Spatial Keyword Query Processing in the Internet of Vehicles

Yanhong Li[1], Lei Shu[2], Jianjun Li[3], Rongbo Zhu[1(✉)], and Yuanfang Chen[2]

[1] College of Computer Science, South-Central University for Nationalities, Wuhan, China
liyanhong@mail.scuec.edu.cn, rbzhu@mail.scuec.edu.cn
[2] Guangdong University of Petrochemical Technology, Maoming, China
{lei.shu,yuanfang_chen}@ieee.org
[3] School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China
jianjunli@hust.edu.cn

Abstract. This paper takes the first step to address the issue of processing Spatial Keyword Queries (SKQ) in the Internet of Vehicles (IoV) environment. As a key technique to obtain location-aware information, the Spatial Keyword Query (SKQ) is proposed. It can search qualified objects based on both keywords and location information. In the IoV, with the popularity of the GPS-enabled vehicle-mounted devices, location-based information is extensively available, and this also enables location-aware queries with special keywords to improve user experience. In this study, we focus on Boolean kNN Queries. And a Spatial Keyword query index for IoV environment (SKIV) is proposed as an important part of the algorithm design to be used to improve the performance of this type of SKQ. Extensive simulation is conducted to demonstrate the efficiency of the SKIV based query processing algorithm.

Keywords: Spatial keyword queries · Internet of vehicles · Wireless data broadcast

1 Introduction

The Internet of Vehicles (IoV) as a new emerging technology has caught widely attention, and has become a research hotspot [1–3]. In a IoV environment, vehicles communicate with each other for data sharing and processing [4–6]. With the prevalence of the geographical applications and devices deployed on vehicles, e.g., vehicle-mounted GPS and Web GIS, the data used in the IoV is associated with location information [7–9]. For example, a driver can locate a restaurant based on its position and some keywords *Japanese seafood takeaway*. Compared with traditional restaurant searching, such restaurant locating provides better user experience. In this scenario, Spatial Keyword Queries (SKQ) are proposed, which can search qualified objects based on keywords and location information [10–12].

L.A. Maglaras et al. (Eds.): INISCOM 2016, LNICST 188, pp. 1–13, 2017.
DOI: 10.1007/978-3-319-52569-3_1

A spatial textual index is an important part in the SKQ processing [13, 14]. Zhou *et al.* [9] did relevant work by combining inverted indexes with R-trees. Three different combining schemes are studied: (i) inverted file and R*-tree double index, (ii) first inverted file then R*-tree, and (iii) first R*-tree then inverted file. The second scheme is better than the other two. As an improvement, Felipe *et al.* [15] proposed an index structure (IR2-tree) that integrates an R-tree and signature files. For each node of the tree, it employs a signature file, in the form of bitmap, to indicate the present of a given term (keyword) in the sub-tree of the node. IR-tree structure proposed by Cong *et al.* [10] augments an R-tree node with inverted lists. It has more powerful pruning ability since it combines R-tree and inverted lists to jointly prune the search space.

In this paper, we propose an algorithm for the spatial keyword query processing in the IoV environment, and we focus on Boolean kNN Queries.

The scientific contributions of this paper are summarized as follows:

- The paper takes the first step to deal with SKQ in IoV. Specifically, we propose an index, SKIV, to deal with SKQ in IoV under wireless broadcast environments.
- By using SKIV, an efficient algorithms is designed for Boolean kNN queries in the IoV environment.
- Extensive simulations are conducted to evaluate the performance of the SKIV based query processing algorithm on a real road network in the IoV environment. The experimental results show that the proposed SKIV based method outperforms the Fixed Partition Based method (FPB) on tuning time and access latency.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 gives some descriptions and preliminaries. Section 4 presents and describes the SKIV index. Section 5 introduces the algorithm for processing BkQ. Section 6 is the evaluation of the proposed algorithm, and we analyze the results in detail. In Sect. 7, we conclude this paper.

2 Related Work

Zhou *et al.* [9] proposed a hybrid index structure, which integrates inverted files and R*-trees, to handle spatial keyword queries. Cong *et al.* [10] proposed an index structure called IR-tree which augments each node of the R-tree with an inverted file. Moreover, a variant of the IR-tree, the DIR-tree, was proposed to incorporate document similarity when computing Minimum Bounding Rectangles.

In [12], Huang *et al.* discussed the problem of processing the moving top-k spatial keyword query. Given a moving spatial query and a set of keywords, it continuously finds the k best objects ranked according to both spatial proximity and text relevance. Li *et al.* [13] discussed the issue of direction-aware spatial keyword query. Given a query with a location, a direction, and a set of keywords,

it finds the k nearest neighbors of the query which are in the query direction and contain all the input keywords.

João *et al.* [16] explored top-k spatial keyword query processing on road networks and described how to rank objects with respect to both network distance and text relevance. Gao *et al.* [11] discussed reverse top-k Boolean spatial keyword query processing in road networks, and proposed filter-and-refinement frame work based algorithms for giving query result sets. To improve the query performance, several pruning heuristics and a data structure called count tree were proposed.

Wang *et al.* [17] discussed spatial query processing in road network for wireless data broadcast and proposed a novel index called ISW. Li *et al.* [14] discussed the problem of processing continuous nearest neighbor queries in road networks on the air. They proposed an NVD-based distributed air index (NVD-DI) to support query processing. Sun *et al.* [18] presented an air index called Network Partition Index (NPI) to support efficient spatial query processing in road networks on the air.

3 Definitions and Preliminaries

3.1 Definitions

In this study, we assume that each geo-textual object has a spatial location and a set of keywords, and we consider Boolean kNN queries in the IoV, under wireless broadcast environments. The frequently used symbols and their definitions are summarized in Table 1.

Table 1. Symbols and definitions

Notation	Definition
D	a set of points with keywords on a road network
q	a spatial query point with a set of keywords
$d_N(p, p')$	the network distance between two points p and p'
$d_E(p, p')$	the Euclidean distance between two points p and p'
$d_N^{min}(R_i, R_j)$	the minimum network distance between regions R_i and R_j
$d_N^{max}(R_i, R_j)$	the maximum network distance between regions R_i and R_j
$d_E^{min}(R_i, R_j)$	the minimum Euclidean distance between regions R_i and R_j

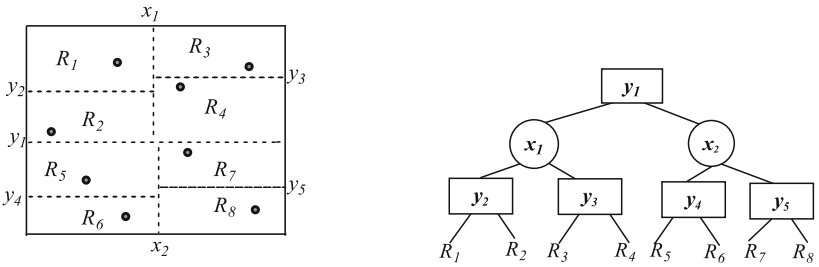
Dataset Setting. Let D be a set of geo-textual objects locating on the edges of a road network. Each geo-textual object $o \in D$ is defined as a tuple $(o.l, o.\varphi)$, where $o.l$ is a spatial location and $o.\varphi$ is a set of keywords.

Boolean kNN query (BkQ). Given a BkQ $q =< l, \varphi, k >$, where $q.l$ is q's location, $q.\varphi$ is a set of keywords, and $q.k$ is the number of result objects, the result of q, $BkQ(q)$ contains k objects such that $\forall o \in BkQ(q)$, there does not exist $o' \in D \backslash BkQ(q)$ such that $d_N(o'.l, q.l) \leq d_N(o.l, q.l) \wedge q.\varphi \subseteq o'.\varphi$.

3.2 Preliminaries

Adaptive Partition and Kd-Tree. There are many space partition methods, such as fixed partition and adaptive partition. The fixed partition method divides the search space into fixed-size grid cells. This method is simple and efficient when the data objects are evenly distributed in the search space. However, in the real world, the objects are usually unevenly distributed in the space. To adapt to skewed object distributions, the adaptive partition scheme adaptively partitions the space by using a kd-tree like partition method [19]. Specifically, it divides the search space into grid cells such that each grid cell contains nearly the same number of data objects.

The adaptive partition method works as follows. The space is partitioned alternatively between horizontal division and vertical division. Suppose the vertical partition is employed, we first use a straight line parallel to x-axis, denoted by $y = y_1$, to partition the whole space into two regions, and make sure that the upper and the lower regions have equal number of objects. Next, these two regions are partitioned by two lines $x = x_1$ and $x = x_2$, respectively. This process is repeated until every grid cell contains the required data objects. As an illustration, Fig. 1 shows an example of adaptive partition and its corresponding kd-tree index structure.



(a) space partition (b) The corresponding kd-tree

Fig. 1. Space partition and the corresponding kd-tree

4 Index for Spatial Keyword Queries in the IoV

In this section, we present a index, SKIV, to deal with SKQ in the IoV under wireless broadcast environments. By SKIV, the mobile client can reduce the data needed to be retrieved for query processing, while keeping the access latency within a reasonable range, and thus can reduce the energy consumption.

4.1 A Road Network with a Set of Geo-Textual Objects

The graph model is often used to simulate road networks. In this work, we model a road network in the IoV environment as an undirected weighted graph

which includes an edge set and a node set. The graph nodes and edges represent road nodes and road segments respectively. Specifically, each node is a road intersection or an end-point in the road network. Each edge has a non-negative weight which represents the length (network distance) of the road segment.

Figure 2 gives a road network and a set of geo-textual objects $D = \{o_1, \cdots, o_{16}\}$ (depicted as solid points) which lie on the edges of the road network. Table 2 shows the word frequencies for each objects. For example, the text information of object o_1 includes three terms "*fish*", "*pizza*" and "*bread*", whose frequencies are 3, 5 and 2, respectively.

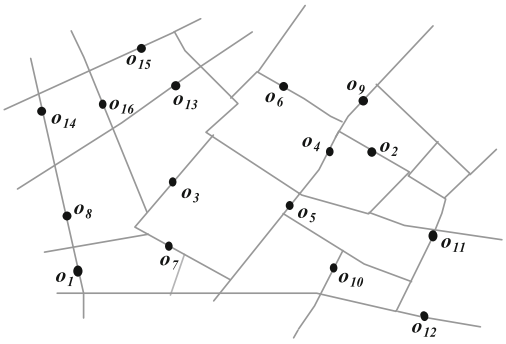


Fig. 2. A road network and a set of geo-textual objects

Table 2. Text information of the objects

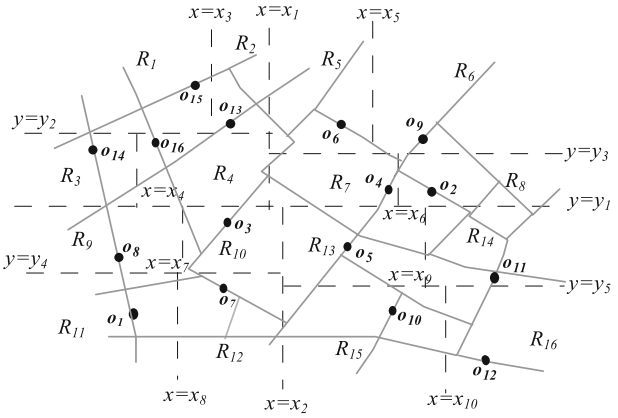
Obj	Terms and term frequencies	Obj	Terms and term frequencies
o_1	(fish, 3) (pizza, 5) (bread, 2)	o_9	(pizza, 4) (Italian, 5) (coffee, 2)
o_2	(pizza, 5) (Italian, 5) (bread, 1)	o_{10}	(Italian, 4) (coffee, 3) (bread, 1)
o_3	(coffee, 4) (Italian, 2) (fish, 3)	o_{11}	(Italian, 4) (fish, 3) (bread, 2)
o_4	(coffee, 4) (Italian, 4) (bread, 3)	o_{12}	(pizza, 4) (bread, 4)
o_5	(Italian, 1) (coffee, 2) (fish, 4) (pizza, 3)	o_{13}	(coffee, 3) (pizza, 2) (bread, 4) (Italian, 2)
o_6	(coffee, 3) (fish, 3) (bread, 4)	o_{14}	(pizza, 3) (bread, 3) (coffee, 2)
o_7	(coffee, 2) (fish, 3) (Italian, 4)	o_{15}	(Italian, 3) (pizza, 2) (bread, 4) (fish, 2)
o_8	(Italian, 4) (pizza, 3) (coffee, 2)	o_{16}	(pizza, 4) (Italian, 5)

4.2 SKIV Index

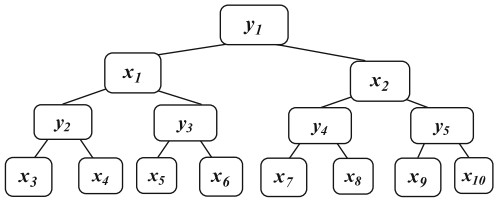
The proposed SKIV index is required to support network space pruning and textual pruning simultaneously The structure of SKIV index includes two components. The first component defines partitions, and maps coordinate points into regions, while the second one specifies minimum/maximum ST scores of the objects between regions.

Firstly, we partition the road network into several disjoint regions. Note that the geo-textual objects may unevenly distributed on the edges of the road network, and we adapt the adaptive partition to divide the search space into grid cells such that each grid cell contains equal number of data objects, and use a kd-tree to keep the partition result [19]. In this way, we can get the partition result and the kd-tree of the road network in Fig. 2, as shown in Fig. 3.

In the client-side, the client needs some information to identify the regions based on coordinate values. Specifically, the first component of the SKIV index includes the splitting values of the kd-tree, which are transmitted in the breadth-first order, and then the client retrieves this index information to reconstruct the tree in Fig. 3(b). Let us go back to the example in Fig. 2, it can be observed that the splitting values is a sequence $< y_1, x_1, x_2, y_2, y_3, y_4, y_5, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10} >$. For ease of query processing at the client side, each node (region) in the kd-tree is assigned an identifier. In particular, the identifier of each region is determined from the leftmost leaf to the rightmost. That is, the identifier of the leftmost leaf is R_1, and increment the region number for the region just at its right side. The derived region numbers are shown in Fig. 3(a).



(a) The partition result



(b) The corresponding kd-tree

Fig. 3. The partition result and kd-tree of the road network in Fig. 2

To efficiently prune the search space during spatial query processing, we need to keep two kinds of information: (i) the road network distance, and (ii) the textual relevance between objects and the query. Firstly, the SKIV index provides the minimum and maximum distances between every pair of regions. This distance part is essentially an $n \times n$ array A, where n is the total number of regions. Every element A_{R_i,R_j} of array A contains two values: the minimum and the maximum network distances from any point in R_i to any point in R_j, i.e., $d_N^{min}(R_i, R_j)$ and $d_N^{max}(R_i, R_j)$[1]. To create array A, we need to pre-calculate the shortest path distances between all possible pairs of nodes from different regions, and the largest weight of the edges in each region. The time complexity of this pre-calculation operation is $O(|N|^3 + |E|)$, where $|N|$ and $|E|$ are the total number of nodes and edges in the road network, respectively.

Secondly, for each region R_i, we keep the following items: (i) the total number of objects in R_i, and (ii) the set of inverted lists containing the objects within R_i. Specifically, one list per different keyword in the description of the objects. The content of each inverted list (for term t_i) includes the *id* of each object o in R_i that have a term t_i in its description. To efficiently search objects which meet the textual requirement of query q, for each region, its inverted lists are arranged in non-increasing order of the number of objects in each list.

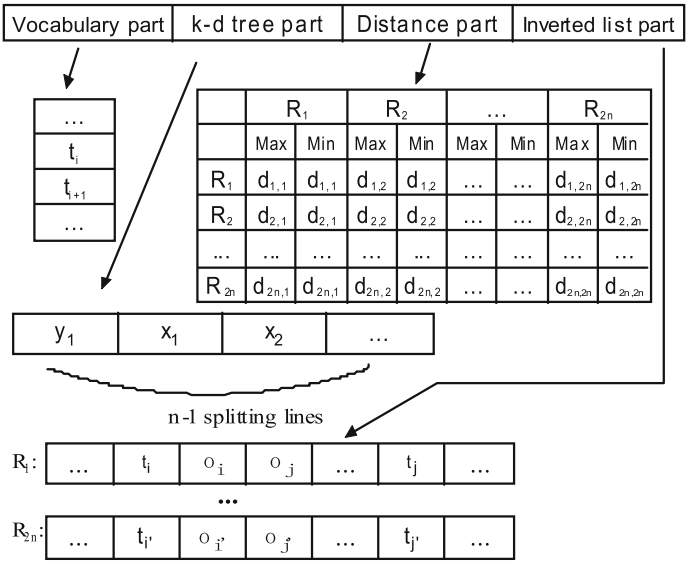


Fig. 4. The index structure of SKIV.

[1] In particular, $d_N^{min}(R_i, R_j)$ equals the minimum network distances from any node in R_i to any node in R_j, and $d_N^{max}(R_i, R_j)$ equals the sum of the maximum network distances from any node in R_i to any node in R_j, the largest weight of the edges in R_i, and the largest weight of the edges in R_j.

In this way, we can get the index segment which includes four parts: the vocabulary file, the k-d tree, the distance array, and the inverted lists. In particular, the k-d tree part includes $n-1$ splitting lines from which the clients can easily reconstruct the space consisting of n regions. For each region R_i, in addition to the network distance bounds from R_i to any other region R_j $(i \neq j, i, j \in [1, n])$ and the textual relevant information, it also contains a pointer to the data of R_i, which is denoted by the offset – the number of packets before the data of R_i is broadcast. The final index structure is shown in Fig. 4. For the sake of clarity, we omit to depict the pointers to the data of the regions.

5 Processing Spatial Keyword Queries on the Client Site

Before discussing the BkQ query processing algorithm in detail, we first introduce three lemmas, which will be used to prune the search space.

Lemma 1. *Given a BkQ query $q =< l, \varphi, k >$ and a region R_i, if $q.\varphi \nsubseteq R_i.\varphi$, then R_i can be safely pruned.*

Proof. For any object o in region R_i, we have $o.\varphi \subseteq R_i.\varphi$. Then by $q.\varphi \nsubseteq R_i.\varphi$, we can get $q.\varphi \nsubseteq o.\varphi$, which means o cannot be a result object of query q. Hence, R_i can be safely pruned.

Lemma 2. *Given a BkQ query $q =< l, \varphi, k >$ and a region R_i, if R_i does not include any object o which belongs to all the invert lists, each of which corresponds to each item in $q.\varphi$, then R_i can be safely pruned.*

Proof. If R_i does not include any object which belongs to all the invert lists corresponding to each item in $q.\varphi$, then for any object o in R_i, the condition $q.\varphi \subseteq o.\varphi$ does not hold, thus R_i can be safely pruned.

Lemma 3. *Given a BkQ query $q =< l, \varphi, k >$ and a region R_i, if $d_N^{\min}(R_q, R_i) > d_k$, then R_i can be safely pruned. In particular, d_k is the minimum $d_N^{\max}(R_q, R_j)$ $(j \in [1, n])$, where R_j satisfies the condition that region R_j together with regions R_l $(l \in [1, n], j \neq l)$ (which meet the condition that $d_N^{\max}(R_q, R_l) \leq d_N^{\max}(R_q, R_j)$) contain at least k objects o, $q.\varphi \subseteq o.\varphi$.*

Proof. Based on the definition of d_k, we know for query q, there exists a set of regions $\mathbb{R}$, and for each region $R \in \mathbb{R}$, there is $d_N^{\max}(R_q, R) \leq d_k$. Moreover, these regions contain at least k objects o which satisfies $q.\varphi \subseteq o.\varphi$. Hence, we can derive that the maximum distance between q and an object that can be the BkQ result of q is d_k. Since $d_N^{\min}(R_q, R_i) > d_k$, we know for any object o' in R_i, there is $d_N(q, o') > d_k$. Therefore, o' cannot be a BkQ object of q, which means R_i does not contain any BkQ object of q and thus can be safely pruned.

Algorithm 1 gives the pseudo-code of BkQ query processing. S is used to keep the regions, which may contain result objects, O_{cand} is used to to keep candidate objects, and float d_k, which is initialized to be infinity, is used to keep the maximum distance value of the k-th nearest neighbor of a query q.

Algorithm 1. BkQ processing

```
1  begin
2      //Step 1: Get the qualified regions;
3      S = ∅, O_cand = ∅, d_k = ∞;
4      Tune into the broadcast channel and receive the index;
5      List all regions R_i (1 ≤ i ≤ n) in ascending order of d_N^min(R_q, R_i);
6      for each region R_i which are listed orderly do
7          if d_N^min(R_q, R_i) > d_k then
8              break; //Lemma 3
9          if q.φ ⊈ R_i.φ then
10             prune R_i; //Lemma 1
11         if R_i includes any object o belonging to all the invert lists, each of which
           corresponds to each item in q.φ then
12             insert each qualified object o into O_cand in the form of
               (o, d_N^max(R_q, R_i));
13             d_k = d_N^max(R_q, R); where R is the region where the k-th object in
               O_cand resides in;
14             insert R_i (R_i, d_N^min(R_q, R_i)) into S;
15         else
16             prune R_i; //Lemma 2;
17     //Step 2: Selectively receive the data and process the query;
18     for each region R_i in S do
19         for each object o in O_cand ( o belong to R_i ) do
20             calculate d_N(q, o) and arrange the object in ascending order of
               d_N(q, o);
21     Put the first k objects in O_cand into BkQ(q);
22     Return BkQ(q);
```

6 Performance Evaluation

6.1 Experimental Setup

This section evaluates the performance of SKIV that is used to support BkQ in the IoV under wireless broadcast environments. All the experiments were executed on a PC with the following configuration: Intel Core 2 Quad, Q8200 2.33 GHz processor and 2 GB main memory, running Linux Ubuntu 9.10. The implementation was written by CPP and compiled by GNU C++ 4.3.3. Each set of experiments was executed on a real road network. Specifically, to model the real-world road network, we used the real data of the traffic network of Oldenburg in Germany (http://www.cs.fsu.edu/lifeifei/spatialdataset.htm), which consists of 6105 nodes and 7035 edges.

To our best knowledge, there is no other work on dealing with spatial keyword queries in the IoV. Hence, we will compare our SKIV with a fixed partition based method (FPB). In FPB, we divide the search space into several equal-sized grid

cells. For each grid cell, we employ inverted files to index the description of the geo-textual objects lying within the cell. Moreover, we calculate and keep the minimum and maximum distances between the cell and any other cells in the road network. In order to reduce the waiting time for receiving the head of the forthcoming index, we also employ the well-known(1, m) interleaving scheme. Suppose the bandwidth of the broadcast channel is fixed, we use the number of bytes transferred over the broadcast channel as the time unit, instead of the real clock time. Moreover, the packet size is set to 256 bytes. We used the generator proposed in [20] to obtain a set of data objects. The description of the objects is obtained from Twitter (http://twitter.com), one tweet per object. We created two data sets namely T_1 and T_2 whose cardinalities (number of objects) are $1k$ and $4k$, respectively. Distinct tweets are used to construct the data set. Table 3 shows the parameters under investigation and the values in bold font are the default values in the experiments. Table 4 gives the characteristics of the data sets.

Table 3. Parameters evaluated in the experiments

Data set	T_1, T_2
Number of NNs (k)	1, 5, **10**, 15, 20
Number of keywords	1, 2, **3**, 4
Number of regions	16, 32, **64**, 128, 256

Table 4. Characteristics of the data sets

Attribute	T_1	T_2
Total size (MB)	2.58	9.98
Total no. of objects	1 k	4 k
Total no. of words	5579	23304
Total no. of distinct words	429	1275
Size of FPB (MB) (64 regions)	0.14	0.60
Size of SKIV (MB) (64 regions)	0.16	0.65

6.2 Experimental Results

We evaluate the performance of SKIV and FPB method for processing BkQ, in terms of tuning time and access latency.

Firstly, the tuning time of the SKIV and FPB method is evaluated. Figure 5(a) depicts the tuning time for BkQ as a function of region number. When we increase the region number, the tuning time of our SKIV first decreases and then increases clearly, and the performance is best at the point of 64 regions. This is because that larger region number means larger index size, thus the time

needed to retrieve the index part increases. On the other hand, if the granularity of region is smaller, more objects are pruned by the pruning power of our method. Similarly, the tuning time of FPB first decreases and then increases as the number of regions increases, and it reaches optimal performance when region number is 128. Our SKIV outperforms FPB. Figure 5(b) plots the tuning time as a function of the keyword number. The cost of these two methods increases as the number of keywords increases, since more query keywords means the result objects may locate in a larger range of the road network. Since SKIV has a strong power to prune unqualified objects both on distance and keyword similarity, its increase tendency is much gentle compared to that of FPB method. Figure 5(c) illustrates the tuning time for BkQ as the value k increases from 1 to 20. The cost of these two method increases for larger k, since larger k means larger search region, larger number of candidate objects, and larger result size.

Then, we also evaluate the access time for BkQ processing. Figure 6(a) depicts the access time for BkQ as a function of region number for data sets T1 and T2. The figure shows that the access time of SKIV and FPB first decreases, and then increases slowly as the number of regions increases, and the access time of FPB is higher than that of our SKIV. On the average, SKIV incurs only 66.4% and 66.8% of access time compared to FPB method for the T1 and T2 data sets, respectively. Figure 6(b) plots the access time as the number of keywords increases. The access time of these two methods increases, when we increase keyword number. The reason lies in that more query keywords means the result

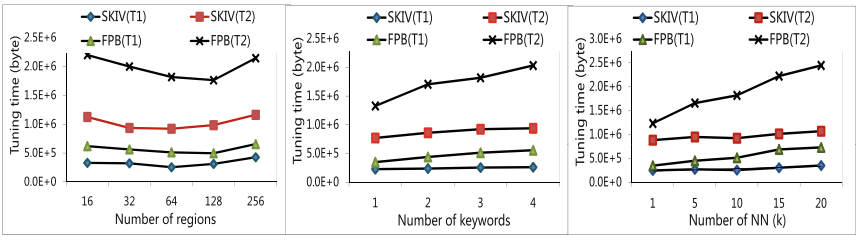


Fig. 5. Tuning time for BkQ

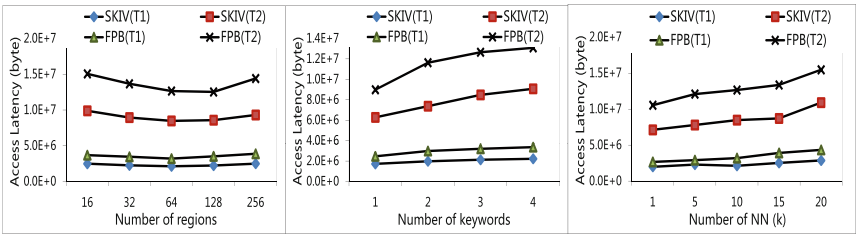


Fig. 6. Access time for BkQ

objects may locate in a larger range of the road network, thus more time is needed to wait for the required data to arrive. Figure 6(c) illustrates the access time for BkQ processing as the value k increases from 1 to 20, and the result shows that the access latency of these two methods grows for larger k value.

7 Conclusion

This paper addresses the issue of processing SKQ in IoV environments. Particularly, we focus on Boolean kNN Queries. A novel SKIV index which includes two parts is proposed. The first part defines space partitions and provides a mapping of coordinate points into regions, while the second part specifies minimum/maximum ST scores of the objects between regions. With the SKIV index, the search space can be efficiently pruned. Based on the SKIV index, a query processing algorithm is developed. Finally, experimental study on a real road network in IoV and two geo-textual data sets demonstrates the efficiency of the proposed index and query processing algorithm. The results show that the proposed method is more efficient than its competitor in both tuning time and access latency.

Acknowledgments. This work is supported by National Science Foundation of China (No. 61309002, No. 61272497).

References

1. Balazs, J.A., Velsquez, J.D.: Opinion mining and information fusion: a survey. Inf. Fusion **27**(C), 95–110 (2016)
2. Yang, F., Wang, S., Li, J., Liu, Z., Sun, Q.: An overview of internet of vehicles. Chin. Commun. **11**(10), 1–15 (2014)
3. Kumar, N., Rodrigues, J.J.P.C., Chilamkurti, N.: Bayesian coalition game as-a-service for content distribution in internet of vehicles. IEEE Internet Things J. **1**(6), 544–555 (2014)
4. Kumar, N., Misra, S., Rodrigues, J., Obaidat, M.S.: Coalition games for spatio-temporal big data in internet of vehicles environment: a comparative analysis. IEEE Internet Things J. **2**(4), 1–1 (2015)
5. Alam, K.M., Saini, M., Saddik, A.E.: Toward social internet of vehicles: concept, architecture, and applications. Access IEEE **3**, 343–357 (2015)
6. Yu, R., Kang, J., Huang, X., Xie, S.: Mixgroup: accumulative pseudonym exchanging for location privacy enhancement in vehicular social networks. IEEE Trans. Dependable Secure Comput. **13**(1), 93–105 (2016)
7. Chen, Y.Y., Suel, T., Markowetz, A.: Efficient query processing in geographic web search engines. In: ACM SIGMOD International Conference on Management of Data, Chicago, Illinois, USA, pp. 277–288, June 2006
8. Christoforaki, M., He, J., Dimopoulos, C., Markowetz, A., Suel, T.: Text vs. space: efficient geo-search query processing. In: ACM Conference on Information and Knowledge Management, CIKM 2011, Glasgow, United Kingdom, pp. 423–432, October 2011

9. 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 (2005)
10. Cong, G., Jensen, C.S., Wu, D.: Efficient retrieval of the top-k most relevant spatial web objects. Proc. VLDB Endow. **2**(1), 337–348 (2009)
11. Gao, Y., Zheng, B., Chen, G.: Efficient reverse top-k boolean spatial keyword queries on road networks. IEEE Trans. Knowl. Data Eng. **PP**(99), 1–14 (2014)
12. Huang, W., Li, G., Tan, K.-L., Feng, J.: Efficient safe-region construction for moving top-k spatial keyword queries. In: Proceedings of the 21st ACM International Conference on Information and Knowledge Management, pp. 932–941 (2012)
13. Li, G., Feng, J., Xu, J.: Desks: direction-aware spatial keyword search. In: Proceedings of the 28th International Conference on Data Engineering, pp. 474–485 (2012)
14. Li, Y., Li, J., Shu, L., Li, Q., Li, G., Yang, F.: Searching continuous nearest neighbors in road networks on the air. Inf. Syst. **42**(2014), 177–194 (2014)
15. De Felipe, I., Hristidis, V., Rishe, N.: Keyword search on spatial databases. In: Proceedings of ICDE, pp. 656–665. IEEE (2008)
16. 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 (2012)
17. Wang, Y., Xu, C., Gu, Y., Chen, M., Yu, G.: Spatial query processing in road networks for wireless data broadcast. Wirel. Netw. **19**(4), 477–494 (2013)
18. Sun, W., Chen, C., Zheng, B., Chen, C., Liu, P.: An air index for spatial query processing in road networks. IEEE Trans. Knowl. Data Eng. **27**(2), 382–395 (2015)
19. Möhring, R.H., Schilling, H., Schütz, B., Wagner, D., Willhalm, T.: Partitioning graphs to speedup Dijkstra's algorithm. J. Exp. Algorithmics (JEA) **11**, 2–8 (2007)
20. Brinkhoff, T.: A framework for generating network-based moving objects. GeoInformatica **6**(2), 153–180 (2002)