Knowledge-Based Systems 93 (2016) 121–134

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Efficient reverse spatial and textual *k* nearest neighbor queries on road networks

Changyin Luo[a], Li Junlin[b], Guohui Li[a], Wei Wei[a,*], Yanhong Li[c], Jianjun Li[a]

[a] School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China
[b] Wuhan Digital Engineering Institute, Wuhan, China
[c] Department of Computer Science, South-Central University for Nationalities, Wuhan, China

ARTICLE INFO

Article history:
Received 14 July 2015
Revised 30 October 2015
Accepted 6 November 2015
Available online 27 November 2015

Keywords:
Spatial keyword query
Reverse *k* nearest neighbor
Road network
Network voronoi diagram

ABSTRACT

The proliferation of geo-positioning technologies boosts the prevalence of GPS-enabled devices, and thus many spatial-textual objects that possess both text descriptions and geo-locations are extensively available in reality. Hence, how to efficiently exploit both spatial and textual description of objects to a spatial keyword query (SKQ) has increasingly become a challenging problem. Previous studies on SKQ problem usually focus on Euclidean space. In the real world, however, most of the spatial-textual objects lie on road networks. This paper takes the first step to investigate a novel problem, namely, reverse spatial and textual *k* nearest neighbor (RST*k*NN) queries on road networks. We formalize the RST*k*NN queries and present several spatial keyword pruning methods to accelerate the query processing. Then two effective verifying techniques are proposed, which can be seamlessly integrated into our RST*k*NN query procedure. Finally, comprehensive experiments on real-world and synthetic data sets are conducted to demonstrate the performance of our approaches.

1. Introduction

With the rapid development of mobile portable devices and location positioning technologies, a large number of user locations are shared on various social platforms, such as Facebook, Twitter, Foursquare, Flickr and Gowalla. Meanwhile, increasing volumes of geo-textual objects that represent Point-of-interests (POIs, e.g., shopping mall, hotel or restaurant) are gaining in prevalence. Generally, a geo-textual object contains a geographical location (i.e., longitude, latitude) and a textual description (e.g., features, reviews, facilities). The massive amount of available geo-textual data enables users to retrieve a set of objects that best matches the user's submitted spatial keyword query (i.e., SKQ, which includes a geographical location and a set of keywords), in terms of both spatial proximity to query location and textual relevance to query keywords.

Reverse *k* Nearest Neighbor (R*k*NN) [1] query, which aims to find a set of objects that take the query as one of their *k*NN based on the spatial distance, has been studied extensively (e.g., [1–12]) over the past decade, due to its importance in a wide range of applications, such as location based service, resource allocation, marketing and decision support, profile-based management, etc. These traditional studies on the retrieval of R*k*NN only consider *spatial distance* as a unique influence factor. However, in real-world applications, both *spatial distance* and *textual relevance* should be taken into account. For example, if one plans to select a location from a given set of potential locations for establishing a new facility (e.g., restaurant, hospital, supermarket), a better choice might be choosing a location that could minimize the average distance among customers, and meanwhile have less textual relevance with their competitors. As another example, assume the customers specify their procurement plans via a set of keywords (e.g., computer, printer, fax) and their locations, a shopping mall can pose an RST*k*NN query to find the potential buyers (customers) whose keywords are relevant to that of the shopping mall and meanwhile have the shopping mall as one of their *k* nearest neighbor.

In recent years, SKQ has become an active topic in database community. Most of the existing studies on SKQ are restricted to Euclidean space [13–22]. According to [23], previous works mainly focus on three types of SKQ in Euclidean space, i.e., Boolean range queries (BRQ) [24,25], Boolean *k*NN queries (B*k*Q) [17,26] and Top-*k* *k*NN queries (T*k*Q) [13,14,21,22,27]. Nevertheless, in reality, the position and accessibility of spatial-textual objects are constrained by network connectivity, and spatial proximity should be determined by the shortest path distance rather than Euclidean distance. Recently, spatial keywords queries on road networks have drawn increasing attention. Rocha et al. [28] pioneer T*k*Q queries on road networks. Range-constrained spatial keyword queries on road networks have

* Corresponding author. Tel.: +8618827440253.
E-mail address: weiwei8329@gmail.com, weiw@hust.edu.cn (W. Wei).

http://dx.doi.org/10.1016/j.knosys.2015.11.009
0950-7051/

been described in [29]. Guo et al. [30] propose a safe segment to continuously monitor T*k*Q queries on road networks. In order to obtain a spatially diversified SKQ result, diversified spatial keywords search (DSKQ) on road networks is investigated in [31], in which a signature-based inverted indexing and an incremental network expansion method are developed for DSKQ search. Gao et al. [32] design an innovative *count-tree* to study reverse top-*k* Boolean spatial keyword (R*k*BSK) retrieval on road networks. In their work, several novel pruning heuristic methods are developed to facilitate R*k*BSK queries processing, but they can only process Boolean spatial keyword query.

Although the traditional R*k*NN queries have been particularly well studied, they only focus on spatial location but ignore text (keywords) relevance. Recently, Lu et al. [33] first take the textual relevance into consideration for R*k*NN queries in Euclidean space. Albeit they design a branch-and-bound search algorithm based on an innovative index called IUR-tree (Intersection-Union R-tree) in their work, their approach cannot be employed to handle RST*k*NN queries on road networks. The key reason is that, IUR-tree is a combination of textual vectors and R-tree that is constructed in Euclidean space, while spatial distance between two objects on road networks should be evaluated by the shortest path distance rather than Euclidean distance. Hence, the pruning methods designed based on IUR-tree cannot work on road networks. As a result, their branch-and-bound search framework cannot be adopted to solve RST*k*NN queries on road networks.

In this work, we investigate RST*k*NN queries on road networks, which pose significant challenges to the existing approaches for processing both conventional RkNN queries (without taking textual relevance into account) and RST*k*NN queries in Euclidean space (its computation cost for the spatial proximity is much lower than that in road networks). Furthermore, RST*k*NN queries in our work belong to *score based* spatial keywords queries. Therefore, the techniques concerning the Boolean SKQ cannot be employed to solve our problem directly.

The contributions of this paper can be summarized as follows:

- We formalize reverse spatial and textual *k* nearest neighbor (RST*k*NN) queries on road networks, and identify the problem of RST*k*NN retrieval. To the best of our knowledge, this is the first work on RST*k*NN queries on road networks.
- We describe several pruning methods to prune non-promising objects at a low cost. The first verifying algorithm in our solutions is based on the network-expansion. In order to avoid expanding road networks multiple times as the first method does, we take advantage of Network Voronoi Diagram (NVD) to develop the second algorithm to obtain RST*k*NN results in an efficient way.
- Comprehensive experiments on real-world and synthetic datasets demonstrate the effectiveness and efficiency of our approach.

The rest of this paper is organized as follows. Section 2 reviews related work and Section 3 gives preliminaries and describes index structure. In Section 4, a basic approach is described. Two efficient algorithms for RST*k*NN queries are developed in Section 5. The experimental results are demonstrated in Section 6. Section 7 makes the conclusion.

2. Related work and background

2.1. RkNN queries on road networks

Korn et al. [1] are the pioneers who first research on RNN queries. They answer RNN query by pre-calculating and adopt three phases, namely pruning, containment and verification, to obtain the final results. After that, numerous literatures concerning the variants of RNN queries in Euclidean space have been particularly well studied [1–10,12]. The following will make an overview of RNN queries on road networks. The snapshot RNN queries in spatial networks are first discussed by Safar et al. [34], in which NVD is utilized to efficiently process RNN queries. In a following work [35], they extend their approach to answer RkNN queries in spatial networks. Sun et al. [36] study the continuous monitoring of RNN queries on road networks. Li et al. [37] design a novel DLM-tree that represents the whole monitoring area of a continuous RkNN (CRkNN) queries to explore CRkNN queries on road networks. Cheema et al. [11] employ a filter and refinement technique to first study CRkNN retrieval (monochromatic and bichromatic) in spatial networks where both the objects and queries continuously change their locations.

2.2. Spatial keyword queries

Retrieving geo-textual objects with query location and keywords has gained increasing attention recently for the popularity of location-based services. There are two types of SKQ, namely, Boolean SKQ and score-based SKQ. The Boolean SKQ is to find the *k* objects nearest to the query *q* among a set of objects whose keyword set covers the query keywords. While score-based SKQ is to obtain the results according to score evaluated by a ranking function that takes into account the spatial proximity and text relevancy (e.g. Eq. (1)). Comparing with Boolean SKQ, it is much more expensive to obtain a score-based SKQ result. A comprehensive experimental evaluation of different SKQ indexing and query processing techniques have been surveyed in [23]. Several geo-textual indices have been developed to efficiently answer T*k*Q, such as IR^2-tree [17], IR-tree [13], S2I [38], I^3 [39] and IL-Quadtree [19]. Top-*k* spatial keyword queries on trajectories are first investigated in [40], in which *k* trajectories whose text descriptions cover the keywords given by the user and that have the shortest match distance are found out. In order to preserve user privacy in text-based search, Wang et al. [41] propose a new dummy query generation method (called HDGA) to deal with various attacks discussed in their work. Literatures [42–44] study *closet keywords search* (Keyword Cover), which retrieves objects that should cover a set of query keywords and have the minimum inter-objects distance. Motivated by the observation of increasing availability and importance of keyword rating in decision marking, Deng et al. [45] investigate a generic version of closet keyword search (called Best Keyword Cover) which considers inter-objects distance as well as the keyword rating of objects. Sometimes, users may wonder why some known object is unexpectedly missing from a result when a SKQ is issued, [46] takes the lead in exploring how to answer why-not questions on spatial keyword top-*k* queries using query refinement. Wang et al. [47] propose a novel adaptive spatial textual partition index (AP-Tree) to support continuous spatial keyword queries over stream. Moreover, many variants of SKQ have been developed such as direction-aware SKQ [48], interactive Top-*k* spatial keyword (IT*k*SK) query [49], temporal spatial-keyword Top-*k* publish/subscribe (TaSK) query [50], approximate keyword query of sematic trajectory [51] and so on. However, all the methods mentioned above cannot be employed to support RST*k*NN retrieval.

3. Preliminaries

In this section, the problem of RST*k*NN queries on road networks as well as the necessary definitions is formally given in Section 3.1, followed by the indexing architecture in Section 3.2.

3.1. Problem definition

Road networks. We model a road network as a weighted graph $G = (V, E, W)$, where V is the set of vertices (i.e., road conjunctions or road borders), E is the set of edges, and W is the set of weights (network distance) that are associated with each edge. Without loss of generality, we assume bidirectional traffic which is pervasive in real life. Unidirectional traffic is also supported by our approach.

Object set. Let O represent a set of spatial-textual objects on the edges E of G. Each object $o \in O$ has a spatial location $o.l$ and a textual description (or called document) $o.d$. Denote $|o, n|$ and $|o, n'|$ as the distance between an object o and two end nodes of the edge (n, n') on which it lies. The shortest path distance between two objects o and o' on G is defined as $d_N(o.l, o'.l)$.

Spatial-textual similarity. Following the previous work [28], we define a spatial-textual similarity in Eq. (1).

$$\tau(o, q) = \frac{\theta(o.d, q.d)}{1 + \alpha \cdot \delta(o.l, q.l)} \tag{1}$$

where $\delta(o.l, q.l)$ represents the network proximity between $o.l$ and $q.l$, and $\theta(o.d, q.d)$ is the text relevance between $o.d$ and $q.d$. $\alpha \in R^+$ is a query preference parameter to balance between network proximity and text relevance. For example, if $\alpha = 0$, it means the network proximity is ignored. And if $\alpha > 1$, it means the network proximity is more important than the textual relevance. Specifically, the network proximity is defined by the network distance between $o.l$ and $q.l$, which is the length of the short path connecting o and q.

$$\delta(o.l, q.l) = d_N(o.l, q.l) \tag{2}$$

Actually, there exist several methods to measure the textual relevance, such as [13], cosine similarity [28,30,38]. In this paper, we adopt the well-known cosine similarity to compute the textual relevance between $o.d$ and $q.d$, as shown in Eq. (3).

$$\theta(o.d, q.d) = \frac{\sum_{t \in q.d} w_{t,o.d} w_{t,q.d}}{\sqrt{\sum_{t \in o.d} (w_{t,o.d})^2 \sum_{t \in q.d} (w_{t,q.d})^2}} \tag{3}$$

where $w_{t,o.d} = 1 + \ln(f_{t,o.d})$, and $f_{t,\, o.d}$ is the number of occurrences (frequency) of term t in $o.d$; $w_{t,\, q.d}$ is computed by $w_{t,q.d} = \ln(1 + \frac{|O|}{df_t})$, in which $|O|$ is the total number of objects in the system and df_t is the number of objects in O containing t. Specifically, the value of $\theta(o.d, q.d)$ is within the range [0, 1], and it is proportional to the textual relevance.

Similar to [38], we also give a definition of the impact $\lambda_{t,\, d}$ of a term t in document d, where d represents a description or a set of keywords of object $o.d$ (or query $q.d$). The impact $\lambda_{t,\, d}$ is the normalized weight of the term in the document [52],

$$\lambda_{t,d} = \frac{w_{t,d}}{\sqrt{\sum_{t \in d} (w_{t,d})^2}} \tag{4}$$

Subsequently, the textual relevance $\theta(o.d, q.d)$ in Eq. (3) can be rewritten as follows.

$$\theta(o.d, q.d) = \sum_{t \in q.d} \lambda_{t,o.d} . \lambda_{t,q.d} \tag{5}$$

Next, we give the definition of Spatial and Textual k Nearest Neighbor (**ST*k*NN**) query and **RST*k*NN** query on road networks.

Definition 1 (ST*k*NN query on road networks)**.** Given a set O of spatial-textual objects and a query object $q =< q.l, q.d, q.k >$, where $q.l$ is the query location, $q.d$ is the query keywords, $q.k$ is the number of requested results. An object $o \in O$ is one of k most similar objects with q, denoted by $o \in STkNN(q)$ if and only if it satisfies the condition: $|\{p \in O | \tau(p, q) \geq \tau(o, q)\}| < k$.

Definition 2 (RST*k*NN query on road networks)**.** Given a set O of spatial-textual objects and a query object $q =< q.l, q.d, q.k >$, RST*k*NN query on the road networks retrieves objects that contain query q in their k most similarity objects, namely, $RSTkNN(q) = \{o \in O | q \in STkNN(o)\}$.

3.2. Indexing architecture

Our work concentrates on RST*k*NN queries on road networks. It incrementally expands the networks from a query point which is similar to Dijkstra's algorithm in essence, but the pruning methods proposed in this work are able to accelerate the query processing, and the expanding stop condition can terminate the networks expansion as early as possible. Applying Dijkstra's approach to expand the road networks, [28] investigates Top-k spatial keyword queries on road networks. Its indexing structure combines IR-tree [13,16,38] and Network R-tree [53]. Likewise, we adopt a similar indexing framework by following the works [28,30]. Next, we will briefly illustrate it.

Fig. 1 presents the indexing architecture, which consists of four components. (a) **Spatial component** combines spatial and network connectivity information as proposed in [53]. It is employed to locate the road edge on which the query lies. (b) **Adjacency component** points to the adjacent vertices (road network nodes) of a given vertex allowing traversing the network from vertex to vertex. It adopts a B-tree to point to block in the adjacency file where the adjacent vertices of a given vertex v_i are stored. The adjacency file stores for each v_i: (i) the id of each edge, and (ii) the length of the edge. (c) **Mapping component** employs a B-tree that maps a key composed of the pair of *edge id* and *a keyword*(a term: t_i) to the inverted list that contains the objects located on the edge with the term in their description. This component also contains the *maximum impact* λ_t^- of a given term t among the description of the objects located on a given edge. The inverted list of a term t on an edge is accessed only if τ derived by minimum distance and maximum impact may turn an object, present on the edge, inside the top-k objects obtained so far. (d) **Inverted file component** contains inverted list and a vocabulary. Each inverted list stores the objects located on an edge with a term in their textual descriptions. For each object, the inverted list stores: (i) the distance between the object and the reference node of the edge, and (ii) the impact of the term in the description of the object. The vocabulary file stores the document frequency df_t of each term.

Based on the indexing architecture, we present a basic approach for RST*k*NN queries in the next section.

4. Basic approach

Although Lu et al. [33] investigate RST*k*NN search problem in Euclidean space, it cannot be applied to road networks directly as mentioned in Section 1. Hence, actually there exists no previous work that researches on the problem of RST*k*NN queries on road networks.

[28] studies T*k*Q on road networks that return k best objects ranked according to the score (Eq. (1)). Its idea can be briefly described as follows: it first locates the edge on which q lies, and expands the adjacencies of q similarly to Dijkstra's algorithm to find objects on the edges. If the keywords of o are not relevant to $q.d$, o will be ignored. Otherwise, it computes the score of o using Eq. (1). The road networks are expanded gradually to check each relevant object until the following conditions are satisfied: (i) the entire network is expanded, or (ii) the unexamined networks cannot have a qualified answer. In other words, the minimum network distance to any remaining object produces an aggregated score that is smaller than or equals to the score of the *kth* object already found. Specifically, when a vertex v_i is visited, we assume v_i and q have the same keyword set, i.e., $\theta(v_i.d, q.d) = 1$, and then we compute an aggregated score $\tau^-(v) = \frac{1}{1+\alpha \cdot \delta(v.l, q.l)}$. If $\tau^-(v)$ is smaller than or equals to the score of the *kth* object already found, we are certain the network expansion via v_i can be safely terminated.

Based on this method, we develop a baseline method (**BM**), in which we first locate the edge on which q lies, then expand the road network utilizing Dijkstra's approach. For each object o located on the each visited edge, if the keyword set of o is relevant to $q.d$, BM employs the method T*k*Q to compute $STkNN(o)$, and if the *kth* result (e.g., o_k) is smaller than the similarity between o and q, i.e., $\tau(o_k, o) < \tau(q, o)$, then o is added to $RSTkNN(q)$.

However, even though BM can obtain a correct RST*k*NN result, its shortcomings are obvious: in order to obtain the complete answer,

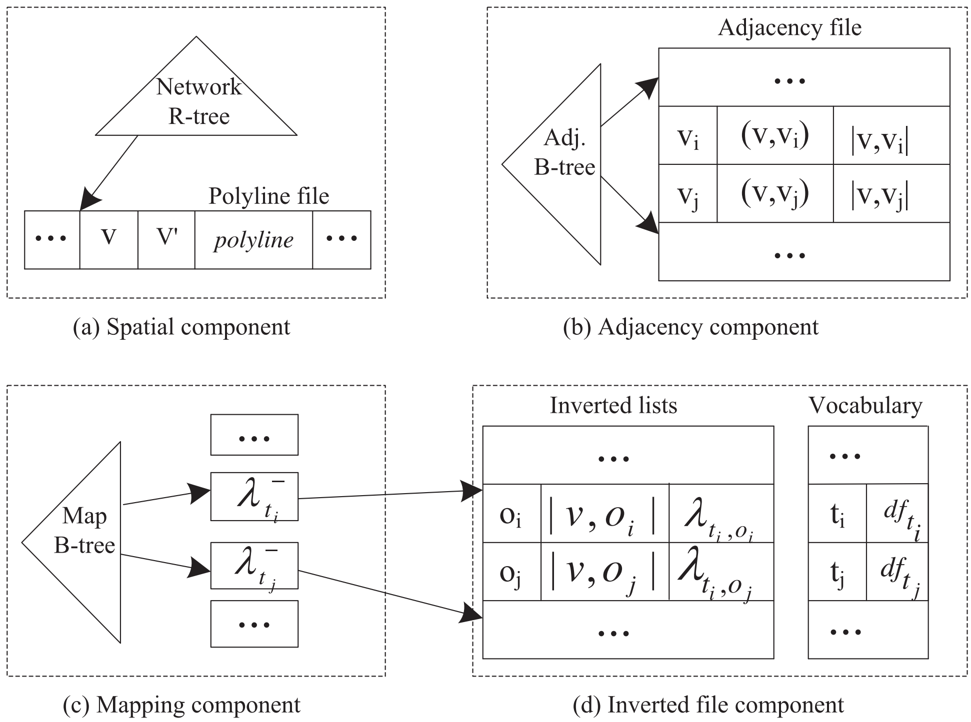


Fig. 1. Indexing architecture.

BM has to expand network gradually and check every relevant spatial-textual object one by one. In particular, it has to compute $STkNN(o_i)$ for each candidate object o_i, and then checks whether $\tau(o_k,\ o_i) < \tau(q,\ o_i)$. Even worst, if all objects on road networks are relevant to q, the whole data set should be traversed $(|O|+1)$ times, i.e., fetching data objects 1 time and verification $|O|$ times by employing T*k*Q method, resulting in high I/O overhead and high CPU cost.

By analyzing Eq. (1), score (i.e., $\tau(o,\ q)$) is determined by *network proximity* and *textual relevance*. Given two candidate objects o_i and o_j, assume both of them have the same textual relevance to q, i.e., $\theta(o_i.d, q.d) = \theta(o_j.d, q.d)$, if q is closer to o_i than o_j, q is more likely to be a member of $STkNN(o_i)$. In other words, if $\theta(o_i.d, q.d) = \theta(o_j.d, q.d)$ and $\delta(o_i.l,\ q.l) \le \delta(o_j.l,\ q.l)$, then $\tau(o_i,\ q) \ge \tau(o_j,\ q)$, which means o_i is more likely to be a valid member of RSTkNN(q) as compared with o_j. As depicted in Fig. 2, the keyword set of each object is in the braces. For instance, both o_3 and o_4 located on the edge $(n_2,\ n_4)$ have the same keyword sets, but $d_N(o_3,\ q) < d_N(o_4,\ q)$, thus $\delta(o_3.l,\ q.l) < \delta(o_4.l,\ q.l)$. Comparing with o_4, o_3 has a higher chance to be a member of $RSTkNN(q)$. If o_3 is verified to be an invalid answer, we can directly ignore o_4 without verification.

Conversely, assume o_i and o_j are located on the same shortest path to q, and both of them have the same distance to q, i.e., $d_N(o_i, q) = d_N(o_j, q)$. If the keyword set of o_i is more relevant to q, i.e., $\theta(o_i.d,\ q.d) > \theta(o_j.d,\ q.d)$, we can obtain $\tau(o_i,\ q) > \tau(o_j,\ q)$, which means o_i is more likely to be a member of $RSTkNN(q)$.

According to the above discussion, we find that it is unnecessary to verify each object when obtaining a candidate set for an RST*k*NN query, namely, part of candidates could be pruned based on some verified candidates. As shown in Fig. 2, for a candidate set: $\{o_1,\ o_2,\ o_3,\ o_4,\ o_5\}$, supposed o_2 is verified, i.e., $o_2 \in RSTkNN(q)$, we can infer $o_1 \in RSTkNN(q)$ due to $\theta(o_1.d, q.d) = \theta(o_2.d, q.d) \wedge d_N(o_1.l, q.l) < d_N(o_2.l, q.l)$. Next, suppose $o_3 \notin RSTkNN(q)$, as $\theta(o_3.d, q.d) = \theta(o_4.d, q.d) \wedge d_N(o_3.l, q.l) < d_N(o_4.l, q.l)$, hence, o_4 is an invalid result. Specifically, it is not difficult to infer that $o_5 \notin RSTkNN(q)$ owning to $o_5.d \subset o_3.d$ (i.e., $\theta(o_5.d,\ q.d) < \theta(o_3.d,\ q.d)$) and $d_N(o_5.l,\ q.l) > d_N(o_3.l,\ q.l)$.

5. RST*k*NN query process

In this section, we first give four lemmas to accelerate the *RSTkNN* queries processing, and then expound on our RST*k*NN Algorithm.

5.1. Pruning methods

For pruning the unpromising objects as many as possible, several effective pruning methods are proposed, which take advantage of both spatial and textual information. We present them as follows.

Lemma 1. *Given a query point $q = \langle q.l, q.d, q.k\rangle$ and a spatial-textual object o whose keywords are relevant to $q.d$, i.e., $o.d \cap q.d \ne \emptyset$. Let SP_{qo} be the shortest path from q to o, and S_{sk} be the set of spatial-textual objects (including o) located on SP_{qo} with their keyword sets the same as $o.d$, i.e., $S_{sk} = \{o' \in O | o' \in SP_{qo} \wedge o'.d = o.d\}$. If $|S_{sk}| > k$, we can infer $o \notin RSTkNN(q)$. If $o \in RSTkNN(q)$, it is certain that $|S_{sk}| \le k$.*

Proof. We prove the first statement by contradiction. That is, if $|S_{sk}| > k$, we have $o \in RSTkNN(q)$. If a ST*k*NN query is issued at o, based on the fact that $S_{sk} = \{o' \in O | o' \in SP_{qo} \wedge o'.d = o.d\}$ and $o.d \cap q.d \ne \emptyset$, we have $\forall o' \in S_{sk}$, $1 = \theta(o'.d, o.d) \ge \theta(o'.d, q.d)$. Furthermore, every point in the set S_{sk} lies on the shortest path SP_{qo}, thus $\forall o' \in S_{sk}$, $\delta(o.l,\ o'.l) < \delta(o.l,\ q.l)$. According to Definition 1, we have $\forall o' \in S_{sk}$, $\tau(o',\ o) > \tau(o,\ q)$. Because of $|S_{sk}| > k$, q cannot be an answer point for $STkNN(q)$, i.e., $o \notin RSTkNN(q)$, which contradicts our assumption that $o \in RSTkNN(q)$. Therefore, our assumption is invalid, and the first statement is correct.

Then, we prove the second statement via contradiction as well. Assume that $o \in RSTkNN(q)$, it can infer $|S_{sk}| > k$. Now suppose $S_{sk} = \{o_1, o_2, \cdots o_k, o_{k+1}, \ldots\}$, and the points in S_{sk} are sorted in ascending order of their distance to q, and assume o is in the last place, i.e., $\forall o_i \in S_{sk}$, $\delta(o_i.l, q.l) < \delta(o_{i+1}.l, q.l) < \delta(o.l, q.l)$. Based on the fact $\forall o_i \in S_{sk}$, $o_i.d = o.d$, and $o.d \cap q.d \ne \emptyset$, we have $\forall o_i \in S_{sk}$, $1 =$

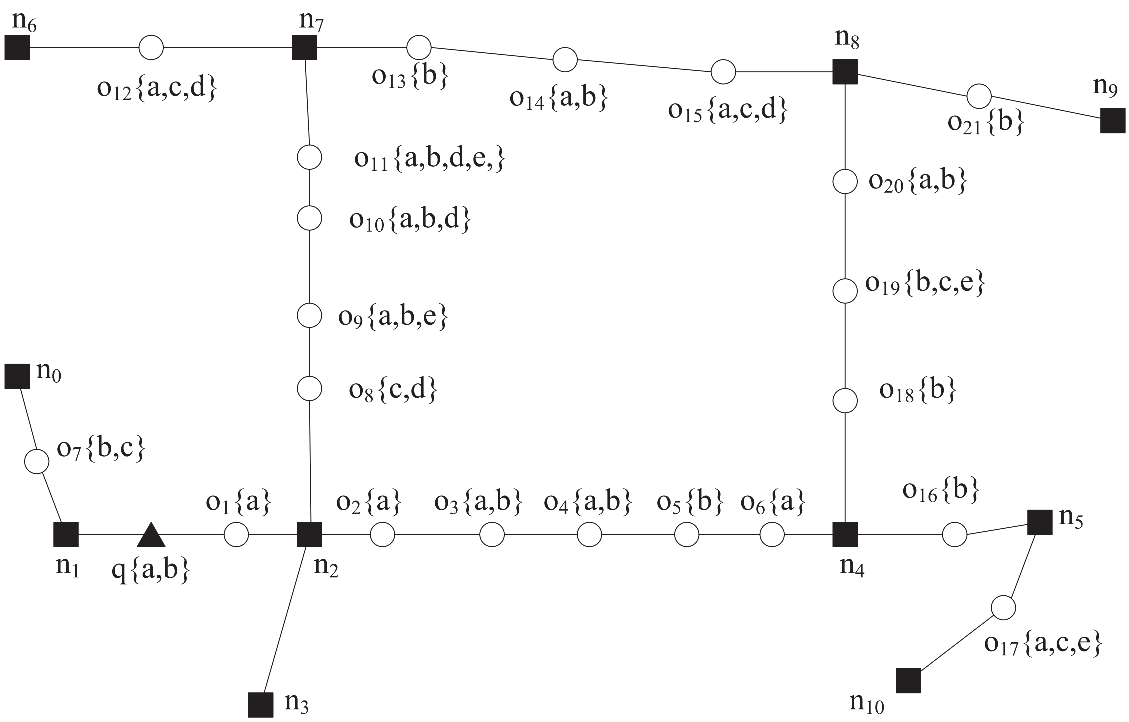


Fig. 2. Road network and spatial-textual objects.

$\theta(o_i.d, o.d) \geq \theta(q.d, o.d)$. Because of $|S_{sk}| > k$, if a STkNN query is issued at o, then $q \notin STkNN(o)$. In other words, $o \notin RSTkNN(q)$. Therefore, our assumption is invalid, and the second statement is correct. The proof completes. □

To better illustrate Lemma 1, let us take the objects depicted in Fig. 2 as an example. q is located on the edge (n_1, n_2), and its keyword set is $\{a, b\}$. If a *RST2NN* (k=2) query is issued at q, we can adopt Dijkstra's approach to extend the road network from q. When the shortest path SP_{qn_4} from q to node n_4 is visited, we find that o_1, o_2 and o_6 located on SP_{qn_4} have the same keyword set, i.e., $o_1.d = o_2.d = o_6.d = \{a\}$, and thus $|S_{sk}| > 2$. According to Lemma 1, we can infer $o_6 \notin RST2NN(q)$. In fact, if o_1 and o_2 have been visited, we can infer that any object with keyword set $\{a\}$ that is located on SP_{qn_4} after o_2 cannot become a member of *RST2NN*(q), they can be ignored directly, which helps to reduce the size of candidate set.

Next, let us still take SP_{qn_4} as example. Suppose we adopt *BM* to check whether o_5 is a valid answer for *RST2NN*(q) or not, specifically, we need to compute *ST2NN*(o_5) first, based on $ST2NN(o_5) = \{o_3, o_4\}$, we can infer $o_5 \notin RST2NN(q)$. Obviously, the verification is costly, yet it can be avoided. Given the fact $o_5.d = \{b\} \subset \{a, b\} = o_3.d = o_4.d = q.d$, as well as o_3, o_4 and o_5 are located on SP_{qn_4}, i.e., $\theta(q.d, o_5.d) = \theta(o_3.d, o_5.d) = \theta(o_4.d, o_5.d)$ and $\delta(q.l, o_5.l) > \delta(o_3.l, o_5.l) > \delta(o_4.l, o_5.l)$, we easily infer $o_5 \notin RST2NN(q)$. Next, we give a lemma that provides the pruning rule used in such example with a guarantee.

Lemma 2. *Given a query point $q = \langle q.l, q.d, q.k\rangle$ and a spatial-textual object o located on the road networks. Let SP_{qo} be the shortest path from q to o, S_{ck} be the set of spatial-textual objects o' located on SP_{qo}, and their keyword set is the subsets of $q.d$, moreover, their keyword is also the superset of $o.d$, i.e., $S_{ck} = \{o' \in O | o.d \subseteq o'.d \subseteq q.d \wedge o' \in SP_{qo}\}$. If $|S_{ck}| \geq k$, it infers $o \notin RSTkNN(q)$. If $o \in RSTkNN(q)$, then $|S_{ck}| < k$.* ***Similarly***, *let S'_{ck} denote the set of spatial-textual objects satisfying condition: $S'_{ck} = \{o' \in O | q.d \subseteq o'.d \subseteq o.d \wedge o' \in SP_{qo}\}$. If $|S'_{ck}| \geq k$, it indicates $o \notin RSTkNN(q)$. If $o \in RSTkNN(q)$, then $|S'_{ck}| < k$.*

Proof. We prove the first statement by contradiction. Suppose the first statement is incorrect. Now, suppose $|S_{ck}| \geq k$, it infers $o \in RSTkNN(q)$. Based on $S_{ck} = \{o' \in O | o.d \subseteq o'.d \subseteq q.d \wedge o' \in SP_{qo}\}$, we have (i) $d_N(o, o') < d_N(o, q)$, i.e., $\delta(o.l, o'.l) < \delta(o.l, q.l)$, and (ii) $\theta(o.d, o'.d) \geq \theta(o.d, q.d)$. If a ST$k$NN query is issued at o, then $\forall o' \in S_{ck}$, $\tau(o, o') > \tau(o, q)$. As $|S_{ck}| \geq k$, therefore, $q \notin STkNN(o)$, i.e., $o \notin RSTkNN(q)$, which contradicts our assumption. Hence, the first statement is correct.

Next, we prove the second statement by contradiction as well. Assume the second statement is invalid, i.e., $o \in RSTkNN(q)$, we have $|S_{ck}| \geq k$. Based on the fact $S_{ck} = \{o' \in O | o.d \subseteq o'.d \subseteq q.d \wedge o' \in SP_{qo}\}$, and $|S_{ck}| \geq k$, if a STkNN query is issued at o, there are at least k spatial-textual objects $o' \in S_{ck}$ satisfying $\tau(o, o') > \tau(o, q)$. Hence, we can infer $q \notin STkNN(o)$, namely, $o \notin RSTkNN(q)$, which contradicts our assumption. Thus, the second statement in lemma is correct.

Finally, we prove the second conclusion in a similar way: based on the given fact $S'_{ck} = \{o' \in O | q.d \subseteq o'.d \subseteq o.d \wedge o' \in SP_{qo}\}$, it is certain that (i) $\delta(o.l, o'.l) < \delta(o.l, q.l)$, and (ii) $\theta(o.d, o'.d) \geq \theta(o.d, q.d)$. Therefore, if $|S'_{ck}| \geq k$, we infer $o \notin RSTkNN(q)$. If $o \in RSTkNN(q)$, it is certain $|S'_{ck}| < k$. The proof completes. □

Let us take Fig. 3 as an example. If a *RST2NN* (k=2) query is issued at q. Suppose we need to verify o_6, the shortest path from q to o_6 is SP_{qo_6}. According to Lemma 2, the corresponding set S_{ck} for o_6 is: $S_{ck} = \{o_1, o_2, o_3, o_4, o_5\}$. Obviously, $|S_{ck}| = 5$ and $|S_{ck}| > k$. Hence, $o_6 \notin RST2NN(q)$. As a result, we can prune o_6 directly. For o_5, we can handle it in a similar way.

Next, let us take Fig. 4 to illustrate the *second conclusion* given in Lemma 2, i.e., $S'_{ck} = \{o' \in O | q.d \subseteq o'.d \subseteq o.d \wedge o' \in SP_{qo}\}$, if $|S'_{ck}| \geq k$, it indicates $o \notin RSTkNN(q)$. For example, if a *RST2NN* (k=2) query is issued at q, suppose o_5 is visited, o_3 and o_4 are located on SP_{qo_5}. The corresponding S'_{ck} for o_5 is: $S'_{ck} = \{o_3, o_4\}$. As $|S'_{ck}| \geq 2$, we can infer $o_5 \notin RST2NN(q)$. In fact, owing to Eq. (1), suppose $\alpha = 1$, $\tau(o_5, o_4) = \frac{4/5}{1+1} = 2/5$. $\tau(o_5, o_3) = \frac{3/5}{1+2} = 1/5$, and $\tau(o_5, q) = \frac{2/5}{1+3} = 1/10$. Hence, $q \notin ST2NN(o_5)$ and $o_5 \notin RST2NN(q)$, which is consistent with the result verified by the second conclusion in Lemma 2.

As shown in Fig. 3, if a *RST2NN* (k=2) query is issued at q, $o_5 \notin RST2NN(q)$ and $o_6 \notin RST2NN(q)$. We find that: (i) o_5 is located on the shortest path SP_{qo_6} from q to o_6, and (ii) $o_6.d \subseteq o_5.d$. If o_5 is not

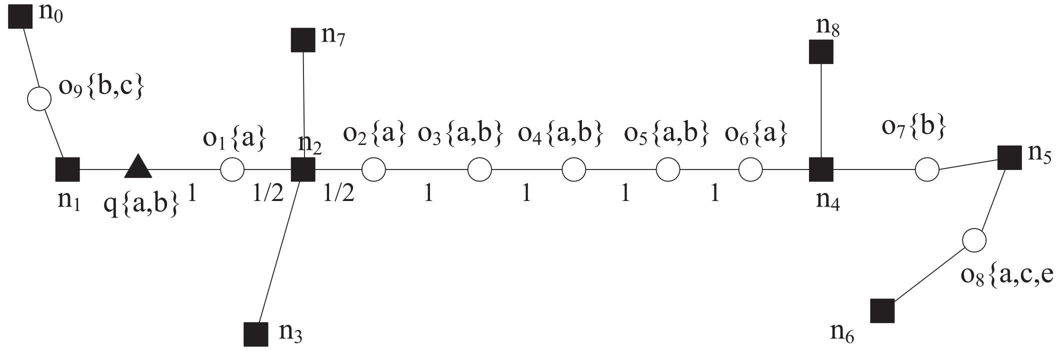


Fig. 3. Illustration of Lemma 2.

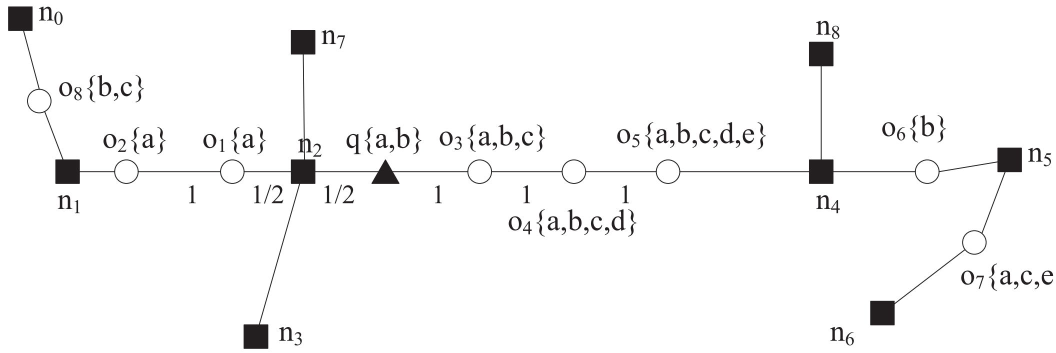


Fig. 4. Supplementary illustration of Lemma 2.

verified to be an answer object, can it be used to prune o_6 directly? We will answer it with Lemma 3.

Lemma 3. *Given a query point $q = < q.l, q.d, q.k >$ and a spatial-textual object o on the road networks. Let S_{sd} indicate all the objects o' satisfying the following conditions: (i) $o'.d \subseteq o.d \subseteq q.d$, and (ii) the shortest path from q to o' passes o, i.e., $S_{sd} = \{o' | o'.d \subseteq o.d \subseteq q.d \wedge o \in SP_{qo'}\}$. If $o \notin RSTkNN(q)$, it can infer $\forall o' \in S_{sd}$, $o' \notin RSTkNN(q)$. **Similarly**, let S'_{sd} denote all the objects o' satisfying: $S'_{sd} = \{o' | q.d \subseteq o.d \subseteq o'.d \wedge o \in SP_{qo'}\}$. If $o \notin RSTkNN(q)$, it is certain that $\forall o' \in S'_{sd}$, $o' \notin RSTkNN(q)$.*

Proof. We prove the first statement by contradiction. Suppose the statement is invalid. That is, if $o \notin RSTkNN(q)$, there is at least one object $o' \in S_{sd}$ belonging to $RSTkNN(q)$, i.e.,$o \notin RSTkNN(q) \to \exists o' \in S_{sd}, o' \in RSTkNN(q)$. Because of $o \notin RSTkNN(q)$, we have $q \notin STkNN(o)$. For each object $o_i \in STkNN(o)$, we can arbitrarily assume their spatial proximity and textual relevance, as long as they guarantee $q \notin STkNN(o)$. Then we can suppose each o_i satisfies the following relationship: $\forall o_i \in STkNN(o)$, $o.d \subseteq o_i.d \subseteq q.d \wedge d_N(o.l, o_i.l) < d_N(o.l, q.l)$. That is, $o.d \subseteq o_i.d \subseteq q.d \to \theta(o, o_i) \geq \theta(o, q)$, and $d_N(o.l, o_i.l) < d_N(o.l, q.l) \to \delta(o.l, o_i.l) < \delta(o.l, q.l)$. Therefore, $\forall o_i \in STkNN(o)$, $\tau(o, o_i) > \tau(o, q)$, which indicates our hypothesis can ensure $q \notin STkNN(o)$. Then, if a *STkNN* query is issued at o', based on the fact $o' \in S_{sd}$, i.e., $o'.d \subseteq o.d \subseteq q.d$, and $\forall o_i \in STkNN(o)$, so we have $o'.d \subseteq o.d \subseteq o_i.d \subseteq q.d$. As a result, $\forall o_j \in \{STkNN(o) \cup o\}$, we have $\theta(o_j.d, o'.d) \geq \theta(q.d, o'.d) \wedge \delta(o'.l, o_j.l) < \delta(o'.l, q.l)$, which indicates that at least $k+1$ objects o_j satisfy $\tau(o', o_j) > \tau(o', q)$. Thus, $o' \notin RSTkNN(q)$, which contradicts our assumption. Hence, the first statement $o \notin RSTkNN(q) \to \forall o' \in S_{sd}, o' \notin RSTkNN(q)$ is correct.

Next, we can adopt the similar method to prove the second state (i.e., $o \notin RSTkNN(q) \to \forall o' \in S'_{sd}, o' \notin RSTkNN(q)$). Based on $o \notin RSTkNN(q)$, we assume that $\forall o_i \in STkNN(o)$, $q.d \subseteq o_i.d \subseteq o.d \wedge d_N(o.l, o_i.l) < d_N(o.l, q.l)$. That is, $\forall o_i \in STkNN(o)$, we have $\tau(o, o_i) > \tau(o, q)$, which can guarantee $o \notin RSTkNN(q)$. Based on fact $S'_{sd} = \{o' | q.d \subseteq o.d \subseteq o'.d \wedge o \in SP_{qo'}\}$, $\forall o' \in S'_{sd}$, we have $q.d \subseteq o_i.d \subseteq o.d \subseteq o'.d \to \theta(o'.d, o_i.d) \geq \theta(o'.d, q.d)$, and $d_N(o'.l, o_i.l) < d_N(o'.l, q.l) \to \delta(o'.l, o_i.l) < \delta(o'.l, q.l)$, which can ensure for each $o' \in S_{sd}$, there are $k+1$ objects $o_j \in \{STkNN(o) \cup o\}$ satisfying $\tau(o', o_j) > \tau(o', q)$. In other words, $o' \notin RSTkNN(q)$. Therefore, $\forall o' \in S_{sd}$, if $o \notin RSTkNN(q)$, it infers $o' \notin RSTkNN(q)$. The proof completes. □

Let us take Fig. 5 to illustrate how to prune objects by Lemma 3. If a *RST2NN* (k=2) query is issued at q, suppose we need to check o_3, we find $ST2NN(o_3) = \{o_2, o_{10}\}$, so $o_3 \notin RST2NN(q)$. Then, the set of objects $\{o_4, o_6, o_7, o_{13}\}$ can be pruned by the *first state* of Lemma 3. And the set of objects $\{o_5, o_8, o_9\}$ can be pruned based on *the second state* of Lemma 3. Therefore, once o_3 is verified that it is an invalid answer, the network expansion via SP_{qn_5} will be terminated immediately.

However, not all candidate objects can be pruned by the above lemmas. Obviously, it is unnecessary to expand the whole network as BM dose. The terminated condition for the expansion of the road networks is given in the lemma as follows.

Lemma 4. *Given a query point $q = < q.l, q.d, q.k >$ on the road networks, SP_{qn} is the shortest path from q to a node n. If there are k objects (excluding q) located on SP_{qn} whose keyword sets are same as $q.d$, then the network expansion via SP_{qn} can be terminated safely.*

Proof. If the networks expansion via SP_{qn} can be terminated, it means there is no qualified answer for $RSTkNN(q)$ on the pruned road networks. We prove this lemma by contradiction. Suppose there is at least one object $o' \in RSTkNN(q)$, but the shortest path $SP_{qo'}$ from q to o' passes the road node n, i.e., $d_N(o', q) > d_N(q, n)$. Based on the fact that k objects (not including q) are located SP_{qn} and their keyword set is the same as $q.d$. Let S denote such k objects, i.e., $S = \{o_i | o_i.d = q.d \wedge o_i \in SP_{qn}\}$ and $|S| = k$, and thus we have $\forall o_i \in S$, $\theta(o'.d, o_i.d) \geq \theta(o'.d, q.d)$ and $\delta(o'.l, o_i.l) < \delta(o'.l, q.l)$. Hence, $\forall o_i \in S$, $\tau(o', o_i) > \tau(o', q)$, therefore, $q \notin STkNN(o') \to o' \notin RSTkNN(q)$, which contradicts our assumption. The proof completes. □

Next, we present the framework of our RST*k*NN algorithm in detail.

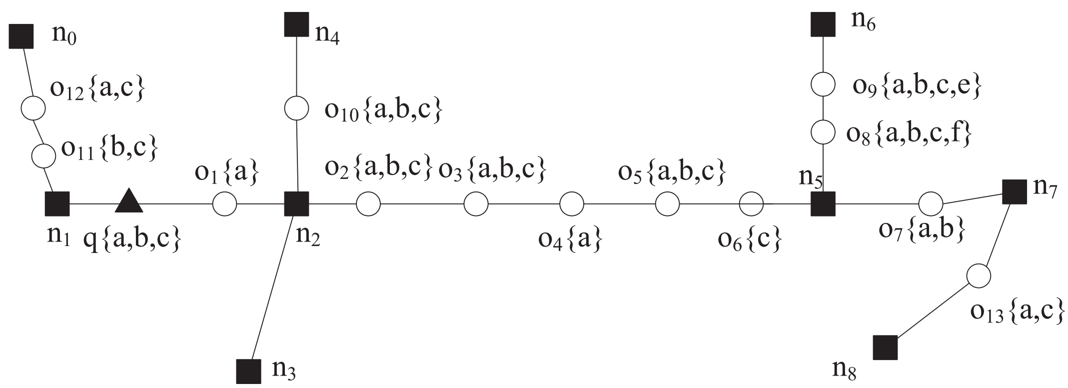


Fig. 5. Illustration of Lemma 3.

5.2. RSTkNN algorithm

The framework of RST*k*NN algorithm consists of three phases: filtering, refinement and pruning. Specifically, (i) **filtering** aims to prune the unqualified spatial-textual objects for an RST*k*NN query. The pruning methods in the last subsection can be adopted to reduce the search space; (ii) **refinement**, in which each remaining candidate o_c that cannot be pruned will be examined by verifying $q \in STkNN(o_c)$; and (iii) **pruning**, employing the verified o_c to further prune the candidate object in S_c.

Algorithm 1: obtain the candidate set S_c of *RSTkNN(q)*.

Input: q, k, a set O of the data objects on road networks
Output: S_c for *RSTkNN(q)*
1 initial $S_c = \emptyset$, Queue $Q = \emptyset$
2 locate the edge (n_i, n_j) that q is located
3 $Q = \{< (q, n_i), d_N(n_i, q) >, < (q, n_j), d_N(n_j, q) >\}$
4 **while** *Q is not empty* **do**
5 $\quad e = (n, n') =$ de-queue(Q)
6 $\quad$**if** *q is located on the edge e* **then**
7 $\quad\quad$**foreach** *$o \in e$ is relevant to q* **do**
8 $\quad\quad\quad n'[o.d].count = 0$
9 $\quad$**else**
10 $\quad\quad$**foreach** *$o \in e$ is relevant to q* **do**
11 $\quad\quad\quad n'[o.d].count = n[o.d].count$
12 $\quad$**foreach** *object o on e* **do**
13 $\quad\quad$**if** $o.d \cap q.d \neq \emptyset \wedge n'[o.d].count < k$ **then**
14 $\quad\quad\quad n'[o.d].count++$ /* */Lemma 1
15 $\quad\quad\quad$search the corresponding set S_{ck} and S'_{ck} on SP_{qo} for o
16 $\quad\quad\quad$**if** $|S_{ck}| < k \wedge |S'_{ck}| < k$ **then** /* */Lemma 2
17 $\quad\quad\quad\quad S_c = S_c \cup \{o\}$
18 $\quad$**if** $n'[q.d].count < k$ **then** /* */Lemma 4
19 $\quad\quad$**foreach** *unvisited adjacent edge (n', n'') in the edge set E* **do**
20 $\quad\quad\quad$en-queue$< (n', n''), d_N(q, n'') >$ to Q
21 return S_c

Algorithm 1 presents the filtering phase of RST*k*NN algorithm. First, it initializes the parameters. S_c is used to preserve the candidate objects for RST*k*NN algorithm. A priority queue Q is adopted to maintain all the visited edges sorted in ascending order of their distances to q. The algorithm should find the edge on which q lies. Suppose q lies on (n_i, n_j), and we assume q is closer to node n_i. Thus, $< (q, n_i), d_N(n_i, q) >$ is firstly stored in Q, and then $< (q, n_j), d_N(n_j, q) >$ is preserved. Second, the algorithm starts to expand the road networks from q until Q is empty (Lines 4–20). At each step, Q de-queues its first element, i.e., $e = (n, n') =$ de-queue(Q). For each object $o \in e$, if $o.d$ is relevant to $q.d$, i.e., $o.d \cap q.d \neq \emptyset$, $n'[o.d].count$ is used to count the number of spatial-objects located on the shortest path $SP_{qn'}$ with their keyword sets that are the same as $o.d$. Specifically, if q is located on the popped edge e, the counter value of $n'[o.d].count$ should be initialized to zero (Line 8). Otherwise, using the counter value of the former node to initialize that of the later one, i.e., $n'[o.d].count = n[o.d].count$ (Line 11). For each object o located on the popped e, if $o.d \cap q.d \neq \emptyset$ and $n'[o.d].count < k$, we can infer that o is a candidate object by Lemma 1. Thus, the value of $n'[o.d].count$ should be increased (Line 14). Furthermore, according to Lemma 2, it needs to retrieve two sets S_{ck} and S'_{ck} for o to judge whether o can be pruned or not (Line 15). If $|S_{ck}| < k$ and $|S'_{ck}| < k$, we are certain o is a candidate object and preserve it in S_c (Line 17). If the counter value of $q.d$ is smaller than k, as guided by Lemma 4, the algorithm should expand the road networks continuously (Lines 18–20). Finally, the candidate set S_c is returned.

Algorithm 2: *STkNN* Algorithm.

Input: q, k, a set O of the data objects on a road network
Output: $STkNN(q)$
1 initial MaxHeap $H = \emptyset$, $\varepsilon = 0$, Queue $Q = \emptyset$
2 locate the edge (n_i, n_j) that q is located
3 use the polyline of (n_i, n_j) to compute $d_N(n_i, q)$ and $d_N(n_j, q)$
4 $Q = \{< (q, n_i), d_N(n_i, q) >, < (q, n_j), d_N(n_j, q) >\}$
5 **while** *Q is not empty and* $\frac{1}{1+\alpha \cdot \delta(n_i.l, q.l)} \leq \varepsilon$ **do**
6 $\quad$**if** $|H| < k$ **then**
7 $\quad\quad$expand network and retrieve k relevant objets ($q.d \cap o.d \neq \emptyset$) to initial H
8 $\quad\quad\varepsilon \leftarrow kth$ score of the objects in H
9 $\quad e = (n_i, n_j) \leftarrow$ de-queue(Q)
10 $\quad$**else if** $S = \{o_c | \forall o_c \in (n_i, n_j) \wedge d_N(o_c.l, q.l) \leq \frac{1-\varepsilon}{\alpha\varepsilon}\}$ **then**
11 $\quad\quad C \leftarrow FindCandidate(S, q, \varepsilon)$
12 $\quad\quad$update H and ε with $o_c \in C$
13 $\quad\quad$**foreach** *unvisited adjacent edge $(n_j, n_j{}')$ of n_j in edge set E* **do**
14 $\quad\quad\quad$en-queue $< (n_j, n_j{}'), d_N(q, n_j{}') >$ to Q
15 return $STkNN(q) \leftarrow H$

Algorithm 2 aims to check whether each object in S_c is a valid answer for an RST*k*NN query. It first initials the parameters (Line 1). The maxheap H is used to store ST*k*NN results. ε is the threshold that is set to the k-th score of objects in H. A priority queue Q is used to maintain all edges to be examined, which is the same as in Algorithm 1. In order to retrieve STkNN, it first has to find the edge (n_i, n_j) on which q lies, then starts to expand the road networks and maintains the edges to be examined (Lines 2–4). The network expansion will terminate when the entire networks are expanded, or there is no valid answer in the unvisited networks any more (Line 5). In other words, the scores of all objects in the unvisited networks are not bigger than ε. The reason is that, when edge (n_i, n_j) is de-queued from Q, suppose the keyword set of n_i is the same as $q.d$, i.e., $\theta(q.d, n_i.d) = 1$, so $\tau(n_i, q) = \frac{1}{1+\alpha.\delta(n_i.l,q.l)}$. $\forall o_i \in (n_i, n_j)$, $d_N(o_i, q) \geq d_N(n_i, q)$, if $\tau(n_i, q) \leq \varepsilon$, we have $\tau(q, o_i) \leq \frac{1}{1+\alpha.\delta(n_i.l,q.l)} \leq \varepsilon$, which indicates the objects in the rest of networks are invalid answers. At the beginning of refinement, the number of objects in H is less than k, we need to find k objects relevant to $q.d$ to initialize H (Line 7), and update the value of ε (Line 8). Once $|H| \geq k$, the *FindCandiate* procedure is employed to retrieve the candidate set C. Meanwhile, H and ε should be updated with $o_c \in C$, and the unexamined adjacent edges are en-queued to Q (Lines 10 - 14)

Note that, not all the objects located on the de-queued edge (n_i, n_j) are needed to be examined, only objects in S are worthy of being verified (Line 10). The reason is that, $\forall o_c \in (n_i, n_j)$, suppose o_c has the maximum textual relevance to $q.d$ (i.e., $\theta = 1$). If $\frac{1}{1+\alpha \cdot d_N(o_c.l,q.l)} \leq \varepsilon$, we can only examine the objects in a limited distance $d_N(o_c.l, q.l) \leq \frac{1-\varepsilon}{\alpha\varepsilon}$.

To better explain how the *FindCandiate* procedure works (Line 11), let's first describe the indexing components depicted in Fig. 1(c) and (d). Specifically, Fig. 1(c) stores the key for each keyword item on an edge, i.e., $\{< ID_{edge}, t_i >, \lambda^-_{t_i}\}$, where ID_{edge} and t_i stand for edge id and a keyword item id, $\lambda^-_{t_i}$ is the maximum impact of t_i among the descriptions of the objects located on edge ID_{edge}, which are mapped to the inverted file component shown in Fig. 1(d). It stores the information $\{< ID_{edge}, t_i >, o_i, d_N(n_i, o_i), \lambda_{t_i,o_i}\}$, where $d_N(n_i, o_i)$ is the distance between o_i and the reference node of the edge, λ_{t_i,o_i} is the impact of the term t_i in the description of o_i. The *FindCandiate* procedure first accesses Fig. 1(c) to compute an upper score τ^- utilizing $\lambda^-_{t_i}$ and the minimum network distance between the edge and $q.l$. Then, if $\tau^- > \varepsilon$, the inverted lists in Fig. 1(d) are accessed. The lists containing the objects are retrieved and the objects whose scores are higher than ε are returned.

For each object $o \in S_c$, if $q \in STkNN(o)$ is verified in Algorithm 2, it means o is an RSTkNN result. Otherwise, as guided by Lemma 3, o can be utilized to prune the objects in the candidate set S_c. The refinement steps are shown in Algorithm 3. Specifically, if $o \notin RSTkNN(q)$, the candidate objects in both S_{sd} and S'_{sd} sets should be pruned (Line 7).

Algorithm 3: Refinement for *RSTkNN(q)* Algorithm.

```
    Input: q, k, S_c
    Output: the result set S_r of an RSTkNN(q)
1   initial S_r = φ
2   foreach o ∈ S_c do
3       if q ∈ STkNN(o) then                          /* Refinement */
4           S_r = S_r ∪ {o}
5       else                                          /* Lemma 3 */
6           search the corresponding sets S_sd and S'_sd for object o
7           S_c = S_c − S_sd − S'_sd
8   return S_r
```

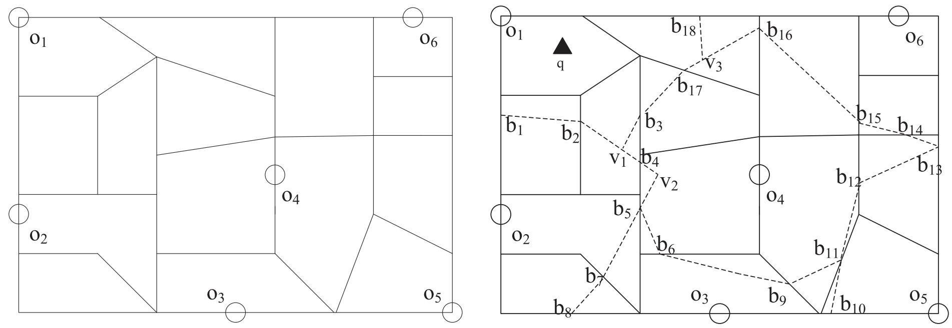
To better understand RSTkNN query processing, let's take Fig. 2 as an example. Suppose $d_N(n_1, q) = 1/2$, $d_N(n_2, q) = 1$, $d_N(n_1, n_0) = 1.2$, $d_N(n_2, n_3) = 2$, $d_N(n_2, n_4) = 5$, $d_N(n_4, n_5) = 1.5$, $d_N(n_5, n_{10}) = 2$, $d_N(n_2, n_7) = 6$, $d_N(n_4, n_8) = 5.5$, $d_N(n_7, n_8) = 6.5$, $d_N(n_7, n_6) = 2.5$, $d_N(n_8, n_9) = 2$. We use the counter $n_i[d].count$ to record the number of objects whose keyword set is d at node n_i. Suppose an edge (n_i, n_j) is examined, a new counter will be constructed at node n_j if there is an object $o \in (n_i, n_j)$ having a different keyword set with that counted at node n_i. If an RST2NN (k=2) query is issued at q, because q is closer to n_1 than n_2, we have $Q = \{< (q, n_1), d_N(q, n_1) >, < (q, n_2), d_N(q, n_2) >\}$. The first de-queued edge is $< (q, n_1), d_N(q, n_1) >$. There is no object located on (q, n_1), hence it is unnecessary to construct a counter at n_1. (n_1, n_0) is the adjacent edge of n_1, and $d_N(n_1, n_0) = 1.2 > 1 = d_N(n_2, q)$, the algorithm en-queues it into Q, i.e., $Q = \{< (q, n_2), d_N(q, n_2) > \pounds\neg < (n_1, n_0), d_N(q, n_0) >\}$. o_1 is located on (q, n_2), once (q, n_2) is de-queued, the counter $n_2[a].count$ for keyword a is constructed, we have $n_2[a].count = 1$ and $S_c = \{o_1\}$. Furthermore, we need to en-queue the adjacent edges of n_2 into Q, i.e., (n_2, n_3), (n_2, n_4) and (n_2, n_7) are en-queued into Q. We have $Q = \{< (n_1, n_0), d_N(q, n_0) >, < (n_2, n_3), d_N(q, n_3) >, < (n_2, n_4), d_N(q, n_4) >, < (n_2, n_7), d_N(q, n_7) >\}$. Then, the first element (n_1, n_0) is de-queued. As o_7 is located on (n_1, n_0), we have $n_0[b, c].count = 1$. Obviously, o_7 belongs to *RST2NN(q)*. The next de-queued edge is (n_2, n_3). There is no object on it. The queue Q continuously de-queues the next element (n_2, n_4). It locates five candidate objects, i.e., $o_2\{a\}$, $o_3\{a, b\}$, $o_4\{a, b\}$, $o_5\{b\}$, $o_6\{a\}$. Due to $o_2.d = \{a\}$, we have $n_4[a].count = 2$. Similarly, we have $n_4[a, b].count = 2$ owning to $o_3.d = o_4.d = \{a, b\}$. As guided by Lemma 2, o_5 and o_6 can be pruned by o_4. Since the counter value for the keyword set $\{a, b\}$ which are the same as $q.d$ reaches to 2, the road networks expansion via SP_{qn_4} can be terminated as guaranteed by Lemma 4. Thus, we have $S_c = \{o_1, o_7, o_2, o_3, o_4\}$.

Subsequently, the priority queue is updated, i.e., $Q = \{< (n_2, n_7), d_N(q, n_7) >\}$. When (n_2, n_7) is de-queued, we find that o_8 is irrelevant to $q.d$ and thus can be ignored. Furthermore, based on $o_9.d = \{a, b, e\}$ and $o_{10}.d = \{a, b, d\}$, when o_{11} is examined, the corresponding set S_{ck}' for o_{11} is : $S_{ck}' = \{o_9, o_{10}\}$, so $|S_{ck}'| \geq k$. Thus, o_{11} can be pruned by Lemma 2. The counters for the new keyword sets are constructed, i.e., $n_7[a, b, c].count = 1$ and $n_7[a, b, d].count = 1$. The evaluation proceeds until $Q = \emptyset$ or the condition listed in Lemma 4 is satisfied.

5.3. Network Voronoi diagram based verification

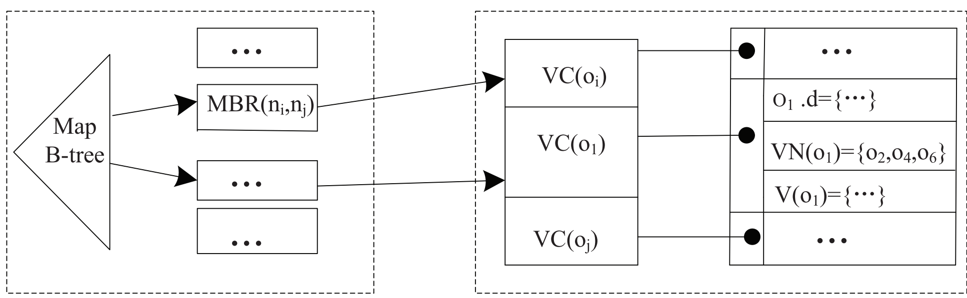
As discussed in the last subsection, when an RSTkNN query is issued, even if k is small, the candidate set S_c is not small. In order to check each candidate object $o_c \in S_c$ whether it is a valid answer point or not, we have to employ Algorithm 2 to verify it. However, the cost of verification process is not low. Because it needs to expand the road networks from each candidate o_c to check whether $q \in STkNN(o_c)$. Obviously, the networks will be expanded $|S_c|$ times. To make the verification process more efficient, Network Voronoi Diagram (NVD) technique is employed in this procedure. NVD is a specialization of Voronoi diagrams, where the locations of objects are restricted to the edges of the graph and the distance between objects is defined as the shortest path connecting them in the network instead of their Euclidean distance.

An NVD divides the network into network Voronoi Cells (VCs). Let $VC(o_i)$ be the network voronoi cell of an object o_i. The network Voronoi Cell $VC(o_i)$ contains all points on edges that are closer to o_i than to any other objects. It is actually a *shortest path tree* generated from o_i [54], and hence o_i is also called the *generator* of $VC(o_i)$. An NVD can be constructed by the methods in literatures [54,55]. Concretely, given a set of objects on the road network, one can construct the NVD by expanding shortest path tree from each object simultaneously until the shortest path trees meet. The meeting points, termed as *border points*, are also on the edges of the road network with the property



(a) Road network (b) Network voronoi diagram

Fig. 6. Road network and NVD.



(a) Mapping component (b)VC component

Fig. 7. NVD indexing.

that the costs (e.g., road network distances) from the meeting point to the two neighboring objects are equal to each other.

Fig. 6(a) presents a road network, its NVD is shown in Fig. 6(b), each network voronoi cell is depicted by the dotted lines, and b_i denotes a border point. Due to $q \in VC(o_1)$, o_1 is thus the nearest neighbor of q. Jing et al. [56] have proven an important property: given the NVD of the dataset O and a query point q on a road network, let $o_1, o_2, \ldots, o_{k-1}$ be the $k-1$ nearest neighbors of q. Then, the k^{th} nearest neighbor of q is among the voronoi neighbors (VN) of $o_1, o_2, \ldots, o_{k-1}$. For example, the 2nd nearest neighbor of q is among the set $VN(o_1) = \{o_2, o_4, o_6\}$. An NVD indexing is shown in Fig. 7, which comprises two components. The first component maps each edge to the corresponding *VC*. Specifically, the record associated with each *VC* in Fig. 7(b) contains the keyword set of the generator, and voronoi neighbors, as well as the vertices of *VC* in a common sequential order.

When an NVD is constructed, for each voronoi cell, we precompute the distance between all the border points of *VC* to its generator as well as the distances of border point-to-border point. As a result, when visiting a new *VC*, we can quickly extend the searched region to the border points without expanding all the internal road segments.

When we start to check a candidate o_c whether $q \in STkNN(o_c)$, if a generator o_i associated with $VC(o_i)$ has no relevance to $o_c.d$, it will be ignored. The verification process starts from $VC(o_c)$, and chooses best object o' from $VN(o_c)$ (i.e., $o' = \arg\max\{\tau(o_i, o_c) | o_i \in VN(o_c)\}$) as the nearest neighbor spatial-textual object of o_c. Thus, o' can be added to the candidate set $STkNN(o_c)$. And then the search expands to the neighboring cells of $VC(o')$. That is, the objects in $VN(o')$ will be examined. The verification continues until no better candidate for $STkNN(o_c)$ exists in unvisited voronoi cells. Concretely, the stopping condition of expansion is set as follows, when we plan to examine the neighboring cells, suppose the keyword sets of the border points that are on the boundary of the visited *VC* match all the keyword set of o_c. If the border point with the highest score is not better than the *kth* object in $STkNN(o_c)$, we can infer all the objects in the unvisited voronoi cells cannot become the valid answers for $STkNN(o_c)$. Therefore, the NVD expansion can be safely terminated.

The verification process employing the NVD method is different from that proposed in Algorithm 2, which is based on the network expansion in a similar way as Dijkstra's approach. The network expansion-based verification is sensitive to the connectivity of road networks and the density of spatial-textual objects. However, NVD-based method can avoid that problem, and the verification only needs to visit from one voronoi cell to another.

6. Experimental evaluation

We systematically evaluate the efficiency and scalability of our methods and the baseline (**BM**). Specifically, our method can be categorized into two classes according to different verifying techniques employed to check each candidate object: (i) **NE-RST*k*NN**, which is based on the network expansion. (ii) **VD-RST*k*NN**, which is based on Network Voronoi Diagram. The experiments are conducted on a modest commodity desktop that is equipped with a Intel-i5 Dual-core 3.4GHz CPU and 8GB RAM. We implement all the algorithms with VC++6.0. The page size is 4KB.

6.1. Experimental setup

The experiments are conducted on both the real-world and synthetic datasets. (i) **Real-world datasets.** The road networks of three US states (DE, ND, LA)[1] are utilized as real-world datasets. For each dataset, the objects with real keyword set are randomly generated in the same way as [28]. Table 1 presents the characteristics of each dataset. (ii) **Synthetic datasets.** Two synthetic datasets are generated

[1] http://www.dis.uniroma1.it/challenge9/data/tiger/

Table 1
Real datasets.

Data	Vertex	Edge	Objects	Keywords
DE	49,109	60,512	0.48M	1,452,288
ND	210,801	260,902	2.0M	6,261,648
LA	413,574	499,254	3.9M	11,982,096

Table 2
Parameter setting.

Parameter	Range	Default
k	10,15,20,25,30	20
No. of query keywords	2,3,4,5,6	4
Query parameter α	0.01,0.1,1,10,100	1
Avg. no. of objects keywords	3,4,5,6,7	4
Avg. no. of objects per edge	4,6,8,10,12	8
Synthetic Dataset1	K1,K2,K3,K4,K5	
Synthetic Dataset2	C1,C2,C3,C4,C5	

by combining *ND* dataset with Twitter[2] messages (tweets) by following the method in [28]. *The first synthetic* dataset is to evaluate the influence of the size of the keyword set of each object on the search performance. We preserve the road networks and the location of the objects of *ND* to create five datasets, i.e., K1, K2, K3, K4, K5. The average number of keywords associated with an object in each set is 3, 4, 5, 6, 7 respectively. *The second synthetic* dataset aims to investigate the impact of object density on the performance of our approaches. We change the number of objects in each edge of the road networks of *ND* to generate five datasets, i.e., C1, C2, C3, C4, C5. The average number of objects on each edge is 4, 6, 8, 10, 12 respectively, and the default number of keywords associated with an object is 3. Table 2 lists parameters used through the experiments. The default values are listed in the last column.

6.2. Experimental results

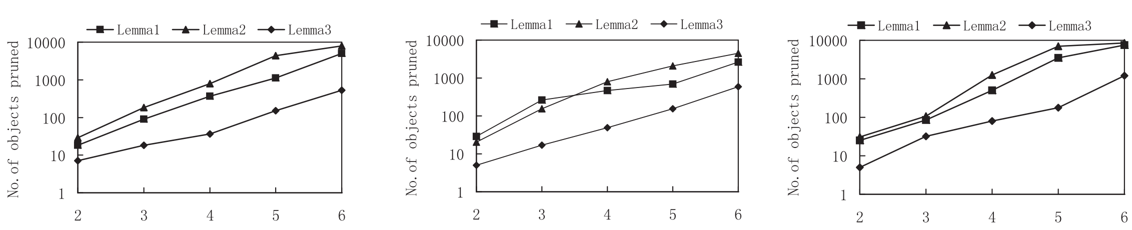
We first evaluate the efficiency of the proposed pruning methods by measuring the number of objects pruned by each pruning method. Next, we evaluate the system performance. In the field of spatial keyword queries, we find that many relevant studies usually adopt *query time* to measure the performance of the proposed algorithms [19,22,33,44,51]. Hence, we use a similar measurement. In our work, we observe that the query times of RST*k*NN queries are affected by *page accesses* and *edges expanded* during the retrieval procedure. Thus, in the experiments, we measure (i) query time; (ii) the number of page accesses by our algorithms during the search; and (iii) the number of edges expanded, i.e., the number of edges expanded before obtaining an RST*k*NN query result. The experiments evaluate the performance of the proposed algorithms under a variety of parameters listed in Table 2. In each experiment, we vary only one parameter and fix other parameters at their default values. 50 random queries are evaluated in every experiment, and their average performance is reported.

6.2.1. Effectiveness of pruning methods

This experiment evaluates the efficiency of three pruning methods (i.e., Lemma 1, 2, 3) proposed in Section 5.1. Fig. 8, where the *y*-axis is in logarithmic scale, shows the number of objects pruned by three lemmas when varying $|q.d|$. As $|q.d|$ increases, the number of objects pruned by the three lemmas increases rapidly. The reason is that, as the number of $q.d$ increases, the number of objects relevant to $q.d$ (i.e., $o.d \cap q.d \neq \emptyset$) increases dramatically. Hence, the size of candidate objects pruned by the three lemmas given in Section 5.1 becomes larger. Furthermore, we observe that Lemma 2 achieves the best performance on both DE and LA datasets. And when $|q.d|$ is small, Lemma 1 outperforms other two lemmas on ND. While the efficiency of Lemma 3 is not so high as others. This is because, our methods first employ both Lemma 1 and Lemma 2 to prune the plenty of relevant objects, and obtain the candidate set S_c. Then Lemma 3 is utilized to further prune objects in S_c using the verified object $o_c \in S_c$. Compared with the number of objects pruned by Lemma 1 and Lemma 2, S_c is smaller. Therefore, Lemma 3 cannot prune so many objects as other two lemmas do.

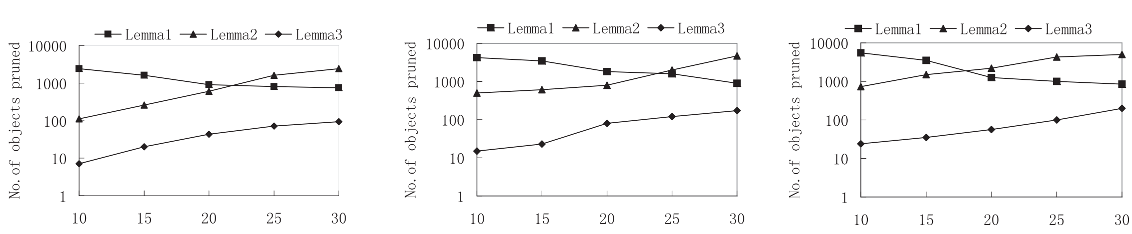
Fig. 9 shows the number of objects pruned by the three lemmas as a function of *k*. Compared with other two lemmas, Lemma 3 solely prune a small number of objects. The reason is similar to the one given in the last experiment. As the value of *k* increases, the number of objects pruned by Lemma 2 is bigger than that of objects pruned

[2] http://twiter.com



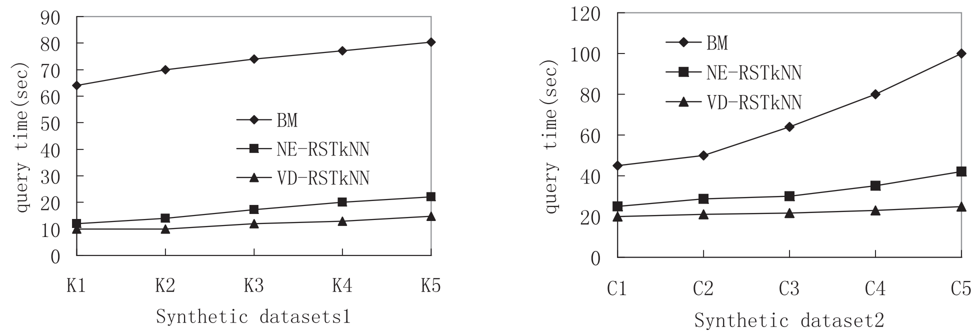
(a) DE (b) ND (c) LA

Fig. 8. Pruning performance of Lemmas vs. $|q.d|$.



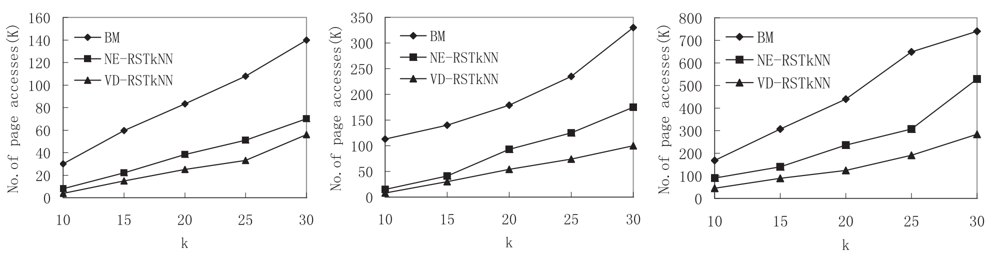
(a) DE (b) ND (c) LA

Fig. 9. Pruning performance of Lemmas vs. *k*.



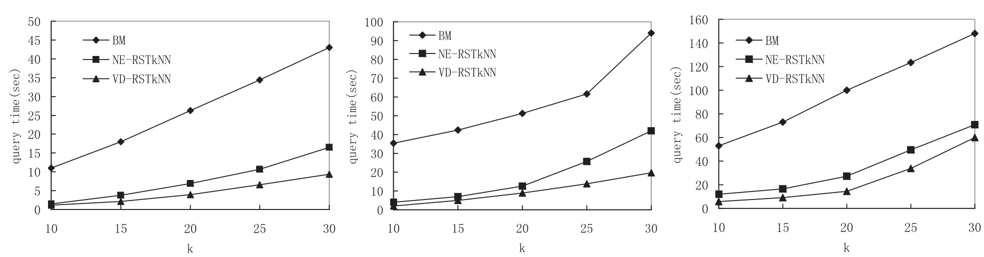
(a) Varying $|o.d|$ (b) Varying object density

Fig. 10. Testing on synthetic datasets.



(a) DE (b) ND (c) LA

Fig. 11. Page accesses vs. k.



(a) DE (b) ND (c) LA

Fig. 12. Query time vs. k.

by Lemma 1. This is because, when the value of k grows larger, the pruning condition given in Lemma 2 (i.e., $|S_{ck}| \geq k$ and $|S'_{ck}| \geq k$) can be easily satisfied. Thus, more candidate objects will be pruned.

6.2.2. Effect of the average number keywords per object

Synthetic Dataset1 is employed to evaluate the performance of our approaches under different average number of keywords per object (i.e., $|o.d|$). In this set of experiments, we set $q.d = 4$ and $k = 20$, and vary $|o.d|$ from 3 to 7. Fig. 10(a) depicts that the query times of all algorithms increase as $|o.d|$ grows. Particularly, VD-RSTkNN outperforms other two algorithms. The main reason is that, when $|o.d|$ grows, the number of the relevant object (i.e., $o.d \cap q.d \neq \emptyset$) increases. Both BM and NE-RSTkNN adopt the network-expanding method to judge the candidate objects, they need more time to do judgement. While VD-RSTkNN adopts NVD-based method to verify the candidate objects, which has a lower cost.

6.2.3. Effect of object density on each edge

Synthetic Dataset2 is utilized to explore the impact of object density on each road segment. The experiment results are depicted in Fig. 10(b). As the density of objects on a road becomes larger, the query time of BM grows significantly. The query time of NE-RSTkNN increases slightly. While the query time of VD-RSTkNN almost remains unchanged. The reason is as follows, as the density of objects is increasing, BM needs more time to filter the unqualified candidate objects. Although NE-RSTkNN could employ the pruning methods to accelerate filtering process, verifying each candidate object in it is still costly. However, NVD-based verifying method can avoid such problem, hence, VD-RSTkNN gains a steady performance.

6.2.4. System evaluation

Since for a query object $q =< q.l, q.d, q.k >$, and the score is calculated as $\tau(o, q) = \frac{\theta(o.d, q.d)}{1+\alpha \cdot \delta(o.l, q.l)}$, in this set of experiments, we try to analyze how system performance is affected by three parameters: the number of returned top results $q.k$ (i.e., k), the number of keywords in queries $q.d$, and the combination ratio α.

Effect of varying *k*. Figs. 11 and12 show the number of page accesses and query times w.r.t k respectively. We vary k from 10 to 30 and fix other parameters at their default values. The results show that the page accesses and the runtime of our algorithms grow with the increase of k. Particularly, VD-RSTkNN performs the best, followed by NE-RSTkNN, BM is the worst. The reason is that, when the value of k increases, the larger number of the candidate objects should be examined, therefore, the required I/O of three algorithms increase. Furthermore, BM has to verify all the relevant spatial keyword object (i.e., $o.d \cap q.d \neq \emptyset$), thus, its runtime increases significantly. Although NE-RSTkNN employs the pruning methods

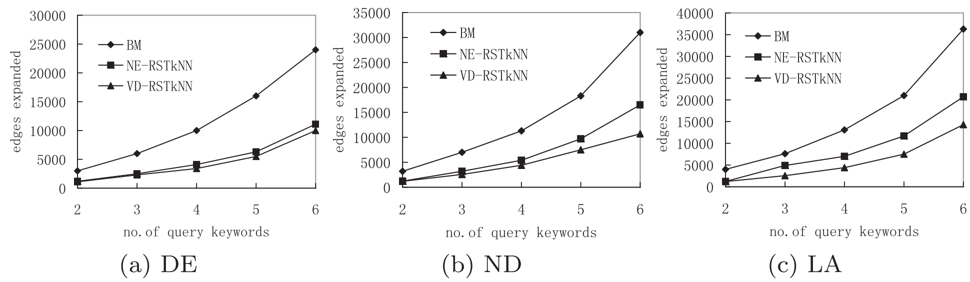


Fig. 13. Edges expanded vs. $|q.d|$.

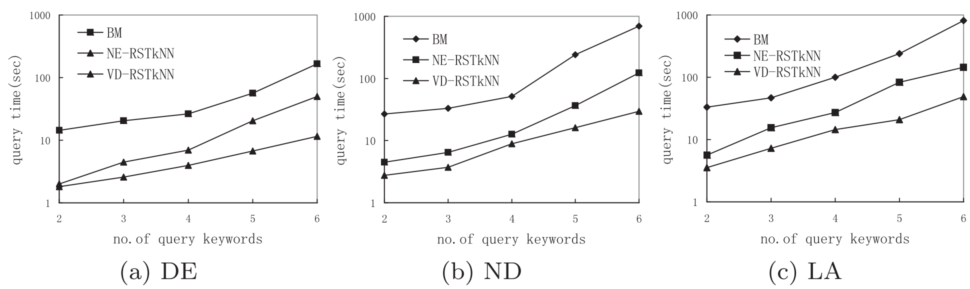


Fig. 14. Query time vs. $|q.d|$.

proposed in Section 5.1 to obtain the candidate object set S_c, it adopts network-expansion method to verify each $o_c \in S_c$, which has to expand the road network multi-times. However, VD-RST*k*NN can avoid these problems, its verification processing only needs to visit from one VC to another. Therefore, VD-RST*k*NN outperforms other two algorithms.

Effect of the number of query keywords $|q.d|$**.** Figs. 13 and14 demonstrate the number of edges expanded and query times w.r.t $|q.d|$ respectively. As $|q.d|$ grows, the number of relevant objects increases remarkably. As expected, our methods need to expand more edges to examine these candidate objects. BM is not as effective as others. The reason is that, both NE-RST*k*NN and VD-RST*k*NN employ the pruning methods and the terminated condition for the network expansion to accelerate query processing. Thus, they need not to expand so many edges as BM does. According to the results depicted in Fig. 13, it is easy to understand the results presented in Fig. 14. Furthermore, VD-RST*k*NN obtains a better performance than NE-RST*k*NN in these two experiments. The reason is that, as discussed in Section 5, once the candidate set S_c is obtained in Algorithm 1, our methods need to verify each candidate object $o_c \in S_c$. NE-RST*k*NN adopts the network-expansion method to verify each o_c, it has to expand network multi-times, whose performance is influenced by the connectivity of road networks and the density of objects located on the road segments. Nevertheless, increasing the density of objects on each road segment has little impact on NVD-based verification process. Hence, VD-RST*k*NN outperforms other two algorithms.

Effect of parameter α**.** Eq. (1) gives how to fuse the network proximity and textual relevance into $\tau(o, q)$. A small value of α gives more preference to the textual description of the objects, while a high value of α gives more preference to the network proximity. To evaluate the impact of α, we vary α from 0.01 to 100 on **ND** data set. Fig. 15 illustrates that as the value of α grows, the query times only decrease slightly, which indicates that the impact of varying α on the query times is not notable.

System performance on synthetic datasets. The experiments shown in above in this subsection are reported based on real-world datasets. We also conduct extensive experiments on synthetic datasets. Due to space limitation, Table 3 only depicts the results on four synthetic datasets (i.e., K1, K5, C1 and C5). We set $q.d = 4$, $k = 20$ and $\alpha = 1$. From the experiment results on K1 and K5, we observe that as $|q.d|$ grows, BM increases its cost significantly, while NE-RST*k*NN and VD-RST*k*NN scale well. Furthermore, according to the results on C1 and C5, as the density of the objects on each road segments grows, BM increases its cost remarkably. Nevertheless, VD-RST*k*NN has steady performance.

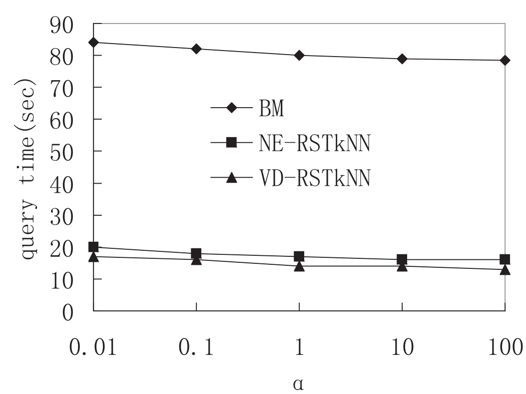


Fig. 15. Query time vs. α.

Table 3
System evaluation on synthetic datasets.

Datasets	Alg.	Page access(K)	Edges expanded	Time(Sec)
K1	BM	225	11,450	64.2
	NE-RSTkNN	90	5080	13.4
	VD-RSTkNN	62	4274	10.1
K5	BM	280	14,675	81.6
	NE-RSTkNN	118	8598	22.2
	VD-RSTkNN	76	5985	14.3
C1	BM	165	15,560	45.4
	NE-RSTkNN	125	8790	25.7
	VD-RSTkNN	90	7468	20.3
C5	BM	320	9878	100
	NE-RSTkNN	170	7056	42.8
	VD-RSTkNN	85	5460	24.5

7. Conclusion

In this paper, we first address the problem of RST*k*NN query on road networks. RST*k*NN query fuses the network proximity and textual relevance together, and it is a score-based spatial keyword query, making it more challenging than Boolean spatial keyword query. Two efficient approaches are developed to support RST*k*NN queries on road networks, in which a road network is modeled by a large graph. The pruning techniques are proposed to prune plenty of unqualified objects at the filter step so that the search space can be minimized efficiently. Increasing the number of the keywords or the density of the spatial-textual object on each road segment has the greater impact on the performance of NE-RST*k*NN, however, they do not present a significant impact on query time of VD-RST*k*NN. An extensive experimental evaluation with both real-world and synthetic datasets has been conducted to verify that VD-RST*k*NN is more efficient than NE-RST*k*NN.

Acknowledgments

This work was substantially supported by the National Natural Science Foundation of China under Grants Nos. 61332001, 61173049, 61300045.

References

[1] F. Korn, S. Muthukrishnan, Influence sets based on reverse nearest neighbor queries, in: ACM SIGMOD Record, 29, ACM, 2000, pp. 201–212.
[2] I. Stanoi, D. Agrawal, A. El Abbadi, Reverse nearest neighbor queries for dynamic databases., in: Proceedings of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, 2000, pp. 44–53.
[3] W. Wu, F. Yang, C.-Y. Chan, K.-L. Tan, Finch: evaluating reverse k-nearest-neighbor queries on location data, Proc. VLDB Endowm. 1 (1) (2008) 1056–1067.
[4] M.A. Cheema, X. Lin, W. Zhang, Y. Zhang, Influence zone: efficiently processing reverse k nearest neighbors queries, in: IEEE 27th International Conference on Data Engineering (ICDE), 2011, IEEE, 2011, pp. 577–588.
[5] Y. Tao, D. Papadias, X. Lian, Reverse knn search in arbitrary dimensionality, in: Proceedings of the Thirtieth international conference on Very large data bases-Volume 30, VLDB Endowment, 2004, pp. 744–755.
[6] E. Achtert, C. Böhm, P. Kröger, P. Kunath, A. Pryakhin, M. Renz, Efficient reverse k-nearest neighbor search in arbitrary metric spaces, in: Proceedings of the 2006 ACM SIGMOD international conference on Management of data, ACM, 2006, pp. 515–526.
[7] Y. Tao, M.L. Yiu, N. Mamoulis, Reverse nearest neighbor search in metric spaces, IEEE Trans. Knowl. Data Eng. 18 (9) (2006) 1239–1252.
[8] M.L. Yiu, D. Papadias, N. Mamoulis, Y. Tao, Reverse nearest neighbors in large graphs, IEEE Trans. Knowl. Data Eng. 18 (4) (2006) 540–553.
[9] M.L. Yiu, N. Mamoulis, Reverse nearest neighbors search in ad hoc subspaces, IEEE Trans. Knowl. Data Eng. 19 (3) (2007) 412–426.
[10] M.A. Cheema, X. Lin, Y. Zhang, W. Wang, W. Zhang, Lazy updates: An efficient technique to continuously monitoring reverse knn, Proc. VLDB Endowm. 2 (1) (2009) 1138–1149.
[11] M.A. Cheema, W. Zhang, X. Lin, Y. Zhang, X. Li, Continuous reverse k nearest neighbors queries in euclidean space and in spatial networks, VLDB J. Int. J. Very Large Data Bases 21 (1) (2012) 69–95.
[12] M.A. Cheema, W. Zhang, X. Lin, Y. Zhang, Efficiently processing snapshot and continuous reverse k nearest neighbors queries, VLDB J. 21 (5) (2012) 703–728.
[13] G. Cong, C.S. Jensen, D. Wu, Efficient retrieval of the top-k most relevant spatial web objects, Proc. VLDB Endowm. 2 (1) (2009) 337–348.
[14] D. Wu, G. Cong, C.S. Jensen, A framework for efficient spatial web object retrieval, VLDB J.ÃíÃ¬ Int. J. Very Large Data Bases 21 (6) (2012) 797–822.
[15] D. Wu, M.L. Yiu, C.S. Jensen, G. Cong, Efficient continuously moving top-k spatial keyword query processing, in: IEEE 27th International Conference on Data Engineering (ICDE), 2011, IEEE, 2011, pp. 541–552.
[16] Z. Li, K.C. Lee, B. Zheng, W.-C. Lee, D.L. Lee, X. Wang, Ir-tree: an efficient index for geographic document search, IEEE Trans. Knowl. Data Eng. 23 (4) (2011) 585–599.
[17] I. De Felipe, V. Hristidis, N. Rishe, Keyword search on spatial databases, in: IEEE 24th International Conference on Data Engineering, 2008., IEEE, 2008, pp. 656–665.
[18] Y. Tao, C. Sheng, Fast nearest neighbor search with keywords, IEEE Trans. Knowl. Data Eng. 26 (4) (2014) 878–888.
[19] C. Zhang, Y. Zhang, W. Zhang, X. Lin, Inverted linear quadtree: efficient top k spatial keyword search, in: Proceedings of the IEEE 29th International Conference on Data Engineering (ICDE), 2013, IEEE, 2013, pp. 901–912.
[20] D. Wu, M.L. Yiu, G. Cong, C.S. Jensen, Joint top-k spatial keyword query processing, IEEE Trans. Knowl. Data Eng. 24 (10) (2012) 1889–1903.
[21] D. Zhang, C.-Y. Chan, K.-L. Tan, Processing spatial keyword query as a top-k aggregation query, in: Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval, ACM, 2014, pp. 355–364.
[22] D. Wu, M.L. Yiu, C.S. Jensen, Moving spatial keyword queries: formulation, methods, and analysis, ACM Trans. Database Syst. (TODS) 38 (1) (2013) 7.
[23] L. Chen, G. Cong, C.S. Jensen, D. Wu, Spatial keyword query processing: an experimental evaluation, in: Proceedings of the VLDB Endowment, 6, VLDB Endowment, 2013, pp. 217–228.
[24] R. Hariharan, B. Hore, C. Li, 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, 2007. SSBDM'07., IEEE, 2007, p. 16.
[25] M. Christoforaki, J. He, C. Dimopoulos, A. Markowetz, T. Suel, Text vs. space: efficient geo-search query processing, in: Proceedings of the 20th ACM International Conference on Information and Knowledge Management, ACM, 2011, pp. 423–432.
[26] A. Cary, O. Wolfson, N. Rishe, Efficient and scalable method for processing top-k spatial boolean queries, in: Scientific and Statistical Database Management, Springer, 2010, pp. 87–95.
[27] X. Cao, G. Cong, C.S. Jensen, Retrieving top-k prestige-based relevant spatial web objects, Proc. VLDB Endowm. 3 (1-2) (2010) 373–384.
[28] J.B. Rocha-Junior, K. Nørvåg, Top-k spatial keyword queries on road networks, in: Proceedings of the 15th International Conference on Extending Database Technology, ACM, 2012, pp. 168–179.
[29] W. Li, J. Guan, S. Zhou, Efficiently evaluating range-constrained spatial keyword query on road networks, in: Proceedings of the Database Systems for Advanced Applications, Springer, 2014, pp. 283–295.
[30] L. Guo, J. Shao, H.H. Aung, K.-L. Tan, Efficient continuous top-k spatial keyword queries on road networks, GeoInformatica 19 (1) (2015) 29–60.
[31] C. Zhang, Y. Zhang, W. Zhang, X. Lin, M.A. Cheema, X. Wang, Diversified spatial keyword search on road networks., in: Proceedings of the EDBT, 2014, pp. 367–378.
[32] Y. Gao, X. Qin, B. Zheng, G. Chen, Efficient reverse top-k boolean spatial keyword queries on road networks, IEEE Trans. Knowl. Data Eng. 27 (5) (2015) 1205–1218.
[33] J. Lu, Y. Lu, G. Cong, Reverse spatial and textual k nearest neighbor search, in: Proceedings of the 2011 ACM SIGMOD International Conference on Management of data, ACM, 2011, pp. 349–360.
[34] M. Safar, D. Ibrahimi, D. Taniar, Voronoi-based reverse nearest neighbor query processing on spatial networks, Multim. Syst. 15 (5) (2009) 295–308.
[35] Q.T. Tran, D. Taniar, M. Safar, Reverse k nearest neighbor and reverse farthest neighbor search on spatial networks, in: Transactions on Large-Scale Data-and Knowledge-Centered Systems I, Springer, 2009, pp. 353–372.
[36] H.-L. Sun, C. Jiang, J.-L. Liu, L. Sun, Continuous reverse nearest neighbor queries on moving objects in road networks, in: Proceedings of the Ninth International Conference on Web-Age Information Management, 2008. WAIM'08., IEEE, 2008, pp. 238–245.
[37] L. Guohui, L. Yanhong, L. Jianjun, L. Shu, Y. Fumin, Continuous reverse k nearest neighbor monitoring on moving objects in road networks, Inf. Syst. 35 (8) (2010) 860–883.
[38] J.B. Rocha-Junior, O. Gkorgkas, S. Jonassen, K. Nørvåg, Efficient processing of top-k spatial keyword queries, in: Advances in Spatial and Temporal Databases, Springer, 2011, pp. 205–222.
[39] D. Zhang, K.-L. Tan, A.K. Tung, Scalable top-k spatial keyword search, in: Proceedings of the 16th International Conference on Extending Database Technology, ACM, 2013, pp. 359–370.
[40] G. Cong, H. Lu, B.C. Ooi, D. Zhang, M. Zhang, Efficient spatial keyword search in trajectory databases (2012) arXiv:1205.2880.
[41] P. Wang, C.V. Ravishankar, On masking topical intent in keyword search, in: IEEE 30th International Conference on Data Engineering (ICDE), 2014, IEEE, 2014, pp. 256–267.
[42] X. Cao, G. Cong, C.S. Jensen, B.C. Ooi, Collective spatial keyword querying, in: Proceedings of the 2011 ACM SIGMOD International Conference on Management of data, ACM, 2011, pp. 373–384.
[43] D. Zhang, B.C. Ooi, A.K. Tung, Locating mapped resources in web 2.0, in: Proceedings of the IEEE 26th International Conference on Data Engineering (ICDE), 2010, IEEE, 2010, pp. 521–532.
[44] D. Zhang, Y.M. Chee, A. Mondal, A.K. Tung, M. Kitsuregawa, Keyword search in spatial databases: towards searching by document, in: Proceedings of the IEEE 25th International Conference on Data Engineering, 2009. ICDE'09., IEEE, 2009, pp. 688–699.
[45] K. Deng, X. Li, J. Lu, X. Zhou, Best keyword cover search, IEEE Trans. Knowl. Data Eng. 27 (1) (2015) 61–73.
[46] L. Chen, X. Lin, H. Hu, C.S. Jensen, J. Xu, Answering why-not questions on spatial keyword top-k queries, in: Proceedings of the IEEE 31st International Conference on Data Engineering (ICDE), 2015, IEEE, 2015, pp. 279–290.
[47] X. Wang, Y. Zhang, W. Zhang, X. Lin, W. Wang, Ap-tree: efficiently support continuous spatial-keyword queries over stream, in: Proceedings of the IEEE 31st International Conference onData Engineering (ICDE), 2015, IEEE, 2015, pp. 1107–1118.
[48] G. Li, J. Feng, J. Xu, Desks: Direction-aware spatial keyword search, in: IEEE 28th International Conference on Data Engineering (ICDE), 2012, IEEE, 2012, pp. 474–485.
[49] K. Zheng, H. Su, B. Zheng, S. Shang, J. Xu, J. Liu, X. Zhou, Interactive top-k spatial keyword queries, in: Proceedings of the IEEE 31st International Conference on Data Engineering (ICDE), 2015, IEEE, 2015, pp. 423–434.
[50] L. Chen, G. Cong, X. Cao, K.-L. Tan, Temporal spatial-keyword top-k publish/subscribe, in: Proceedings of the IEEE 31st International Conference on Data Engineering (ICDE), 2015, IEEE, 2015, pp. 255–266.

[51] B. Zheng, N.J. Yuan, K. Zheng, X. Xie, S. Sadiq, X. Zhou, Approximate keyword search in semantic trajectory database, in: Proceedings of the IEEE 31st International Conference on Data Engineering (ICDE), 2015, IEEE, 2015, pp. 975–986.
[52] G. Salton, C. Buckley, Term-weighting approaches in automatic text retrieval, Inf. Process. Manag. 24 (5) (1988) 513–523.
[53] D. Papadias, J. Zhang, N. Mamoulis, Y. Tao, Query processing in spatial network databases, in: Proceedings of the 29th International Conference on Very Large Data Bases-Volume 29, VLDB Endowment, 2003, pp. 802–813.
[54] A. Okabe, B. Boots, K. Sugihara, S.N. Chiu, Spatial tessellations: concepts and applications of Voronoi diagrams, 501, John Wiley & Sons, 2009.
[55] M. Kolahdouzan, C. Shahabi, Voronoi-based k nearest neighbor search for spatial network databases, in: Proceedings of the Thirtieth International Conference on Very Large dData bases-Volume 30, VLDB Endowment, 2004, pp. 840–851.
[56] Y. Jing, L. Hu, W.-S. Ku, C. Shahabi, Authentication of k nearest neighbor query on road networks, IEEE Trans. Knowl. Data Eng. 26 (6) (2014) 1494–1506.