Effective Spatial Keyword Query Processing on Road Networks

Hailin Fang[1], Pengpeng Zhao[1(✉)], Victor S. Sheng[2], Jian Wu[1], Jiajie Xu[1], An Liu[1], and Zhiming Cui[1]

[1] School of Computer Science and Technology, Soochow University, Suzhou 215006, People's Republic of China
{hlfang,ppzhao,jianwu,xujj,anliu,szzmcui}@suda.edu.cn
[2] Computer Science Department, University of Central Arkansas, Conway, USA
ssheng@uca.edu

Abstract. Spatial keyword query plays an important role in many applications with rapid growth of spatio-textual objects collected. In this context, processing boolean spatial keyword query on road networks is one of the most interesting problems. When giving a query which contains a location and a group of keywords, our aim is to return k objects containing all the query keywords which are the nearest to the query location. Though the research on this problem has received extensive studies in Euclidean space, little is done to deal with it on road networks. We first propose novel indexing structures and algorithms that are able to process such query efficiently. Experimental results on multiple real-word datasets show that our methods achieves high performance.

Keywords: Road networks · Spatial keyword search · Spatial indexing

1 Introduction

With the increasing pervasiveness of the mobile devices and geo-location services, there are large amounts of spatio-textual data available in many applications. For instance, many applications (e.g, Twitter, Yellow page etc) provide target location information with a short text description. People can publish contents with geographical position information in these applications every day. For these huge spatial textual data, how to establish an effective real time query mechanism is a great challenge. The current approaches processing spatial keyword queries are mostly developed in Euclidean space [1–4,13,14,16]. In reality, our daily travels are usually constrained by the road networks, which leads to the network distance between two locations may be larger than their Euclidean distance. For example, the network distance between two hotels on the opposite banks of a river is completely different from their Euclidean distance. Then the result obtained under the Euclidean space may not be close to a query location on its road network.

For example, Fig.1 illustrates a part of road network and some spatio-textual objects residing on some roads. Fig.2 provides text descriptions of the all objects

M.A. Sharaf et al. (Eds.): ADC 2015, LNCS 9093, pp. 194–206, 2015.
DOI: 10.1007/978-3-319-19548-3_16

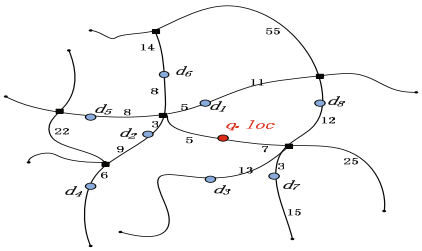


Fig. 1. Road network

d_1	*steam-bath,laundry-service,lounge*
d_2	*steam-bath,restaurant*
d_3	*steam-bath,restaurant,safe*
d_4	*Wi-Fi,restaurant,laundry-service*
d_5	*steam-bath,safe,wellness*
d_6	*Wi-Fi,restaurant,lounge*
d_7	*steam-bath,restaurant,flowershop*
d_8	*casino,restaurant,fitness root*

Fig. 2. The descriptions of objects on Road network

(represent hotels) shown in Fig.1. On this partial road network, there are 6 vertices(road intersections) and 8 objects, which are denoted as grids and circles, respectively. Each digit on edge presents the length of it. Given a query such as q={*q.loc, q.term, k*} in Fig.1, *q.loc* indicates the location of the query, *q.term*={*steam-bath,restaurant*} presents a set of keywords, and k is the limited number of returns. Supposed that the user intends to find two nearest hotels to *q.loc*. According to the Euclidean distance, the result of this query is $\{d_2, d_3\}$. But due to the constrains of the road network, the road distance between *q.loc* and the objects d_2 is 8, denoted as $\delta(q, d_2) = 8$. Similarly we have $\delta(q.loc, d_3) = 20$. However, we find d_7 also meets the text description of the query and its distance $\delta(q.loc, d_7) = 10$ is shorter than the distance to the object d_3. Therefore the best query result set is $\{d_2, d_7\}$ rather than $\{d_2, d_3\}$.

This example shows that the results of query on road networks are very different from Euclidean space. Compared with Euclidean space, it is much more challenging on road networks. The reason is that what we need is to compute the network distance between two objects rather than Euclidean distance. Thus existing methods in Euclidean space cannot be directly applied to road networks. Although spatial keyword query on road networks have attracted some research efforts in recent years [8,10], which are focused on different type of query like *range*, *top-k* query, but their solutions can not directly applied to boolean spatial keyword query(BSKQ) on road networks. To the best of our knowledge, there are no work studying the BSKQ on road networks.

In this paper, we propose two novel approaches to deal with boolean spatial keyword on road networks. The basic spatial keyword query method on road networks is to expanse the network from the query location using Dijkstra's algorithm, but during the process of traversing a network, the performance is very poor. So we propose a new method First Inverted file Then G-Tree, called FITG, by combining inverted index with spatial index G-Tree [15]. According to the principle of first text pruning then spatial pruning, it is avoid traversing the network in a point-to-point manner. Furthermore, we also propose a hybrid index named SG-Tree(Signature based G-Tree). The idea of SG-Tree is to create a signature for each node in G-Tree and partitions the whole road network into a group of interconnected sub-networks and organizes them in a hierarchy structure, which improves query efficiency tremendously.

The rest of the paper is organized as follows. Section 2 introduces related work in this field. Section 3 formally defines basic concepts and notations used in this paper. Section 4-5 elaborates the proposed methods, i.e., FITG and SG-Tree, respectively. Section 6 conducts the evaluation on three real-word datasets and shows our experimental results. Finally, we conclude our paper in Section 7.

2 Related Works

Currently there are three types of spatial keyword query(SKQ): boolean spatial keyword query (BSKQ), top-k spatial keyword query (Top-k SKQ) and range constrained spatial keyword query (RC-SKQ). In the past decades, researchers have done a lot of work on k-nearest neighbors(KNN) on road networks [5–7,9,11,15].

In recent years, SKQ has received extensive studies in Euclidean distance space [1–4,13,14,16], and achieved very significant results(e.g., [1,3] for a comprehensive survey). BSKQ is one of the most important query problems and many efficient query approaches have been proposed such as inverted R-tree [16], information retrieval R-tree [4], Inverted Linear Quad-tree [13], and some solve other types of query(e.g., top-k) method were proposed, like I^3 [14] and so on. However, all of these algorithms are based on Euclidean space, but real-life travel trajectories are constrained by road networks.

Recently, the problem of SKQ on road networks has been studied by [8–10, 12]. Shekhar et al. [12] proposed the CCAM method which reduces 2-dimensional data of a node to a single dimensional and effectively organizes the adjacent list of road nodes. We can take advantage of the access locality in the query processing that can reduce the I/O costs to improve query efficiency. Papadias et al. [9] proposed an efficient framework to store road networks and spatio-textual objects. Recently, Rocha-Junior et al. [10] raised efficient methods to address Top-k SKQ on road networks. They designed a framework of the index for using overlay networks to prune the regions of the network. Li et al. [8] proposed a new query that range-constrained SKQ on road networks and put forward several different indexing strategies to address this type of SKQ. Their proposed approaches are very similar, although addressing different types of problems. But all the above works have not been studied the BSKQ problem on road networks. Furthermore, we exploit a new elegant and efficient road networks index G-Tree[15] and the ubiquitous text index signature to propose a novel hybrid index called SG-Tree that can scale to large road networks.

3 Problem Statement

In this section, we introduce a graph model to represent a road network. Then, we define the related concepts which will be used in the following of this paper.

Road Networks: This paper uses a weighted graph to describe a road network, which is denoted as $G = (V, E, W)$, where V is a set of nodes that represent a

road segment, the edge set is denoted as E and W is a set of weights denoting the cost on the corresponding edge, such as travel time or distance. $(\upsilon, \nu) \in E$ denotes an edge, and $w_{\upsilon,\upsilon}$ is a weight on this edge. The shortest path between two nodes υ and ν is denoted as $|\upsilon, \nu|$, an $\|\upsilon, \nu\|$ is the minimum length between υ and ν, i.e., $\|\upsilon, \nu\|{=}w_{\upsilon,\upsilon}$. Let $q.loc$ be a query, and o be a spatio-textual object on the road network, then the shortest distance between the query and o is denoted as follows:

$$\|q.loc,\ o\| = \begin{cases} \|q.loc,\ o\| & both\ q.loc\ and\ o\ are\ on\ the\ same\ edge \\ min\{\|q.loc, \upsilon\| + \|o,\ \upsilon\|, \|q.loc, \nu\| + \|o, \nu\|\} & otherwise \end{cases} \quad (1)$$

Spatio-Textual Object: A spatio-textual object is normally expressed by a point with coordinates and a set of keywords, which is described in a two dimensional space. For example, $\{loc, term\}$ presents a spatio-textual object, where $o.loc$ is the location of the object, including its latitude and longitude, and $o.term$ presents a set of keywords describing some text expressions, such as $term = \{t_1, t_2 .... t_f\}$. For simplicity, each object lies in its corresponding edge.

Spatial Keyword Query: According to the spatio-textual object definition, we use D to represent all objects in a spatial database as follows: $D = \{o|\forall o \in D, o = \{o.loc, o.term\}\}$. Given a road network as a weight graph G and a query point q, a spatial keyword search can retrieve k objects, each of which contains all keywords of the query and whose network distance is the shortest to the query location.

The Baseline Approach: Traditional spatial keyword query methods use the network expansion on road networks. On this basis, we combine the signature index on edge with network expansion and develop an enhance method called Signature based Network Expansion (SNE) as our baseline method. The structure of SNE we adopt a very popular data structure connectivity clustered access method (CCAM) [12] used to store a road network. We build a network R-tree to identify the road segment where the location of query is. In order to avoid loading a large number of irrelevant objects, we use the signature technology to organize all objects on each edge. We use $I(e, term)$ as the signature of each edge e. If there is at least one object contains the query keywords lying on the edge e, $I(e, term) = 1$; otherwise $I(e, term) = 0$, thus only partial edges containing query keywords can be loaded.

After providing the definitions of the related concepts, we will discuss the three proposed methods for spatial keyword query on road networks in details.

4 The First Inverted File Then G-Tree Approach

In the above baseline method, if the road network data is very complicated with a huge number of spatio-textual objects or locations of objects satisfying the condition are far away from the query, it needs to spend a lot of time on the edge expansion process. It means that, the time complexity is very high.

In order to solve this problem, we employ an excellent spatial road network index technique by combing G-Tree and a current traditional text inverted index. Using two separated indexes to present a new method called First Inverted file Then G-Tree (FITG in short). The index structure is as follows.

4.1 Index Structure

FITG method is executed according to the principle of first text pruning then spatial pruning. It avoids a point-to-point manner to traverse the network which can save much cost and improve query efficiency. In this algorithm, we first use a text index to find all objects which contain all the keywords of the query, and then set them as candidates to calculate the shortest network distance to query location. Since inverted index has been explained in many papers, we will not repeat it in this paper. G-Tree index is a novel and efficient spatial road network index structure, proposed by Zhong et al. [15]. G-Tree has two core features. The first feature is that it has a highly balanced tree structure which can recursively divide the road network into a plurality of sub networks and map each vertex to a corresponding sub network. The other is that it uses the best-first search algorithm, greatly improving the performance. For other details of G-Tree, please refer to the paper [15].

For example, given a query q whose location is show in Fig.1 and it contains the text description $q.term = \{steam\text{-}bath,\ restaurant\}$, whose aim is to find $k = 2$ spatio-textual objects. The steps of the FITG algorithm are as following:

step 1: *According to the inverted index, the returned candidates of the keyword "steam-bath" are:* d_1, d_2, d_3, d_5, d_7.
step 2: *According to the inverted index, the returned candidates of the keyword "restaurant" are:* $d_2, d_3, d_4, d_6, d_7, d_8$.
step 3: *After the intersection, three candidates are returned, which contain all the keywords:* d_2, d_3, d_7.
step 4: *Initialize the occurrence lists of* d_2, d_3, d_7 *by G-Tree.*
step 5: *Add* d_2 *to the result set* $R = < d_2, 8 >$.
step 6: *Add* d_7 *to the result set* $R = < d_2, 8 >, < d_7, 10 >$.
step 7: *Because the size of* $R.size = 2$ *is equal to* k*, so the process terminates and returns the result.*

4.2 Query Processing

In Algorithm 1, for lines 1-3 we use the inverted index technique to prune the whole text dimension first to find the object list L_i which contains any one of the keywords in a query. Then we make an intersection for these lists denoted as L. So the L is contain all objects which contains all query keywords. Then the priority queue, the result set and the occurrence list of candidates are initialized respectively in lines 4-6. Second, we use the spatial index G-Tree to calculate the network distance between any objects in L and the location of the query in lines 7-22, If a priority queue is null or the number of the result set is more than k, the process terminates.

Algorithm 1. *FITGTreeQueryProcessing(q.loc,q.term,k,I,T_g)*

Input: *q.loc, q.term, k.I*:is the Inverted index, T_g a G-Tree index.
Output: $\Re$ objects satisfying the query condition.

```
1: for each word t_i in q.term do
2:     L_i ← I.getDocListByTerm(t_i );
3: InvL ← the intersection of object pointers in L_i;
4: Initialize : ℜ := ∅; Q := ∅;
5: Γ ← InvL; initialize the occurrence list from the candidate set
6: Q.Enqueue(⟨T_{g_root}, 0⟩)
7: while Q ≠ ∅ and |ℜ| ≤ k do
8:     n = Q.pop();
9:     if n is a leaf node then
10:        if q.loc ∈ n then
11:            mindist_inside_leaf(q.loc, n);
12:        else mindist_outside_leaf(q.loc, n);
13:        for each τ ∈ Γ(n) do
14:            Q.Enqueue(τ, SPDist(n, τ));
15:    else if n is a non-leaf node then
16:        for each child node c ∈ Γ(n) do
17:            if q.loc is in c then
18:                Q.Enqueue(c, SPDist(q.loc, c) = 0);
19:            else
20:                mindist_outside_nonleaf(q.loc, c);
21:                Q.Enqueue(c, SPDist(q.loc, c));
22:    else if n is is an object then insert then insert n into ℜ
   return ℜ;
```

5 The Signature Based G-Tree Index Approach

From the above algorithm, the efficiency of the FITG approach is much higher than that of the baseline method SNE. FITG method can not only enhance the text pruning ability but also reduce the time on network distance computing. It dose not need to traverse the entire network. It only needs to calculate the network distance between the query and candidates, which reduces the time complexity and the computation cost. However, the FITG algorithm will cause a serious performance deterioration of algorithm if a great deal of object contain the query keywords or the query condition contains a large number of keywords. A large number of objects will be retrieved, which results in a high computation cost on finding the intersection from the candidates for each query keyword. So we propose an efficient and elegant method which can support an efficient spatial keyword query on large road networks. We integrate the popular textual index signature into spatial index G-Tree named Signature based G-Tree(SG-Tree). We will detail our index structure as follows.

5.1 Index Structure

In this approach, we create a signature for each node in G-Tree which node represents a sub-tree root node. The objects that each root node contains consist of those objects of its children nodes. We use the signature to determine whether the root node contains the query keywords. It will prune the entire sub-tree if it dose not match with query signature. The reason is that the signature of non-leaf node is composed of all children signatures. In order to enhance the spatial pruning ability, we integrate the traditional Incremental Nearest Neighbor(INE) method into the G-Tree. By integrating the traditional INE with the signature, the distance-first SG-Tree access nodes and spatio-textual objects have a minimum distance away from the location of the query, which contributes to improve the query efficiency of the distance-first spatial keyword.

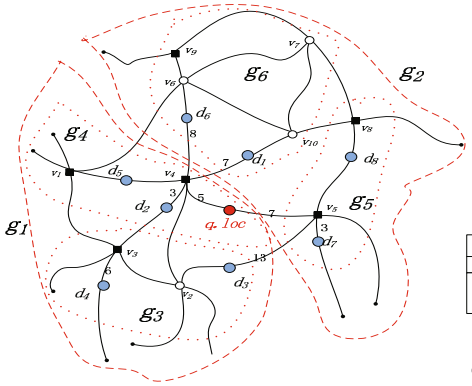


Fig. 3. Graph partition

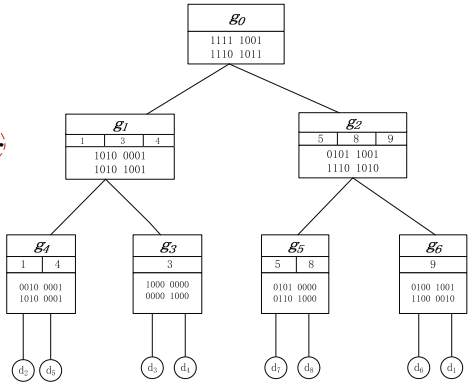


Fig. 4. SG-Tree structure

For example, Fig.4 shows an example of SG-Tree which integrates the G-Tree of the partial road network in Fig.3 with the spatial-textual objects in Fig.2. Given a query point *q.loc* as is shown in Fig.3 and its keywords description such as *q.term*={*steam-bath, restaurant*}, k objects which are the nearest to the query location and containing all keywords of the query will be returned. The sequence of steps is as follows.

step 1: *Enqueue* g_0*:* $Q = \{g_0, 0\}$*;*
step 2: *Enqueue* g_0*: match the signature, Enqueue* g_1, g_2*;*$Q = \{(g_1, 0), (g_2, 7)\}$*;*
step 3: *Enqueue* g_1*: match the signature, Enqueue* g_4*,* $Q = \{(g_4, 0), (g_2, 7)\}$*;*
step 4: *Enqueue* g_3*: is a leaf node,* d_2 *contains keyword, Enqueue* d_2*,* $Q = \{(d_2, 8), (g_2, 7)\}$*;*
step 5: *Enqueue* g_2*: match the signature,Enqueue* $g_5 Q = \{(g_5, 7), (d_2, 8)\}$*;*
step 6: *Enqueue* g_5*: is a leaf node then* d_7 *contains keywords, Enqueue* d_7*,* $Q = \{(d_2, 8), (d_7, 10)\}$*;*
step 7: *Enqueue* d_2*:* d_2 *is an object and added to* $\Re$*,* $Q = \{(d_7, 10)\}$*;*
step 8: *Enqueue* d_7*:* d_7 *is an object and added to* $\Re$*;*
step 9: $\Re.size = 2$*, then algorithm terminate and return* $\Re$

5.2 Query Processing

The SG-Tree query method has two advantages. One is using the text index signature to prune on text and the other is to conduct spatial pruning using the distance-first algorithm. It is SG-Tree that simultaneously prunes on the spatial and text two dimensions. In algorithm 2, all node signatures, occurrence lists and the priority queue are initialized in lines 1-4. For the objects in the queue, we iteratively dequeue and handle each element in the queue separately with three possibilities, such as a leaf node, a non leaf node and a spatial object respectively in lines 7, 15, 21 respectively. The priority queue is employed to keep objects accessed during the G-Tree traversing. It is used to determine the signature of a node whether it matches the query signature. If it does not match the whole sub-trees, it will be pruned in lines 11, 17. If the node matches the signature of the query, then it will be pushed into the priority queue according to their network distance. The algorithm will safely terminate when k answers are returned or the priority queue is empty.

Algorithm 2. *SignatureGtreeQueryProcessing(q.loc,q.term,k,T_g)*

Input: *q.loc, q.term, k, T_g.*
Output: $\Re$ objects satisfying the query condition.

1: **Initialize** : $S \leftarrow$ node; initialize all node signature
2: **Initialize** : $\Re := \emptyset; Q := \emptyset$; $Gamma = \emptyset$;
3: $Q.Enqueue(\langle T_{g_{root}}, 0\rangle)$
4: $W \leftarrow$ signature($q.term$)
5: **while** $Q \neq \emptyset$ and $|\Re| \leq k$ **do**
6: $n = Q$.pool();
7: **if then**n is a leaf node then
8: **if** $q.loc \in n$ **then**
9: $mindist\_inside\_leaf(q.loc, n)$;
10: **else** $mindist\_outside\_leaf(q.loc, n)$;
11: **if** S matches W **then**
12: **for** each $o \in n$ **do**
13: **if** o contain $q.term$ **then**
14: $\Gamma \leftarrow o$; $Q.Enqueue$(o,$SPDist(q.loc, o)$);
15: **else if** n is a non-leaf node **then**
16: **for** each child node $c \in \Gamma(n)$ **do**
17: **if** S matches W **then** $\Gamma \leftarrow c$;
18: **if** $q.loc$ is in c **then**
19: $Q.Enqueue(c, SPDist(q.loc, c) = 0)$;
20: **else** $mindist\_outside\_nonleaf(q.loc, c)$;
21: **else if** n is is an object **then** insert then insert n into $\Re$;
return $\Re$;

6 Experimental Evaluation

In this section, we will investigate our three approaches on three real-world road networks according to different evaluation criterions.

6.1 Setup

In this section, we evaluate the performance of our three approaches SNE, FITG and SG-Tree. The experiments are conducted on three real datasets which are road networks of CAL, NA, and SF(shown in Table 1) respectively. All datasets are obtained from the webset[1]. The spatio-textual objects are obtained from the US Board on Geographic Names[2] in which each objects is composed of a short text description with a geographic location. Table 1 summaries the detail of three datasets. Note that the scalability of G-Tree have been proved in paper [15], so do not repeat it. Our experiments were executed on Linux computer with 3.0 GHz CPU Inter processor and 4G RAM.

Table 1. Summary of the three Datasets

Data	Description	Vertices	Edge	Spatio textual object size(MB)
CAL	California	21,048	21,693	13.1
NA	North America	175,812	179,178	25.7
SF	San Francisco	174,956	223,001	62.9

6.2 Experimental Results

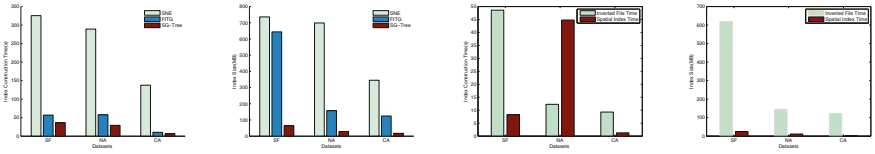
In this section, we will evaluate three approaches of spatial keyword query on road networks from different perspectives, such as index construction time, index size, a varied number of results, and a varied number of query keywords.

Evaluation on Index Construction: First, we evaluate the space and time consumption overhead of the index construction of three methods. Fig.5(a) illustrates the time consumption of the three index methods on the three different real datasets. From Fig.5(a), we can see that the index construction time of the baseline method is significantly more than that of the other two methods proposed in this paper. This is because SNE method utilizes the CCAM structure, it needs to take the longest time to partition the edges and to create the signature of each edge. As is shown in Fig.5(c), FITG method creating the spatial index for a road network dose not need to spend too much time so that the most of that time can be consumed for text indexing. The greater the spatio-textual object on a road network, the more time it requires. Note that its time spending on index creating index increases when the text data is relatively large. SG-Tree builds signatures based on a pseudo-document for each node on the G-Tree, so

[1] http://www.cs.utah.edu/~lifeifei/SpatialDataset.htm
[2] http://geonames.usgs.gov

it is more efficient than FITG. Both FITG and SG-Tree approaches spend less time creating index than of the baseline approach.

Evaluation on Index Size: Fig.5(b) shows that the index size of three approaches on three real different datasets. It can be seen that both baseline and FITG approaches need more external memory space to store index, which dues to that two methods need store more index information easy to search. The Baseline method, utilizing the CCAM structure, which needs more space to store adjacency information of each vertex and each edge. Besides, it also has to store signatures, so that its external memory consumption is very high. As we can see from Fig.5(d) that FITG needs more external memory because it keeps the inverted index. Fig.5(d) shows that the spatial index G-Tree only needs small extra space, while the Inverted File requires more space. In contrast, our proposed SG-Tree method only needs a very small space to store nodes and the signature on each node of G-Tree.

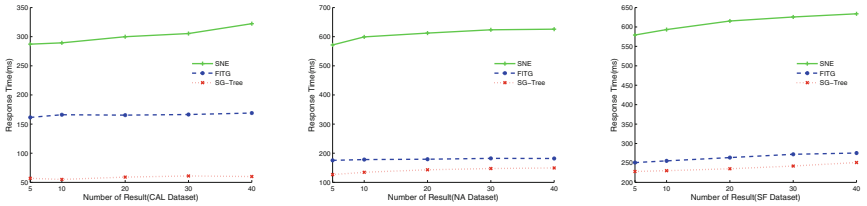


(a) Index construction time (b) Index size (c) FITG Index construction time (d) FITG Index size

Fig. 5. Index Construction Size and Index Size

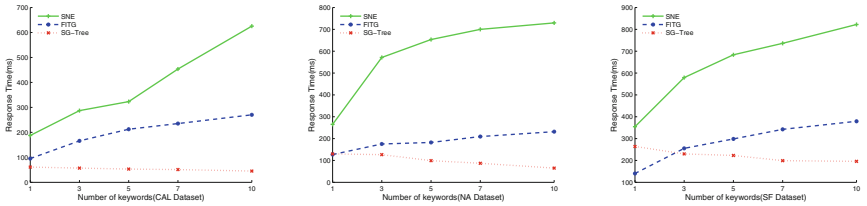
Evaluation on the varied Number of Results: Fig.6 depicts the response time within various numbers of results on the different datasets and the query has three keywords. We compare these three methods with varying number of results in our experiments. Fig.6(a), Fig.6(b) and Fig.6(c) show that both SG-Tree and FITG have better performance than that of SNE on three datasets. We can also find that the response time of SNE has little increment with the increment of the number of results. The reason is that SNE method in the query process needs to expand access edges. Thus, it needs to spend more time to check edges whether they contain the keywords when the result numbers are increased. However, the response time of both FITG and SG-Tree is almost unchanged. This is because both FITG and SG-Tree only need to calculate the shortest distance of few objects and this process does not need to spend much time.

Evaluation on the Number of Query Keywords: We evaluate the response time under the varying number of query keywords on different real datasets. The experimental results are shown in Fig.7. It shows that SG-Tree method performs much better than SNE and FITG. It can be ascribed to that it bypasses the hierarchy pruning on tree. With the keywords increasing, more and more nodes are pruned, so that the number of candidates becomes small. In Fig.7(a), we



(a) Dataset: CAL (b) Dataset: NA (c) Dataset: SF

Fig. 6. Varying Numbers of Results on Different three Datasets



(a) Dataset: CAL (b) Dataset: NA (c) Dataset: SF

Fig. 7. Varying Numbers of Keywords on Different three Datasets

can see that with the increase in the number of keywords, the processing time in method FITG and SNE are growing gradually, while in method SG-Tree it is reducing. The same performance also can be seen in Fig.7(b) and Fig.7(c). The reason is that SG-Tree Method can prune lots of unrelated objects through eliminating many more G-Tree nodes. On the contrary, FITG and SNE need to check whether the object contains the keyword repeatedly. When the query has one keyword, SNE performance is reasonable. With the increment of the number of query keywords, its response time rises rapidly because it has to expand large edges. Compared to SNE, FITG performs better because it dose not expand edges and can save a lot of time.

In all, our experimental results show that our proposed two hybrid indexes and query algorithms outperform the baseline method. In particular, the SG-Tree index exploits the advantages of the hierarchical tree pruning on the text and spatial dimensions, and improves the query efficiency tremendously.

7 Conclusion

In this paper, we analyzed the advantages and disadvantages of the existing methods on road networks. After that, we proposed three novel approaches (SNE, FITG, and SG-Tree) to achieve rapid and efficient spatial keyword search. Our experimental results on real-world road networks show that the SG-Tree index structure is the most efficient method.

In the future, since real-time is quite important for urban planning, transportation planning, route planning with heavy traffic, we will integrate temporal information into spatial keyword query on road networks.

Acknowledgments. This work was partially supported by Chinese NSFC project (61170020, 61402311, 61440053), and the US National Science Foundation (IIS-1115417).

References

1. Cao, X., Chen, L., Cong, G., Jensen, C.S., Qu, Q., Skovsgaard, A., Wu, D., Yiu, M.L.: Spatial keyword querying. In: Atzeni, P., Cheung, D., Ram, S. (eds.) ER 2012 Main Conference 2012. LNCS, vol. 7532, pp. 16–29. Springer, Heidelberg (2012)
2. Cao, X., Cong, G., Jensen, C.S., Ooi, B.C.: Collective spatial keyword querying. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data, pp. 373–384. ACM (2011)
3. Chen, L., Cong, G., Jensen, C.S., Wu, D.: Spatial keyword query processing: an experimental evaluation. Proceedings of the VLDB Endowment **6**(3), 217–228 (2013)
4. De Felipe, I., Hristidis, V., Rishe, N.: Keyword search on spatial databases. In: IEEE 24th International Conference on Data Engineering, ICDE 2008, pp. 656–665. IEEE (2008)
5. Jensen, C.S., Kolářvr, J., Pedersen, T.B., Timko, I.: Nearest neighbor queries in road networks. In: Proceedings of the 11th ACM International Symposium on Advances in Geographic Information Systems, pp. 1–8. ACM (2003)
6. Lee, K.C., Lee, W.C., Zheng, B.: Fast object search on road networks. In: Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, pp. 1018–1029. ACM (2009)
7. Lee, K.C., Lee, W.C., Zheng, B., Tian, Y.: Road: a new spatial object search framework for road networks. IEEE Transactions on Knowledge and Data Engineering **24**(3), 547–560 (2012)
8. Li, W., Guan, J., Zhou, S.: Efficiently evaluating range-constrained spatial keyword query on road networks. In: Han, W.-S., Lee, M.L., Muliantara, A., Sanjaya, N.A., Thalheim, B., Zhou, S. (eds.) DASFAA 2014. LNCS, vol. 8505, pp. 283–295. Springer, Heidelberg (2014)
9. Papadias, D., Zhang, J., Mamoulis, N., Tao, Y.: Query processing in spatial network databases. In: Proceedings of the 29th International Conference on Very Large Data Bases, vol. 29, pp. 802–813. VLDB Endowment (2003)
10. Rocha-Junior, J.B., Nørvåg, K.: Top-k spatial keyword queries on road networks. In: Proceedings of the 15th International Conference on Extending Database Technology, pp. 168–179. ACM (2012)
11. Samet, H., Sankaranarayanan, J., Alborzi, H.: Scalable network distance browsing in spatial databases. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 43–54. ACM (2008)
12. Shekhar, S., Liu, D.R.: Ccam: A connectivity-clustered access method for networks and network computations. IEEE Transactions on Knowledge and Data Engineering **9**(1), 102–119 (1997)

13. Zhang, C., Zhang, Y., Zhang, W., Lin, X.: Inverted linear quadtree: efficient top k spatial keyword search. In: 2013 IEEE 29th International Conference on Data Engineering (ICDE), pp. 901–912. IEEE (2013)
14. Zhang, D., Tan, K.L., Tung, A.K.: Scalable top-k spatial keyword search. In: Proceedings of the 16th International Conference on Extending Database Technology, pp. 359–370. ACM (2013)
15. Zhong, R., Li, G., Tan, K.L., Zhou, L.: G-tree: an efficient index for knn search on road networks. In: Proceedings of the 22nd ACM International Conference on Conference on Information & Knowledge Management, pp. 39–48. ACM (2013)
16. Zhou, Y., Xie, X., Wang, C., Gong, Y., Ma, W.Y.: Hybrid index structures for location-based web search. In: Proceedings of the 14th ACM International Conference on Information and Knowledge Management, pp. 155–162. ACM (2005)