) or end (n_e) point of an edge
$W(e)$	Weight of edge (n_s, n_e)
q	Query point in road network
k	A number that represents q can be among k number of closest facilities to a data object d
D	Set of data objects $D = d_1, d_2, \dots, d_{ D }$
$D_{(n_s, n_e)}$	Set of data objects in an edge
p_a	Anchor point that corresponds to start point of expansion
p_{SE}	Safe exit point where safe and non-safe regions of q intersect
α	query parameter
$\psi(d)$	Score of data object d
$\mu(d, t, q, l)$	textual relevance of data object d with query keywords
$\lambda(d, l, q, l)$	Spatial relevance of data object d with query location
D^+	Set of answer objects
D^-	Set of non-answer objects
d_i^+	Lowest answer object
d_h^-	Highest non-answer object

algorithm [26]. Algorithm 1 returns the top- k data objects with the highest scores according to their joint textual and spatial relevance to the query. The algorithm begins by exploring the active edge where query object q is located, and expands the network in an increasing order of distance from q . Each entry in the min-heap has the form $(p_a, edge)$, where p_a indicates the anchor point in the edge. For an active edge, q becomes the anchor point. Otherwise, for directed edges, ending node n_e becomes the anchor point. For bidirectional edges, either of the adjacent boundary nodes, i.e., n_s or n_e , becomes the anchor point. Let D_k be the current set of top- k data objects and s_k be the score of the k -th data object in D_k . The $candsearch((e_{id}, t_{id}), s_k)$ function retrieves the candidate data objects D_c located in an edge with a better score $\psi(d)$ than s_k . Next, the D_k set is updated with the data objects in

D_c , and so does s_k . The algorithm continues its expansion and inserts the adjacent edges of the boundary node until the heap is exhausted or the upper-bound score of the remaining data objects cannot have a better score than s_k . The upper-bound score $\psi(n)$ of node n is computed using $dist(n, q)$ and the maximum textual relevance ($\mu = 1$). Therefore, if $\psi(n) \leq s_k$, it means that even if there is unexplored data object d matching all query keywords, its score can be better than the k -th object in D_k because $dist(d, q, l) \geq dist(n, q, l)$. This is certain owing to the fact that the algorithm strictly expands the node with a minimum distance to the query location.

Algorithm 2 presents the $candsearch((e_{id}, t_{id}), s_k)$ procedure, which finds the candidate data objects. This procedure has two main steps. In the first step, the upper-bound score of the edges is computed using a significance factor (θ_t) of a term

```

(1) Input: Top- $k$  spatial keyword query  $Q_N = (q.l, q.t, k)$ 
(2) Output: Top- $k$  data objects with highest score
(3)  $D_c \leftarrow \emptyset$  /*set of candidate data objects
(4)  $max\text{-}heap\ D_k \leftarrow \emptyset$  /*current Top- $k$  set
(5)  $s_k \leftarrow 0$  /* $k$ -th score in  $D_k$ 
(6)  $min\text{-}heap \leftarrow \emptyset$ 
(7)  $explored \leftarrow \emptyset$ 
(8)  $min\text{-}heap.insert(q.l, edge_{active})$ 
(9)  $D_c \leftarrow candsearch((e_{id}, t_{id}), s_k)$ 
(10) update  $D_k$  and  $s_k$  with  $d \in D_c$ 
(11) while  $min\text{-}heap \neq \emptyset$  and  $(1/(1 + \alpha \cdot \lambda(d.l, q.l)) < s_k)$  do
(12)   for each unexplored adjacent edge of  $(p_a, edge)$  do
(13)      $explored \leftarrow explored \cup (p_a, edge)$ 
(14)      $D_c \leftarrow candsearch((e_{id}, t_{id}), s_k)$ 
(15)     update  $D_k$  and  $s_k$  with  $d \in D_c$ 
(16)   end
(17)    $min\text{-}heap.insert(adjacent\ node, edge)$ 
(18) end
(19) return  $D_k$ 

```

ALGORITHM 1: EvaluateSnapshotQuery(Node n_i , Edge e_i).

```

(1) Input: Edge ID:  $e_{id}$ , Term ID:  $t_{id}$ , score of  $k$ -th object  $s_k$ 
(2) Output: candidate list  $D_c$ 
(3) compute  $\theta_t(e_i)$ 
(4) if  $\theta_t(e_i) > 0$  then
(5)    $max.score(e_i) \leftarrow computemax.score(\theta_t, dist(e_i, q.l))$ 
(6) end
(7) if  $max.score(e_i) > s_k$  then
(8)   for each data object in  $e_i$  do
(9)     compute  $d.score$ 
(10)   end
(11)   if  $d.score > s_k$  then
(12)      $D_c \leftarrow D_c \cup d$ 
(13)   end
(14) end
(15) return  $D_c$ 

```

ALGORITHM 2: CandidateSearch($(e_{id}, t_{id}), s_k$).

$t \in q.t$ and the shortest distance $sdist(e_i, q.l)$ between the edge and the query location. In the next step, the inverted lists of term t are fetched if their upper-bound score is greater than s_k . In the inverted lists, the objects with score $\psi(d)$ greater than s_k are returned.

To understand the proposed algorithm, consider the road network presented in Figure 1. Assume that a query q generated a top-1 keyword query with $q.d$ "Italian Restaurant." For ease of presentation, we assume $\alpha = 1$ and the textual relevance μ is the number of occurrences of query keywords in $d.t$ divided by the number of keywords in the document (description of data object). For example, $\psi(d_4) = \mu(d_4.t, q.t) / (1 + \lambda(d_4.l, q.l)) = 0.5/8 = 0.06$. The algorithm starts the network expansion from an active edge $\overrightarrow{(n_2, n_3)}$ where q is the anchor point. Note that the direction of the edge $\overrightarrow{(n_2, n_3)}$ is from n_2 to n_3 . Therefore, the algorithm explores

only $\overrightarrow{(q, n_3)}$. There is no data object found in $\overrightarrow{(q, n_3)}$. Then, n_3 becomes the anchor point and edges $\overrightarrow{(n_3, n_4)}$, $\overrightarrow{(n_3, n_5)}$, and $\overrightarrow{(n_3, n_7)}$ are inserted in min-heap. Next, the *candsearch* function retrieves the candidate data objects on edges $\overrightarrow{(n_3, n_4)}$, $\overrightarrow{(n_2, n_3)}$, and $\overrightarrow{(n_3, n_7)}$, whose score is better than s_k . On edge $\overrightarrow{(n_3, n_5)}$, data object d_3 is retrieved with $\psi(d_3) = 0.2$. Data object d_3 is inserted in the D_k set, and the value of s_k is set to 0.2. For edges $\overrightarrow{(n_3, n_4)}$ and $\overrightarrow{(n_3, n_7)}$, there is no candidate object found because $d_{2.t}$ ("Cafe") and $d_{7.t}$ ("Cafe and Bakery") do not match with $q.t$. The algorithm continues expanding the edges whose upper-bound score is greater than s_k . The edge $\overrightarrow{(n_7, n_2)}$ is explored next. The upper-bound score of $\overrightarrow{(n_7, n_2)}$ is $1/7$, which is less than s_k . Similarly, for edge $\overrightarrow{(n_6, n_5)}$, the upper-bound score is $0.5/8 < s_k$. Therefore, the algorithm terminates and reports d_3 as the top-1 result.

FIGURE 3: Illustration of directed road network.

FIGURE 4: Illustration of directed road network.

5. Moving Top- k Spatial Keyword Queries

In this section, we present our method to monitor the moving top- k spatial keyword queries where query objects are moving in a directed road network. Figure 3 provides an example of TkSK in road networks, where query point q issues a TkSK query at point p_1 . Note that the numbers on the arrows in the figure indicate the order of the steps. To obtain top- k results at p_1 , the server executes Algorithm 1 as mentioned in Section 4.2. Now, consider that the query object is moved to p_2 as shown in Figure 4 to retrieve the top- k results at point p_2 . The simple method is to repeat the procedure executed at p_1 . However, the use of recomputation whenever query q changes its location significantly increases the computation cost. Furthermore, it also increases the communication overhead because the query object must report its location whenever it moves, and the server must send the results set. To address these issues, we introduce the safe exit approach.

In the proposed framework, the server computes safe exit points for a query object. The server maintains a set of moving queries, and the query result remains valid until the query objects remain inside their respective safe exit points. Whenever a query object leaves its safe exit points, the server recomputes the TkSK and safe exit points for the query object.

Next, we present our method to compute the safe exit points for a query object. The safe exit point represents a point in the segment where a safe region and nonsafe region meet. We compute the safe exit point using the divide-and-conquer technique. Before presenting the detailed methodology, we define the terminologies used in this section.

Definition 1 (safe region). A portion of a road segment that can guarantee that, as long as the query point lies in it, its top- k results remain valid.

Definition 2 (answer objects D^+). A data object d is called an answer object of query q if the score of data object d ($\psi(d) > \psi(d^a)$), where d^a represents any other data object in the directed road network. Similarly, we can generalize this definition for TkSK: a data object d is called an answer object

of query q if the score of a data object d ($\psi(d) > \psi(d_{k+1})$), where d_{k+1} represents the $(k+1)$ th data object in the directed road network. In other words, we can state that all answer objects are top- k results of query q .

Definition 3 (nonanswer objects D^-). A data object d is called a nonanswer object of query q if the score of data object d ($\psi(d) < \psi(d^a)$), where d^a represents any other data object in the directed road network. Similarly, we can generalize this definition for TkSK: a data object d is called a nonanswer object of query q if the score of data object d ($\psi(d) < \psi(d_k)$), where d_k represents the k th data object in the directed road network. That is, we can say that all answer objects are top- k results of query q . Therefore, we can state that none of the nonanswer objects are in the top- k results of query q .

Definition 4 (lowest answer object D_l^+). An answer object $d^+ \in D^+$ is called a lowest answer object to a point $p \in G$ such that $\psi(d_l^+)_p = \min(\psi(d_1^+)_p, \psi(d_2^+)_p, \dots, \psi(d_{|d^+|}^+)_p)$, where $\psi(d_l^+)_p$ represents the score of the lowest answer object at point p . In other words, $\psi(d_l^+)_p < \psi(d_a^+)_p$ at point p , where d_a^+ is any other answer object in the D^+ set.

Definition 5 (highest nonanswer object D_h^-). A nonanswer object $d^- \in D^-$ is called a highest nonanswer object to a point $p \in G$ such that $\psi(d_h^-)_p = \max(\psi(d_1^-)_p, \psi(d_2^-)_p, \dots, \psi(d_{|d^-|}^-)_p)$, where $\psi(d_h^-)_p$ represents the score of the highest nonanswer object at point p . In other words, the $\psi(d_l^-)_p < \psi(d_a^-)_p$ at point p , where d_a^- is any other nonanswer object in the D^- set.

As discussed earlier, the main challenge in the continuous processing of moving TkSK is to maintain the validity of the result set because the movement of query objects can nullify the result set. To monitor the validity of the result set, we propose a safe-region-based approach.

5.1. Computation of Safe Exit Points. In this section, we present our technique to compute the safe exit points. The main goal is to find a point in the road network where the

query result set will change. The result set will change when the score of highest nonanswer D_h^- surpasses the score of D_l^+ . Generally, the textual relevance score does not change. Therefore, the score of data objects only changes because of the spatial relevance score, which can only change by the movement of query objects. The computation of the safe exit point is based on two key observations:

Observation 1. If $D_{n_\beta}^+ = D_{p_a}^+$, there is no safe exit point in the segment

Explanation. $D_{p_a}^+$ represents the set of answer objects at anchor point p_a , whereas $D_{n_\beta}^+$ represents the set of answer objects at boundary node n_β . As discussed earlier, the safe exit point is the particular point where the query results changed. If the query results at the starting node are the same as the ending node of any segment/edge, there does not exist any point where the query result is changing. Hence, we do not search the safe exit point in that segment.

Observation 2. If $D_{p_a}^+ \neq D_{n_\beta}^+$, there is a safe exit point in the segment

Explanation. In contrast to Observation 1, if the query results are different at the starting and ending points, then there exists a point where the query results are changing. Hence, there is a safe exit point in the segment.

To find the safe region, we observe the following cases:

Case 1 (when $\alpha = 1$ and the textual relevance of the highest nonanswer object and lowest answer object is the same). In this case, both the textual and spatial relevance have the same importance (i.e., $\alpha = 1$). In addition, the top- k result depends only on the spatial relevance because the textual relevance of both objects is the same. The data object that is closer to query point q becomes the answer object. For an undirected edge, the safe exit point p_{se} is the center point, i.e., $\max(\text{dist}(p_{se}, d_1^+), \text{dist}(p_{se}, d_2^+), \dots, \text{dist}(p_{se}, d_{|d^+|}^+)) = \min(\text{dist}(p_{se}, d_1^-), \text{dist}(p_{se}, d_2^-), \dots, \text{dist}(p_{se}, d_{|d^-|}^-))$, between the lowest answer object and the highest nonanswer object. However, in case of a directed edge where $\text{dist}(p_a, n_\beta) \neq \text{dist}(n_\beta, p_a)$, the safe exit point is either d_l^+ or p_a . If $d_l^+ \in (p_a, n_\beta)$, then the safe exit point is d_l^+ ; otherwise, the safe exit point is p_a .

Case 2 (when $\alpha \neq 1$ and the textual relevance of the highest nonanswer object and lowest answer object is different). In this case, the top- k result depends on all functions that are the α , spatial, and textual relevance. Clearly, for the undirected edges, the midpoint between the lowest answer object and the highest nonanswer object does not provide a valid safe exit point. Therefore, we introduce the divide-and-conquer technique. This will keep dividing the search space until we get the point where the score of the nonanswer is greater than that of the answer object. Typically, the safe exit point should be closer to the data object whose score is lower. Based on this observation, first we compute the midpoint in a similar fashion to Case 1, and then we continue dividing the search

space until we find the point. For undirected edges, the safe exit point can be computed in a similar fashion to Case 1.

Case 2 also works for other cases when the safe exit point is not the mid point between the lowest answer object and the highest nonanswer object. In these cases the safe exit point depends on two or more functions. Therefore, the safe exit point can be easily computed using the aforementioned divide-and-conquer technique. Following are the scenarios where the safe exit point can be computed using Case 2.

- (a) When $\alpha = 1$ and textual relevance of the nearest non-answer object and farthest answer object is different.
- (b) When $\alpha \neq 1$ and textual relevance of the nearest nonanswer object and farthest answer object is same.

Case 3 (when $\alpha = 0$). This means the spatial relevance has no effect on the score of data objects. Hence, no monitoring is required for this scenario.

Algorithm 3 retrieves the safe exit points using the observations we discussed earlier. The core function in this algorithm is $\text{ComputeSafeExit}(p_a, n_\beta)$, which finds the safe exit point in a segment between p_a and n_β . The detailed $\text{ComputeSafeExit}(p_a, n_\beta)$ is described in Algorithm 4. First, Algorithm 4 determines d_l^+ and d_h^- at point $p \in [p_a, n_\beta]$. Recall that d_l^+ is the lowest answer object to p , where d_h^- is the highest nonanswer object to p . Algorithm 4 computes the safe exit point based on the cases we discussed earlier. There are a further two scenarios for Cases 1 and 2. For Case 1, if $\text{dist}(p_a, n_\beta) = \text{dist}(n_\beta, p_a)$, then the safe exit point is the midpoint between d_l^+ and d_h^- . If $\text{dist}(p_a, n_\beta) \neq \text{dist}(n_\beta, p_a)$, then the edge is directed, and therefore the safe exit point is either p_a or d_l^+ . If d_l^+ lies on the edge $[p_a, n_\beta]$, then d_l^+ is the safe exit point. Otherwise, p_a is the safe exit point.

Similarly, for Case 2, if $\text{dist}(p_a, n_\beta) = \text{dist}(n_\beta, p_a)$, then the safe exit point is computed by dividing the search space by half until we find the closest point such that $\psi(d_h^-) > \psi(d_l^+)$. The safe exit point is computed in the same way as in Case 2 if $\text{dist}(p_a, n_\beta) \neq \text{dist}(n_\beta, p_a)$.

5.2. Computation of Safe Exit Points for Example. Consider the same example in Figure 1, where the query point q issues a top-1 keyword query with $q.t$ "Italian restaurant." For this example, let us consider $\alpha = 1$. The monitoring algorithm starts exploring from the active edge containing the query object q . Therefore, (q, n_3) is explored first. As shown in Table 3, for (q, n_3) , $D_q^+ = d_3$ and $D_{n_3}^+ = d_3$. According to Observation 1, no safe exit point exists in this segment. Therefore, edges adjacent to n_3 are explored, and n_3 becomes the new p_a . The edge (n_3, n_4) is explored next. Similarly, the answer object at n_3 and n_4 is the same: $D_{n_3}^+ = D_{n_4}^+ = d_3$. Therefore, a safe exit point does not exist in (n_3, n_4) . The edge (n_3, n_7) is explored next. As shown in Table 3, $D_{n_3}^+ = d_3$ and $D_{n_7}^+ = d_6$. By Observation 2, there is a safe exit point in (n_3, n_7) . As shown in Figure 1, $d_{3,t} = d_{6,t} = \text{"ItalianRestaurant"}$ and $\text{dist}(n_3, n_7) = \text{dist}(n_7, n_3)$.


```

(1) Input: Same as Algorithm 1
(2) Output:  $P_{SE}$ : a set of safe exit points
(3)  $P_{SE} \leftarrow \emptyset$  /* set of safe exit points
(4)  $D_{p_a}^+ \leftarrow \text{EvaluateSnapshotQuery}(p_a, (p_a, n_\beta))$ 
(5) /* Results calculated using Algorithm 1
(6)  $D_{n_\beta}^+ \leftarrow \text{EvaluateSnapshotQuery}((n_\beta, (p_a, n_\beta)))$ 
(7) /* Results calculated using Algorithm 1
(8) if  $D_{p_a}^+ = D_{n_\beta}^+$  then
(9)   no safe exit point /* refer to Observation 1
(10) end
(11) if  $D_{p_a}^+ \neq D_{n_\beta}^+$  then
(12)    $P_{SE} \leftarrow P_{SE} \cup \text{ComputeSafeExit}(p_a, n_\beta)$  /* safe exit point
      exist - refer to Observation 2
(13) end
(14) return  $P_{SE}$ 

```

ALGORITHM 3: COSK: monitoring algorithm.

```

(1) Input: same as Algorithm 1
(2) Output: se: safe exit point in  $(p_a, n_\beta)$ 
(3)  $D_l^+ \leftarrow \langle p, D_l^+ \rangle$  | for each point  $p \in [p_a, n_\beta]$ ,  $d_l^+$  such that
       $\psi(d_l^+)_p = \min(\psi(d_1^+)_p, \psi(d_2^+)_p, \dots, \psi(d_{|d^+|}^+)_p)$ 
(4)  $D_h^- \leftarrow \langle p, D_h^- \rangle$  | for each point  $p \in [p_a, n_\beta]$ ,  $d_h^-$  such that
       $\psi(d_h^-)_p = \max(\psi(d_1^-)_p, \psi(d_2^-)_p, \dots, \psi(d_{|d^-|}^-)_p)$ 
(5) if Case 1 then
(6)   if  $\text{dist}(p_a, n_\beta) = \text{dist}(n_\beta, p_a)$  then
(7)      $p_{se} = \max(\text{dist}(se, d_1^+), \text{dist}(se, d_2^+), \dots, \text{dist}(se, d_{|d^+|}^+)) =$ 
       $\min(\text{dist}(se, d_1^-), \text{dist}(se, d_2^-), \dots, \text{dist}(se, d_{|d^-|}^-))$ 
(8)   end
(9)   if  $\text{dist}(p_a, n_\beta) \neq \text{dist}(n_\beta, p_a)$  then
(10)     $p_{se} = p_a$  or  $p_{se} = d_l^+$  where  $d_l^+ \in (p_a, n_\beta)$ 
(11)   end
(12) end
(13) if Case 2 then
(14)   if  $\text{dist}(p_a, n_\beta) = \text{dist}(n_\beta, p_a)$  then
(15)     $p_{se} = \text{closest point to } p_a \text{ such that } \psi(d_h^-) > \psi(d_l^+)$ 
(16)   end
(17)   if  $\text{dist}(p_a, n_\beta) \neq \text{dist}(n_\beta, p_a)$  then
(18)    Same as Line (10)
(19)   end
(20) end
(21) return  $p_{se}$ 

```

ALGORITHM 4: ComputeSafeExit(p_a, n_β).

Therefore, according to Case 1, the safe exit point s_1 is the midpoint between d_3 and d_6 . That is, $\text{dist}(p_{se1}, d_3) = \text{dist}(p_{se1}, d_6)$, where $\text{dist}(p_{se1}, d_3) = x+3$ and $\text{dist}(p_{se1}, d_6) = -x+5$ for $0 < x < 3$. Consequently, $x = 1$, which means that the distance from n_3 to p_{se1} is 1.

Next, we determine a safe exit point in (n_3, n_5) . As shown in Table 3, the answer object at n_5 is also the same as n_3 . Hence, no safe exit point exists in this edge. Next, (n_6, n_5) is explored with $p_a = n_5$. According to Table 3, $D_{n_7}^+ = d_4$ and

$D_{n_5}^+ = d_3$. Therefore, a safe exit point exists in this edge. This edge is directed, and for each point $p \in (n_6, n_5)$, the shortest distance from p to d_3 is from $p \rightarrow n_6 \rightarrow n_2 \rightarrow n_3 \rightarrow d_3$. Therefore, n_5 is the safe exit point.

The bold lines in Figure 5 indicate the safe region of q . The top-1 result remains d_3 until the query q lies in the safe region.

Next, we analyze the time complexity for determining a set of safe exit points using a set of qualifying objects $d \in D_{p_a}^+ \cup D_{n_\beta}^+ \cup D(\overline{p_a, n_\beta})$. Note that $D_{p_a}^+$ ($D_{n_\beta}^+$) indicates

TABLE 3: Computation of safe exit points for example scenario.

Edge/Segment	p_a	$D_{p_a}^+$	$D_{n_\beta}^+$	p_{se}
$\overrightarrow{(q, n_3)}$	q	$D_q^+ = d_3$	$D_{n_3}^+ = d_3$	none
$\overrightarrow{(n_3, n_4)}$	q	$D_{n_3}^+ = d_3$	$D_{n_4}^+ = d_3$	none
$\overrightarrow{(n_3, n_7)}$	n_3	$D_{n_3}^+ = d_3$	$D_{n_7}^+ = d_6$	p_{se1}
$\overrightarrow{(n_3, n_5)}$	n_3	$D_{n_3}^+ = d_3$	$D_{n_5}^+ = d_3$	none
$\overrightarrow{(n_6, n_5)}$	n_5	$D_{n_5}^+ = d_3$	$D_{n_6}^+ = d_4$	p_{se2}

FIGURE 5: Illustration of safe region of q .

the set of k data objects that satisfies the query condition at p_a (n_β). According to Dijkstras algorithm [26], the time complexity $O(D_q^+)$ for computing a set of answer objects at a query point q is $O(D_q^+) = O(|E| + |N| \log |N|)$. This means that $O(D_{p_a}^+) = O(D_{n_\beta}^+) = O(|E| + |N| \log |N|)$ holds for endpoints p_a and n_β . Thus, time complexity $O(\Omega_{kth})$ when determining the skyline Ω_{kth} with the k -th highest score is $O(\Omega_{kth}) = C_{kth} \cdot O(|D_{p_a}^+ \cup D_{n_\beta}^+ \cup D(\overline{p_a, n_\beta})|)$ where C_{kth} is the number of qualifying objects that participate in the constitution of the skyline with the k -th highest score. Therefore, the time complexity of determining a safe exit point coincides with the time complexity of determining the two skylines, i.e., the skyline D_l^+ with the k -th highest (or lowest) score for answer objects and the skyline D_h^- with the highest score for nonanswer objects. This is because the safe exit point is found at the cross point between these skylines.

Figure 6, represents the skyline graph for $k = 1$ in an edge $\overline{(n_7, n_3)}$. Let us draw the score function for d_3 and d_6 for the road segment $\overline{(n_7, n_3)}$ where a safe exit point exists. This is because $D_{n_3}^+ = d_3$ and $D_{n_7}^+ = d_6$ for $k = 1$. For each point $p \in \overline{(n_7, n_3)}$, the distance between d_3 and point p can be represented as $dist(d_3, p) = dist(d_3, n_3) + len(n_3, p) = 6 - len(n_7, p)$. Similarly, for each point $p \in \overline{(n_7, n_3)}$, the distance between d_6 and point p can be represented as $dist(d_6, p) = dist(d_6, n_7) + len(n_7, p) = 2 + len(n_7, p)$. Let, $len(n_7, p)$ be

FIGURE 6: Skyline graph for $k = 1$ on the road segment $\overline{(n_7, n_3)}$.

a variable x ($0 \leq x \leq 3$). We can write $\lambda(d_3, p) = dist(d_3, p) = 6 - x$ and $\lambda(d_6, p) = dist(d_6, p) = 2 + x$. Then, we can represent score function $\psi(d_3)$ and $\psi(d_6)$ as follows:

$$\psi(d_3) = \mu(d_3, t, q, t) / (1 + \alpha \cdot \lambda(d_3, p)) = 1 / (7 - x) \text{ for } (0 \leq x \leq 3)$$

$$\psi(d_6) = \mu(d_6, t, q, t) / (1 + \alpha \cdot \lambda(d_6, p)) = 1 / (3 + x) \text{ for } (0 \leq x \leq 3)$$

Finally, we present the lemma to prove that safe exit points computed by COSK are correct.

Lemma 8. *The COSK algorithm correctly computes a set of safe exit points.*

Proof. We will prove the correctness of the COSK algorithm by contradiction. We assume that if $D_{p_a}^+ \neq D_{n_\beta}^+$, there is no safe exit point in a road segment $(\overline{p_a n_\beta})$. This means that, for each point p in the road segment $(\overline{p_a n_\beta})$, the query result at p equals $D_{p_a}^+$, i.e., $D_p^+ = D_{p_a}^+ \forall p \in (\overline{p_a n_\beta})$. However, it leads to a contradiction that $D_{n_\beta}^+ = D_{p_a}^+$ when $p = n_\beta$. Therefore, if $D_{p_a}^+ \neq D_{n_\beta}^+$, a safe exit point exists in $(\overline{p_a n_\beta})$. In addition, a safe exit point is determined using the skyline D_l^+ for answer objects and the skyline D_h^- with the highest score for nonanswer objects when $D_{p_a}^+ \neq D_{n_\beta}^+$. The first skyline is a composite polyline drawn from answer objects in $D_{p_a}^+$. The second skyline is a composite polyline drawn from nonanswer objects in $D_{n_\beta}^+ \cup D(\overline{p_a n_\beta}) - D_{p_a}^+$. $\square$

6. Monitoring Query Results and Safe Regions in Dynamic Directed Road Networks

In this section, we discuss the monitoring of spatial keyword queries in dynamic road networks where the network distance changes depending on the traffic conditions. The updates on weight of some edges may invalidate the query results or safe region of q , even though the query object q remains within their respective safe region. Figure 7 illustrates an example of changing the weights edges $(\overline{n_1, n_2})$ and $(\overline{n_1, n_6})$. For convenience we consider $\alpha = 1$ and $q.t = \text{"Italian restaurant."}$ In Figure 7(a), the top-1 result is d_1 and bold lines show the safe region of query q . Now consider at time t_j the weights of two edges $(\overline{n_1, n_2})$ and $(\overline{n_1, n_6})$ changed due to heavy traffic condition as shown in Figure 7(b). The update in weight of edges may invalidate the query result or safe region of q . Therefore, it is necessary to monitor the validity of results and safe region when the changes occur.

Next, we introduce a monitoring region to monitor the validity of the safe region effectively when the weight of an edge is changed. Monitoring region MR contains all the points between query point q and lowest answer object and highest nonanswer object. Formally, it is defined as $MR = \text{dist}(q, D_l^+) \cup \text{dist}(q, D_h^-)$, where $\text{dist}(q, D_l^+)$ is the distance between q and lowest answer object and $\text{dist}(q, D_h^-)$ is highest nonanswer object. In given example, the $D_l^+ = d_1$ and $D_h^- = \{d_2, d_3\}$. Therefore, the dotted lines in Figure 8(a) shows the monitoring region of query object q .

Now at time t_j , the update to edges $(\overline{n_1, n_6})$ and $(\overline{n_1, d_1})$ which is not part of monitoring region can safely be ignored. However, the updated on segment $(\overline{n_2, d_1})$ which is associated with monitoring region may nullify the results. As shown in

Figure 8(b), after update the top-1 result becomes d_2 and bold lines represents the new safe region of q .

Algorithm 5 monitors the validity of result set and safe region of query object q when the weight of any edge changes. Let us consider weight of edge $(\overline{n_i, n_j})$ changes at time t_j . First, algorithm checks whether edge $(\overline{n_i, n_j})$ is associated with monitoring region or not. If it is not part of monitoring region then algorithm simply ignores the update in edge $(\overline{n_i, n_j})$ and query results and safe region remains valid. In contrast, if edge is associated with monitoring region (i.e., $MR \cap (\overline{n_i, n_j}) \neq \emptyset$) then algorithm evaluates the query results. Consequently, the top- k results and safe region of query q needs to be updated. Finally, the algorithm updates the monitoring region of q .

7. Performance Evaluation

In this section, we evaluate the performance of COSK through simulation experiments. We describe our experimental settings in Section 7.1, and we present our experimental results for static and dynamic road networks in Sections 7.2 and 7.3, respectively.

7.1. Experimental Settings. All of our experiments were performed using real road networks, namely, Oldenburg, San Francisco, and San Joaquin. All three road networks were obtained from [27]. The original road network of San Francisco had 21,047 nodes and 21,692 edges. We reformat the network, pruned approximately 30% of the nodes, and adjusted the edges and their weights accordingly. This resulted in a network with 14,732 nodes and 14,316 edges. Both the direction of edges and data objects on the edges were generated randomly. The description of each data object was extracted from Twitter messages [28], and we assigned one tweet per data object. Table 4 presents the characteristics of the data sets used in the experimental evaluation. We simulated moving query objects by using a spatiotemporal data generator [29]. The input to generator was the road network of the data set used, and the output was the set of query objects moving on the road network. Each experiment had 100 moving queries which were continuously monitored for 100 timestamps (1 timestamp = 1 second), and the average result was reported in the experiments.

As a benchmark for COSK in static road network, we implemented a CMTkSK+ algorithm [22] which also continuously monitored the moving top- k spatial keyword queries in the road networks. However, this algorithm was originally designed for undirected road networks. To make a fair comparison, we modified CMTkSK+ to process top- k spatial keyword queries in directed road networks and called it CMTkSK+. Specifically, we modified the distance computation method between two points such that in directed road networks, $\text{dist}(p_1, p_2) \neq \text{dist}(p_2, p_1)$. Since CMTkSK+ does not handle top- k spatial queries in dynamic road roads, we compared the performance of COSK with basic algorithm which recomputes the results whenever query object changes its location. All algorithms were implemented in Java and were executed on a desktop PC 2.80-GHz Intel Core i5 with

FIGURE 7: Updating the weight of edges in a dynamic road network, where $t_i < t_j$. FIGURE 8: Monitoring region and updated safe region at time t_j .

```

(1) Input: Monitoring region:  $MR$ , updated edge:  $\overleftarrow{(n_i, n_j)}$ 
(2) Output: none
(3) if  $MR \cap \overleftarrow{(n_i, n_j)} = \emptyset$  then
(4)     /* edge  $\overleftarrow{(n_i, n_j)}$  is not part of monitoring region
(5)     ignore the change in the weight of edge  $\overleftarrow{(n_i, n_j)}$ 
(6) end
(7)  $P_{SE} \leftarrow \emptyset$  /* set of safe exit points
(8) else
(9)      $D_{k_{upd}} \leftarrow EvaluateSnapshotQuery(n_i, e_i)$  /* update set of
        top-k results
(10)     $P_{SE_{upd}} \leftarrow ComputeSafeExit(P_a, n_\beta)$  /* update safe exit
        points
(11)     $MR_{upd} \leftarrow ComputeMonitoringRegion(D_l^+, D_h^-)$ 
        /* update monitoring region
(12) end

```

ALGORITHM 5: MonitoringSafeRegion($MR, \overleftarrow{(n_i, n_j)}$).

TABLE 4: Summary of datasets.

Attribute	Oldenburg	San Francisco	San Joaquin
Total no. of nodes	6104	14732	18262
Total no. of edges	7034	14316	23876
Percentage of directed edges	30%	30%	30%
Total no. of objects	5627	11453	19098
Average no. of objects per edge	0.8	0.8	0.8
Total no. of words	49517	103649	166153

TABLE 5: Experimental parameter settings.

<i>Parameter</i>	<i>Range</i>
Number of results (k)	5, 10, 15 , 20, 25
Number of keywords (n)	1, 2, 3 , 4, 5
Query parameter (α)	0.01, 0.1, 1 , 10, 100
Dataset	Oldenburg, San Francisco , San Joaquin
Number of data objects (N_D)	10, 20, 30 , 40, 50 (x1000)
Speed of query objects (V_{qry})	25, 50, 75 , 100, 125 (km/h)
Mobility (M_{qry})	20, 40 , 60, 80, 100
Ratio of directed edges (E_{dir})	10, 20 , 30, 40, 50
Ratio of updated edges (E_{upd})	15, 30, 60, 80, 100

8 GB of memory. In the experiments, we compared (1) query processing times; (2) edges processed, i.e., the number of edges processed for retrieving query results; and (3) index sizes. Table 5 summarizes the parameters used in the experiments. In each experiment, we varied a single parameter within the range that is shown in Table 5 while maintaining the other parameters at the bolded default values.

We evaluated the performance of the algorithms by using the following measures: (1) total amount of server CPU time, which indicates the query processing time, and (2) total communication cost as the total number of points (i.e., the location updates sent by query objects, and the query results and safe exit points returned by the server) transferred between clients and the server. The battery power and wireless bandwidth consumption typically increase with the amount of data transferred between objects (clients) and servers. Thus, we used the amount of transferred data as a metric to evaluate the communication cost.

7.2. Experimental Results of Top- k Spatial Keyword Queries in Static Road Networks

7.2.1. Effect of k . Figure 9 indicates the effect of the number of results on the query processing time and communication cost for both algorithms. Figure 9(a) indicates that the query processing time increases for both algorithms as the value of k increases. This is expected because with an increase in k , more data objects are required to be explored and verified. Nevertheless, COSK significantly outperforms CMTkSK+ for two main reasons. First, a relevant object search is very efficient when using the highest significant factor; and second, COSK does not need to verify the set of answer objects as long as the query object lies in a safe region. On the other hand, the CMTkSK+ query processing time increases significantly because it has to monitor and verify the set of candidate objects periodically. In Figure 9(b), the communication costs for both algorithms increase as the number of objects increases. However, the proposed algorithm demonstrates superior performance compared to CMTkSK+ because client-server communication is not required when the query object lies within the safe exit points, whereas in CMTkSK+, the query object is required to report its location to the server whenever it moves.

7.2.2. Effect of N_D . This experiment was conducted on dataset San Joaquin. This dataset included 19,098 data objects; therefore, we randomly generated approximately 30,000 additional data objects on different edges. In Figure 10, we evaluate the performance of COSK and CMTkSK+ by varying the cardinality of the data objects. Note that $N_D = 10K$ corresponds to a low density of data points, while $N_D = 50K$ corresponds to a high density. In Figure 10(a), it is interesting to notice that the query processing times of both algorithms decrease as the cardinality of the data objects increases. For CMTkSK+, this is because with high density, the monitoring range of a query decreases. However, for COSK, it is mainly because when the data density is high, fewer edges are required to be expanded, which decreases the query processing time. In Figure 10(b), we study the influence of the cardinality of the data objects on the communication costs. The experimental results indicate that the communication costs of CMTkSK+ incur almost constant communication costs regardless of data object cardinality. However, the communication costs of COSK increase in proportion to the N_D value. This is expected because the safe region becomes smaller as the density of the data objects increases, which increases the communication costs.

7.2.3. Effect of Query Keywords (n). Figure 11 shows the query processing time and communication for COSK and CMTkSK+ as a function of the number of query keywords. Figures 11(a) and 11(b) show the trend that the performance of both algorithms degrades when the number of keywords increases. This is mainly because by increasing the number of query keywords, the number of relevant objects may also increase, resulting in a higher query processing time and communication cost. However, the safe-region-based algorithm COSK scales better than CMTkSK+ because of its less expensive monitoring technique.

7.2.4. Effect of α . Figure 12 demonstrates the impact of query parameter α on the query processing time and on the communication cost. A small value of α indicates a greater importance of textual relevance, whereas a high value of α gives more preference to the spatial relevance. It is interesting to note that the query processing time is lower for higher

FIGURE 9: Effect of k on query processing time and number of edges processed. FIGURE 10: Effect of N_D on query processing time and communication cost.

values of α , which indicates more importance to the spatial relevance. This is mainly because when the spatial relevance is higher, fewer edges and objects are required to be explored and processed to determine the top- k data objects. Observe that in Figure 12(b), the number of messages sent by COSK decreases sharply with an increase in α .

7.2.5. Effect of Speed. Figure 13(a) demonstrates the influence of the speed of the query objects on the query processing time of the COSK and CMTkSK+ algorithms. The experimental results indicate that the performance of CMTkSK+ is not significantly influenced by the speed of the query

objects because the candidate objects must be continuously monitored after a regular interval of time, regardless of the speed. On the other hand, for COSK, the performance gradually decreases as the speed of the query objects increases because the objects leave their respective safe regions more frequently. Figure 13(b) shows the communication costs of COSK and CMTkSK+ with respect to the speed of the query objects. CMTkSK+ incurs almost constant communication costs because a server-initiated request to verify the candidate objects does not depend on the speed. For COSK, the query objects cross safe regions more frequently when the speed is high, which increases the communication costs.

FIGURE 11: Effect of number of keywords on query processing time and communication cost.

FIGURE 12: Effect of α on query processing time and communication cost.

7.2.6. Effect of Mobility. Figure 14 shows the effect of mobility M_{qry} (mobility refers to the percentage of query objects that are moving at any timestamp) on the performance of COSK and CMTkSK+ algorithms. As expected, the query processing time and communication costs for both algorithms increase with M_{qry} . Nevertheless, COSK performs better than CMTkSK+ in terms of query processing time and communication costs.

7.2.7. Effect of Directed Edges. Figure 15 shows the impact of percentage of directed edges E_{dir} on the performance of COSK and CMTkSK+ algorithms. The query processing time

increases with E_{dir} because algorithm needs to explore more edges to retrieve the top- k keyword queries. However, the communication cost is not significantly affected by the value of E_{dir} for both the algorithms.

7.2.8. Effect of Datasets. Figure 16 demonstrates the index sizes of the COSK and CMTkSK+ approaches for different datasets. As shown in Figure 16, both algorithms have similar index sizes. However, COSK has minor space overhead because it stores additional information of the highest significance factor θ_i of edges. More important, this space overhead is minimal as compared to the gain achieved by COSK in query processing time and communication costs.

FIGURE 13: Effect of speed on query processing time and communication cost.

FIGURE 14: Effect of mobility on query processing time and communication cost.

7.3. Experimental Results of Top-k Spatial Keyword Queries in Dynamic Road Networks. In this section, we evaluate the performance of COSK and basic algorithm for dynamic road networks. The E_{upd} indicates the percentage of all edges that change their weight at each timestamp. The length of an updated edge is randomly selected between 0.1 to 10 times the original length. Figure 17(a) depicts the query processing time of COSK and basic algorithm. It is evident from the figure that query processing time of basic algorithm is not significantly affected by E_{upd} . This is mainly because the query objects issue top-k spatial queries at each timestamp. However, query

processing time of COSK increases with the value of E_{upd} because the probability that the updated edge may associated with the monitoring region of query q increases with E_{upd} . Therefore, when E_{upd} becomes large the results need to be frequently updated which increases the query processing time. Figure 17(b) shows the communication costs of COSK and basic algorithm with respect to E_{upd} . Basic algorithm incurs almost constant communication costs regardless of the value of E_{upd} . In contrast, the communication cost of COSK increases with E_{upd} because the query result and safe regions needs to be frequently updated.

FIGURE 15: Effect of E_{dir} on query processing time and communication cost.

FIGURE 16: Effect of dataset on index size.

8. Conclusion

In this paper, we investigated moving top- k spatial keyword queries in directed and dynamic road networks. We presented an efficient indexing framework using inverted files that indexes the data objects on edges, allowing for the effective searching of data objects relevant to queries in terms of both textual and spatial relevance. We also presented a safe-exit-based algorithm called COSK to monitor moving top- k spatial keyword queries. We demonstrated that the query results remain valid as long as the query object resides within a safe region. Furthermore, COSK can effectively monitor the validity of query results and safe regions in dynamic road networks. Finally, an experimental evaluation conducted on

real road networks demonstrated that COSK significantly reduced the query processing time and communication costs compared to the CMTkSK+ algorithm.

Data Availability

The real road network data used in this study are also used in many previous studies. The road network data is cited in the manuscript and it is available at <https://www.cs.utah.edu/~lifeifei/SpatialDataset.htm>. To simulate the moving queries the authors used the spatiotemporal data generator which is also used in previous studies. The research article of generator is cited in the manuscript. The documentation and source

FIGURE 17: Effect of E_{upd} on query processing time and communication cost.

files of generator are available at <https://iapg.jade-hs.de/personen/brinkhoff/generator/>. They used the Twitter tweets for generating the description of data objects and also query keywords. The tweets used can be accessible at <http://follow-thehashtag.com/datasets/free-twitter-dataset-usa-200000-free-usa-tweets/>.

Conflicts of Interest

The authors declare that there is no conflicts of interest regarding the publication of this paper.

Acknowledgments

Hyung-Ju Cho was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean Government (MSIP) (NRF-2016R1A2B4009793) and this research was partially supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2016RID1A1B03934129).

References

- [1] D. Papadias, N. Mamoulis, J. Zhang, and Y. Tao, "Query processing in spatial network databases," in *Proceedings of the 29th International Conference on Very Large Data Bases (VLDB '03)*, pp. 802–813, September 2003.
- [2] H.-J. Cho, K. Ryu, and T.-S. Chung, "An efficient algorithm for computing safe exit points of moving range queries in directed road networks," *Information Systems*, vol. 41, pp. 1–19, 2014.
- [3] G. Tsatsanifos and A. Vlachou, "On processing Top-k spatio-textual preference queries," in *Proceedings of the 18th International Conference on Extending Database Technology (EDBT '15)*, pp. 433–444, March 2015.
- [4] R. Li, A. X. Liu, A. L. Wang, and B. Bruhadeshwar, "Fast range query processing with strong privacy protection for cloud computing," *Proceedings of the VLDB Endowment*, vol. 7, no. 14, pp. 1953–1964, 2014.
- [5] G. Cong, C. S. Jensen, and D. Wu, "Efficient retrieval of the Top-k most relevant spatial web objects," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 337–348, 2009.
- [6] Z. Li, K. C. K. Lee, B. Zheng, W.-C. Lee, D. Lee, and X. Wang, "IR-tree: An efficient index for geographic document search," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 4, pp. 585–599, 2011.
- [7] Y. Zhou, X. Xie, C. Wang, Y. Gong, and W. Ma, "Hybrid index structures for location-based web search," in *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, pp. 155–162, Bremen, Germany, October 2005.
- [8] J. Zobel and A. Moffat, "Inverted files for text search engines," *ACM Computing Surveys*, vol. 38, no. 2, 2006.
- [9] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger, "R*-an efficient and robust access method for points and rectangles," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, vol. 19, pp. 322–331, May 1990.
- [10] R. Hariharan, B. Hore, C. Li, and S. Mehrotra, "Processing spatial-keyword (sk) queries in geographic information retrieval (gir) systems," in *Proceedings of the 19th International Conference on Scientific and Statistical Database Management (SSDBM '07)*, July 2007.
- [11] I. De Felipe, V. Hristidis, and N. Rishe, "Keyword search on spatial databases," in *Proceedings of the 24th International Conference on Data Engineering (ICDE '08)*, pp. 656–665, April 2008.
- [12] J. B. Rocha-Junior, O. Gkorgkas, S. Jonassen, and K. Nørvåg, "Efficient processing of top-k spatial keyword queries," in *Proceedings of the International Symposium on Spatial and Temporal Databases*, pp. 205–222, Springer, 2011.
- [13] D. Zhang, K.-L. Tan, and A. K. Tung, "Scalable top-k spatial keyword search," in *Proceedings of the 16th International Conference on Extending Database Technology*, pp. 359–370, 2013.

- [14] J. B. Rocha-Junior and K. Nørnvåg, "Top-k spatial keyword queries on road networks," in *Proceedings of the 15th International Conference on Extending Database Technology*, pp. 168–179, Berlin, Germany, March 2012.
- [15] H.-J. Cho, S. J. Kwon, and T.-S. Chung, "A safe exit algorithm for continuous nearest neighbor monitoring in road networks," *Mobile Information Systems*, vol. 9, no. 1, pp. 37–53, 2013.
- [16] D. Yung, M. L. Yiu, and E. Lo, "A safe-exit approach for efficient network-based moving range queries," *Data & Knowledge Engineering*, vol. 72, pp. 126–147, 2012.
- [17] M. Attique, H. Cho, R. Jin, and T. Chung, "Efficient Processing of Continuous Reverse k Nearest Neighbor on Moving Objects in Road Networks," *ISPRS International Journal of Geo-Information*, vol. 5, no. 12, p. 247, 2016.
- [18] H. G. Elmongui, M. F. Mokbel, and W. G. Aref, "Continuous aggregate nearest neighbor queries," *GeoInformatica*, vol. 17, no. 1, pp. 63–95, 2013.
- [19] D. Wu, M. L. Yiu, C. S. Jensen, and G. Cong, "Efficient continuously moving top-k spatial keyword query processing," in *Proceedings of the IEEE International Conference on Data Engineering (ICDE '11)*, pp. 541–552, Hannover, Germany, April 2011.
- [20] W. Huang, G. Li, K.-L. Tan, and J. Feng, "Efficient safe-region construction for moving top-k spatial keyword queries," in *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pp. 932–941, 2012.
- [21] L. Guo, J. Shao, H. H. Aung, and K.-L. Tan, "Efficient continuous top-k spatial keyword queries on road networks," *GeoInformatica*, vol. 19, no. 1, pp. 29–60, 2014.
- [22] Y. Li, G. Li, L. Shu, Q. Huang, and H. Jiang, "Continuous monitoring of top-k spatial keyword queries in road networks," *Journal of Information Science and Engineering*, vol. 31, no. 6, pp. 1831–1848, 2015.
- [23] M. Attique, A. Khan, and T.-S. Chung, "ESPAK: Top-k spatial keyword query processing in directed road networks," in *Proceedings of the Workshops of the International Conference on Extending Database Technology and the International Conference on Database Theory (EDBT/ICDT '17)*, March 2017.
- [24] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing & Management*, vol. 24, no. 5, pp. 513–523, 1988.
- [25] V. N. Anh, O. de Kretser, and A. Moffat, "Vector-space ranking with effective early termination," in *Proceedings of the 24th Annual International ACM SIGIR Conference*, pp. 35–42, New Orleans, LO, USA, 2001.
- [26] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.
- [27] "Real datasets for spatial databases," <https://www.cs.utah.edu/lifeifei/SpatialDataset.htm>.
- [28] "Twitter," <https://twitter.com>.
- [29] T. Brinkhoff, "A framework for generating network-based moving objects," *GeoInformatica*, vol. 6, no. 2, pp. 153–180, 2002.