Efficiently Evaluating Range-Constrained Spatial Keyword Query on Road Networks

Wengen Li[1], Jihong Guan[1(✉)], and Shuigeng Zhou[2]

[1] Department of Computer Science and Technology,
Tongji University, Shanghai, China
lwengen@gmail.com, jhguan@tongji.edu.cn
[2] School of Computer Science, Fudan University, Shanghai, China
sgzhou@fudan.edu.cn

Abstract. With the rapid development of geo-positioning technologies, spatial information retrieval plays an important role in a wide spectrum of applications, e.g., online maps and location-based services. Specifically, spatial keyword query (*SK* query), considering both spatial proximity to the query location and textual relevance to the query keywords, is now a hot research topic in database community. This paper addresses a specific type of *SK* query, termed *range constrained spatial keyword query* (*RC-SK* query), which searches for all the POIs (points of interest) whose textual description is relevant to the query keywords within a specified area. Though *RC-SK* query has received extensive studies in Euclidean space, little is done to deal with it on road networks. In this paper, alternative approaches with different indexing strategies are proposed to solve this problem. Extensive empirical studies on multiple real datasets demonstrate the efficiency of these proposed approaches.

Keywords: Range-constrained spatial keyword query · Road networks · Hierarchy indexing

1 Introduction

The rapid development of techniques for both geo-positioning and mobile communication has made location aware query a necessary part in many applications. In this paper, we consider a specific type of such query called *range constrained spatial keyword query*, *RC-SK* query for short, on road networks. Concretely, a *RC-SK* query, specified with a spatial location and a set of query keywords, is targeted for finding all the POIs whose textual relevance to the query keywords is larger than a specified threshold and location is within a specified distance to the query location. For instance, a visitor poses a query to search for all the banks offering exchange service within 2 km from his or her current location. Here, a bank is a POI with a spatial location (e.g., longitude and latitude) and a piece of textual description about the services it offers.

Actually, there have been some works on *RC-SK* query in Euclidean space. In reality, however, people's trajectories are usually constrained by road networks.

W.-S. Han et al. (Eds.): DASFAA 2014, LNCS 8505, pp. 283–295, 2014.
DOI: 10.1007/978-3-662-43984-5_21,

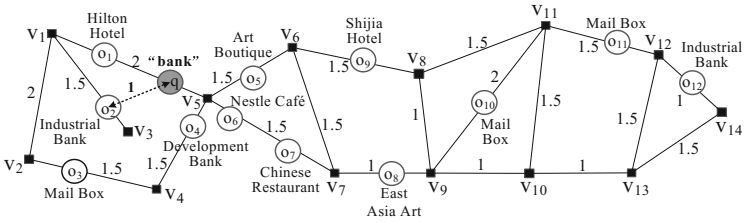


Fig. 1. An example of *RC-SK* query. Here, $dis(q, o_2) = 2.5$.

Figure 1 illustrates an example of *RC-SK* query on a small road network which has 14 vertices $v_i (i = 1, \ldots, 14)$ and 12 POIs $o_j (j = 1, \ldots, 12)$ denoted as black grids and circles, respectively. Each edge is labeled with its length. The query q, illustrated by a grey filled circle, searches for all the banks within 2 km. In Euclidean space, both o_2 and o_4 will be returned as the result. However, we cannot reach o_2 within 2 km along the road network. Hence, it is more practicable to conduct *RC-SK* query based on network distance than Euclidean distance.

However, conducting *RC-SK* query on road networks is much more challenging than that in Euclidean space because the shortest path between query location and any candidate POI should be computed. Especially for a larger query range, we need to enumerate many POIs and compute their shortest distances to the query location.

In this paper, three approaches are proposed to deal with *RC-SK* query on road networks. The first one is *expansion-based approach* (*EA*), a baseline approach, which traverses the road network from the query location with the same flavor as Dijkstra's algorithm. The second approach is *Euclidean heuristic approach* (*EHA*) which is an improvement of *EA* approach and employs Euclidean heuristic to accelerate query processing. As both *EA* and *EHA* have to traverse the road network vertex by vertex, they are inefficient for a large query range. To solve this problem, the third approach called *Rnet Hierarchy* [1] *based approach* (*RHA*) is proposed. *RHA* partitions the whole road network into a group of interconnected subnets and organizes them in a hierarchy structure, which greatly improves the query efficiency.

The remainder of this paper is organized as follows. Section 2 reviews the related work and Sect. 3 formally defines the problem. Sections 4–6 elaborate *EA*, *EHA* and *RHA*, respectively. Section 7 empirically evaluates the proposed approaches and Sect. 8 concludes the paper.

2 Related Work

Generally, there are two types of widely used spatial keyword queries [7], i.e., top-k spatial keyword query (top-k *SK* query) [13,14], searching for the k best POIs based on both spatial proximity and textual relevance, and range-constrained spatial keyword query (*RC-SK* query) [8,16], searching for all the POIs satisfying the required textual relevance within a specified area.

Table 1. Hybrid indices.

References	Hybrid index	Spatial index	Textual index
[15]	IR2-tree	R-tree	Signature file
[11]	IR-tree	R-tree	Inverted file
[8,16]	KR*-tree	R*-tree	Inverted file
[18]	bR*-tree	R*-tree	Bitmap

During the past decade, *RC-SK* query has received extensive studies in Euclidean space. The original solution [8] to *RC-SK* query retrieves all the POIs within the query range area and conducts a detailed examination on these POIs based on their textual relevance, which is inefficient for large-size datasets. To solve this problem, previous works try to embed traditional textual indices, such as inverted file and signature file [12] into an R-tree [9], a widely-used index structure for multi-dimensional data, or its variants. Table 1 shows major hybrid schemes that merge text index and spatial index.

Almost all the proposed approaches for *RC-SK* query in Euclidean space are based on these hybrid indices. During query processing, spatial proximity and textual relevance are computed simultaneously, which make it efficient to prune irrelevant branches as soon as possible. However, all these index structures and processing algorithms are devised for spatial keyword queries in Euclidean space and cannot be directly used for *RC-SK* query on road networks.

In addition, Rocha-Junior et al. [10] proposed several efficient approaches to address top-k *SK* query on road networks, which is the most related work to ours. The framework of their overlay approach is similar to that of our *RHA*. However, both the partition strategy and index structure of *RHA* are different from those of the overlay approach. More importantly, we aim at evaluating *RC-SK* query instead of top-k *SK* query on road networks.

3 Problem Statement

Formally, a road network is represented as an undirected graph $G = (V, E)$, where V and E are the sets of vertices and edges, respectively. Each vertex $v \in V$ represents a road intersection or a road endpoint; each edge $e_{i,j} \in E (i \neq j)$ represents the road segment connecting v_i and v_j and its length is denoted as $|e_{i,j}|$. The distance between two vertices u and v, $dis(u, v)$, is the length of the shortest path between them.

A POI o is represented as $o = (l, e, d, K)$, where $o.l$ is the spatial location consisting of longitude and latitude, $o.e$ is the edge on which o resides, $o.d$ is the distance from $o.l$ to the beginning vertex of $o.e$, and $o.K$ is a set of keywords which describe the details of o.

A *RC-SK* query q over G is defined as $q = (l, K, \tau, r)$, where $q.l$ is the query location, $q.K$ is a set of query keywords, $q.\tau \in (0, 1]$ is a predefined textual

relevance threshold and $q.r$ specifies the query range. The answers to q are the set of POIs on G such that each of them satisfies

$$dis(o.l, q.l) \leq q.r \wedge \theta(o.K, q.K) \geq q.\tau$$

where $dis(o.l, q.l)$ is the distance between $o.l$ and $q.l$, $\theta(o.K, q.K)$ is the textual relevance between $o.K$ and $q.K$ and defined as follows [3].

$$\theta(o.K, q.K) = \frac{\sum_{k \in q.K} w_{k,o.K} \cdot w_{k,q.K}}{\sqrt{\sum_{k \in o.K} (w_{k,o.K})^2 \cdot \sum_{k \in q.K} (w_{k,q.K})^2}} \quad (1)$$

where $w_{k,o.K} = 1 + ln(f_{k,o.K})$, $f_{k,o.K}$ is the occurrences of query keyword $k \in q.K$ in $o.K$; $w_{k,q.K} = ln(1 + \frac{|P|}{df_k})$, where $|P|$ is the number of POIs on G, df_k is the number of POIs containing k.

Although the definition above and the following approaches are based on undirected road networks, they can be extended to directed road networks with only a little modification.

4 The Expansion-Based Approach

This baseline approach processes *RC-SK* query in an expansion fashion like Dijkstra's algorithm.

4.1 Index Structure

An R*-tree [2] is employed to index all edges in E as illustrated in Fig. 2 where each edge is represented as a minimum bounding rectangle that totally encloses it. With the help of the R*-tree, the edge on which $q.l$ resides can be quickly determined with a spatial point query.

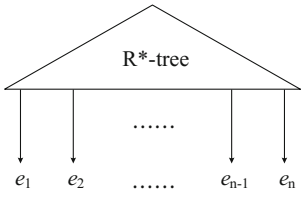


Fig. 2. R*-tree for edges.

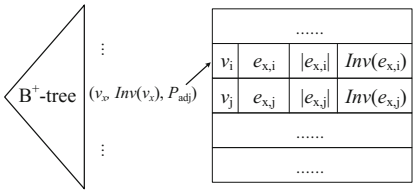


Fig. 3. Index for road networks.

Meanwhile, a B^+-tree is used to index the modified adjacent lists as shown in Fig. 3 which keeps the connectivity of G. Each entry in leaf node is a triple

$(v_x, Inv(v_x), P_{adj})$, where v_x is a vertex, $Inv(v_x)$ is a pointer to the inverted file [3] (called *vertex inverted file*) which covers all the POIs on v_x's adjacent edges. P_{adj} is another pointer pointing to the adjacent list of v_x. Each entry in the adjacent list is a quadruple $(v_i, e_{x,i}, |e_{x,i}|, Inv(e_{x,i}))$, where v_i is a neighbor vertex of v_x, $e_{x,i}$ is the edge between v_x and v_i with length $|e_{x,i}|$, and $Inv(e_{x,i})$ is a pointer to the inverted file (called *edge inverted file*) covering all POIs on $e_{x,i}$.

4.2 Query Processing

Initially, a priority queue U is created to store visited vertices during expansion based on their network distances to $q.l$. Meanwhile, a list L is created to store query results. First, the edge $e_{i,j}$ on which $q.l$ resides is located by using the R*-tree built for edges. Then a verification is conducted on $e_{i,j}$ to check whether it contains any POI whose textual relevance to $q.K$ is larger than $q.\tau$. Next, both v_i and v_j are inserted into U with their distances to $q.l$. By obtaining vertices from U and checking their adjacent edges, we can traverse all edges within $q.r$ from $q.l$ and verify them in the same way as we do for $e_{i,j}$. During the verification, POIs satisfying the textual relevance threshold are added to L.

Consider the query q over the road network in Fig. 1, where $q.l$ is the filled circle, $q.K =$ "bank", $q.r = 2$. For presentation simplicity, we ignore $q.\tau$ and only require that each returned POI contains $q.K$. First, we find that $q.l$ is located on $e_{1,5}$ which has no desirable POIs. Then, v_1 and v_5 are inserted into U with $(v_5, 0.5)$ and $(v_1, 1.5)$, respectively. Here, we assume $|q.l, v_1| = 1.5$ and $|q.l, v_5| = 0.5$. Next, we get $(v_5, 0.5)$ from U. By checking the inverted file $Inv(v_5)$ for v_5, we find that $e_{5,4}$ contains query keyword "bank". Then $Inv(e_{5,4})$ is checked and o_4 is inserted into L. As the distance from $q.l$ to v_4 is 2, we do not search beyond v_4, which is same for v_6 and v_7. Following that, we get $(v_1, 1.5)$ from U and its adjacent edges $e_{1,2}$ and $e_{1,3}$ are checked, and no POIs (assume $|o_2, v_1| = 1$, we have $dis(q.l, o_2) = 2.5 > 2$) are inserted into L. Now U is empty and the algorithm terminates. Finally, we get the query result $L = \langle o_4 \rangle$.

5 The Euclidean Heuristic Approach

EA is efficient enough for *RC-SK* queries with a small $q.r$. However, if $q.r$ is very large and numerous edges and POIs are covered, *EA* will incur a considerable overhead to obtain all desirable POIs because it has to verify the inverted files of all relevant edges (containing any query keyword). In general, however, a large portion of keywords cover only a small number of POIs. In such a situation, to check the inverted files for all relevant edges is unnecessary. To overcome this drawback, we propose the *Euclidean heuristic approach* (*EHA*).

Intuitively, Euclidean distance between any two vertices on a road network is always smaller, if not equal to, than the network distance between them. Therefore, if a POI belongs to the query result based on network distance, then it must be in the query result based on Euclidean distance. Based on this observation, we propose the *EHA* to first retrieve all the candidate POIs according to any

state-of-the-art Euclidean distance based approach. Here we employ IR-tree [11] which augments each node of the R-tree with an inverted file. The set of edges having candidate POIs (satisfying textual relevance threshold) is recorded as E_q. During the expansion process, we avoid verifying the inverted file of a particular edge by checking whether it is contained in E_q. Thus, edges containing no desirable POIs are filtered. *EHA* is implemented based on *EA* by adding a *SK* query on the IR-tree at the beginning and an edge set check during expansion. We omit the detail here due to space limit.

6 The Rnet Hierarchy-Based Approach

Essentially, both *EA* and *EHA* expand from one vertex to another on G within $q.r$, which makes it very expensive to evaluate *RC-SK* queries with a large $q.r$. To solve this problem, we introduce the Rnet Hierarchy [1] to index road networks and further propose *Rnet Hierarchy-based approach*, *RHA* for short.

6.1 Indexing Structure

Rnet Hierarchy partitions a road network into connected subnets called Rnets (regional nets) and organizes them in a hierarchy structure as depicted in Fig. 4. An Rnet R is defined as (V_R, E_R, B_R) where V_R, E_R, and B_R are the sets of vertices, edges and border vertices of R, respectively. B_R are the vertices shared by two or more Rnets (e.g., v_5). In Fig. 4, there are four Rnets R_{11}, R_{12}, R_{21} and R_{22} at level 1 and two larger Rnets R_1 and R_2 at level 2. R_1 encloses R_{11} and R_{12}, and R_2 encloses R_{21} and R_{22}. Level 0 is the original road network. Therefore, a road network is organized as a group of connected Rnets at each level.

In order to skip over Rnets, *shortcut* is introduced. A *shortcut* is the shortest path between two border vertices of an Rnet, e.g., $SP(v_5, v_9)$. With the help of shortcuts on different levels, a search expands quickly with different step sizes. Here, a challenge is how to partition a road network into a group of Rnets with a minimum number of border vertices. In this paper, we first consider the *equal-size partition* [1], which adopts the geometric approach [4] and *KL* algorithm [5], to partition the whole road network into Rnets of the similar size. *Equal-size*

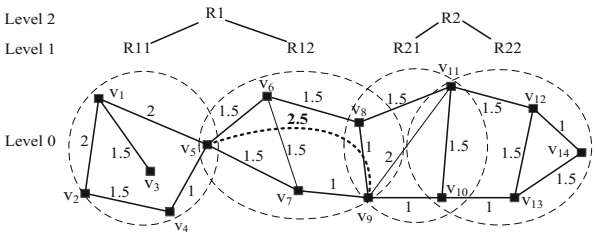


Fig. 4. An example of Rnet Hierarchy of a road network.

partition first partitions the whole road network into two parts with almost the same number of edges, and then tunes them by exchanging edges to reduce the border vertices. By doing this recursively, G is partitioned into a set of Rnets with almost the same size.

However, the partition method above ignores the road network's semantics. In reality, POIs on road networks are often clustered [6] in some hot areas like commercial centers. For example, area around v_5 in Fig. 1 has more POIs than areas around other vertices (e.g., v_{13}). Figures 5 and 6 display the POI distribution on the road network of London. Obviously, most vertices have less than 5 POIs (POIs residing on the edges adjacent to the vertex). Accordingly, we consider partitioning a road network based on the distribution of POIs, i.e., *distribution-aware partition* and aim to partition as many as POIs into the same Rnet. To this end, we first collect all the vertices with more POIs than a specified parameter (e.g., 10) and then merge these vertices to form larger areas based on their spatial proximity until a Rnet is generated. Meanwhile, the other areas are partitioned by using the *equal-size partition*.

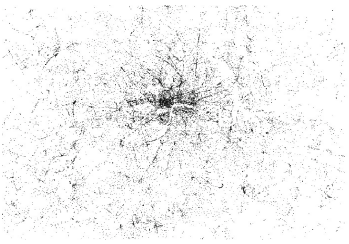


Fig. 5. POI distribution of London

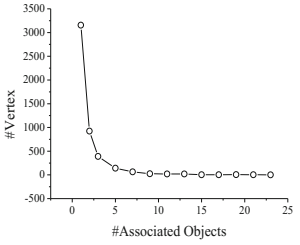


Fig. 6. POI distribution statistics

Rnet Hierarchy is organized using a B^+-tree as illustrated in Fig. 7. The B^+-tree indexes all vertices based on their identifiers and each entry in a leaf node points to an adjacent list or a *hierarchy tree*. Concretely, if a vertex v_i (e.g., v_2) is not a border vertex, the entry for v_i has a pointer pointing to an adjacent list just as *EA*. Otherwise, the entry for v_i has a pointer pointing to a *hierarchy tree* which records the organization of all the Rnets associated with v_i at different levels. For example, v_5 is a border vertex of R_{11} and R_{12}, and it has a *hierarchy tree* T_{v_5}. The root node of T_{v_5} contains two entries E_{R11} (for R_{11}) and E_{R12} (for R_{12}) and a pointer pointing to the inverted file for them. In a *hierarchy tree*, each entry in the intermediate nodes also stores all the shortcuts within the corresponding Rnet while each entry in the leaf nodes points to the adjacent list of the border vertex.

In addition, to utilize Euclidean heuristic to quickly find out those Rnets that contain desirable POIs, we also try to organize all Rnets at different levels into a variant IR-tree as illustrated in Fig. 8.

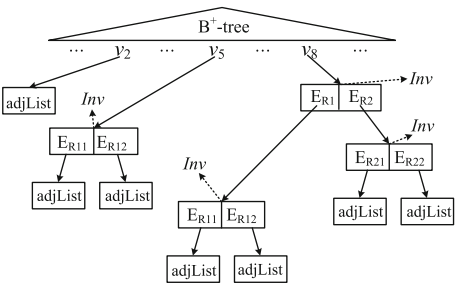


Fig. 7. Index for the Rnet Hierarchy

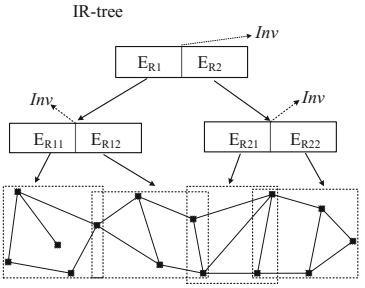


Fig. 8. IR-tree for Rnets

6.2 Query Processing

Given a query q, *RHA* searches for POIs in an expanding fashion as detailed in Algorithm 1. If a vertex v_i is not a border vertex, it proceeds in the same way as *EA*. Otherwise, the hierarchy tree T_{v_i} of v_i is checked. First, we examine the inverted file for the root node of T_{v_i} to check whether it contains desirable POIs. If not, we skip over the entire Rnet through the shortcuts without checking its inner edges. Otherwise, we check its child nodes to further retrieve desirable POIs.

Algorithm 1. The Rnet Hierarchy based Approach

Input: $G = (V, E)$, $q = (q.l, q.K, q.\tau, q.r)$
Output: Any POI o such that $dis(o.l, q.l) \leq q.r \wedge \theta(o.K, q.K) \geq q.\tau$

```
1: U=newPriorityQueue();     // used to store visited vertices during processing
2: L=newList();     // used to store final result
3: e_q=locateEdge(q.l);
4: U.enqueue(e_q.v1, |q.l, e_q.v1|);
5: U.enqueue(e_q.v2, |q.l, e_q.v2|);
6: checkEdge(e_q);     // check Inv(e_q)
7: while not U.isEmpty() do
8:   v=U.Dequeue();
9:   if not v.isBorderVertex()   // v is not a border vertex
10:    for each adjacent edge e of v
11:      if e contains q.K
12:        checkEdge(e);
13:      if(v.currentDistance+e.length< q.r)
14:        U.enqueue(e.anotherVertex);
15:  else   // v is a border vertex
16:    check v.HierarcyTree;
17: return L;
```

Table 2. Statistics of datasets.

Attributes	Dublin	London	Australia	BritishIsles
#vertices	62,975	209,406	1,223,171	3,760,213
#edges	82,730	282,267	1,682,182	4,865,094
#POIs	5,297	34,341	70,064	300,891
#keywords	15,216	97,824	193,106	842,369
#distinct keywords	3,563	12,522	18,789	60,558

Table 3. Parameters in experiments.

Parameters	Values
$q.r$	1, 3, **5**, 7, 9 (km)
$q.\tau$	0.3, **0.5**, 0.7, 0.9, Boolean
$\|q.K\|$	1, **2**, 3, 4, 5
Datasets	Dublin, London, Australia, **British Isles**
Partition strategy	***Equal-size partition***, *Distribution-aware partition*

For example, we consider the same query in Fig. 1 with a larger $q.r = 3$. First, we verify the edge $e_{1,5}$ on which $q.l$ resides. Then, $(v_5, 0.5)$ and $(v_1, 1.5)$ are inserted into U. Next, we obtain $(v_5, 0.5)$ from U. As v_5 is a border vertex, we evaluate R_{11} and R_{12}. R_{11} contains *bank* and a detailed examination is conducted. Then $e_{5,4}$ is checked and o_4 is added to L. Meanwhile, $(v_4, 2)$ is inserted into U. R_{12} contains no desirable POIs and its shortcuts from v_5 to v_8 and v_9 go beyond the query range. Then, we get $(v_1, 1.5)$ and $(v_4, 2)$ in order, and no more POIs are added to L. Finally, the result $L = \langle o_4 \rangle$ is returned.

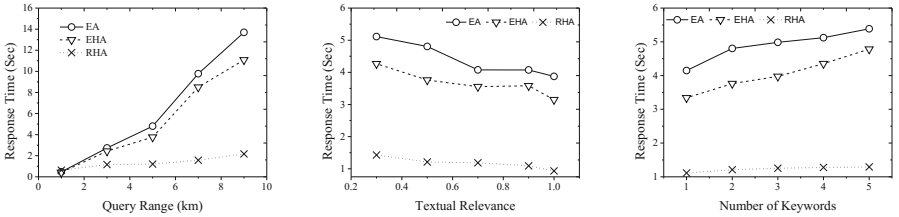
As an expected improvement, we employ the IR-tree in Fig. 8 to accelerate query processing by getting all Rnets containing desirable POIs in advance, which avoids checking the inverted file for each Rnet.

7 Experimental Evaluation

7.1 Setup

The performance of proposed approaches is evaluated on four real datasets[1] which are road networks of Dublin, London, Australia, and British Isles, respectively. Table 2 displays some statistics of the four datasets. For each dataset, we randomly generate 500 locations within the road network area as query locations and 500 sets of keywords of size 1, 2, 3, 4, and 5, separately. Table 3 lists the parameters used in the experiments and marks their default values in bold. Experiments run on a PC with a 3.1 GHz Intel processor and a 4 GB RAM. The index structures of the three approaches are disk-resident and the buffer is set at 4 MB.

[1] http://www.idi.ntnu.no/~joao/publications/EDBT2012/



(a) Vary $q.r$ (b) Vary $q.\tau$ (c) Vary $|q.K|$

Fig. 9. Response time while varying $q.r$, $q.\tau$, and $|q.K|$

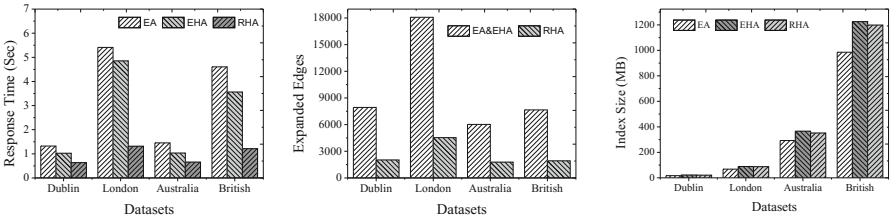
7.2 Experimental Results

Varying $q.r$. Figure 9(a) illustrates the response time while varying $q.r$. *EA* performs well for queries with a small $q.r$. With the increase of $q.r$, however, the response time increases rapidly because *EA* has to expand all the edges within $q.r$ from $q.l$. Compared to *EA*, *EHA* performs better because it avoids checking the inverted file for every edge that contains any query keyword. This differs from the conclusion in [17] that network expansion algorithms performs better than Euclidean distance heuristic based algorithms, because *EHA* just uses Euclidean heuristic to avoid unnecessary edge examination. However, both *EA* and *EHA* expand vertex by vertex, which inevitably incurs a high overhead. *RHA* performs much better than *EA* and *EHA* because it bypasses the Rnets containing no desirable POIs and avoids a detailed examination on their inner edges. Additionally, with the help of different layers and different size of Rnets, *RHA* works well with different $q.r$.

Varying $q.\tau$. Figure 9(b) shows the response time while varying the textual relevance $q.\tau$. A smaller $q.\tau$ covers more POIs and it consumes more time to evaluate these POIs. Both *EA* and *EHA* cost more than 3 s to evaluate a *RC-SK* query while varying $q.\tau$ from 0.3 to 0.9. As for *RHA*, only about one second is required. Because no textual relevance is computed, Boolean query $(q.\tau = 1)$[2] takes less time than other queries.

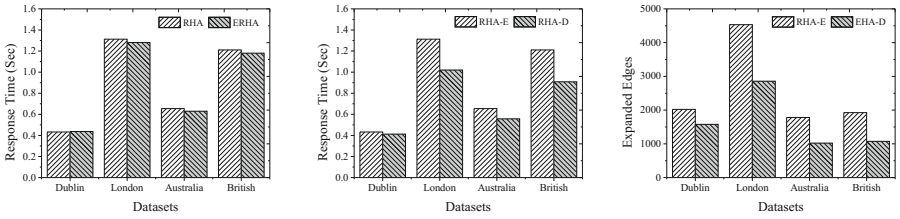
Varying$|q.K|$. Figure 9(c) presents the response time while varying the number of query keywords $|q.K|$. With the increase of $|q.K|$, both *EA* and *EHA* have to verify more POIs relevant to $q.K$, which leads to an increase in query time. Although *RHA* also needs more time to evaluates *RC-SK* queries with more keywords, the increase of response time is very slow because *RHA* computes the textual relevance between an Rnet and $q.K$, and there is no detailed examination if the relevance is less than $q.\tau$. Therefore, increasing query keywords affects only a small portion of Rnets and the other Rnets are still skipped through shortcuts.

[2] This value is just a label for Boolean query instead of a textual relevance value.



(a) Response time (b) Expanded edges (c) Index size

Fig. 10. Response time, expanded edges and index size for different datasets



(a) ERHA (b) Response time (c) Expanded edges

Fig. 11. Response time using ERHA, response time and expanded edges while varying partition strategies

Different Datasets. Figure 10(a) shows the response time for different datasets. *RHA* evaluates *RC-SK* queries on the four datasets of different size in about one second. We can find that the response time on Australia network is smaller than that on London network. This is because the London network is denser than Australia network. In general, given the same query range, London network usually has more POIs and edges than Australia network. This can also be seen from Fig. 10(b), which illustrates the number of edges expanded during query processing.

Index Size. Figure 10(c) shows the index size of the three approaches on different datasets. *EHA* has a larger index size than *EA* because it constructs an IR-tree for all POIs on the road network. *RHA* and *EHA* have all most the same index size.

Euclidean Heuristic Based RHA. Figure 11(a) presents the response time of the Euclidean heuristic based *RHA* (*ERHA*) that indexes all Rnets with an IR-tree as illustrated in Fig. 8. As *RHA* indexes a road network in a hierarchy and is already able to prune unrelated Rnets, *RHA* and *ERHA* have no much difference in response time.

Partition Strategy. Figure 11(b) and (c) illustrate the response time and number of expanded edges while *distribution-aware partition* (called *RHA-D*) is adopted. Compared to *RHA* using *equal-size partition* (*RHA-E*), *RHA-D* has a better performance. By partitioning a road network based on POI distribution, some Rnets have more POIs than others and accordingly have more keywords. Besides, areas containing few POIs can be partitioned into Rnets of larger granularity. Hence, this partition strategy makes *RHA-D* more advantageous in pruning unrelated Rnets than *RHA-E*.

8 Conclusion

In this paper, we define the *RC-SK* query on road networks and devise three approaches, i.e., *EA*, *EHA* and *RHA*, to deal with this problem. Both *EA* and *EHA* are suitable for *RC-SK* queries with a small query range while *RHA* also works excellently for *RC-SK* queries with a large query range.

In the future, we will consider some temporal spatial keyword queries over road networks because it is quite important for a city with heavy traffic.

Acknowledgement. This work was supported by National Natural Science Foundation (NSFC) under grant No. 61373036 and the Research Innovation Program of Shanghai Municipal Education Foundation under grant No. 13ZZ003.

References

1. Lee, K.C.K., Lee, W.-C., Zheng, B.: Fast object search on road network. In: Proceedings of EDBT, pp. 1018–1029 (2009)
2. Beckman, N., Kriegel, H.-P., Scheider, R., Seeger, B.: The R*-tree: an efficient and robust access method for points and rectangles. In: Proceedings of SIGMOD, pp. 322–331 (1990)
3. Zobel, J., Moffat, A.: Inverted files for text search engines. ACM Comput. Surv. **38**(2), 1–56 (2006)
4. Huang, Y.-W., Jing, N., Rundensteiner, E.A.: Effective graph clustering for path queries in digital map. In: proceedings of CIKM, pp. 215–222 (1996)
5. Kernighan, B., Lin, S.: An efficient heuristic procedure for partitioning graphs. Bell Syst. Tech. J. **49**(2), 291–308 (1970)
6. Yiu, M.L., Mamoulis, N.: Clustering objects on a spatial network. In: Proceedings of SIGMOD, pp. 443–454 (2004)
7. 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)
8. Zhou, Y., Xie, X., Wang C., Gong, Y., Ma, W.: Hybrid index structures for location-based web search. In: Proceedings of CIKM, pp. 155–162 (2005)
9. Guttman, A.: R-trees: a dynamic index structures for spatial searching. In: Proceedings of SIGMOD, pp. 47–57 (1984)
10. Rocha-Junior, J.B., Norvag, K.: Top-k spatial keyword queries on road networks. In: Proceedings of EDBT, pp. 168–179 (2012)

11. Li, Z., Lee, K.C.K., Zheng, B., Lee, W.-C., Lee, D.L., Wang, X.: IR-tree: an efficient index for geographic document search. IEEE TKDE **23**(4), 585–599 (2011)
12. Faloutsos, C., Christodoulakis, S.: Signature files: an access method for documents and its analytical performance evaluation. ACM TODS **2**(4), 267–288 (1984)
13. Rocha-Junior, J.B., Gkorgkas, O., Jonassen, S., Nørvåg, K.: Efficient processing of top-k spatial keyword queries. In: Pfoser, D., Tao, Y., Mouratidis, K., Nascimento, M.A., Mokbel, M., Shekhar, S., Huang, Y. (eds.) SSTD 2011. LNCS, vol. 6849, pp. 205–222. Springer, Heidelberg (2011)
14. Cong, G., Jensen, C.S., Wu, D.: Efficient retrieval of the top-k most relevant spatial web objects. In: Proceedings of SIGMOD, pp. 337–348 (2009)
15. Felipe, I.D., Hristidis, V., Rishe, N.: Keyword search on spatial databases. In: Proceedings of ICDE, pp. 656–665 (2008)
16. Hariharan, R., Hore, B., Li, C., Mehrotra, S.: Processing spatial-keyword (SK) queries in geographic information retrieval (GIR) systems. In: Proceedings of SSDBM, pp. 1–10 (2007)
17. Papadias, D., Zhang, J., Mamoulis, N., Tao, Y.: Query processing in spatial network databases. In: Proceedings of VLDB, pp. 802–813 (2003)
18. Zhang, D., Chee, Y.M., Mondal, A., Tung, A.K.H., Kitsuregawa, M.: Keyword search in spatial databases: towards searching by document. In: Proceedings of ICDE, pp. 688–699, (2009)