Top-k Spatial Keyword Query Based on Reachability

Yanhong Li
South-Central Minzu University
Jiayu Ren
South-Central Minzu University
Changyin Luo (changyinluo@ccnu.edu.cn)
Central China Normal University
Yu Feng
South-Central Minzu University
Xiaokun Du
South-Central Minzu University
Jianjun Li
Huazhong University of Science and Technology

Research Article

Keywords: Reachability, Spatial keyword query, Road network, Trajectory data

Posted Date: April 25th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1573042/v1

Top-k Spatial Keyword Query Based on Reachability

Yanhong Li[†], Jiayu Ren[†], Changyin Luo[♯],Yu Feng[†], Xiaokun Du[†], Jianjun Li[§]

[†]*College of Computer Science, South-Central Minzu University, Wuhan, China*
[♯]*School of Computer, Central China Normal University, Wuhan, China*
[§]*School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China*

Abstract

Spatial keyword query (SKQ) is a research hotspot in the field of spatio-temporal databases. The spatial proximity and text similarity between objects and the query are considered during query processing. In order to better meet the personalized needs of users and improve the accuracy and real-time of spatial keyword query results, we propose the Top-k spatial keyword query based on reachability (RSKQ). Such query returns the k best objects that satisfy the reachability and query keyword constraints according to a comprehensive score, which considers the spatial proximity, text similarity, and accessibility between the query and the object. Firstly, an efficient index called SRTR-tree is proposed, which can intelligently organize road network structure information, keywords and location information of spatio-textual objects, and vehicle trajectory information. Moreover, several pruning techniques are designed to prune massive objects irrelevant to the query according to accessibility, spatial proximity, and textual similarity, so as to speed up query processing. Based on the SRTR-tree and pruning strategies, a non-trivial basic algorithm is proposed to process RSKQ queries. In addition, we introduce a more efficient optimized algorithm to improve the efficiency of query processing. Finally, a series of experimental evaluations are carried out to show the effectiveness of our methods.

Keywords: Reachability, Spatial keyword query, Road network, Trajectory data.

[*]Corresponding Author: changyinluo@ccnu.edu.cn

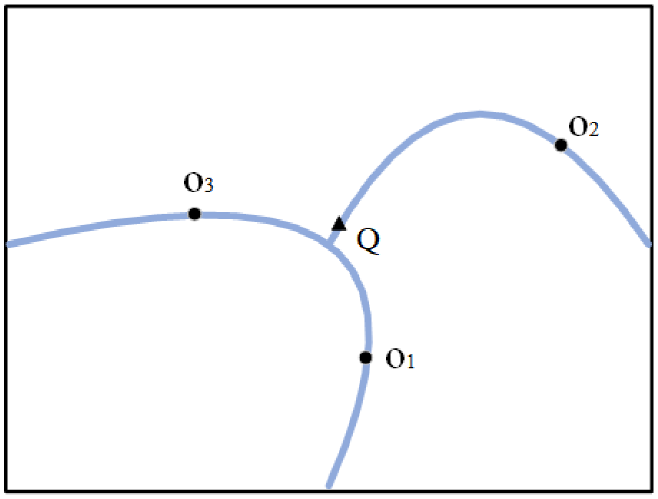


Figure 1: Road network and spatio-textual objects

1. Introduction

With the rapid development of the mobile network and geo-location technology, location-based services have been widely used in daily life, and the network data objects containing spatial and textual information are growing rapidly. As a key technology of location-based services, the spatial keyword query has attracted widespread attention and becomes a research hotspot in the field of database. At present, spatial keyword query processing is extended from Euclidean space to the road network, but the existing research is usually limited to the static distance between the query and spatio-textual objects, does not consider the possibility of users reaching the target object from the query point within a given duration, that is, the accessibility of the target object. However, in real life, the accessibility of spatio-textual objects affects people's satisfaction with query results to a great extent.

As shown in Figure 1, there are three spatio-textual objects o_1, o_2, o_3 and a query point Q in the road network. Assuming that objects o_1 and o_2 meet the query keyword constraints, and the road network distance between query Q and object o_1 is less than that between Q and o_2, o_1 should be better than o_2. However, sometimes, due to traffic conditions or some other reasons, the probability of reaching o_1 from Q is less than that from Q to o_2, so o_1 is not necessarily better than o_2 at this time. Therefore, the accessibility from the query point to objects should be considered in Top-k spatial keyword query processing.

At present, spatial keyword query processing no longer only focuses on the spatial proximity and text similarity between the query and objects, many other factors affecting query results are also considered. For example, direction-aware spatial keyword queries [1, 2] and time-sensitive spatial keyword queries [3–6] are proposed and solved. Meanwhile, the accessibility query in the static road network has been deeply studied. It can predict the accessibility in different periods according to massive historical trajectory data [7, 8], and pay close attention to weather, traffic conditions [9, 10], holidays [11], and other factors [12–14]. Both spatial keyword queries and accessibility queries are constantly explored to better meet the needs of users in the real world, but no one has considered the combination of the two types of queries. In view of the actual impact of object accessibility on spatial keyword query results, we introduce accessibility into s-patial keyword queries for the first time, and propose the Top-k spatial keyword queries (RSKQ) based on accessibility.

Our RSKQ problem presents three main challenges. The first challenge is to reasonably and efficiently organize complex road network data, and a large amount of spatio-textual object information and trajectory information for subsequent calculation. Secondly, many users may initiate queries at the same time. The proposed query processing method should be efficient and intelligent enough to send the information of the matching objects to the relevant query users as soon as possible. Furthermore, taking the reachability factor into account will further increase the difficulty of query processing.

To overcome the above challenges, we first present a novel indexing structure named SRTR-tree. The SRTR-tree uses an R-tree to organize the whole road network, and each tree node maintains the set of keywords contained in all its child nodes or segments and an inverted list to facilitate keyword matching in query processing. For each road segment in leaf nodes, the Temporal-information component is constructed to calculate its reachability. Moreover, three pruning techniques are proposed to prune large amounts of road segments and spatio-textual objects unrelated to the RSKQ query. Based on the SRTR-tree and three pruning strategies, a basic algorithm and an optimized algorithm are designed to process the RSKQ query.

The key contributions of this paper are as follows.

- This paper takes the first step towards studying Top-k spatial keyword queries based on reachability (RSKQ). RSKQ distinguishes itself from the existing spatial keyword queries since it considers the reachability factor under the current time and traffic conditions, which will enhance the effectiveness of query results.

- An efficient index structure called SRTR-tree is proposed. In SRTR-tree, the road network, keywords, location information of spatio-textual objects, and the trajectory information of vehicles are smartly organized. Moreover, several lemmas are proposed to prune huge amounts of irrelevant spatio-textual objects for RSKQ queries.
- On the basis of SRTR-tree and pruning strategies, a non-trivial basic algorithm based on road network expansion and an optimized algorithm that is more efficient for RSKQ query processing are proposed.
- Experiments on a real road network and a trajectory dataset reveal the efficiency of our query processing methods and the associated SRTR-tree index.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 gives basic concepts and problem descriptions. In Section 4, the index structure SRTR-tree is discussed in detail, and Section 5 presents the basic RSKQ query processing method based on SRTR-tree. Section 6 proposes the optimized method and analyses the time complexity of our algorithms. Section 7 shows the experimental study, and finally, Section 8 concludes this paper.

2. Related Work

This paper makes the first attempt to study Top-k spatial keyword queries based on reachability. In this section, we discuss two topics that are closely related to our work, including (1) reachability query processing, (2) spatial keyword query.

2.1. Reachability Query Processing

The conventional reachability query is one of the fundamental graph operations, asking if two nodes are connected in a directed graph. The methods in [15, 16] construct a small-sized but effective index with a low construction cost to solve such reachability queries. Zhou *et al.* [17] introduced a graph reduction method to speed up the reachability calculation, while other works [18, 19] proposed different labeling methods to reduce the index size.

Recently, reachability queries are commonly used to detect whether there is a path from one point to another in an urban road network. Based on the historical trajectory dataset, Wu and Ding *et al.* [7] resegmented the road and matched the trajectory to the map, then found out all reachable road segments by maximum bounding region search and backtracking search, thereby mining spatio-temporal reachable regions. Their method is effective for the single-location spatio-temporal reachability query (S-Query) and the union-of-multi-location spatio-temporal reachability query (U-Query). Later, Ding *et al.* [8] extended their previous ideas by introducing a new multi-source reachability query, namely, the intersection-of-multi-location spatio-temporal reachability Query (I-Query). They utilized an indexing schema composed of a spatio-temporal index and a connection index to answer I-Queries, which is vital for many urban applications, such as location-based recommendation, advertising, and business coverage analysis.

Then, some accessibility related applications have been discussed. Tribby A *et al.* [12] constructed a GIS-based high-resolution spatio-temporal bus network model. The accessibility of public transportion from the starting point to the destination is measured by calculating the total travel time, including the walking time to the bus station, the typical waiting time at the bus station, the travel time on the bus network, and the transfer time between routes. Tang *et al.* [10] proposed a spatio-temporal reachable area calculation scheme called STRC. Two boundary segment selection policies are designed for the time-sensitive applications and the distance-related applications, respectively, which improve the applicability of their scheme in real urban transportation services. Ruan *et al.* [11] proposed a linear integer programming model on the space-time network structure to maximize the system-wide transportation accessibility from different origins to activity locations at special event sites. Tong *et al.* [13] improved reachability by constructing a time-dependent space-time network to maximize the number of accessible activity locations within the travel time budget for road users. When people participate in collective social activities, they will consider the distance and time cost from their current location to the activity location. Ning *et al.* [14] proposed a multi-user location recommendation scheme based on the spatial characteristics of urban road networks and user time requirements.

2.2. Spatial Keyword Query

The typical spatial keyword query consists of the query location, query keywords, the constraint set, the sorting function, and the number of returned results, and returns result objects that meet the constraints and rank high comprehensively. Objects with geo-location and textual properties are called spatio-textual objects [20]. Inverted files are often constructed to organize the textual information of objects. R-trees, grid indexes, and their variants are often used to organize spatial information of spatio-textual objects.

In order to deal with Top-k spatial keyword queries, various index structures are proposed based on R-tree. Zhou *et al.* [20] used R-tree to index the geographic location of all objects. For each leaf node, an inverted file is created to index the text of objects it contains. Cong *et al.* [21] designed IR-tree by combining inverted lists and R-tree. Then, a variant of IR-tree, DIR tree, was designed to merge text similarity and spatial proximity, and improve the index framework. WIR-tree [22] is another variant of IR-tree, which groups objects according to the contained keywords, so that each group shares as few keywords as possible, speeding up the pruning of unrelated object groups. Felipe *et al.* [23] introduced an indexing structure called IR^2-Tree, which combines an R-Tree with superimposed text signatures to answer Top-k spatial keyword queries. At the same time, some other famous indexing techniques were also proposed, such as S2I[24], I^3 [25], and IL-Quadtree [26].

Moreover, some variants of the spatial keyword query have also been discussed. Wu *et al.* [27] studied the efficient processing of continuously moving Top-k spatial keyword queries. They proposed two algorithms for computing safe zones, which can guarantee the correct results at any time. Huang *et al.* [28] proposed an effective method to support mobile Top-k spatial keyword queries. Li *et al.* [29] explored the direction-aware query processing, which returns the k nearest neighbors of the query that meet query direction and keyword constraints. Liu *et al.* [30] studied the generic location-aware rank query over a set of location-aware objects. Considering the time factor, Chen *et al.* [3] defined a new query, namely time-aware collective spatial keyword query, which considers the positional relevance, textual relevance, and temporal relevance between objects and the query at the same time.

3. Preliminaries

This section first clarifies the key terms used in this paper, and then provides a formal definition of the Top-k keyword query based on reachability. Table 1 summarizes the symbols and their definitions frequently used in this paper.

3.1. Basic Concepts

Definition 1. *(Road Network) The road network can be represented by the graph $G = (V, E)$, where E is a set of edges, and V is the set of vertices. A vertex $v \in V$ is a road intersection or an endpoint in the road network and an edge $e \in E$ represents a road segment.*

Definition 2. *(Spatio-textual Object) A spatio-textual object $o \in O$ is located on the edge of the road network, and has its spatial location information $o.l$ (latitude and longitude) and textual description $o.d$. RSKQ queries proposed in this paper return spatio-textual objects as results.*

Definition 3. *(Trajectory) The trajectory is the route of vehicles on the road network. It is composed of a series of spatio-temporal points, each containing the trajectory ID, spatial position (latitude and longitude), timestamp, and other attributes, such as the travel speed and direction.*

Definition 4. *(Trajectory Reachability) Given the query start position $q.s$, start time $q.t$, query duration $q.l$, and the road segment r_i, the trajectory reachability refers to whether any historical trajectory has passed through the road segment r_i from the start position $q.s$ within the given duration $q.l$. If yes, the trajectory accessibility is 1, otherwise 0.*

Definition 5. *(Reachable Road Segment) Given the query start position $q.s$, start time $q.t$, and query duration $q.l$, if the trajectory reachability from $q.s$ to r_i is 1, r_i is a reachable road segment.*

Based on the definition of reachable road segment, we introduce the road segment probability.

Definition 6. *(Road Segment Reachability) The road segment probability describes the percentage of days in the historical trajectory dataset that support the fact that the road segment r_i (the road segment where the object o is located) is reachable from the start road segment r_0 (the road segment where query q is located) within the given duration. The road segment reachability is between 0 and 1.*

Definition 7. *(Spatio-textual Objects Reachability) The reachability of a spatio-textual object o is represented by the reachability of the road segment where o is located.*

3.2. Problem Definition

Definition 8. *(Top-k Spatial Keyword Query based on Reachability (RSKQ)) An RSKQ query q is formally defined as $q = \langle q.s, q.d, q.t, q.l, q.r, k\rangle$, where $q.s$ is the query location, $q.d$ is a set of query keywords, $q.t$ is the start time, $q.l$ is the query duration, $q.r$ is the user-specified reachability and k is the number of target results to be returned. An RSKQ query q returns the top-k spatio-textual objects ranked*

Table 1: Symbols and Definitions.

Notation	Definition
q	a Top-k spatial keyword query based on reachability
$q.s$	the spatial information of query q including latitude and longitude
$q.d$	the keyword set of query q
$q.t$	the start time of query q
$q.l$	the duration of query q
$q.r$	the reachable probability specified by query q, which is the minimum value of the reachability that result objects need to meet
k	the number of target results to be returned by query q
O	a set of spatio-textual objects in a specified road network
o	a spatio-textual object in a road network
$o.l$	the spatial information of o including latitude and longitude
$o.d$	the set of keywords of o
$Prob$	the reachability from query q to object o
Tr	the text similarity between query q and object o
Sr	the spatial proximity between query q and object o

by the comprehensive score satisfying the reachability constraint. In particular, the comprehensive score of object o is determined by the following three factors: (1) the reachability from $q.s$ to $o.l$ during the time interval $[q.t, q.t + q.l]$; (2) the spatial proximity between q and o; (3) the text similarity between q and o.

Definition 9. *(Comprehensive scoring function, $Rank(q, o)$) For the query q and an object o, the scoring function Rank(q,o) returns the comprehensive score that considers the reachability from q to o, the spatial proximity between $q.s$ and $o.l$, and the text similarity between $q.d$ and $o.d$, as follows:*

$$Rank(q, o) = \alpha \times Prob(q, o) + \beta \times Sr(q.s, o.l) + (1 - \alpha - \beta) \times Tr(q.d, o.d) \tag{1}$$

$Prob(q, o)$ is the reachability from q to o in the time interval [$q.t$, $q.t + q.l$]; $Sr(q.s, o.l)$ is the spatial proximity between $q.s$ and $o.l$ and $Tr(q.d, o.d)$ is the text similarity between $q.d$ and $o.d$. α, β, $(1 - \alpha - \beta) \in$ (0,1) represent the user's preference parameters, which are used to measure the importance of reachability, spatial proximity, and text similarity in the scoring function. For example, when

$\alpha = 0$ and $\beta \neq 0$, it means that users only consider the spatial proximity and text similarity between q and o; When $\alpha \neq 0$ and $\beta = 0$, users only focus on whether $o.l$ can be reached from $q.s$ within the query duration and whether $o.d$ and $q.d$ are textual similar, but do not care about the road network distance between $q.s$ and $o.l$.

Definition 10. *(Reachability value $Prob(q, o)$) In this paper, $Prob(q, o)$ describes the reachability from $q.s$ to $o.l$ in interval $[q.t, q.t + q.l]$. Assume that the road segment where q is located is r_0 and the road segment where o is located is r_i, $Prob(q, o)$ can be calculated as follows:*

$$Prob(q, o) = Prob(r_0, r_i) = \frac{\sum_{j=1}^{m} Prob_j(q,o)}{m} \times 100\% \tag{2}$$

Here m is the total number of days covered by the trajectory data, and the reachability $Prob(q, o)$ is the average value of $Prob_j(q, o)$ ($1 \leq j \leq m$), which is defined as follows.

$$Prob_j(q, o) = Prob_j(r_0, r_i) = \begin{cases} \frac{|tr_e|}{|tr_s|} \times 100\% & r_i \neq r_0 \\ (1 - \frac{|tr_e|}{|tr_s|}) \times 100\% & r_i = r_0 \end{cases} \tag{3}$$

tr_s is a set of trajectories that pass through r_0 at the start time $q.t$ and finally pass through r_i, and $| \bullet |$ is the number of elements in the set $\bullet$. If $r_i \neq r_0$, tr_e is a set of trajectories that pass through r_0 at the start time $q.t$ and finally pass through r_i within the interval $[q.t, q.t + q.l]$; otherwise, i.e., $r_i = r_0$, tr_e is a set of trajectories that never leave the road segment r_0 within $[q.t, q.t+q.l]$. In particular, tr_e can be calculated as follows.

$$tr_e = \begin{cases} \sum_{l=0}^{n} tr_l & r_i \neq r_0 \\ \bigcap\limits_{l=0}^{n} tr_l & r_i = r_0 \end{cases} \tag{4}$$

If $r_i \neq r_0$, tr_l is a subset of tr_s, including the trajectories which pass through r_0 at the start time $q.t$ and pass through r_i within the l-th period of the query duration $q.l$. The time interval $q.l$ is divided according to the time granularity of SRTR-tree: if $q.l$ is larger than the time granularity Δt of $q.t$ in the SRTR-tree, $q.l$ is divided into n small periods by the time granularity, and road segment r_i may be passed by within any of the periods; otherwise, the reachability of r_i within $[q.t, q.t + q.l]$ is replaced by its reachability within $[q.t, q.t + \Delta t]$. If $r_i = r_0$, tr_l is a set of trajectories that did not leave r_0 within the $l - th$ period of the query duration $q.l$.

Definition 11. *(Spatial proximity $Sr(q.s, o.l)$) The spatial proximity $Sr(q.s, o.l)$ describes the proximity between query q and object o on the road network distance, which is defined as follows:*

$$Sr(q.s, o.l) = 1 - \frac{d_R(q.s, o.l)}{d_{max}}, \tag{5}$$

where, $d_R(q.s, o.l)$ is the road network distance between $q.s$ and $o.l$, and d_{max} is the maximum road network distance between any two points in the given road network.

Definition 12. *(Text Similarity $Tr(q.d, o.d)$) The text similarity between query q and object o can be calculated by common text similarity measure functions. In this paper, the cosine similarity is used for similarity calculation, as defined below:*

$$Tr(q.d, o.d) = \frac{\sum_{t \in q.d} w_{t,o.d} \cdot w_{t,q.d}}{\sqrt{\sum_{t \in o.d} (w_{t,o.d})^2} \cdot \sqrt{\sum_{t \in q.d} (w_{t,q.d})^2}} \tag{6}$$

The weight of keyword t in the object keyword set $o.d$ is $w_{t,o.d}$=$1 + \ln(f_{t,o.d})$, $f_{t,o.d}$ is the number of occurrences of t in $o.d$. $w_{t,q.d} = \ln(1 + |O| / df_t)$, $|O|$ is the number of spatio-textual objects in O, and df_t is the number of occurrences of keyword t in the keyword set of O. When the query keyword set $q.d$ and the keyword set of object o don't have any same keywords, namely $|q.d \cap o.d| = 0$, their text similarity is 0.

4. System Structure

4.1. Data Pre-processing

Firstly, the original road network and trajectory data are preprocessed to facilitate the subsequent calculation. (1) Road re-segmentation. In this paper, the accessibility of the object is expressed by that of the road segment where it is located, so long road segments will affect the accuracy of query results. Therefore, we re-segment the original roads according to a certain spatial granularity (e.g. 500m). When the original road is re-segmented, if the length of the remaining part is less than half of the given spatial granularity, the remaining part is merged into its adjacent road segment. Otherwise, the remaining part is regarded as a separate road segment. Then, we record the length of the new segments after re-segmentation, and insert intersections to connect new segments generated to maintain the connectivity of original roads. (2) Trajectory-map matching.

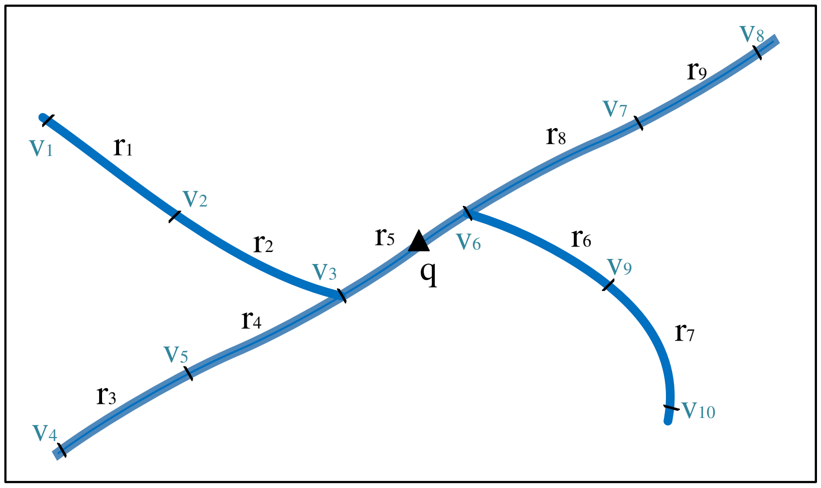


Figure 2: Partial road network structure after road re-segmentation

The raw trajectories are mapped to the re-segmented road network. Firstly, the GPS points of a moving object are mapped to the corresponding segments, and then all segments are connected to form a mapping trajectory. For each segment, the instantaneous speed, vehicle ID, and timestamp values are added as their attributes. A moving object has only one trajectory per day, which is composed of GPS points recorded at different time stamps during the day. Figure 2 gives the partial road network structure after road re-segmentation, which is composed of nine road segments.

4.2. Index Structure

In order to efficiently organize the road network structure information, trajectory information, and the spatial, temporal, textual, and reachability information of spatio-textual objects, a new index structure called SRTR-tree is constructed based on the R-tree.

Firstly, an R-tree is used to divide the road network and its objects, and the partition results are saved. For example, Figure 3 gives the division results of the road network in Figure 2. In particular, the road network (represented by the root node of the R-tree) is first divided into three parts (nodes) E_1, E_2 and E_3, which will be the children of the root node and further subdivided. The above spatial division is repeated until the number of road segments in each node does not exceed a predefined threshold δ, and the spatio-textual objects are organized

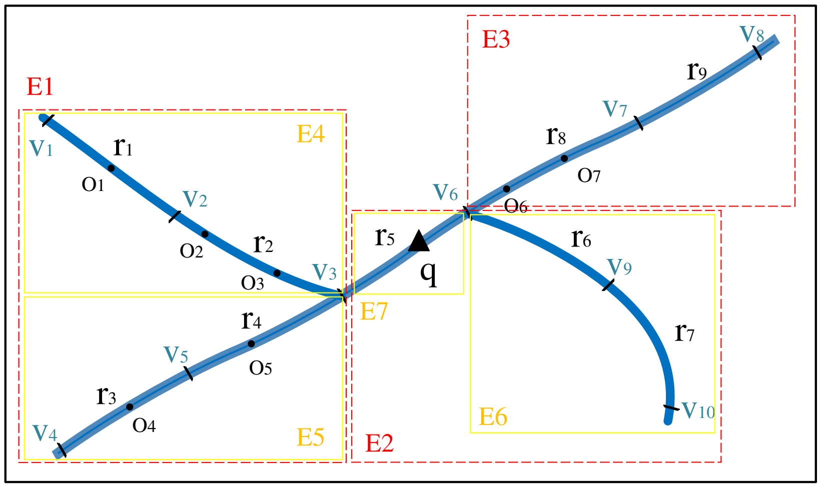


Figure 3: Intersections and spatio-textual objects in the road network

according to their segments. At the same time, each node of the SRTR-tree (except the root node) has a pointer to its parent node and several pointers to its contained child nodes or segments. Each object is located on a segment and has location information and keyword information used to calculate the spatial proximity and text similarity between the query and the object. The location information of an object is represented by its distance to the vertex of the road segment where it is located.

For the root node of the SRTR-tree, an Adjacency component is maintained, which records the adjacent road segment information of each road segment in the road network. For each node of the SRTR-tree, an Inverted file based on keywords is constructed to accelerate the selection of areas or segments related to the query. Each leaf node of the SRTR-tree also stores the information about the road segments it contains and all spatio-textual objects on those segments. Specifically, a segment list is linked to each leaf node, and a Temporal-information component is connected to each road segment. Figure 4 shows the overall structure of the SRTR-tree, and Figures 5, 6, and 7 show the structures of Adjacent components, Inverted files, and Temporal-information components, respectively.

Adjacency component The adjacency component records the adjacent segments and related intersections of each segment in the road network to find which segments the path from the query q to a specific spatio-textual object will pass through. For example, we can retrieve the entries corresponding to segment r_2 in

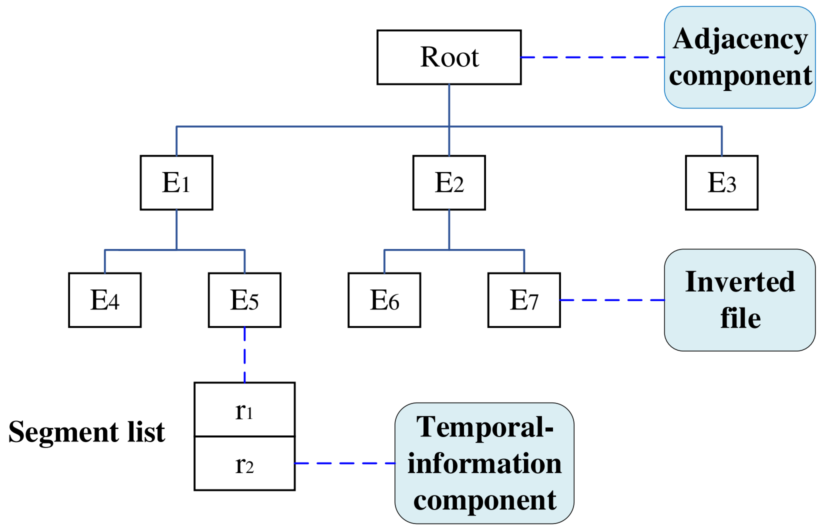
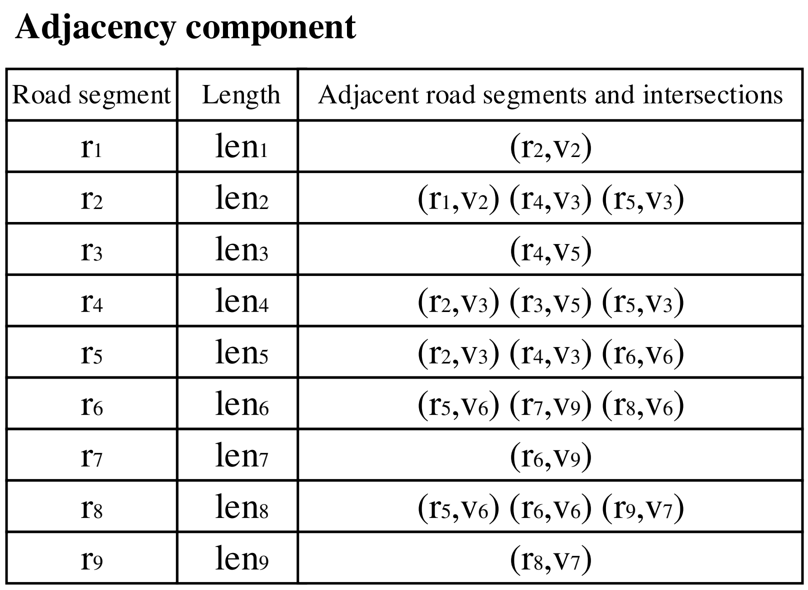


Figure 4: SRTR-tree

the Adjacent component to obtain its adjacent segment r_1, r_4, and r_5, and nodes v_2, v_3, and v_3, which are the intersections of these three adjacent segments and segment r_2, respectively.

Inverted file The Inverted file is connected to the node of SRTR-tree, and for each keyword contained in the node, it records the ID of the child node containing the keyword. For example, when the user launches a query with “shopping” as the query keyword, we can prune the region E_4 according to the inverted file of E_1, because the child node containing the keyword “shopping” in E_1 is only E_5.

Temporal-information component Temporal information has two dimensions - date and time of day. The date information corresponds to the date of the historical trajectory data, and the time of a day is firstly divided into “morning rush period” (7:00-9:00), “evening rush period” (17:00-20:00), and “flat peak period” (other periods except the first two periods) according to the traffic situation. Secondly, the two hours in the morning rush period and three hours in the evening rush period are divided into several 10-minute intervals, and the flat peak period is divided into several 30-minute intervals. This is because in the morning-evening rush period of every day, people plan the time more accurately than other periods, and have higher requirements for the reachability of objects on the road network. The smaller the time granularity, the more accurate the calculation result is. Finally, according to the principle of “time first, date late”, the IDs of all trajectories



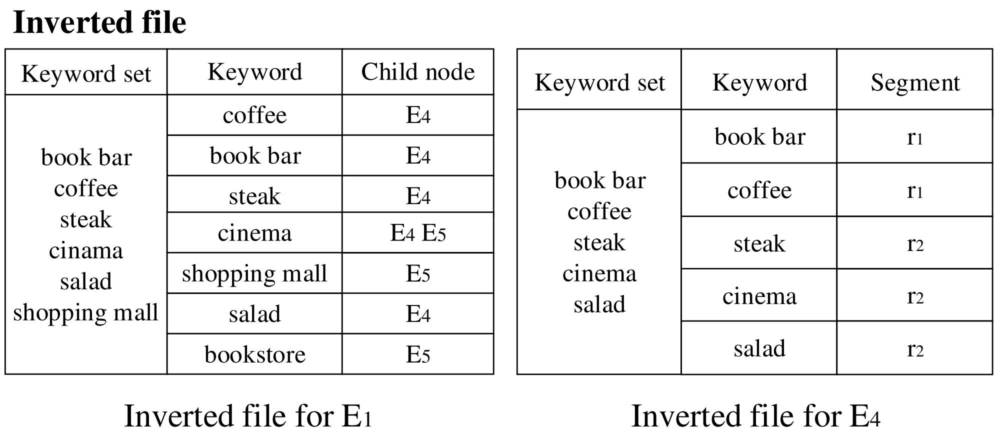
Adjacency component

Road segment	Length	Adjacent road segments and intersections
r_1	len_1	(r_2,v_2)
r_2	len_2	(r_1,v_2) (r_4,v_3) (r_5,v_3)
r_3	len_3	(r_4,v_5)
r_4	len_4	(r_2,v_3) (r_3,v_5) (r_5,v_3)
r_5	len_5	(r_2,v_3) (r_4,v_3) (r_6,v_6)
r_6	len_6	(r_5,v_6) (r_7,v_9) (r_8,v_6)
r_7	len_7	(r_6,v_9)
r_8	len_8	(r_5,v_6) (r_6,v_6) (r_9,v_7)
r_9	len_9	(r_8,v_7)

Figure 5: Adjacency component

passing through a certain road segment will be stored in the corresponding date and time information table. The specific information of the trajectory includes the road segment it passes through, travel speed, and time stamps.

The temporal information is mainly used to calculate the reachability of the road segment within an interval. We take the calculation of reachability from road segment r_a to r_b as an example. Firstly, we locate the corresponding time period of r_a's time table in the SRTR-tree according to the start segment r_a and the start time $q.t$, and then explore the trajectory information set of the time period to find the set SR_d of trajectories which will pass through r_b later in the day for each day d. Similarly, the trajectory set ER_d passing through road segment r_b during the period $[q.t, q.t + q.l]$ of each day d can be found. Finally, by calculating the value of $|SR_d \cap ER_d| \,/\, |SR_d|$, the reachability of day d from road segments r_a to r_b is obtained. The final result of accessibility is the average of the calculated daily accessibility.


Inverted file

Keyword set	Keyword	Child node
book bar coffee steak cinama salad shopping mall	coffee	E4
	book bar	E4
	steak	E4
	cinema	E4 E5
	shopping mall	E5
	salad	E4
	bookstore	E5

Inverted file for E1

Keyword set	Keyword	Segment
book bar coffee steak cinema salad	book bar	r1
	coffee	r1
	steak	r2
	cinema	r2
	salad	r2

Inverted file for E4

Figure 6: Inverted file

5. Query Processing Algorithm

5.1. The Pruning Strategy

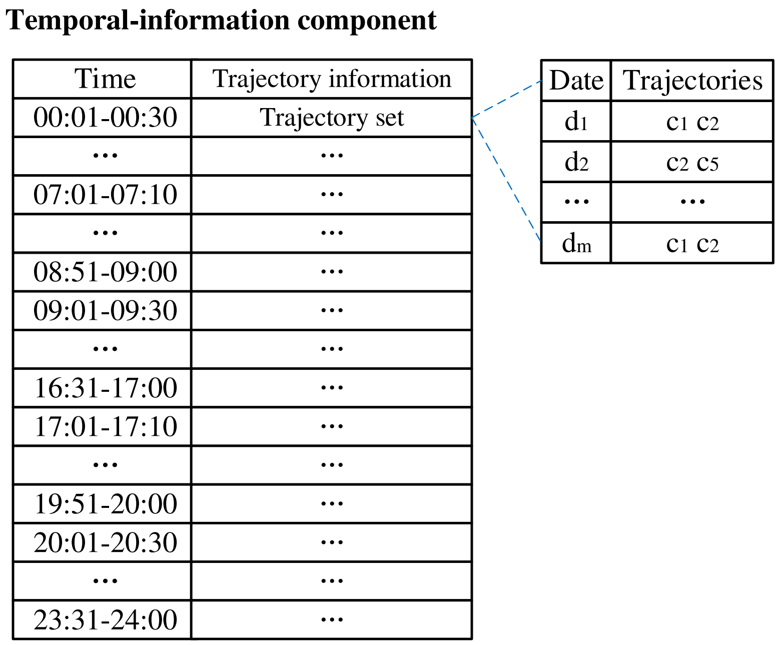
Lemma 1. *Given a query q= $\langle q.s, q.d, q.t, q.l, q.r, k\rangle$ and a segment R, R can be safely pruned if $Prob(q, R) < q.r$, where $Prob(q, R)$ is the reachability from $q.s$ to R within interval $[q.t, q.t + q.l]$. In addition, the adjacent segment R' of R that has not been retrieved can also be safely pruned.*

Proof. According to the SRTR-tree, for any object o on segment R, $Prob(q, o) = Prob(q, R)$. If $Prob(q, R) < q.r$, then $Prob(q, o) < q.r$, thus all objects on segment R cannot meet the reachability requirement. As a result, R can be safely pruned.

Because R' is not encountered during the road network exploration from $q.s$ to R, the distance between R' and the query q is farther than that between R and q, so $Prob(q, R') \leq Prob(q, R) < q.r$. As a result, R' can also be safely pruned. □

Lemma 2. *Given a query q=$\langle q.s, q.d, q.t, q.l, q.r, k\rangle$ and a region E (or a segment R), if $q.d \cap E.d = \varnothing$ (or $q.d \cap R.d = \varnothing$), region E (or segment R) can be safely pruned.*

Proof. For any object o in region E (or segment R), there is $o.d \subseteq E.d$ (or $o.d \subseteq R.d$). If $E.d \cap q.d = \varnothing$ (or $R.d \cap q.d = \varnothing$), then $o.d \cap q.d = \varnothing$, which means that object o is irrelevant to query keywords and can not be a result object. □



Temporal-information component

Time	Trajectory information
00:01-00:30	Trajectory set
···	···
07:01-07:10	···
···	···
08:51-09:00	···
09:01-09:30	···
···	···
16:31-17:00	···
17:01-17:10	···
···	···
19:51-20:00	···
20:01-20:30	···
···	···
23:31-24:00	···

Date	Trajectories
d_1	c_1 c_2
d_2	c_2 c_5
···	···
d_m	c_1 c_2

Figure 7: Temporal-information component

Lemma 3. *Given a query* q=$\langle q.s, q.d, q.t, q.l, q.r, k\rangle$ *and a segment* R*, if* $d_R(q.s, R) > d_{max} \cdot [1 - Rank(q, o_k)] / \beta$*, segment* R *can be safely pruned, where* $d_R(q.s, R)$ *is the road network distance between* q *and* R*,* d_{max} *is the maximum distance between any two points in the given road network,* o_k *is the current* k^{th} *result object, and* $Rank(q, o_k)$ *is the comprehensive score of object* o_k*. In addition, the adjacent segment* R' *of* R *that has not been retrieved can also be safely pruned.*

Proof. If $d_R(q.s, R) > d_{max} \cdot [1 - Rank(q, o_k)] / \beta$, for any object o in segment R, $d_R(q.s, o) > d_{max} \cdot [1 - Rank(q, o_k)] / \beta$. Through transformation, we have $Rank(q, o)$=$1 - \beta \cdot d_R(q.s, o) / d_{max} < 1 - \beta \cdot (d_{max} \cdot [1 - Rank(q, o_k)] / \beta) / d_{max}$ = $Rank(q, o_k)$. Thus o can not be a result object and segment R can be safely pruned. Similar to the proof of the second half of Lemma 1, the adjacent segment R' of R that has not been retrieved can also be safely pruned.

□

Lemma 4. *Given a query* q=$\langle q.s, q.d, q.t, q.l, q.r, k\rangle$ *and a segment* R*, if* $d_R(q.s, R) > v_{max} \times q.l$*, the road segment* R *and the adjacent segments of* R *that have not been* *retrieved can be safely pruned, where v_{max} is the upper limit of the driving speed for all road segments.*

Proof. $v_{max} \times q.l$ is the farthest distance that can be traveled from the start location $q.s$ within the duration $q.l$. If the road network distance between q and R is greater than $v_{max} \times q.l$, the road segment R and its adjacent segments that have not been retrieved are all unreachable, thus they can be safely pruned. □

5.2. Basic Algorithm of RSKQ Query Processing

Algorithm 1 gives an SRTR-tree based method for calculating the reachability from the query q to the object o within the duration $q.l$ starting from $q.t$. As mentioned earlier, the reachability from q to o is represented by that from the start road segment r_0 where the query q is located to the segment r_b where the object o is located. The SRTR-tree based reachability calculation method takes as inputs an SRTR-tree index, a $RSKQ$ query $q = \langle q.s, q.d, q.t, q.l, q.r, k\rangle$ whose start road segment is r_0, and the segment r_b, and outputs $probability$, the reachability value which is between 0 to 1.

Firstly, variable $probability$ is initialized to $-\infty$, and trajectory sets tr_s and tr_e are initialized to empty to keep the IDs of trajectories that pass through the start road segment r_0 at $q.t$ and finally pass through r_b, and the IDs of trajectories that pass through r_0 at the start time $q.t$ and finally pass through r_b within the interval $[q.t, q.t + q.l]$, respectively (Line 4). m is the total number of days covered by the trajectory data, and Δt is the time granularity of the time interval containing the start time $q.t$ in the SRTR-tree.

Lines 5-22 give the processing steps for calculating reachability from the start road segment r_0 to the segment r_b. In particular, we obtain the reachability for each day d ($1 \leq d \leq m$), and take the average value as the final result.

For each day d, the trajectories in the Temporal-information component of SRTR-tree are processed orderly as follows:

(1) Constructing set tr_s. If $r_0 \neq r_b$, find the trajectory list of the time period containing $q.t$ in the Temporal-information component. Then, by checking the specific information of trajectories, the IDs of the trajectories that will pass through r_b are added to set tr_s (Lines 6-7). Otherwise, add the IDs of all trajectories in this period to set tr_s (Lines 8-9).

(2) Constructing set tr_e. If $q.l > \Delta t$, $q.l$ is divided into several small time intervals $\Delta t_1, \Delta t_2, ..., \Delta t_n$ based on the time granularity of the Temporal-information component (Lines 10-11). If query q and object o are on different road segments, i.e., $r_0 \neq r_b$, the IDs of the trajectories in set tr_s that pass through r_b within any

Algorithm 1: Reachability calculation $PROB(q, r_b)$

```
1  Input: the query $q$=$\langle q.s, q.d, q.t, q.l, q.r, k\rangle$ (assume that the start road segment of
   $q$ is $r_0$), the SRTR-tree, and the road segment $r_b$;
2  Output: $probability$, the value of reachability from $r_0$ to $r_b$ (also the reachability
   from the query $q$ to the object $o$) within $q.l$ starting from $q.t$;
3  begin
4      float $probability \leftarrow -\infty$, $tr_s \leftarrow \varnothing$, $tr_e \leftarrow \varnothing$;
5      for int $d \leftarrow 1$ to $m$ do
6          if $r_0 \neq r_b$ then
7              Add IDs of trajectories that pass through $r_0$ at the start time $q.t$ and
               finally pass through $r_b$ into $tr_s$ according to the Temporal-information
               component in the SRTR-tree;
8          else
9              Add IDs of trajectories that pass through $r_0$ at $q.t$ into set $tr_s$
10         if $q.l > \Delta t$ then
11             $q.l$ is divided into several small time intervals $\Delta t_1, \Delta t_2 ... \Delta t_n$;
12             if $r_0 \neq r_b$ then
13                 $tr_e = \sum_{c=1}^{n} tr_c$ ($tr_c \subseteq tr_s$);
14             else
15                 $tr_e = \bigcap_{c=1}^{n} tr_c$ ($tr_c \subseteq tr_s$);
16         else
17             Add IDs of trajectories in $tr_s$ that pass through $r_b$ within
               $[q.t, q.t + \Delta t]$ into set $tr_e$ ;
18         if $r_0 \neq r_b$ then
19             $Prob_d(r_0, r_b) = \frac{|tr_e|}{|tr_s|} \times 100\%$;
20         else
21             $Prob_d(r_0, r_b) = (1 - \frac{|tr_e|}{|tr_s|}) \times 100\%$;
22     $probability = Prob(r_0, r_b) = \frac{\sum_{d=1}^{m} Prob_d(r_0, r_b)}{m} \times 100\%$;
23     Return $probability$;
```

of those small time intervals are added to tr_e (Lines 12-13). Otherwise, set tr_e consists of the IDs of the trajectories that do not leave the start segment r_0 within the duration $q.l$ (Lines 14-15). If $q.l \leq \Delta t$, set tr_e keeps the IDs of the trajectories in tr_s that pass through r_b within the time period $[q.t, q.t + \Delta t]$ (Lines 16-17).

(3) Calculating $Prob_d(r_0, r_b)$ for day d. If r_0 and r_b are not the same road segment, $Prob_d(r_0, r_b) = |tr_e| \,/\, |tr_s| \times 100\%$ according to Eqn.(3) (Lines 18-19). Otherwise, $Prob_d(r_0, r_b) = (1 - |tr_e| \,/\, |tr_s|) \times 100\%$ (Lines 20-21), because intuitively, the more trajectories that never leave the road segment r_0 within $[q.t, q.t + q.l]$, the lower the reachability of objects on other segments except r_0.

Finally, the average value of daily accessibility will be calculated and returned as $probability$ (Lines 22-23).

Algorithm 2 illustrates the detailed implementation of handing $RSKQ$ queries based on the SRTR-tree (SRTR-tree method for short). It takes as inputs the SRTR-tree index, a $RSKQ$ query $q = \langle q.s, q.d, q.t, q.l, q.r, k \rangle$, and outputs the result set R, where the number of result objects is between 0 to k. This approach adopts the idea of road network expansion. Starting from the segment r_0 where q is located, it explores the adjacent road segments in ascending order of road network distance until k result objects are found or all possible road segments are explored.

Set R, queue NR, and pointer $LNode$, initialized to empty, are used to keep at most k result objects, candidate road segments not visited, and the SRTR-tree leaf node (i.e., the segment in the road network) being accessed, respectively. In addition, a float $Rank(q, o_k)$, which is initialized to $-\infty$, is used to keep the comprehensive score of the current k^{th} result object in set R (Line 4).

Firstly, it locates the start road segment r_0 where the query q is located according to its location information $q.s$ (line 5). Lines 6-14 detail the processing of spatio-textual objects on the start road segment r_0. If $r_0.d \cap q.d \neq \varnothing$, there is at least one object related to query keywords on segment r_0, and the reachability $Prob(q, r_0)$ of the start segment r_0 is calculated by calling algorithm 1. If $Prob(q, r_0) < q.r$, we believe that all segments on the road network are unreachable, so there are no result objects for the query q, and the query is terminated (Lines 7-8). Otherwise, for each object o on segment r_0, if it meets the query keyword requirement, its comprehensive score will be calculated. If the score is greater than that of the current k^{th} result object in R , it will be added to the result set R. The value of $Rank(q, o_k)$ is updated accordingly (Lines 10-12). When the number of objects in R reaches k, the query terminates and k result objects are returned (Lines 13-14).

Lines 15-26 give the processing steps for other road segments except for r_0 and objects on them. Firstly, the adjacent road segments of r_0 are pushed into queue NR in ascending order of road network distance (Line 15). When queue NR is not empty, its elements are processed orderly as follows: (1) the head element of NR is popped out and pointed by $LNode$ (Line 17). (2) N-

Algorithm 2: Basic RSKQ Query Algorithm Based on Network Expansion

```
1  Input: the $RSKQ$ query $q$=$\langle q.s, q.d, q.t, q.l, q.r, k\rangle$, the SRTR-tree;
2  Output: the result set $R$;
3  begin
4      $R \leftarrow \varnothing$, $NR \leftarrow \varnothing$, $LNode \leftarrow \varnothing$, $Rank(q, o_k) \leftarrow -\infty$;
5      Locate the start road segment $r_0$ where $q$ is located;
6      if $r_0.d \cap q.d \neq \varnothing$ then
7          if $Prob(q, r_0) = PROB(q, r_0) < q.l$ then
8              No result object exists, and the query terminates;
9          else
10             for each object $o$ on the segment $r_0$ do
11                 if $o.d \cap q.d \neq \varnothing$ && $Rank(q, o) > Rank(q, o_k)$ then
12                     Put $o$ into $R$, and update the value of $Rank(q, o_k)$
                       accordingly;
13                 if $|R| = k$ then
14                     The query terminates, and the result set $R$ returns;
15     Push adjacent road segments of $r_0$ into $NR$ in ascending order of road
       network distance;
16     while $NR \neq \varnothing$ do
17         $LNode$ = Dequeue($NR$);
18         if $d_R(q.s, LNode) < d_{max}\left[1 - Rank(q, o_k)\right]/\beta$ &&
           $d_R(q.s, LNode) < v_{max} \times q.l$ &&
           $Prob(q, LNode) = PROB(q, LNode) > q.r$ then
19             Enqueue un-visited adjacent road segments of $LNode$ in ascending
               order of road network distance;
20             if $LNode.d \cap q.d \neq \varnothing$ then
21                 for each object $o'$ on $LNode$ do
22                     if $o'.d \cap q.d \neq \varnothing$ && $Rank(q, o') > Rank(q, o_k)$ then
23                         Put $o'$ into $R$, and update $Rank(q, o_k)$ accordingly;
24                     if $|R| = k$ then
25                         The query terminates, and the result set $R$ returns;
26     The result set $R$ returns;
```

ode $LNode$ may contain result objects if it meets the following requirements: i) $d_R(q.s, LNode) < d_{max}\left[1 - Rank(q, o_{k-th})\right]/\beta$ and $d_R(q.s, LNode) < v_{max} \times q.l$, where $d_R(q.s, LNode)$ is the road network distance between q and $LNode$; ii) $Prob(q, LNode)$, the reachability from the query q to $LNode$ calculated by algorithm 1, is greater than $q.r$ (Lines 17-18). Thus, the un-visited adjacent road segments of $LNode$ are also candidate road segments and are pushed into NR in ascending order of road network distance (Line 19). (3) If $LNode.d \cap q.d \neq \varnothing$, there is at least one object on node $LNode$ contains any query keyword and needs to be processed (Line 20). Therefore, for each object o' on the qualified $LNode$, if $o'.d \cap q.d \neq \varnothing$ and $Rank(q, o') > Rank(q, o_k)$, we put the object o' into R and update $Rank(q, o_k)$ accordingly (Lines 21-23). If there are k objects in the result set R, the query terminates (Lines 24-25). After retrieving and processing all possible road segments, the result set R is obtained and returned (Line 26).

6. The Optimized Approach

6.1. Optimized Algorithm of Query Processing

The basic algorithm discussed in Section 5 can return the result set R, which contains the Top-k result objects of the query. However, Algorithm 2 is based on network expansion and will incur a lot of time overhead if the qualified objects are distributed in a large area of the underlying road network. To this end, we will propose an optimized query processing algorithm, which can prune irrelevant regions and objects by using the SRTR-tree and corresponding pruning strategies, so as to greatly improve the efficiency of query processing. Algorithm 3 gives detail steps of the optimized $RSKQ$ query processing method based on SRTR-tree. Its input parameters and their meaning are similar to that of Algorithm 2, and the result set R is output.

Set R, queue NR, and pointer $TNode$ are initialized to empty, to keep result objects, candidate areas and road segments on the road network, and an area or a road segment being processed, respectively. $Rank(q, o_k)$ is initialized to $-\infty$ (Line 4).

Firstly, the start road segment r_0 where the query is located is identified (Line 5). The processing steps for objects on the start road segment r_0 are the same as Lines 6-14 of Algorithm 2. After that, if $|R| < k$, other areas and road segments on the road network will be processed.

Specifically, the node containing the start segment r_0 is pushed into the queue NR (Line 7). When queue NR is not empty, its elements are processed orderly as follows: (1) The head element of NR is popped out and pointed by $TNode$

Algorithm 3: Optimized RSKQ Query Algorithm Based on SRTR-tree

```
1  Input: the $RSKQ$ query $q$ = $\langle q.s, q.d, q.t, q.l, q.r, k\rangle$, the SRTR-tree;
2  Output: the result set $R$;
3  begin
4      $R \leftarrow \varnothing$, $NR \leftarrow \varnothing$, $TNode \leftarrow \varnothing$, $Rank(q, o_k) \leftarrow -\infty$;
5      Locate the start road segment $r_0$ where the query $q$ is located according to $q.s$;
6      Same as lines 6-14 of Algorithm 2 ;
7      Push the node that $r_0$ is contained into queue $NR$;
8      while $NR \neq \varnothing$ do
9          $TNode$ = Dequeue($NR$);
10         if $TNode.d \cap q.d \neq \varnothing$ && $d_R(q.s, TNode) < v_{max} \times q.l$ &&
           $d_R(q.s, TNode) < d_{max}\left[1 - Rank(q, o_k)\right]/\beta$ then
11             if $TNode$ is a non-leaf node then
12                 Push un-visited child nodes of $TNode$ into queue $NR$;
13                 if $TNode$ is not the root node then
14                     Push parent node of $TNode$ into queue $NR$;
15             else
16                 for each un-visited segment $r$ in $TNode$ do
17                     if $r.d \cap q.d \neq \varnothing$ && $Prob(q, r) > q.r$ then
18                         for each object $o'$ on $r$ do
19                             if $o'.d \cap q.d \neq \varnothing$ && $Rank(q, o') > Rank(q, o_k)$
                               then
20                                 Put $o'$ into $R$, and update $Rank(q, o_k)$
                                   accordingly;
21                                 if $|R| = k$ then
22                                     The query terminates, and the result set $R$
                                       returns;
23         else
24             if $TNode$ is the root node then
25                 The query terminates;
26             else
27                 Push parent node of $TNode$ into queue $NR$;
28     The result set $R$ returns;
```

(Line 9); (2) We first determine whether $TNode$ meets the keyword and distance constraint, which means that the area covered by $TNode$ may contain result objects; (3) When the spatial and textual requirements are satisfied, i) if $TNode$ is a non-leaf node, its unvisited child nodes are pushed into queue NR for further consideration (Lines 10-12). Then, we continue to determine whether $TNode$ is the root node of SRTR-tree. If not, its parent node is pushed into queue NR (Lines 13-14); ii) When $TNode$ is a leaf node, the pruning strategies are used to determine whether the segments it contains are related to query q. For each segment r, if it meets the query keyword and accessibility requirements, it may contain result objects that need further processing (Line 16-17). For each object o' on r, if $o'.d \cap q.d \neq \varnothing$ and $Rank(q, o') > Rank(q, o_k)$, add o' into the result set R and update the value of $Rank(q, o_k)$ accordingly (Lines 18-20). If there are k objects in the result set R, the query terminates (Lines 21-22). (4) When the keywords of $TNode$ are irrelevant to $q.d$ or the distance between $TNode$ and q is too far, all objects on $TNode$ are not considered. If $TNode$ is the root node, the query will be terminated (Lines 24-25); otherwise, the parent node of $TNode$ will be pushed into queue NR (Lines 26-27). After retrieving and processing all possible nodes, the result set R is obtained and returned (Line 28).

6.2. Algorithm Complexity Analysis

Now we discuss the time complexity of our RSKQ processing algorithms.

As mentioned earlier, the reachability from the query q to the object o starting from $q.t$ in duration $[q.t, q.t{+}q.l]$ is represented by that from the start road segment r_0 to the segment r_b where o is located. According to Algorithm 1, for each day d in historical trajectory data, the calculation of reachability includes three main steps: (1) Construct set tr_s; (2) Construct set tr_e; (3) Calculate the reachability. Let $|TR_{start}|$ be the average number of trajectories that pass through the start road segment r_0 at $q.t$, $|TR_{end}|$ be the average number of trajectories that pass through r_0 at $q.t$ and finally pass through r_b, $|TR_{target}|$ be the average number of trajectories that pass through r_0 at $q.t$ and pass through r_b in duration $[q.t, q.t{+}q, l]$, m be the total number of days covered by the trajectory data. Therefore, it takes $O((|TR_{start_d}| + |TR_{end_d}| + |TR_{target_d}|) \times m)$ to calculate the reachability for m days and takes the average reachability value as the reachability from q to o for day d.

Our basic and optimized algorithms only take $O((|TR_{start}|{+}|TR_{end}|{+}|TR_{target}|)\times m)$ to process RSKQ queries if the reachability of r_0 is less than $q.r$. This is obvious because if the reachability of the road section where q is located is lower than $q.r$, no object in the road network can meet the query reachability requirement. Otherwise, let $|S_{visit}|$ be the average number of road segments to be retrieved for the query, $|S_{diatance}|$, $|S_{reach}|$, $|S_{keyword}|$ be the number of segments to be retrieved that meet the distance, reachability, and keyword requirement, respectively, $|S_{remain}|$ be the number of segments not pruned by Lemmas 1, 2, 3, and 4, $|o_{cand}|$ be the average number of objects related to $q.d$ in a segment R, where $q.d \cap R.d \neq \varnothing$. The time cost to judge whether a segment satisfy the distance and keyword constraint is $O(1)$ and $O(|q.d| \times |R.d|)$, respectively.

According to Algorithm 2, the cost of obtaining the segments that satisfy the distance constraint is $O(|S_{visit}|)$. On this basis, it takes $O(|S_{distance}| \times (|TR_{start}| + |TR_{end}| + |TR_{target}|) \times m)$ to find reachable segments, and then $O(|q.d| \times |R.d|) \times |S_{diatance} \cap S_{reach}|$ is costed to get candidate segments. In addition, $|S_{remain}| \times |o_{cand}|$ is for retrieving candidate objects. To sum up, the time complexity of the basic algorithm is $O(|S_{visit}| + |S_{distance}| \times (|TR_{start}| + |TR_{end}| + |TR_{target}|) \times m + O(|q.d| \times |R.d|) \times |S_{diatance} \cap S_{reach}| + |S_{remain}| \times |o_{cand}|)$.

For the optimized algorithm, we prune irrelevant nodes according to query keyword and distance constraints, and on the basis retrieve the objects in leaf nodes that satisfy the keyword, distance, and reachability requirements. The time complexity of the optimized algorithm is $O(|S_{visit}|/\delta + |S_{distance}|/\delta \times |q.d| \times |N.d| + |S_{diatance} \cap S_{keyword}| \times (|TR_{start_d}| + |TR_{end_d}| + |TR_{target_d}|) \times m + |S_{remain}| \times |o_{cand}|)$. Firstly, it takes $O(|S_{visit}|/\delta)$ to find the nodes that satisfy the distance requirement and $O(|S_{distance}|/\delta \times |q.d| \times |N.d|)$ to get the nodes related to query keywords, where $|N.d|$ is the average number of distinct keywords contained by each node. Then, $O(|S_{diatance} \cap S_{keyword}| \times (|TR_{start_d}| + |TR_{end_d}| + |TR_{target_d}|) \times m)$ is costed to find candidate segments and $O(|S_{remain}| \times |o_{cand}|)$ is used to retrieve the objects related to the query. As shown in algorithm 3, we can prune the subtree rooted at $TNode$ if $TNode$ is irrelevant to $q.d$ or is too far from q. Therefore, compared with the basic algorithm, the optimized algorithm needs to retrieve less segments and has lower time complexity.

7. Performance Evaluation

This section evaluates the performance of our methods for the Top-k spatial keyword query based on reachability. It first describes the experimental setup in Section 7.1, and then discusses the experimental results in Section 7.2.

Table 2: Parameters Evaluated in the Experiments.

Parameter	Values
number of query keywords ($\vert q.d\vert$)	1, **2**, 3, 4
start time $q.t$ (o'clock)	8, 12, **16**, 20, 24
duration $q.l$ (min)	**10**, 20, 30, 40
reachability $q.r$	0, 0.2, 0.4, 0.6, **0.8**, 1
number of objects ($\vert O\vert$)	2, **4**, 6, 8 (M)
number of results (k)	**1**, 5, 9, 13
parameter α	0, **0.3**, 0.6, 0.9
parameter β	0, **0.3**, 0.6, 0.9

7.1. Experimental Settings

We adopt three datasets in our experiments: (1) **Road Network.** The road network of Shenzhen, China is extracted from OpenStreetMap. (2) **Objects.** The synthetic object set includes a set of spatio-textual objects, and the locations of objects are randomly distributed on the edges of the road network in Shenzhen. The keywords of objects are obtained from OpenStreetMap. (3) **Trajectories.** We use a trajectory dataset collected from taxis in Shenzhen. The dataset was collected for 30 days in November 2014. These trajectories represent 21,385 unique taxis in Shenzhen. It contains 407,040,083 GPS points of 21,385 taxis, with an average sampling rate of 30 seconds.

To our best knowledge, there is no other work on dealing with $RSKQ$ queries up to now. IR-tree based methods are often used to solve traditional Top-k spatial keyword queries, and the single-source maximum boundary search algorithm (SQMB) and the trace back search algorithm (TBS) work together to search the reachable area. Therefore, we combine the above three methods as the comparison baseline method (called "IR-tree+SQMB+TBS") to verify the performance of the two methods based on SRTR-tree index (called Basic method and Optimized method respectively). The baseline method first finds reachable segments under given conditions, and then finds the Top-k spatio-textual objects that satisfy spatial and textual constraints, which are located on these candidate segments.

The following experiments verify the performance of the three methods by varying the start time, the number of keywords, the query duration, the number of spatio-textual objects, the number of results, reachability, parameters α and β, as shown in Table 2.

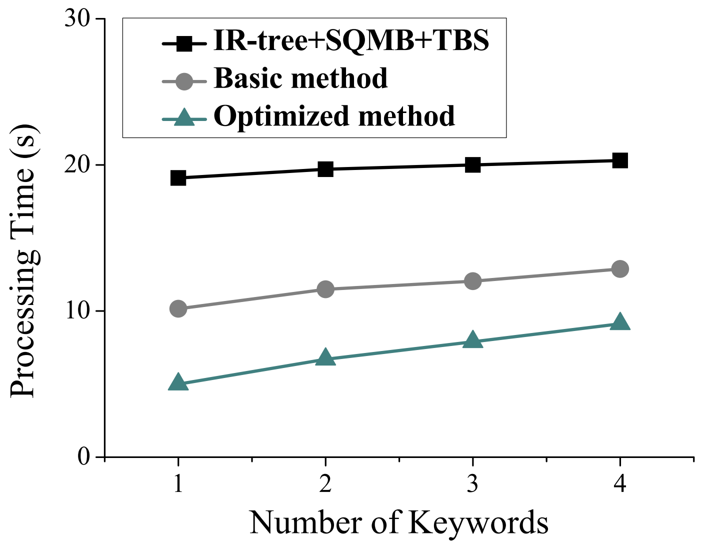


Figure 8: The effect of $|q.d|$

7.2. *Experimental Results*

7.2.1. *Varying the number of keywords*

As shown in Figure 8, the processing time of these three methods increases with the increment of the query keyword number. This is because as the number of keywords increases, candidate regions or road segments in the IR-tree and SRTR-tree increase, and correspondingly more time is required to calculate the comprehensive score of the candidate objects. However, the processing time of IR-tree+SQMB+TBS is much higher than our methods, because no matter how the number of query keywords changes, it has to obtain reachable road segments by the SQMB+TBS algorithm first. Our optimized method prunes irrelevant areas and road segments faster, which is superior to the basic method.

7.2.2. *Varying the query start time*

The change of query start time has no effect on the traditional Top-k spatial keyword query, and it mainly affects the efficiency of reachability calculation. We set the same time granularity as the SRTR-tree for the index structure of the reachable area computing algorithm "SQMB+TBS" according to rush periods and other periods of a day. From Figure 9, we observe that the running time of "IR-tree+SQMB+TBS" changes with the query start time. The running time around 8

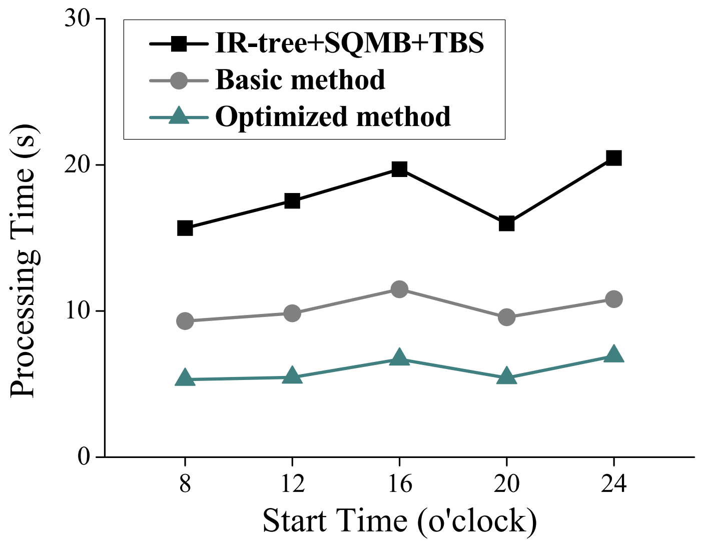


Figure 9: The effect of $q.t$

am and 8 pm is significantly lower than that in other periods, mainly due to traffic congestion during rush hours. When the query start time is set to 4 pm or 12 pm, the processing time is basically the same. The better the traffic condition, the larger the reachable area, and the longer the calculation time of reachability. Our basic method shows the same change trend as the "IR-tree+SQMB+TBS" method, but the change range is small. For the optimized method, it first prunes irrelevant areas according to query keywords, and has less reachability calculation, so its running time is shorter than the other two methods.

7.2.3. Varying the query duration

To evaluate the effect of the query duration $q.l$ on the processing time of the baseline method and our two algorithms, we increase $q.l$ from 10 to 40 minutes. As shown in Figure 9, the running cost of these methods increases with the increase of $q.l$.

For "IR-tree+SQMB+TBS", the reason lies in two aspects. Firstly, the "SQMB+TBS" algorithm can skip the area near the query starting position $q.s$, and start the search from the road segments on the farthest boundary of the maximum boundary region, which is the area that can be reached within the query duration $q.l$ starting from $q.s$. Its processing time increases as $q.l$ increases, because of

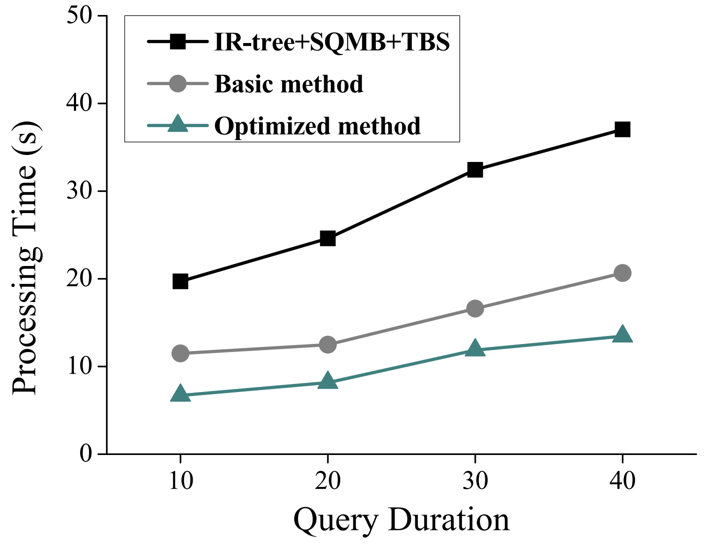


Figure 10: The effect of $q.l$

the expansion of the maximum boundary region. At the same time, for the IR-tree based method of solving spatial keyword queries, there are more candidate objects to be retrieved in a larger reachable area, so the running time increases. When $q.l$ increases, the processing time of our basic method increases due to more reachability calculations of road segments. In the optimized method, the priority use of pruning techniques based on query keywords and road network distance reduces the reachability calculation of irrelevant areas and segments, thus the processing time is the shortest.

7.2.4. Varying the reachability

The parameter $q.r$ is the lower limit value of reachable probability for result objects of $RSKQ$ queries. We fix other parameters to study how different reachability $q.r$ influence the performance of our query processing algorithm, as shown in Figure 11.

The "SQMB+TBS" algorithm finds the furthest and closest road segments that can be reached based on the maximum and minimum speed in all directions, so the query processing time is less dependent on the reachability $q.r$. As the reachability increases, the number of reachable segments reduces, and the number of candidate objects decreases, which shortens the query processing time. Our basic

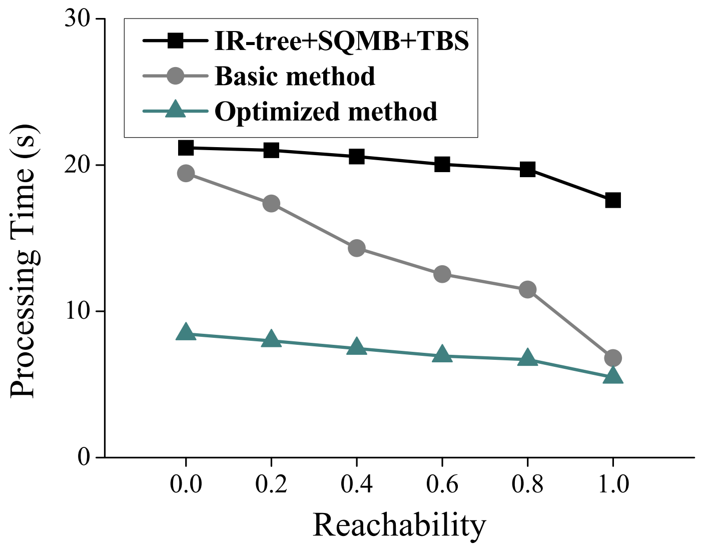


Figure 11: The effect of $q.r$

algorithm starts from the segment r_0 where the query q is located and explores adjacent reachable road segments in ascending order of road network distance to find result objects. When $q.r$ is 0, the number of road segments to be explored is the largest and the processing time is the longest. When the reachability $q.r$ increases, the processing time decreases continuously. For the optimized method, fewer reachable road segments lead to fewer candidates and its processing time is slightly reduced, which is always less than that of the other two methods.

7.2.5. Varying the number of objects

As shown in Figure 12, when the number of spatio-textual objects in the system increases, the running time of the baseline method increases. Although the increase in the number of objects hardly affects the processing time of reachability calculation, it takes more time to retrieve candidate objects. For our basic method, when the number of spatio-textual objects increases, they are more densely distributed in the road network, which leads to fewer road segments to be retrieved, so the processing time is reduced. Although the processing time of the optimized method increases slightly with the increase of objects, it is still superior to other methods.

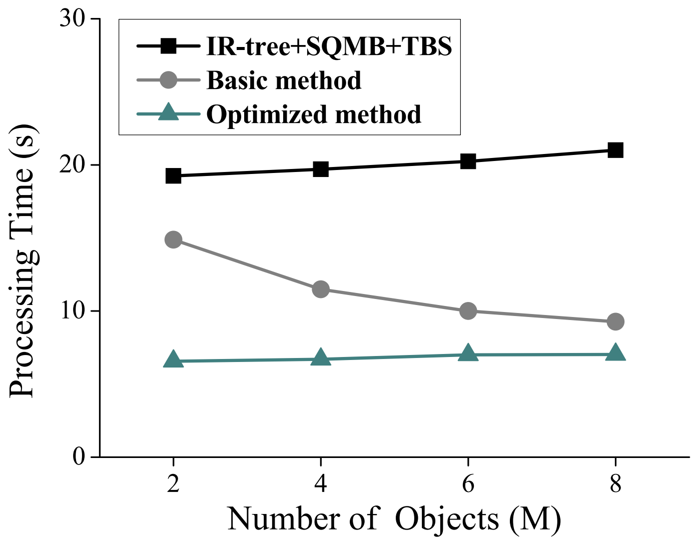


Figure 12: The effect of $|O|$

7.2.6. Varying the number of query results

We study the impact of varying k (i.e., the number of result objects requested) with the results plotted in Figure 13. For the SQMB+TBS algorithm that handles reachability computation, the change of k does not affect its processing time. The increase of k makes the IR-tree based method has to retrieve more objects, but the time required for the spatial keyword query is several orders of magnitude less than that required for reachability calculation. Therefore, the processing time of the baseline method changes little with the change of k value. For our methods, the more the number of results, the more sections to be retrieved and the greater the amount of reachability calculation, which makes the processing time longer. But our methods are still superior to the baseline method. The optimized method first prunes the road segments irrelevant to the query keywords to reduce the reachability calculation, so as to shorten the query processing time.

7.2.7. Varying parameter α

In this experiment, we evaluate the impact of the query preference parameter α as illustrated in Figure 14. The parameter α is used to measure the importance of reachability in the scoring function. When α is 0, it means that only the road network distance between the query q and the object o is concerned, and whether

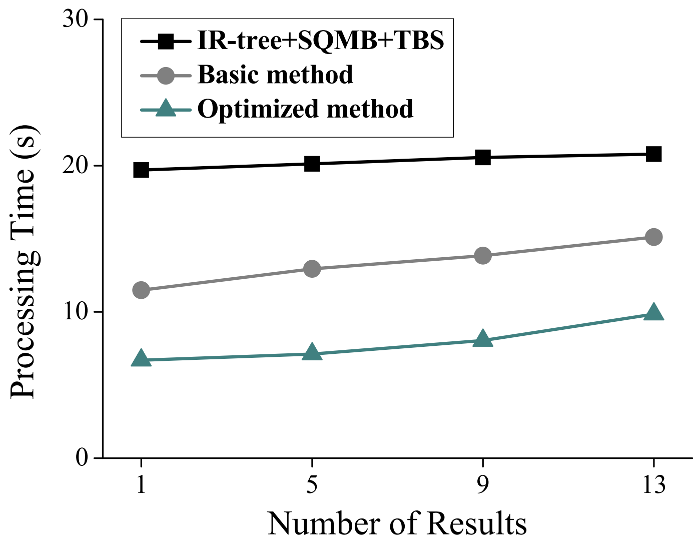


Figure 13: The effect of k

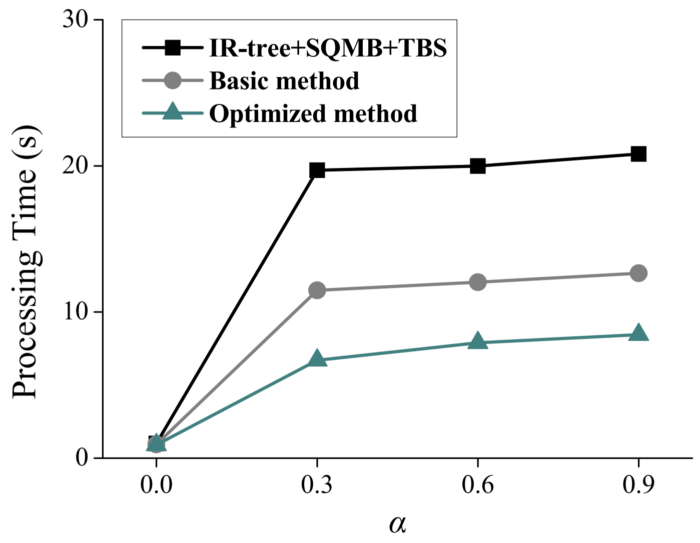
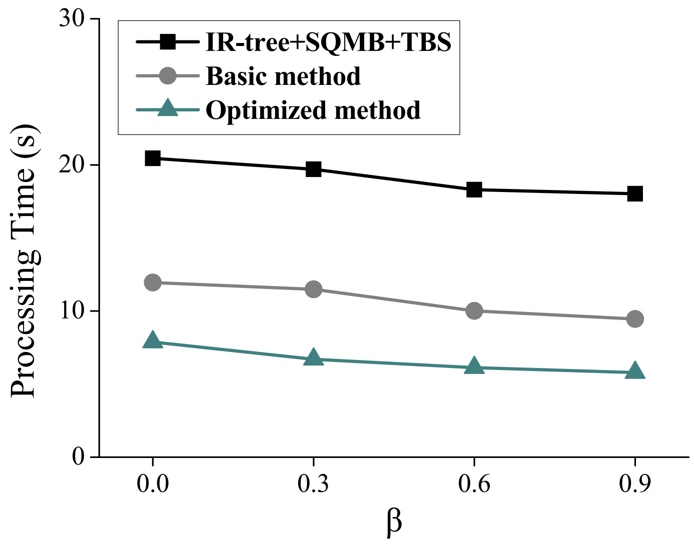


Figure 14: The effect of α



.

Figure 15: The effect of β

the object can be reached within the duration $q.l$ is not concerned. In this case, the RSKQ query is equivalent to the traditional Top-k spatial keyword query, and the processing time of three methods is almost the same. When α is greater than 0, the accessibility of objects should be considered in query processing, and the processing time is much higher than the traditional Top-k spatial keyword query. However, with the continuous increase of α, the increase of processing time tends to be gentle.

7.2.8. Varying parameter β

Next, the effect of varying the query preference parameter β on the performance of these three methods is evaluated. Figure 15 shows that the processing time of the methods does not decrease very much with increasing β. When β is 0, only the reachability from query q to object o is concerned, without considering the distance between q and o. As β gradually increases, the weight of the spatial proximity between the query q and the object o increases. More areas and road segments on the road network which are far from the query q are pruned and the processing time decreases.

8. Conclusion

This paper addresses the issue of processing the Top-k spatial keyword query based on reachability (RSKQ). The traditional spatial keyword query only considers the text similarity and spatial proximity between the query and spatio-textual objects. The RSKQ query introduces object accessibility into traditional spatial keyword queries to determine whether the result object can be reached within a given time interval from the start position and time point of a given query, so as to improve the effectiveness of the query result. To solve RSKQ queries, an efficient index called SRTR-tree is designed. The SRTR-tree effectively organizes the information of the road network, trajectories, and spatio-textual objects on the road network. Among them, the road network and trajectory information are used to calculate reachability. Moreover, several lemmas are proposed to prune huge amounts of irrelevant spatial objects for queries. Based on the SRTR-tree, we propose two algorithms and demonstrate their efficiency through extensive experiments. The results show that the proposed methods are feasible and the optimized method is more efficient. In the future, we will introduce accessibility into other SKQ queries, such as mobile SkQ queries, direction-aware SKQ queris, and collective SKQ queries.

9. Ethical Approval and Consent to participate (Not applicable)

10. Human and Animal Ethics (Not applicable)

11. Availability of supporting data (Not applicable)

12. Funding (Not applicable)

13. Authors' contributions

The contributions of the authors are as follows:

The first three authors (including Yanhong Li, Jiayu Ren and Changyin Luo): conceptualization, propose core ideas,write manuscripts, review and editing.

The fourth and fifth authors (Yu Feng, Xiaokun Du): algorithm optimization, some code implementation and testing, formal analysis.

The last author (Jianjun Li): validation, review.

14. Consent for publication (Not applicable)

15. Acknowledgments

This work is supported by National Science Foundation of China (NO.61309002, 61522208), Science Foundation of Hubei Province (NO.2017CFB135), Ministry of education of Humanities and Social Science project (No.20YJCZH111), and Fundamental Research Funds for the Central Universities (No.CCNU20ZT013).

16. Authors' information

Yanhong Li received the Ph.D. degree in computer science from Huazhong University of Science and Technology, Wuhan, China, in 2011. She is currently a Professor with the College of Computer Science, South-Central University for Nationalities, Wuhan, China. Her research interests include spatial information and communication, and multimedia network technology.

Jiayu Ren and Yu Feng are currently studying for a master's degree in computer science at the school of computer science, Central South University for nationalities, Wuhan, China. Their research interests include spatial information and communication and multimedia network technology.

Changyin Luo received the Ph.D. degree in computer science from Huazhong University of Science and Technology, Wuhan, China, in 2015. He is currently a Lecturer with the School of Computer, Central China Normal University,Wuhan, China. His research interests include spatial database, spatial keyword search, and social media data processing.

Xiaokun Du received the Ph.D. degree in computer science from Huazhong University of Science and Technology, Wuhan, China, in 2010. He is currently a Professor with the College of Computer Science, South-Central University for Nationalities, Wuhan, China. His research interests include spatial information and multimedia network technology.

Jianjun Li received the Ph.D. degree in computer science from Huazhong University of Science and Technology, Wuhan, China, in 2012. He is currently an Associate Professor the School of Computer, Huazhong University of Science and Technology, Wuhan, China. His research interests include spatial database and big data.

References

[1] G. Li, J. Feng, and X. Jing. Desks: Direction-aware spatial keyword search. In *2012 IEEE 28th International Conference on Data Engineering*, 2012.

[2] C. Lei, Y. Li, J. Xu, and C. S. Jensen. Direction-aware why-not spatial keyword top-k queries. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, 2017.

[3] G. Chen, J. Zhao, Y. Gao, L. Chen, and R. Chen. Time-aware boolean spatial keyword queries. *IEEE Transactions on Knowledge and Data Engineering*, 29(11):2601–2614, 2017.

[4] Jingwen Zhao, Yunjun Gao, Gang Chen, and Rui Chen. Towards efficient framework for time-aware spatial keyword queries on road networks. *ACM Transactions on Information Systems*, 36(3):24.1–24.48, 2018.

[5] Z. Chen, T. Zhao, and W. Liu. Time-aware spatial keyword cover query. *Data & Knowledge Engineering*, 122(JUL.):81–100, 2019.

[6] Zijun Chen, Tingting Zhao, and Wenyuan Liu. Time-aware collective spatial keyword query. *Computer Science and Information Systems*, (00):34–34, 2021.

[7] G. Wu, Y. Ding, Y. Li, J. Bao, Y. Zheng, and J. Luo. Mining spatio-temporal reachable regions over massive trajectory data. *Masters Theses*, pages 1283–1294, 2017.

[8] Y. Ding, X. Zhou, G. Wu, Y. Li, and J. Luo. Mining spatio-temporal reachable regions with multiple sources over massive trajectory data. *IEEE Transactions on Knowledge and Data Engineering*, PP(99):1–1, 2019.

[9] R. Li, J. Bao, H. He, S. Ruan, and Y. Zheng. *Discovering Real-Time Reachable Area Using Trajectory Connections*. Database Systems for Advanced Applications, 2020.

[10] X. Tang, M. Chai, X. Chen, and W. Chen. Spatio-temporal reachable area calculation based on urban traffic data. *IEEE Systems Journal*, PP(99):1–12, 2020.

[11] J. M. Ruan, B. Liu, H. Wei, Y. Qu, N. Zhu, and X. Zhou. How many and where to locate parking lots? a spacetime accessibility-maximization modeling framework for special event traffic management. *Urban Rail Transit*, 2(2):59–70, 2016.

[12] Calvin P. Tribby A and Paul A. Zandbergen B. High-resolution spatio-temporal modeling of public transit accessibility. *Applied Geography*, 34(4):345–355, 2012.

[13] L. Tong, X. Zhou, and H. J. Miller. Transportation network design for maximizing spacetime accessibility. *Transportation Research Part B*, 81:555–576, 2015.

[14] J Ning, X Wu, and Z Liu. Multi-user location recommendation considering road accessibility and time-cost. *Geomatics Inf. Sci. Wuhan Univ.*, 44(5):633–639, 2019.

[15] Stephan Seufert, Avishek Anand, Srikanta Bedathur, and Gerhard Weikum. Ferrari: Flexible and efficient reachability range assignment for graph indexing. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pages 1009–1020. IEEE, 2013.

[16] Renê Rodrigues Veloso, Loïc Cerf, Wagner Meira Jr, and Mohammed J Zaki. Reachability queries in very large graphs: A fast refined online search approach. In *EDBT*, pages 511–522. Citeseer, 2014.

[17] Junfeng Zhou, Shijie Zhou, Jeffrey Xu Yu, Hao Wei, Ziyang Chen, and Xian Tang. Dag reduction: Fast answering reachability queries. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 375–390, 2017.

[18] Hao Wei, Jeffrey Xu Yu, Can Lu, and Ruoming Jin. Reachability querying: An independent permutation labeling approach. *Proceedings of the VLDB Endowment*, 7(12):1191–1202, 2014.

[19] Jiao Su, Qing Zhu, Hao Wei, and Jeffrey Xu Yu. Reachability querying: can it be even faster? *IEEE Transactions on Knowledge and Data Engineering*, 29(3):683–697, 2016.

[20] Y. Zhou, X. Xie, C. Wang, Y. Gong, and W. Y. Ma. Hybrid index structures for location-based web search. *ACM*, 2005.

[21] Gao Cong, Christian S. Jensen, and Dingming Wu. Efficient retrieval of the top-k most relevant spatial web objects. *Proc. VLDB Endow.*, 2(1):337–348, 2009.

[22] Dingming Wu, Man Lung Yiu, Gao Cong, and Christian S Jensen. Joint top-k spatial keyword query processing. *IEEE Transactions on Knowledge and Data Engineering*, 24(10):1889–1903, 2011.

[23] Ian De Felipe, Vagelis Hristidis, and Naphtali Rishe. Keyword search on spatial databases. In *2008 IEEE 24th International Conference on Data Engineering*, pages 656–665. IEEE, 2008.

[24] João B. Rocha-Junior, Orestis Gkorgkas, Simon Jonassen, and Kjetil Nørvåg. Efficient processing of top-k spatial keyword queries. In Dieter Pfoser, Yufei Tao, Kyriakos Mouratidis, Mario A. Nascimento, Mohamed F. Mokbel, Shashi Shekhar, and Yan Huang, editors, *Advances in Spatial and Temporal Databases - 12th International Symposium, SSTD 2011, Minneapolis, MN, USA, August 24-26, 2011, Proceedings*, volume 6849 of *Lecture Notes in Computer Science*, pages 205–222. Springer, 2011.

[25] Dongxiang Zhang, Kian-Lee Tan, and Anthony KH Tung. Scalable top-k spatial keyword search. In *Proceedings of the 16th international conference on extending database technology*, pages 359–370, 2013.

[26] Chengyuan Zhang, Ying Zhang, Wenjie Zhang, and Xuemin Lin. Inverted linear quadtree: Efficient top k spatial keyword search. *IEEE Transactions on Knowledge and Data Engineering*, 28(7):1706–1721, 2016.

[27] Dingming Wu, Man Lung Yiu, Christian S. Jensen, and Gao Cong. Efficient continuously moving top-k spatial keyword query processing. In Serge Abiteboul, Klemens Böhm, Christoph Koch, and Kian-Lee Tan, editors, *Proceedings of the 27th International Conference on Data Engineering, ICDE 2011, April 11-16, 2011, Hannover, Germany*, pages 541–552. IEEE Computer Society, 2011.

[28] Weihuang Huang, Guoliang Li, Kian Lee Tan, and Jianhua Feng. Efficient safe-region construction for moving top-k spatial keyword queries. In *Acm International Conference on Information & Knowledge Management*, 2012.

[29] Guoliang Li, Jianhua Feng, and Jing Xu. Desks: Direction-aware spatial keyword search. In *2012 IEEE 28th International Conference on Data Engineering*, pages 474–485. IEEE, 2012.

[30] X. Liu, L. Chen, and C. Wan. Linq: A framework for location-aware indexing and query processing. *IEEE Transactions on Knowledge and Data Engineering*, 27(5):1288–1300, 2015.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- WWWJ.rar