

Spatial query processing in road networks for wireless data broadcast

Yanqiu Wang · Chuanfei Xu · Yu Gu ·
Mo Chen · Ge Yu

Published online: 6 July 2012
© Springer Science+Business Media, LLC 2012

Abstract Recently, wireless broadcast environments have attracted significant attention due to its high scalability to broadcast information to a large number of mobile subscribers. It is especially a promising and desirable dissemination method for the heavily loaded environment where a great number of the same type of requests are sent from the users. There have been many studies on processing spatial queries via broadcast model recently. However, not much attention is paid to the spatial queries in road networks on wireless broadcast environments. In this paper, we focus on three common types of spatial queries, namely, k nearest neighbor (k NN) queries, range queries and reverse nearest neighbor (RNN) queries in spatial networks for wireless data broadcast. Specially, we propose a novel *index for spatial queries in wireless broadcast environments (ISW)*. With the reasonable organization and the effectively pre-computed bounds, *ISW* provides a powerful framework for spatial queries. Furthermore, efficient algorithms are designed to cope with k NN, range and RNN queries separately based on *ISW*. The search space can be obviously reduced and subsequently the client can download as less as possible data for query processing, which can conserve the energy while not significantly influence the efficiency. The detailed theory analysis of cost model and the experimental results are presented for verifying the efficiency and effectiveness of *ISW* and our methods.

Keywords Wireless data broadcast · Spatial queries · Road networks · *ISW*

Y. Wang (✉) · C. Xu · Y. Gu · M. Chen · G. Yu
College of Information Science and Engineering, Northeastern University, Shenyang 110819, China
e-mail: wangyanqiu@ise.neu.edu.cn

1 Introduction

With the popularity of smart mobile devices and increasing requirements for ubiquitous information access, there has been an increasing interest in wireless data services from both industrial and academic communities in recent years [1–3]. There are two basic approaches for information access through the wireless technology: on-demand access and wireless data broadcast [4, 5]. For on-demand access, a mobile client initiates a query to the server which in turn processes the query and returns answers to the client through a pre-established point-to-point channel. As a traditional client-server approach, the on-demand mode is suitable for answering customized queries from users. Wireless data broadcast can be considered as a way of disseminating data to a massive number of users, where the query processing tasks are entirely executed at the client sides. In this mode, the server only monitors the information of the data objects, but it is unaware of the clients and their queries because there is no uplink from clients to the server. In this way, wireless data broadcast is expected to have high scalability when dealing with some pre-defined popular applications, such as weather forecast. In recent years, many wireless data broadcast systems have been applied. For example, the Ambient Information Network [6] broadcasts the realtime data packets such as stocks, weather, traffic and sports. It expects that a small amount of wireless data can provide valuable information when delivered in a timely manner to the mobile devices. However, complex spatial query services have not been offered by these systems.

Location based services are considered as the most important applications via wireless broadcasting model. Recently, a lot of research efforts have been conducted on the spatial queries via data broadcast model, such as

k nearest neighbor (k NN) query [4, 5, 7], range query [5, 7], transitive nearest neighbor search [8], continuous nearest neighbor query [9] etc. Most of the existing methods are developed based on Euclidean space, while spatial queries in road networks have not been sufficiently studied. In practice, objects usually move along the pre-defined paths, and therefore the spatial query over road networks are more practical and can acquire higher accuracy [10]. In this paper, we focus on dealing with spatial queries in road networks via wireless data broadcast mode. In road network, the distance computation is more complicated than in Euclidean space. Besides, great amount of information is needed to be broadcast. Furthermore, many existing pruning strategies are designed based on Euclidean metric and invalid in the road network. Therefore, these difficulties render the existing methods impractical for networks. Based on this background, we focus on three common snapshot spatial queries, in particular, k NN queries, range queries and reverse nearest neighbor (RNN) queries in spatial networks combining the context of wireless data broadcast. Three examples of snapshot spatial queries are given as follows to further illustrate our query scenarios. If a continuous query is issued, our solution is executing snapshot queries at each timestamp. Obviously, our system does not perform well for continuous queries as for snapshot queries unless some optimized mechanism are added.

Example 1 A driver would like to query the k nearest restaurants when he/she arrives at an unacquainted city. The pre-stored map does not contain the information of this city, so he/she has to request the answers through the wireless data broadcast.

Example 2 A car would like to search the gas stations within the range that it can reach with the remaining gas. The answer can be acquired through the wireless broadcast systems in which a server stores the information of all the gas stations nearby.

Example 3 A restaurant would like to attract customers from the markets which have the query point as the nearest restaurant.

There are three steps for the wireless broadcast system to handle these queries. Firstly, the server periodically broadcasts the information (for example, in example 3 the information contain the current positions of the passengers) to the clients within its coverage, where the clients denote the mobile devices. Secondly, when the user invokes a spatial query, the client device tunes in the broadcast channel and downloads the information disseminated by the server. Thirdly, after receives the information, the client executes the query.

Note that existing algorithms for processing spatial queries in road networks [11–13] are mostly designed for

on-demand access, and can not be applied or simply extended to wireless broadcast systems. That is because these existing techniques are designed for the random access disk which will incur a significant access latency for broadcast model where data are sequentially transmitted. As an important step forward, Georgios Kellaris et al. [14] focus on the shortest path computation in road networks for wireless data broadcast. However, the technique can not be extended to specific spatial queries such as k NN, range and RNN queries. Hence, we argue that novel techniques are necessary for solving spatial queries in spatial networks for wireless data broadcast. To the best of our knowledge, this is the first paper aiming to provide a general framework for this problem. Specifically, our contributions can be summarized as follows.

- We design a novel *Index for Spatial queries for wireless data broadcast (ISW)*, to provide a general framework for spatial queries in road networks via wireless broadcast. *ISW* can provide a powerful guideline for the client to only download the necessary data, which is realized by its reasonable organization and the tight pre-computed bounds.
- Based on *ISW*, algorithms for k NN, range and RNN queries in road networks on data broadcasting, are separately developed. Effective filter-refinement mechanisms are proposed for these queries.
- Detailed analysis of the cost model for *ISW* is given. Besides, extensive experiments are implemented for verifying the effectiveness of *ISW* and our methods.

The rest of this paper is organized as follows. Section 2 overviews related work. Section 3 proposes the structure of *ISW*. Sections 4, 5 and 6 present algorithms for processing k NN, range and RNN queries, respectively. Section 7 illustrates the experimental results and Sect. 8 concludes the paper.

2 Related work

2.1 Wireless data broadcast

In a wireless broadcast system, the access efficiency and energy conservation are two critical issues [4, 7] for measuring the performance of client devices. First, a user expects that the query can be responded in specified time. Besides, most mobile devices have to work with a battery, so power consumption may become the bottleneck. To save the power, client devices usually support active mode and doze mode. Devices receive data in the active mode with more power consumed. When a system becomes idle, devices tune in the doze mode and only a little power is consumed. In the literature [1, 7, 15], two performance

metrics are usually used to measure the access efficiency and energy conservation respectively.

- *Tuning time.* The time period a mobile client stays active to receive the requested data.
- *Access latency.* The time elapsed from the moment a query is invoked to the moment the answers are received.

If there is no auxiliary information to indicate the schedule of data, a client has to receive all data for spatial queries. In this case, a lot of energy will be consumed since the client is demanded to keep active during the whole broadcast cycle. Air indexing mechanism [16] is the most common organization of the broadcast cycle. The basic idea is that the server constructs the index, and interleaves it with the data. Since the index size is much smaller than the data size, the client is expected to download less data by accessing the index and pre-determining the arrival time of the required data. The client can switch to the active mode once the acquired data has arrived. In order to reduce the waiting time for receiving the root node of a forthcoming index, the index is copied m times and interleaved with the data as shown in Fig. 1. As analyzed in [16], $m = \sqrt{\frac{\text{size of data}}{\text{size of index}}}$ can reach the optimal balance between the tuning time and access latency.

2.2 Spatial queries in wireless broadcast systems

There are many studies about processing spatial queries based in wireless broadcast environments from the database communities in literature. Zheng et al. [7] propose a Hilbert-curve based index, which can be used for k NN queries and window queries via wireless data broadcast. A cost model is developed to compare the performance of the Hilbert-curve index with others. Furthermore, the Hilbert-curve Index is also used for searching continuous k nearest neighbors [9]. Mouratidis et al. [17] focus on the continuous spatial queries in wireless broadcast environments. They propose the Broadcast Grid Index (BGI) method for both snapshot and continuous queries. For efficiently detecting the update of objects, a dirty grid is

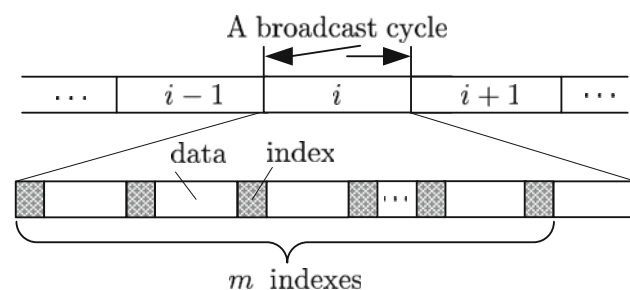


Fig. 1 $(1, m)$ index mechanism

built. Park et al. [2] aim to process the Mobile Continuous Nearest Neighbor Query (MCNNQ) where service users and target objects can move freely. The Guaranteed Region (GR) is defined for dividing the query line into disjoint lines where the nearest neighbor of any point inside a GR is the same. Specially, an ESS_GR algorithm is proposed to process the queries. Gedik et al. [4] propose the exact k NN search on conventional sequential-access R-trees, and optimize the k NN search algorithm. They use the histograms to guide the search, and the analytical results on the maximum queue size and node access count are derived. Besides, Distributed spatial index (DSI) [18] has a linear yet fully distributed structure, and facilitates multiple search paths to be naturally mixed together by sharing links over the whole broadcast cycle. DSI allows a search to start right after a client tunes in the channel so it is very efficient compared with other types of indexes. This work also develops algorithms for snapshot window queries, snapshot k nearest neighbor queries, continuous window queries, and continuous nearest neighbor search based on DSI. However, these spatial query methods are designed for Euclidean space and they are not fit for processing spatial queries in road networks for wireless data broadcast.

In real-life applications, some data may be more popular than others and be frequently accessed by clients. Thus, the nonuniform broadcast provides a good performance on reducing waiting time for the clients. Xu et al. [19] propose a novel parameterized index, called the exponential index, which can optimize the access latency with the tuning time bounded by a given limit. Shen et al. [20] propose an efficient nonuniform index called the skewed index, which is built according to skewed access patterns of clients and the popular data are allocated index modes more times than less popular ones in a broadcast cycle. Zhong et al. [21] firstly introduce Huffman tree into wireless broadcast environments. The Huffman tree based index performs better than B-tree based index. These nonuniform indexing techniques are built based on the assumption that the access frequency of data are known. While in our work, the frequently accessed data are difficult to be obtained so the uniform index is more suitable.

Kellaris et al. [14] address the shortest path computation in road networks for wireless data broadcast, and they propose two methods for this problem. The elliptic boundary (EB) provides the client with an upper bound of the shortest path distance between two regions, and then prunes nodes that lie too far away to affect the shortest path search. To improve the efficiency, the next region (NR) further reduces the volume of received data. Their EB-index records the minimum/maximum distance between two regions, this is efficient for shortest path computation. While for spatial queries, in order to significantly reduce

the search space, more efficient organization of index structure is necessary.

2.3 Spatial queries in road networks

There have been many solutions on processing spatial queries over road networks. Papadias et al. [11] propose a Euclidean restriction (IER) and a network expansion (INE) method for k NN queries. Consider the IER (Incremental Euclidean Restriction) method [11] which applies the multi-step k NN methodology. In the broadcast setting, once the user invokes a query, the device is waken up to determine the first Euclidean nearest neighbor o_{E1} . Then it computes the network distance $d_N(q, o_{E1})$, and retrieves the nearest neighbor in the area where the distances from q to any objects are within $[d_E(q, o_{E1}); d_N(q, o_{E1})]$. However, the data of objects lying in such region may have already been broadcasted and in such situation the device needs to wait until the next broadcast cycle. This incremental restriction procedure has to be repeated for at most k times, and hence the access latency of the IER method is very large as the device may have to wait at most k cycles.

Now consider the INE (Incremental Network Expansion) [11] in the broadcast setting. The INE algorithm performs network expansion starting from q , and examines objects in the order they are encountered. The objects are organized by R-tree which induces backtracking, so for each expansion the needed data might have been broadcasted and the device has to wait for the next broadcast cycle. The access latency of the INE method may last for several cycles at most as the number of the encountered objects.

The Voronoi-based k NN search method [12] adopts an iterative filter/refinement process, and it saves on both storage and computation by performing across-the-network computation for only the border points of the neighboring regions. The filter/refinement step must be invoked k times for searching the k NN for q , which also means k cycles in the broadcast setting. Quad tree is utilized for k NN search by best-first manner in road networks [13], which dramatically reduces the storage. The shortest path computation needs the quad trees of all vertexes at the path if we apply this method into wireless broadcast model. The quad trees are needed to transmit for all vertexes because the source vertex and the destination vertex cannot be pre-specified. The quad trees can not be received until the previous vertex is determined so we cannot filter the unqualified vertexes and their quad trees. By this way, the data volume we need to transmit is about $O(n^{2.5})$ (n is the number of nodes in road networks), and therefore this approach would cause long broadcasting cycle if it is adapted to wireless broadcast model. As shown in the above analysis, the existing methods may incur significant

access latency if they are adapted into wireless broadcast systems, so the new efficient method is necessary.

Range queries on road networks are also studied in [11]. First, it initiates a range query at the datasets which returns objects within Euclidean distance r from q (r is the desired range value). Then the Range Network Expansion (RNE) method refines the results by qualifying segments within the network range r from q and then retrieves the objects falling in these segments. If we adapt the RNE into the wireless broadcast system, two challenges have to be faced. First, the Euclidean bound of distance r is too loose to contain a large number of false hits. Second, examining each subset of query segments means accessing R-tree nodes that overlap some unqualified segments, which invokes backtracking and raises the access latency.

Yiu et al. [22] first address the pruning methods for RNN query in graphs. They propose the *eager* and *lazy* algorithms, where the *eager* algorithm minimizes the I/O cost and can be more CPU-intensive than *lazy* algorithm for certain networks. The pruning granularity of these algorithms is object, while in wireless broadcast model data are transmitted in the form of packets so they can not perform well if they are adapted into wireless broadcast model.

In a word, the existing k nearest neighbor, range and RNN query approaches designed for random access disk are not suitable for the wireless broadcast setting where data are transmitted sequentially.

3 Index for spatial queries in wireless data environments

To address the demand of spatial queries in road networks through the data broadcast environments, we design a general spatial index structure, namely *ISW*, to allow a client to download as less as possible data for query processing which can conserve the energy without significantly lengthening the access latency. *ISW* can provide an effective guideline for greatly reducing the search space, which is realized by its reasonable organization and the pre-computed bounds.

In the following, we first introduce the index structure of *ISW*, and then we propose the computation of the effectively pre-computed bounds. At last, the cost model of *ISW* is analyzed.

3.1 The structure of *ISW*

Generally, the road network is modeled as a weighted directed graph $G = \{E, V, W\}$, where $v_i \in V$ denotes a conjunction node or vertex in the network, and $e_{ij} \in E$ represents the edge connecting v_i and v_j in the network. Given $e_{ij} \in E$, $w(e_{ij})$ denotes the weight value of e_{ij} , where

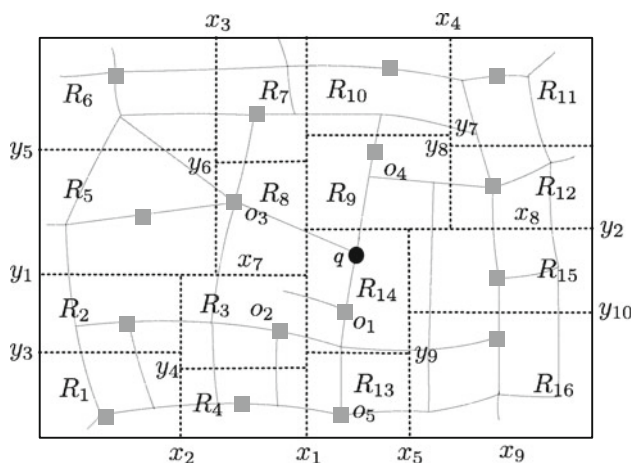


Fig. 2 Partition the whole spatial networks into small cells which contain equal number of objects

the weight can be distance, travel time, cost etc. The objects are denoted as $\mathcal{O} = \{o_1, o_2, \dots, o_n\}$, which lie at some conjunction nodes or edges of the road networks (as the solid square in Fig. 2). In road networks, the network distance between two objects is determined by the length of the shortest path connecting two objects, which is denoted as $d_N(v_i, v_j)$ between v_i and v_j . If there is an object o_i in the edge e_{ij} , then $d_N(o_i, q) = \min\{d_N(o_i, v_i) + d_N(q, v_i), d_N(o_i, v_j) + d_N(q, v_j)\}$, where $d_N(o_i, v_i)(d_N(o_i, v_j))$ denotes the network distance from o_i to $v_i(v_j)$.

The query results are usually centralized in some small domain, while searching the overall space will dramatically affect the performance. Thus, we partition the space into some disjoint regions in order to reduce the search space for a query and subsequently reduce the volume of accessed data. In the real world, the objects are usually not uniformly distributed in the road networks. For example, the density of restaurants is higher in downtown than in the industrial area. Therefore, the straightforward method that partitions the space into cells with the equal size leads to the density imbalance of the cell containment and affects the benefits of the partitions. We adopt the *kd*-tree partitioning [14, 23] which is simple and effective. The *kd*-tree partitioning method divides the whole space into several regions with the equal number of objects, which will results in a lower tuning time in average compared with the straightforward method.

As shown in Fig. 2, firstly the network is partitioned into two regions by a straight line parallel to y -axis. In our case, this line is denoted as $x = x_1$ and it ensures that the left and the right parts have equal number of objects. Then the left part of the network is divided into two regions by $y = y_1$, and the right part of the network is divided into two regions by $y = y_2$. This process continues and alternates between the two axes, until the desired number of regions is reached. Figure 3 depicts the *kd*-tree structure

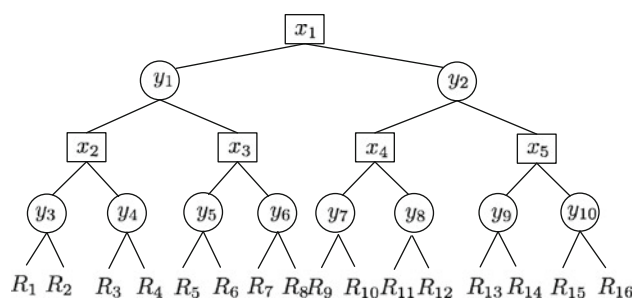


Fig. 3 *kd*-tree structure according to the partition of Fig. 2

corresponding to the partitions of Fig. 2, where each leaf node denotes a region and an intermediate node denotes a line that is used to divide the upper-level space. For example, the leaf node R_1 denotes the region R_1 and the intermediate node x_2 denotes that the child nodes are divided by the line $x = x_2$. *ISW* based on *kd*-tree is transmitted in the breadth-first order, and the identifier of each region is determined from the leftmost leaf to the right. For example, the regions in Fig. 2 are identified from R_1 to R_{16} according to the *kd*-tree structure in Fig. 3. Once the clients receive the index, they can easily reconstruct the space.

The *kd*-tree partition is also adopted by Kellaris and Mouratidis [14], which focuses on the shortest path computation in road networks for wireless data broadcast. They give a maximum/minimum distance table as the bounds, which are effective and efficient for pruning unnecessary cells when the shortest path computation is executed. However, our *ISW* has two advantages compared with the *EB*-index. Firstly, the *ISW* reduces the tuning time of the client devices, because the *ISW* can prune more objects. *ISW* provide tighter bounds than the *EB*-index, which can prune more regions especially when the database is of large size(will be illustrated in Sect. 3.2). By the *ISW*, we can obtain the specified bounds for different query q , which is important since we focus on location-dependent query. For *EB*-index, the bounds may be the same if the positions of q are at the same region. In the case that the region is large, the bounds given by the *EB*-index will be coarse. Therefore, the pruning power of *ISW* is much better than *EB*-index when the cell is large. Secondly, the *ISW* shortens the access latency compared with the *EB*-index, because less data are needed to receive by the client devices. As illustrated above, the *ISW* has higher pruning power and more regions can be pruned, which will reduce the volume of data needed to be downloaded and hence the average access latency is shortened.

Figure 4 illustrates the structure of *ISW*. Take the leaf node which stores R_{14} as an example. The cardinality of the objects inside this cell ($R_{14,c}$) is also recorded. Besides, the leaf node contains a pointer to the data of R_{14} , which is denoted by an offset set as the number of packets before the data of R_{14} are broadcasted. The data store the adjacent list

of objects inside each cell. The leaf node stores N_c (number of cells) pairs of parameters for R_{14} and any other cell.

3.2 Effective bounds

Partitioning the overall space into cells provides a chance for shrinking the search space, based on which effective bounds can be proposed for pruning the unqualified cells. We propose a new method for providing effective bounds $[\varphi^-(q, R_j), \varphi^+(q, R_j)]$ which are tighter than the maximum/minimum distance bounds and ensure that no cells are missed during pruning. Further, our bounds can restrict the distance between the query point and the other cells, and thus can be efficiently applied into pruning process.

For evaluating the bounds, we first define a pair of parameters, namely η^- and η^+ for R_i and R_j ($1 \leq i, j \leq N_c$). Eqs. (1, 2) show the computation of the two parameters:

$$\eta^-(R_i, R_j) = \min_{1 \leq i, j \leq N_c} \left\{ \frac{d_N(o_k^i, o_l^j)}{d_E(o_k^i, R_j)} \right\}, \tag{1}$$

$$\eta^+(R_i, R_j) = \max_{1 \leq i, j \leq N_c} \left\{ \frac{d_N(o_k^i, o_l^j)}{d_E(o_k^i, R_j)} \right\}, \tag{2}$$

where o_k^i denotes an object lying in R_i , o_l^j denotes an object lying in R_j , and $d_E(o_k^i, R_j)$ denotes the Euclidean distance between o_k^i and any points in R_j . Notice that $\eta^-(R_i, R_j)$ is different from $\eta^-(R_j, R_i)$ and vice versa. For instance, $d_N(o_2, q) = 8$, $d_N(o_2, o_1) = 6$ (as shown in Fig. 2), $\max(d_E(q, R_3)) = 12$, $\max(d_E(o_1, R_3)) = 10$, and hence $\eta^-(R_{14}, R_3) = \min\{\frac{8}{12}, \frac{6}{10}\} = 0.6$. Similarly, $\min(d_E(q, R_3)) = 6$, $\min(d_E(o_1, R_3)) = 5$, and hence $\eta^+(R_{14}, R_3) = \max\{\frac{8}{6}, \frac{6}{5}\} = 1.33$. For each pair of R_i and R_j , the values of $\eta^-(R_i, R_j)$ and $\eta^+(R_i, R_j)$ are pre-computed and stored in the leaf node of ISW, as shown in Fig. 4. When a query is issued, the position of q is known, and therefore the bounds restricting the network distance between q and any cell R_j can be computed as follows.

$$\varphi^-(q, R_j) = \eta^-(R_i, R_j) \cdot \min(d_E(q, R_j)) \tag{3}$$

$$\varphi^+(q, R_j) = \eta^+(R_i, R_j) \cdot \max(d_E(q, R_j)) \tag{4}$$

Continuing the above example, $[\eta^-(R_{14}, R_3), \eta^+(R_{14}, R_3)] = [0.6, 1.33]$ (as shown in Fig. 4), $\min(d_E(q, R_3)) = 6$ and $\max(d_E(q, R_3)) = 12$, and the bounds of q to R_3 are $[\varphi^-(q, R_3), \varphi^+(q, R_3)] = [3.6, 15.96]$. Lemma 1 and Lemma 2 illustrate that $\min(d_N(R_q, R_j)) \leq \varphi^-(q, R_j) \leq \min(d_N(q, R_j))$ and $\max(d_N(q, R_j)) \leq \varphi^+(q, R_j) \leq \max(d_N(R_q, R_j))$, which ensures the pruning effect of $[\varphi^-(q, R_j), \varphi^+(q, R_j)]$.

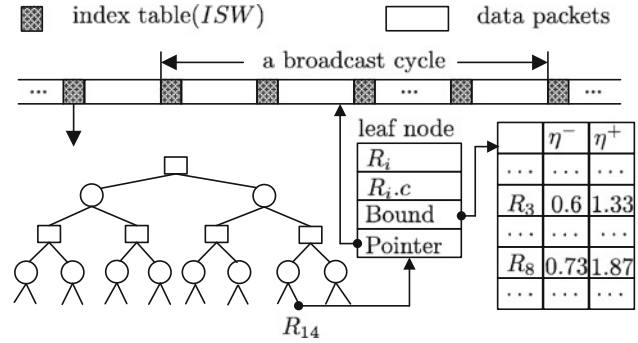


Fig. 4 Content of ISW

Lemma 1 For the query point q and the region R_j , the bounds of $[\varphi^-(q, R_j), \varphi^+(q, R_j)]$ is tighter than $[\min(d_N(R_q, R_j)), \max(d_N(R_q, R_j))]$, where R_q denotes the region in which q lies.

Proof By Eqs. (1, 3), $\varphi^-(q, R_j) = \eta^-(R_q, R_j) \cdot \min(d_E(q, R_j)) = \min_{1 \leq j \leq N_c} \left\{ \frac{d_N(o_k^q, o_l^j)}{d_E(o_k^q, R_j)} \right\} \cdot \min(d_E(q, R_j))$.

If q is located in R_i , then $\varphi^-(q, R_j) = \min_{1 \leq j \leq N_c} \left\{ \frac{d_N(o_i, o_j)}{d_E(o_i, R_j)} \right\} \cdot \min(d_E(q, R_j))$. For two pairs of nodes (o_i, o_j) and (o'_i, o'_j) , if $d_N(o_i, o_j) \geq d_N(o'_i, o'_j)$, then we suppose that $\frac{d_N(o_i, o_j)}{d_E(o_i, o_j)} \geq \frac{d_E(o_i, o_j)}{d_E(o'_i, o'_j)}$. Such assumption is made according to the observation that the network distance connecting two points are much longer than the straight-line distance connecting two points especially when the distances are large. Therefore, $\frac{d_N(o_k^q, o_l^j)}{\min(d_N(R_q, R_j))} \geq \frac{d_E(o_k^q, o_l^j)}{\min(d_E(q, R_j))} \Rightarrow \min_{1 \leq j \leq N_c} \left\{ \frac{d_N(o_k^q, o_l^j)}{d_E(o_k^q, R_j)} \right\} \cdot \min(d_E(q, R_j)) \geq \min(d_N(R_q, R_j))$. Hence, $\varphi^-(q, R_j) \geq \min(d_N(R_q, R_j))$.

Likewise, $\frac{\max(d_N(R_q, R_j))}{d_N(o_k^q, o_l^j)} \geq \frac{\max(d_E(q, R_j))}{d_E(o_k^q, o_l^j)}$, and $\varphi^+(q, R_j) = \max_{1 \leq j \leq N_c} \left\{ \frac{d_N(o_k^q, o_l^j)}{d_E(o_k^q, R_j)} \right\} \cdot \max(d_E(q, R_j)) \leq \max(d_N(R_q, R_j))$. Therefore, $\min(d_N(R_q, R_j)) \leq \varphi^-(q, R_j) \leq \varphi^+(q, R_j) \leq \max(d_N(R_q, R_j))$ and the bounds of $[\varphi^-(q, R_j), \varphi^+(q, R_j)]$ are tighter than $[\min(d_N(R_q, R_j)), \max(d_N(R_q, R_j))]$. \square

Lemma 2 For the query point q and the region R_j , $\varphi^-(q, R_j) \leq \min(d_N(q, o_k^j))$ ($o_k^j \in R_j$), and $\max(d_N(q, o_k^j))$ ($o_k^j \in R_j$) $\leq \varphi^+(q, R_j)$.

Proof By Eqs. (1, 3), $\varphi^-(q, R_j) = \eta^-(R_q, R_j) \cdot \min(d_E(q, R_j)) = \min_{1 \leq j \leq N_c} \left\{ \frac{d_N(o_k^q, o_l^j)}{d_E(o_k^q, R_j)} \right\} \cdot \min(d_E(q, R_j))$. Because $\min_{1 \leq j \leq N_c} \left\{ \frac{d_N(o_k^q, o_l^j)}{d_E(o_k^q, R_j)} \right\} \leq \frac{\min(d_N(q, R_j))}{\min(d_E(q, R_j))}$, $\varphi^-(q, R_j) \leq \min(d_N(q, o_k^j))$.

Likewise, $\varphi^+(q, R_j) \geq \max(d_N(q, o_k^j))$ and $\varphi^-(q, R_j) \leq \min(d_N(q, o_k^j))$ ($o_k^i \in R_j \leq \max(d_N(q, o_k^i))$ ($o_k^i \in R_j \leq \varphi^+(q, R_j)$). \square

The value of $[\eta^-(R_i, R_j), \eta^+(R_i, R_j)]$ can be computed by traversing all the objects inside R_i and R_j , and the time complexity is the same as the computation of maximum/minimum distance bounds.

3.3 Cost model

We conduct a performance analysis based on the assumption that the clients tune into the channel randomly and each object has the same probability to be accessed. In order to facilitate the following analysis, Table 1 lists the symbols and their definitions. We utilize the number of access data packets to denote the average access latency and the average tuning time.

Theorem 1 *The access latency of a spatial query can be represented as follows:*

$$Acl = \frac{1}{2} \left(I + \frac{D}{m} \right) + \left(I + \frac{D}{m} \right) * i * \sum_{i=0}^m P_i + \left(I + \frac{D}{2m} \right), \tag{5}$$

$$P_i = \begin{cases} 0 & i = 0, \dots, \tau - 1 \\ C_{i-1}^{\tau-1} \left(\frac{1}{m} \right)^\tau & i = \tau, \dots, m. \end{cases} \tag{6}$$

Proof The first term of Eq. (5) is the duration of the initial probe which starts when a client tunes into the broadcast channel till the moment the index is received. In average, it takes $\frac{1}{2} \left(I + \frac{D}{m} \right)$ time.

As defined in Table 1, in order to receive necessary data for processing a spatial query such as the k NN, range or RNN query, we must to access τ data blocks. We denote

Table 1 Commonly used symbols

Symbol	Definition
N	Number of objects
N_c	Number of cells
D	Size of data used to include all the objects Information (number of data packets)
Acl	Average access latency of a spatial query
Tun	Average tuning time of a spatial query
B_i	The i -th data block in a cycle, and size of B_i Is equal to D/m
P_i	Probability that some desired data reside In B_i
I	Size of ISW (number of data packets)
τ	Number of data packets needed to be accessed When a spatial query is executed

the last one of τ data packets as τ_{last} , and then the probability that τ_{last} resides at B_i can be represented by Eq. (6). The τ_{last} data block can not reside at B_i when $i = 0, \dots, \tau - 1$. The probability that τ_{last} resides at B_i ($i = \tau, \dots, m$) is $P_\tau = \left(\frac{1}{m} \right) \left(\frac{1}{m-1} \right) \dots \left(\frac{1}{m-\tau} \right)$. Because $\tau \ll m, P_\tau = \left(\frac{1}{m} \right)^\tau$. Therefore, the expectation value that τ_{last} resides at B_i can be represented as $i * \sum_k^i P_i$. Consequently, before we access the last object, $\left(I + \frac{D}{m} \right) * i * \sum_{i=0}^m P_i$ value of data have been disseminated.

The term of $I + \frac{D}{2m}$ represents the time consumed by the final probe process. According to the above three parts, we can infer the average access latency as in Theorem 1. \square

Theorem 2 *The average tuning time of a spatial query can be represented as:*

$$T = T_i + T_d, \tag{7}$$

$$T_i = \frac{m}{ml + D} \cdot (I + 1) + \frac{m(I - 1) + D}{ml + D} \cdot (1 + I) = I + 1, \tag{8}$$

$$T_d = \tau. \tag{9}$$

Proof The average tuning time T includes T_i and T_d , where T_i denotes the time to download the index and T_d denotes the time to download the relative data for the queries. When a user issues a spatial query, the client tunes into the broadcast channel, and searches for the root of the index. Since the client can starts to probe the broadcast channel at any time, we analyze the first phase by two cases:

Case 1 The first visited data packet is exactly the one before the packet containing the root node of the index. In this case the client can directly receive the index. The probability of this case is $\frac{m}{ml+D}$, and the average tuning time of this case is $\frac{m}{ml+D} \cdot (I + 1)$ (suppose that the initial probe costs one data packet).

Case 2 The first visited packet is a data packet or an index bucket excluding the root node. The client needs to wait for the next root node of the index. This has a probability of $\frac{m(I-1)+D}{ml+D}$, and average tuning time is $\frac{m(I-1)+D}{ml+D} \cdot (1 + I)$ (suppose that the initial probe costs one data packet).

For the phase of downloading data, the client tunes in the doze mode until the required data arrives, and then tunes in the active mode again to download the relative data. Suppose that the average data packets of the spatial query is τ . Thus, the tuning time is represented by Eq. (7). \square

Choosing a suitable partition for the space is an important problem for spatial queries via wireless data broadcast. If the partition number is too small, then the

effect of pruning is limited and the tuning time is influenced as well. If the number of cells is too large, then the size of index is large and the access latency will be long. Therefore, we need to choose a suitable partition for achieving the optimal tradeoff between tuning time and access latency. In order to choose a suitable value for partitioning the space, we have to consider two factors: the size of the road networks and the volume of objects. For example, if the road network and the number of objects are large, we had better partition the space into more regions. Note that in theory, the optimal partition number can not be efficiently evaluated. Alternatively, once a new data set is given, we need to do some experiments to choose a suitable value by which the system can acquire optimal performance. Then, this size is utilized to the system until new dataset is given. We will analyze the proper partition number in our experiment section.

4 k nearest neighbor query processing

Given a query point q , and a spatial dataset \mathcal{O} , a k nearest neighbor (k NN) query retrieves the k objects of \mathcal{O} closest to q according to the network distance. This section presents an efficient algorithm for k NN queries, based on *ISW*.

4.1 Filter by *ISW*

Since a k NN query only cares about the k objects with the minimal network distance from q , efficient algorithms should avoid the retrieval of any unnecessary objects. Thus, an important goal for processing k NN search is to obtain a small search range in the estimate phase in order to reduce the tune-in time. We develop efficient k NN query processing algorithm based on *ISW*. The detailed algorithm comprises two steps: (1) The client downloads the index, and analyzes the index to determine the bound d_k that contains at least k nearest neighbors of q . (2) The client downloads all the necessary data according to the precious step, and then examines the candidate set to obtain the exact answer set. We first develop Lemma 3 for filtering.

Lemma 3 *Given a query point q and a cell R_i , if $\varphi^-(q, R_i) > d_k$, then any objects in R_i will not appear in the answer set, where d_k is the minimum of $\varphi^+(q, R_i)$ ($i = 1, \dots, N_c$) satisfying that R_i and the regions dominated by R_i contain at least k objects.*

Proof For q , we define that R_i dominates R_j (denoted as $R_i \preceq R_j$), if $\varphi^+(q, R_i) \leq \varphi^-(q, R_j)$. If $\varphi^-(q, R_i) > d_k$ holds,

the object $o_i^i (o_i^i \in R_i)$ can not belong to the k nearest neighbors of q since there always exist at least k objects with distances smaller than or equal to d_k . Therefore, the answer set $kNN(q)$ will not contain such an $o_i^i (o_i^i \in R_i)$ according to the k NN definition. In other words, the fact that o_i^i is the k nearest neighbors of q implies $\varphi^-(q, R_i) \leq d_k (o_i^i \in R_i)$. \square

We utilize the bound d_k to eliminate the regions that do not contain any k NN of q . It is interesting to note that, the bound of d_k is not a circle around q , while d_k denotes a range which is represented by the network distance. If any region R_j with $\varphi^-(q, R_j) > d_k$, then R_j can be safely pruned because any objects lie in R_j will not be the result. As depicted at Section 3, for each region R_i and R_j , the $[\eta^-(R_i, R_j), \eta^+(R_i, R_j)]$ have been pre-computed and stored in the corresponding leaf node. With the location of q and the pre-computed information, the bounds of R_i and q can be computed as $[\varphi^-(q, R_i), \varphi^+(q, R_i)]$. This pair of bounds can be used for determining the suitable d_k . For regions R_i, R_j , and q , if $\varphi^+(q, R_i) \leq \varphi^-(q, R_j)$, R_i and any region $R_l (R_l \preceq R_i)$ contain at least k objects, then at least k objects are nearer to q than objects in R_j . Similarly, at least k objects are nearer to q than objects in R_j . According to this, if d_k is determined, then a region R_i does not have to be considered if $\varphi^-(q, R_j) > d_k$. Suppose $\varphi^+(q, R_3) = d_k (\varphi^+(q, R_3) = 15.96$ as computed in Section 3.2), and $\varphi^-(q, R_{12}) = 16.34$. Because $\varphi^-(q, R_{12}) > d_k$, R_{12} can be safely pruned. Two Lemmas are proposed for quickly determining the value of d_k .

Lemma 4 *For the current d_k , if $\exists R_i, \varphi^+(q, R_i) < d_k$, and R_i contains at least k objects, then d_k can be updated as $\varphi^+(q, R_i)$.*

Proof If R_i contains at least k objects, then there are at least k objects whose distances to q are smaller than d_k due to $\varphi^+(q, R_i) < d_k$. Because d_k is defined as a safe bound that contains at least k objects which have the shortest distance from q , the value of d_k can be safely updated as $\varphi^+(q, R_i)$. \square

Lemma 5 *For the current $d_k, \exists R_i, R_j, \varphi^-(q, R_i) < \varphi^-(q, R_j) < d_k, \varphi^+(q, R_j) < d_k, R_i$ contains l objects and R_j contains l' objects. If R_i dominates R_j and $l + l' > k$, then d_k can be updated as $\varphi^+(q, R_j)$.*

Proof If R_i dominates R_j , then $\varphi^+(q, R_i) < \varphi^-(q, R_j) < \varphi^+(q, R_j) < d_k$. Suppose an object o_k^i locates in R_j , then any objects o_k^i residing in R_i is nearer to q . According to R_i containing l objects, if R_j contains $k - l$ objects, then $\varphi^+(q, R_j)$ contains at least k objects that have shorter distance to q than the objects outside $\varphi^+(q, R_j)$. If $l + l' > k$, then d_k can be updated as $\varphi^+(q, R_j)$. \square

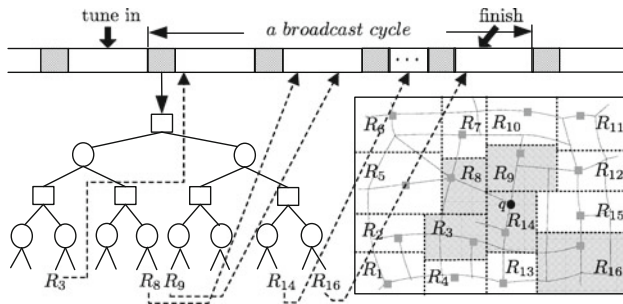


Fig. 5 *k*NN query processing

4.2 *k*NN query processing

Figure 5 depicts the procedure of a *k*NN query. Once the user invokes a *k*NN query, the client device tunes in the broadcast channel, and receives the index. Through the analysis of the index, the client learns which cells may contain the results, such as the grey regions in Fig. 5. The time when to wake up and receive the information is determined by the pointers in the leaf nodes, and by them the client device receives the required data and executes the query.

In order to compute the distance between q and the object o_i , the intermediate nodes in $path(q, o_i)$ (path from q to o_i is denoted as $path(q, o_i)$) and the distance of each interval between two nodes are also needed to be received. Lemma 6 illustrates that by our pruning strategies all intermediate nodes for computing *k*NN are also received.

Lemma 6 $\forall o \in C$, if $d_N(o, q) < d_k, \forall o' \in path(q, o) \rightarrow o' \in C$.

Proof Because $o' \in path(q, o), d_N(o', q) < d_N(o, q) < d_k$. Suppose o' resides at R'_i , then $\varphi^-(q, R'_i) < d_N(o', q) < d_k$, so objects reside at R'_i are also in candidate set C . \square

Algorithm 1 demonstrates the detail steps of *k*NN query processing. As the client device receives *ISW*, it first initializes a heap with the root node of *ISW* (line 2). To accelerate the query processing, this algorithm next estimates a candidate set by d_k to prune the unnecessary regions (line 3–15). According to d_k , the client device can determine which regions are necessary for the query processing, and it finds the needed data by some pointers. In the second step, the client device receives the necessary data as pre-determined, namely, the region R_i which satisfies $\varphi^-(q, R_i) < d_k$ are needed to be received (line 24). The objects in these regions are put into the candidate set C (line 25). After all necessary data are received, the client device executes the *k*NN query (line 28) locally by the IER method. As illustrated by Lemma 6, for any object $o_j \in C$, if $d_N(o, q) < d_k$, the nodes that pass by the $path(q, o_j)$ are also in C . Hence, *k*NN can be computed inside C .

Algorithm 1 *k*NN query processing

```

Input : query point  $q$ 
Output:  $k$  nearest neighbors
1 //Step 1: Initially obtain the candidate set
2 initialize the heap  $H$  with the root node of  $ISW$ ;
3  $d_k \leftarrow \infty$ ;
4 initialize a list  $L, L \leftarrow \emptyset$ ;
5 for  $H$  is not empty do
6   pop a node  $P$  from  $H$ ;
7   if  $P$  is the root node or middle node then
8     take out the child nodes of  $P$ , and insert them
9     into  $H$ ;
10  end
11 if  $P$  is the leaf node then
12   retrieve  $R_i$  stored in  $P$ ;
13   compute  $[\varphi^-(q, R_i), \varphi^+(q, R_i)]$ ;
14   if  $\varphi^-(q, R_i) \leq d_k$  and  $|L| < k$  then
15     insert  $R_i$  into  $L$  with ascending  $\varphi^-(q, R_i)$ ;
16      $d_k \leftarrow \varphi^+(q, R_i)$ ;
17   end
18   if  $\varphi^-(q, R_i) > d_k$  then
19     prune  $R_i$ ;
20   end
21 end
22 //Step 2: Selectively receive the data and process the
23   query in client devices
24 for each region  $R_i$  that  $\varphi^-(q, R_i) < d_k$  do
25   receive the objects  $o_k^i$  at  $R_i$ ;
26   put  $o_k^i$  into the candidate set  $C$ ;
27 end
28 for all objects in  $C$  do
29   search the  $k$  nearest neighbors for  $q$ .
30 end

```

5 Range query processing

Given a query point q , a value r and a spatial dataset \mathcal{O} , a range query retrieves all objects from \mathcal{O} that are within network distance r from q , that is, $RQ(q, r) = \{o_i | o_i \in \mathcal{O}, d_N(o_i, q) < r\}$. In the Euclidean space, the range queries are easily answered by comparing the coordinates of objects with the bounds of query windows. However, in road networks, whether the object o belongs to $RQ(q, r)$ is determined by the network distance between o and q . The shape of the query window is irregular and the results are difficult to be acquired.

The naive solution for range queries over road networks first examines each segments within r from q and then puts the encountered object o into the result set if $d_N(o, q) < r$. Obviously, this solution would traverse the whole segment set and the cost is very large. We propose a filter-refinement mechanism for efficiently executing the range queries by *ISW*, which greatly reduces the volume of data reception and reduces the computations. Firstly, the client tunes in the broadcast channel and receives the index, and analyzes the index to prune unnecessary cells. The cells can be

classified into three types according to the value of $[\varphi^-(q, R_i), \varphi^+(q, R_i)]$ with r , where $[\varphi^-(q, R_i), \varphi^+(q, R_i)]$ denote the bounds of distance from q to the cell R_i . If $\varphi^-(q, R_i) > r$, then R_i can be safely pruned (Lemma 7). If $\varphi^+(q, R_i) < r$, then $\forall o_k^i \in R_i$ can be put into the result set (Lemma 8). If $\varphi^-(q, R_i) < r$ and $\varphi^+(q, R_i) > r$, then R_i should be put into the candidate set. The data of the second and the third type of cells should be received for answering queries. Secondly, the client analyzes the candidate objects by examining each $path(o, n_i)$, where n_i is a point on a path and $d_N(o, n_i) = r$. All objects in segment $path(o, n_i)$ are put into the result set.

Lemma 7 For any region R_i , if $\varphi^-(q, R_i) > r$, then R_i can be safely pruned.

Proof For any object o_k^i in $R_i, d_N(q, o_k^i) > \varphi^-(q, R_i) > r$, and hence R_i can be safely pruned. \square

Lemma 8 For any region R_i , if $\varphi^+(q, R_i) < r$, then the objects o_k^i within R_i can be put into the result set.

Proof For any object o_k^i which resides in $R_i, d_N(q, o_k^i) < \varphi^+(q, R_i) < r$. Therefore, o_k^i is the result. \square

Figure 6 shows the filter step, where $r = 9$, $R_1, R_2, R_4, R_5, R_6, R_7, R_{10}$ and R_{11} can be pruned according to the Lemma 7. Due to $\varphi^+(q, R_{14}) = 7.86 < r$, the objects lie in R_{14} can be directly put into the result set. Besides, $R_3([\varphi^-(q, R_3), \varphi^+(q, R_3)] = [3.6, 15.96]), R_8, R_9, R_{12}, R_{15}$ and R_{16} should be received as the candidate set. Figure 7 shows the refinement step of range queries, where several segments are examined and the final results are returned.

Algorithm 2 gives the detail steps of the range queries. Firstly, a client device receives the index and looks up the bounds of $[\varphi^-(q, R_i), \varphi^+(q, R_i)]$. According to Lemma 7, the regions whose bound of $\varphi^+(q, R_i)$ are larger than r can be safely pruned (line 2–4). According to Lemma 8, the regions with the bound of $\varphi^-(q, R_i)$ smaller than r can be received (line 5–7), and the objects residing in these regions should be put into the result set (line 10–12). The other regions whose bound of $\varphi^-(q, R_i)$ are smaller than r and the bound of $\varphi^+(q, R_i)$ larger than r should be received as the candidate set C (line 8). For regions in the candidate set, all objects should be received and the distances from them to q are computed and compared with r (line 13–17).

6 Reverse nearest neighbor query processing

Reverse nearest neighbor (RNN) query is very useful in applications such as the decision support and the resource allocation. Given a dataset \mathcal{O} and a query point q , a RNN query retrieves all objects $o \in \mathcal{O}$ that have q as their nearest neighbor. The naive method for searching RNN is

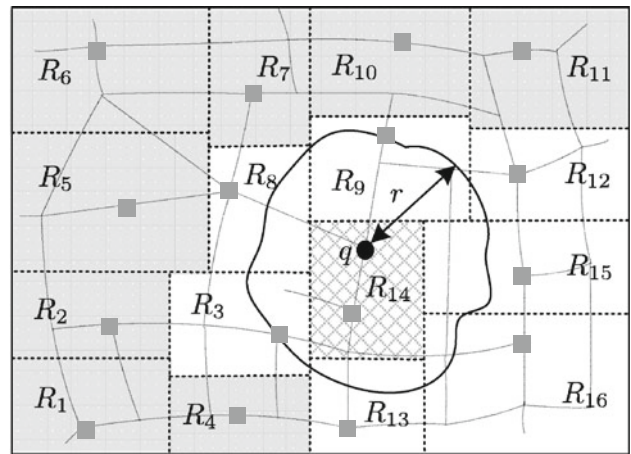


Fig. 6 Filter step for range query

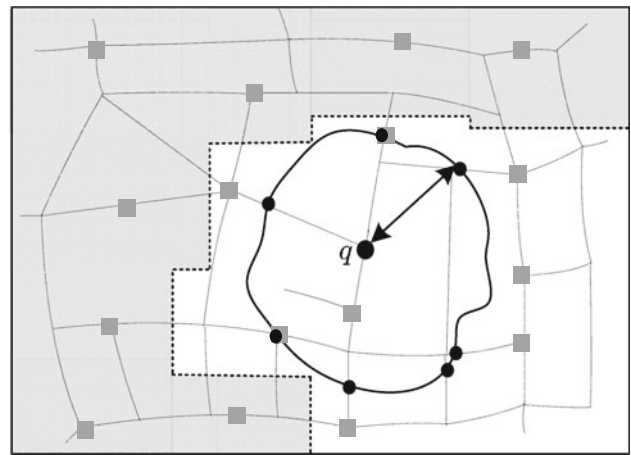


Fig. 7 Refinement step for range query

traversing the network from q , and searching the nearest neighbor for each encountered object $o \in \mathcal{O}$. If there is no o' satisfying $d_N(o, o') > d_N(o, q)$, then $o \in RNN(q)$. The number of objects in $RNN(q)$ is not fixed, and the distance from o to q may be very far, so every object in the road networks has to be examined.

We adopt *ISW* to support the RNN query and to optimize the system performance. It is difficult to determine a bound for RNN query, but two Lemmas can be utilized to prune some unqualified regions and further reduce the tuning time.

Lemma 9 For regions R_i and R_j , if $\varphi^+(R_i, R_j) < \varphi^-(q, R_i)$, then R_i can be safely pruned.

Proof $\varphi^+(R_i, R_j)$ is defined as $\eta^+(R_i, R_j) \cdot \max(d_E(R_i, R_j))$. Because $\eta^+(R_i, R_j) = \max\left\{\frac{d_N(o_k^i, o_l^j)}{d_E(o_k^i, R_j)}\right\}, \eta^+(R_i, R_j) \cdot \max(d_E(R_i, R_j)) \geq \max(d_N(o_k^i, o_l^j))$. For any object o which

Algorithm 2 Range query processing

```

Input : query point  $q$ , distance  $r$ 
Output: result set  $RQ(q, r)$ 
1 for each region  $R_i$  do
2   if  $\varphi^-(q, R_i) > r$  then
3     prune  $R_i$ ;
4   end
5   if  $\varphi^+(q, R_i) < r$  then
6     put  $R_i$  into  $C'$ ;
7   end
8   put  $R_i$  into  $C$ ;
9 end
10 for each region  $R_i$  in  $C'$  do
11   put object  $o$  which locates in  $R_i$  into  $RQ(q, r)$ ;
12 end
13 for each region  $R_i$  in  $C$  do
14   for each object  $o_k^i$  in  $R_i$  do
15     if  $d_N(o_k^i, q) < r$  then
16       put  $o_k^i$  into  $RQ(q, r)$ ;
17     end
18   end
19 end
20 return  $RQ(q, r)$ .

```

in R_i , if $\varphi^+(R_i, R_j) < \varphi^-(q, R_i)$, then there is an object o' which in R_j that $d_N(o, o') \leq \max(d_N(o_k^i, o_j^i)) \leq \varphi^+(R_i, R_j) < \varphi^-(q, R_i) \leq \min(d_N(q, o_k^i)) \leq d_N(o, q)$. Due to $d_N(o, o') < d_N(o, q)$, the nearest neighbor of o could not be q , and the RNN of q would not contain o . Hence, R_i can be safely pruned. As shown in Fig. 8, $\varphi^+(R_6, R_8) < \varphi^-(q, R_6)$, so R_6 can be pruned. \square

Lemma 10 For the region R_i , if $\varphi^+(R_i, R_i) < \varphi^-(q, R_i)$ and R_i contains at least two objects, then R_i can be safely pruned.

Proof Suppose o and o' are located in R_i , $d_N(o, o') \leq \max(d_N(R_i, R_i)) \leq \varphi^+(R_i, R_i) < \varphi^-(q, R_i) \leq \min(d_N(q, o_k^i)) \leq d_N(o, q)$, $d_N(o, o') \leq \max(d_N(R_i, R_i)) \leq \varphi^+(R_i, R_i) < \varphi^-(q, R_i) \leq \min(d_N(q, o_k^i)) \leq d_n(o', q)$, so q can not be the nearest neighbor of o or o' . Hence, R_i can be safely pruned. As shown in Fig. 9, $\varphi^+(R_3, R_3) < \varphi^-(q, R_3)$, so R_3 can be pruned. \square

Algorithm 3 describes the procedure of RNN query algorithm, which is executed in three steps. The first step analyzes *ISW* and prunes some regions by Lemmas 9 and 10 (line 4–8). The second step receives the objects in pre-selected regions, and these objects form a small size candidate set C (line 11, 12). The third step refines the candidate set by examining the nearest neighbor of each object. To accelerate the local query, the *lazy* algorithm [22] is adopted.

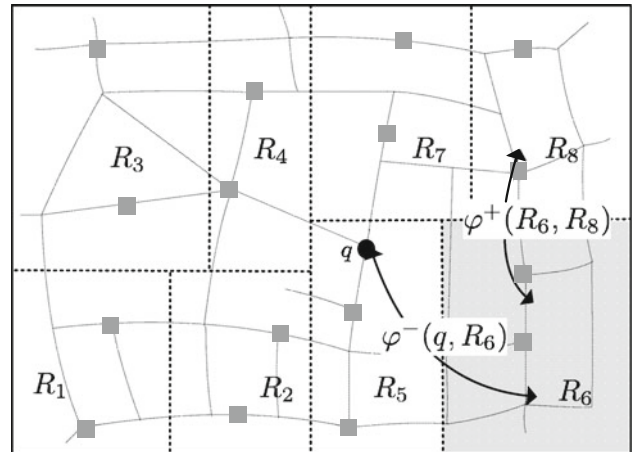


Fig. 8 Example for Lemma 9

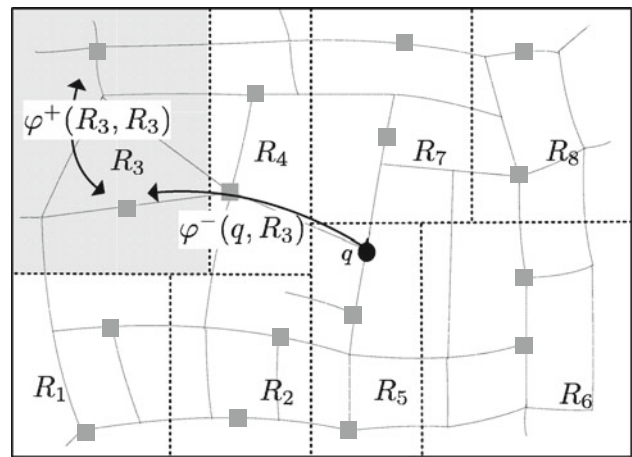


Fig. 9 Example for Lemma 10

7 Performance evaluation

This section evaluates the performance of *ISW* for supporting *kNN* queries, range queries and RNN queries. All the experiments were executed on a PC with 2.5 GHz CPU and 2 GB main memory and the algorithms are implemented with C++. Each set of experiments are executed on the real road network dataset, CA, obtained from the Digital Chart of the World Server. In particular, CA captures the road networks in California, and contains 21,047 nodes and 21,692 edges [24]. The packet size is set to 128 bytes. We assume two types of bandwidth for the broadcast channel, 2 Mbps and 384 Kbps respectively, which are typical in 3G networks for static and moving devices.

In order to compare with *ISW* index, we also evaluate the search algorithm based on the following two indexing techniques and test their performance.

Algorithm 3 RNN query processing

```

Input : query point  $q$ 
Output: result set  $RNN(q)$ 
1 //Step 1: Traverse  $ISW$ 
2 for each leaf node  $R_i$  in  $ISW$  do
3   if  $\varphi^+(R_i, R_i) < \varphi^-(q, R_i)$  and  $R_i$  contains at
   least two objects then
4     prune  $R_i$ ;
5   end
6   if exist  $R_j$  that  $\varphi^+(R_i, R_j) < \varphi^-(q, R_i)$  then
7     prune  $R_i$ ;
8   end
9 end
10 //Step 2: Selectively receive the data
11 receive the objects in pre-selected regions;
12 put these objects into candidate set  $C$ ;
13 //Step 3: Query processing in client devices
14 for each object  $o$  in  $C$  do
15   if the nearest neighbor of  $o$  is  $q$  then
16     put  $o$  into  $RNN(q)$ ;
17   end
18 end
    
```

- *EB-index* [14], which adopts the same space partitioning mechanism with *ISW*, and the maximum/minimum distance table is computed as the bounds.
- R-tree. We employ the (1, m) interleaving scheme, and adopt the best-first manner for kNN search.

We first test the effect of the region number on the index size. We set the number of objects as 5K and vary the number of regions from 32 to 512. The size of both *ISW* and *EB-index* increases with the increasing number of regions as shown in Fig. 10. The size of *ISW* is a little larger than that of *EB-index*, because *ISW* stores two pairs of parameters for a pair of cells while *EB-index* stores a pair of bounds for a pair of cells. The size of R-tree has no relationship with the region number so that the index size does not change with the increasing region number.

7.1 Evaluation of kNN queries

In this section we compare *ISW* against *EB-index* and R-tree for processing kNN query, in terms of tuning time and access latency, respectively.

The first experiment studies the tuning time for kNN queries as a function of region number, which is shown in Fig. 11 (k is set to 50, and the number of objects is set to 5K). The tuning time equals to the volume of data packets received by the clients. As the region number increases, the volume of packets received by the clients with *EB-index* and *ISW* first reduce then keep stable. The reason is threefold. Firstly, with the smaller granularity of region, the clients can receive less data through the pruning methods.

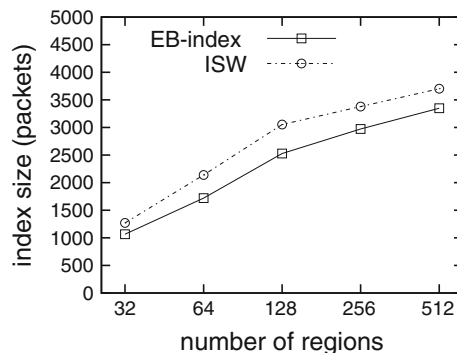


Fig. 10 Effect of region number on index size (kNN query)

Secondly, the finer partition of space results in that many data packets are not full, which weakens the benefits of the smaller region. Thirdly, larger number of regions will result in larger size of index. Therefore, both *ISW* and *EB-index* perform best when the number of regions is set as 128. The tuning time of *ISW* is a little lower than that of *EB-index* because the pruning strategy of *ISW* is more powerful. The tuning time of R-tree is stable and is much higher than the others (We do not draw the line of R-tree because it is too high than the other two indexes), because the organization of R-tree does not take into account the space partitioning.

Figure 12 illustrates the tuning time of kNN queries for various database sizes, ranging from 1 to 10K objects. The performance of *ISW* is better than *EB-index* due to its more effective pruning bounds. As the data size increases, the density of objects increases and hence more data are needed to be received. R-tree receives most data in a fixed area which increases slowly with the increasing data size, so the tuning time increases slowly as well (Due to the space limitation, the line of R-tree is not painted).

Figure 13 investigates the effect of the query selectivity (k) on the tuning time for kNN queries. As k increases from 20 to 500, the tuning time of all indexing techniques grows due to the larger search region and result size. *ISW* is consistently better than its competitors.

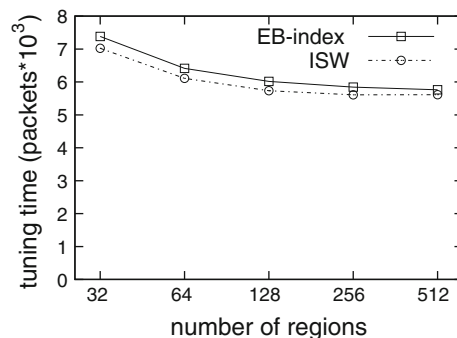


Fig. 11 Effect of region number on tuning time (kNN query)

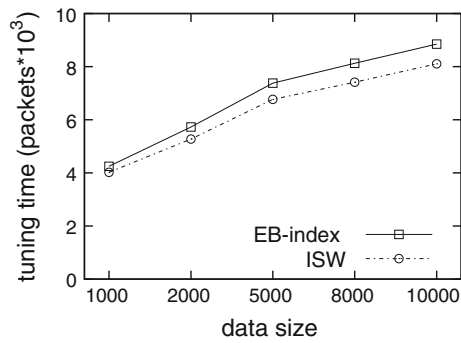


Fig. 12 Effect of data size on tuning time (kNN query)

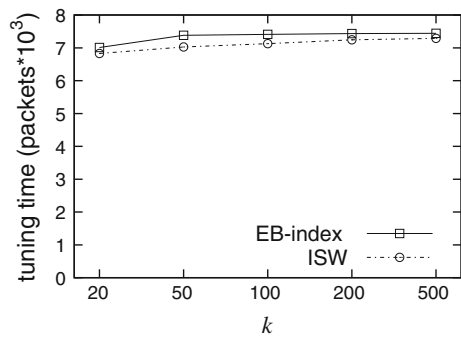
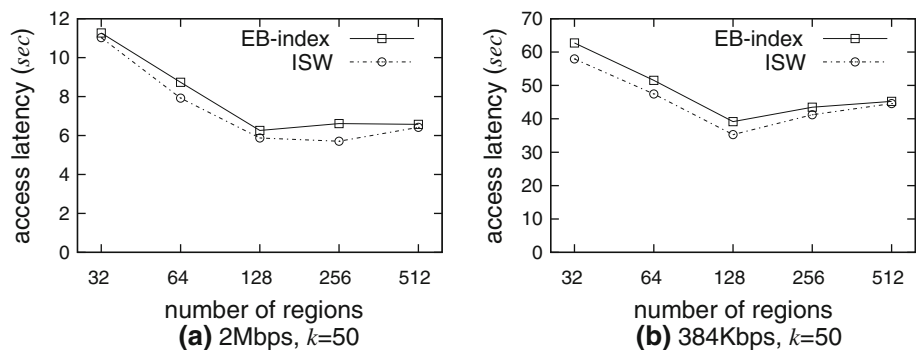


Fig. 13 Effect of k on tuning time (kNN query)

Figure 14(a, b) depicts the access latency for kNN queries as a function of region number, under 2 Mbps (Fig. 14(a)) and 384 Kbps (Fig. 14(b)) broadcast channels respectively. The latency equals to the time interval between the first time the clients tune in the broadcast channel to the time the clients receive the last data for a query. As the region number increases, the access latency of EB-index and ISW first reduce then increase slowly. When the space is partitioned into 128 regions, the system performs best. Even though the index segment is smaller when the region number is small, the latency is larger because the clients need to receive more objects. On the other hand, a finer partition means a larger index and hence

Fig. 14 Effect of region number on access latency



lengthens the broadcast cycle and consequently the access latency. The performance of ISW is better than EB-index.

Figure 15(a, b) illustrates the effect of datasize on the access latency under 2 Mbps (Fig. 15(a)) and 384 Kbps (Fig. 15(b)) broadcast channels respectively. The access latency of all indexing techniques increases as the data size increases from 1 to 10 K. The access latency of ISW and EB-index is much lower than that of R-tree because they can execute the queries in one cycle, while the methods adopting R-tree have backtracking and consume more than one broadcasting cycle.

Figure 16(a, b) shows the effect of k on access latency under 2 Mbps (Fig. 16(a)) and 384 Kbps (Fig. 16(b)) broadcast channels respectively. As the k increases, so does the latency. The reason is that more objects are needed to be received with the increasing k. ISW performs better than EB-index due to its more effective bounds. The access latency by adopting R-tree are much larger than by the other index. The reason is, with the increasing k, the required data and the volume of backtrack increase.

7.2 Evaluation of range queries

In this subsection we investigate the performance of the tuning time and access latency for the range query. As shown in Fig. 17, with the increasing number of regions, the tuning time by adopting both EB-index and ISW first increases significantly and then increases slowly. The reason is the same with Fig. 11. The tuning time by adopting ISW-index is less than the EB-index, and this graph describe the difference of a single query. Therefore, if the clients invoke queries frequently, the saving power is considerable.

Figure 18 depicts the effect of data size on tuning time. The tuning time by adopting R-tree is significantly larger than by ISW and EB-index because the search space can not be pruned by R-tree and all data are needed to be received. The tuning time of both ISW and EB-index increases when data size increases from 1K to 10K. As the data size increases, the density of objects increases, so that for the same query more data are needed to be received for ISW and EB-index. As the involving data increases, more

Fig. 15 Effect of data size on access latency (k NN query)

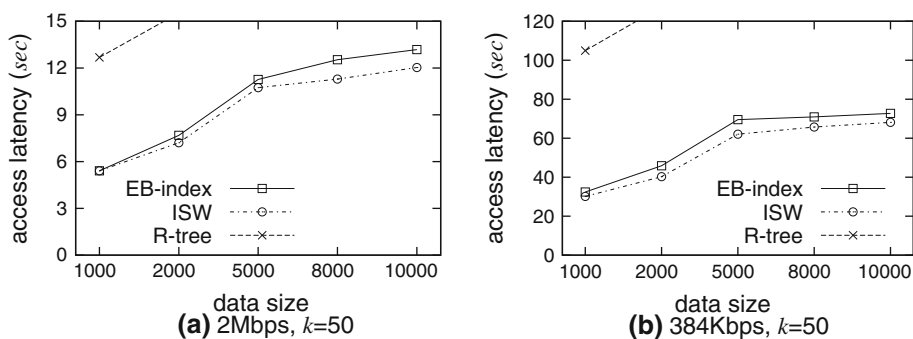


Fig. 16 Effect of k on access latency (k NN query)

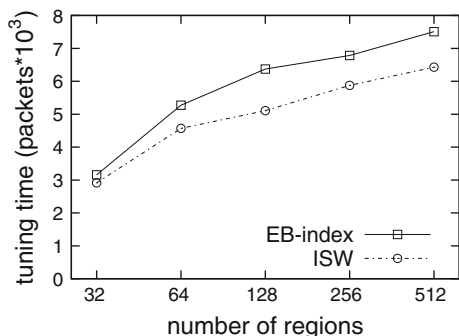
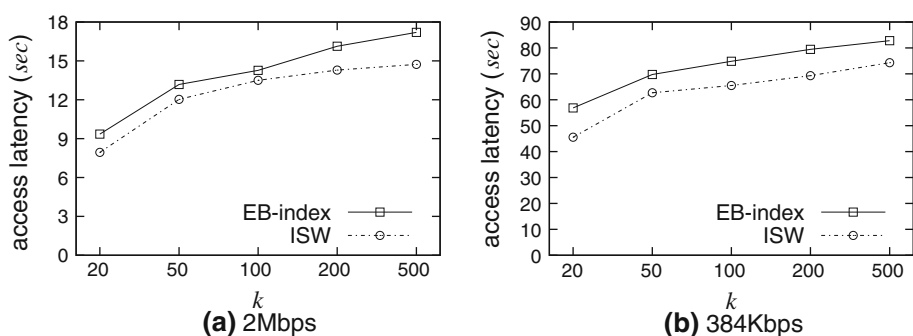


Fig. 17 Effect of number of regions on tuning time (range query)

unfilled packets are needed to be received, leading to increased tuning time.

We furthermore test the effect of the query range on the tuning time in Fig. 19. We vary the range from 2 to 15 % of the whole space and set the number of objects to 5K. The tuning time by adopting *ISW* increases with the increasing range, because larger range means more regions are needed to be downloaded. The tuning time by adopting *R-tree* is large and stable due to it receives all data no matter how large the range is.

We illustrate the access latency with the function of region number for the range query, under the 2 Mbps (Fig. 20(a)) and 384 Kbps (Fig. 20(b)) broadcast channels respectively. The latency of *ISW* and *EB-index* acquire the

minimum value when the region number equals to 128. The reason is the same with the effect of region number on latency for k NN query.

Figure 21 shows the access latency for various data sizes ranging from 1 to 10 K, under the 2 Mbps (Fig. 21(a)) and 384 Kbps (Fig. 21(b)) broadcast channels respectively. The latency of *ISW* and *EB-index* increases slowly. As the density of objects increases with the increasing data size, the number of objects inside the same range increases with the data size. The range is very small comparing to the whole search space so that the increasing objects inside the range do not significantly increase the latency of *ISW* and *EB-index*.

Finally, we test the effect of the query range on the access latency for range queries, under the 2 Mbps (Fig. 22(a)) and 384 Kbps (Fig. 22(b)) broadcast channels respectively. As the query range increases from 2 to 15 % of the whole space, the latency of all algorithms lengthens due to the larger search region and result size. *ISW* and *EB-index* are consistently better than *R-tree*, for the reason that both *ISW* and *EB-index* can provide pruning strategies and *R-tree* can not.

7.3 Evaluation of RNN queries

In this subsection we study the performance of tuning time and access latency for RNN query. Figure 23(a) depicts the

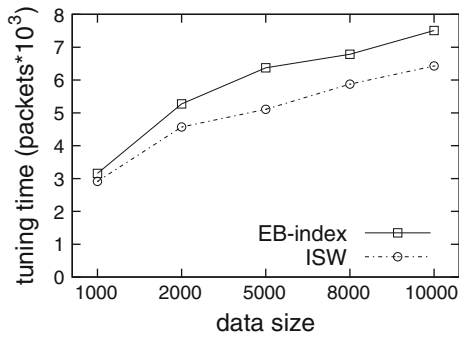


Fig. 18 Effect of data size on tuning time (range query)

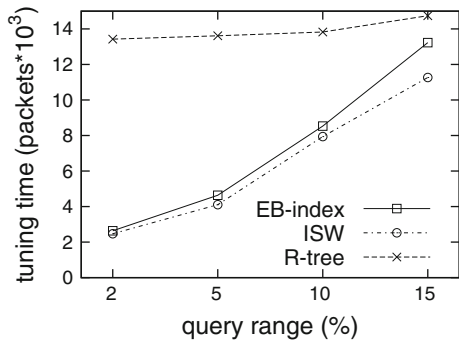


Fig. 19 Effect of query range on tuning time (range query)

Fig. 20 Effect of region number on access latency (range query)

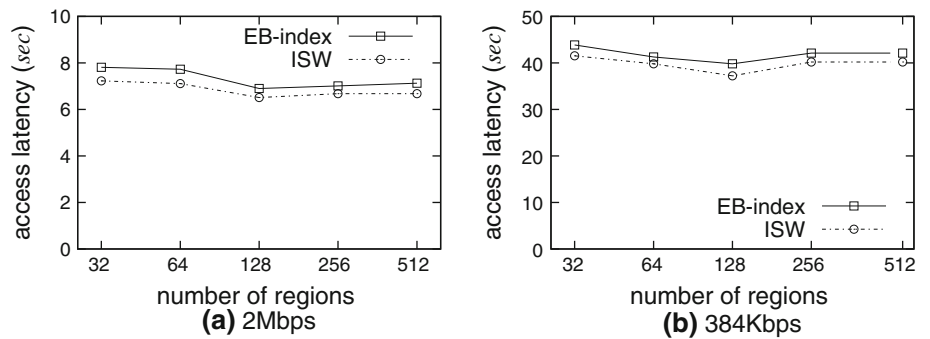
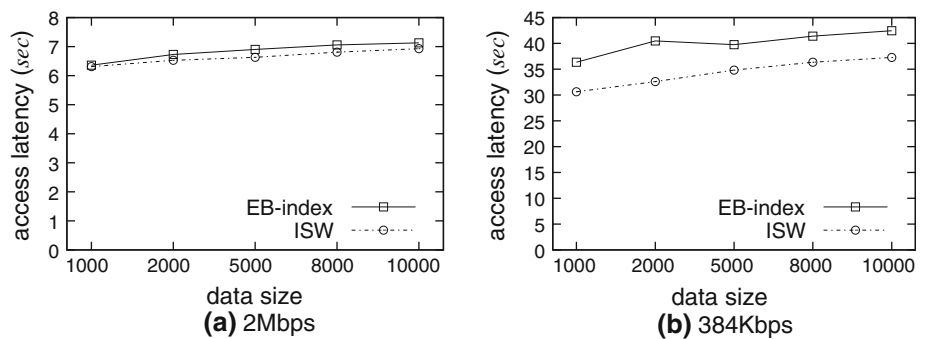


Fig. 21 Effect of data size on access latency (range query)



tuning time of RNN queries by adopting *EB-index*, *ISW* and R-tree. *ISW* performs better than any other index with varying region number. The main reason is, through the partition of the whole space, some unqualified regions are pruned and the tuning time is reduced by *ISW* index. Figure 23(b) shows the tuning time for RNN queries with the varying data size. Likewise, *ISW* performs better than the other indexes because *ISW* can prune more data that are not the query results.

In the next set of experiments, we compare *ISW* with other indexes in RNN queries with the varying number of objects (data size) from 1 to 10K. The results are shown in Fig. 24(a, b). We can observe that the access latency by adopting *ISW* is much lower than others. The main reason is that the methods adopting *ISW* can execute the queries in one cycle, while the methods adopting R-tree have backtracking and consume more than one broadcasting cycle. Moreover, the bounds of *ISW* are tighter than those of *EB-index*.

Lastly, we illustrate the access latency with the change of the region number for the range query, under the 2 Mbps (Fig. 25(a)) and 384 Kbps (Fig. 25(b)) broadcast channels respectively. The access latency of *ISW* is lower than that of *EB-index* and R-tree. And the latency of *ISW* and *EB-index* acquire the minimum value when the region number equals to 128. The reason is the same with the

Fig. 22 Effect of query range on access latency (range query)

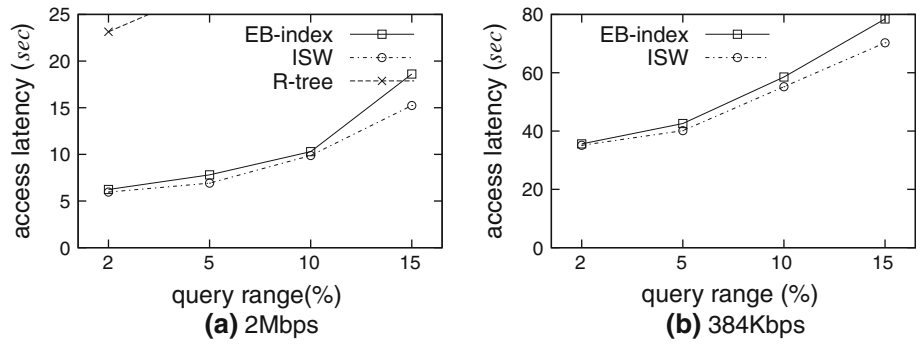


Fig. 23 Performance of tuning time for RNN query

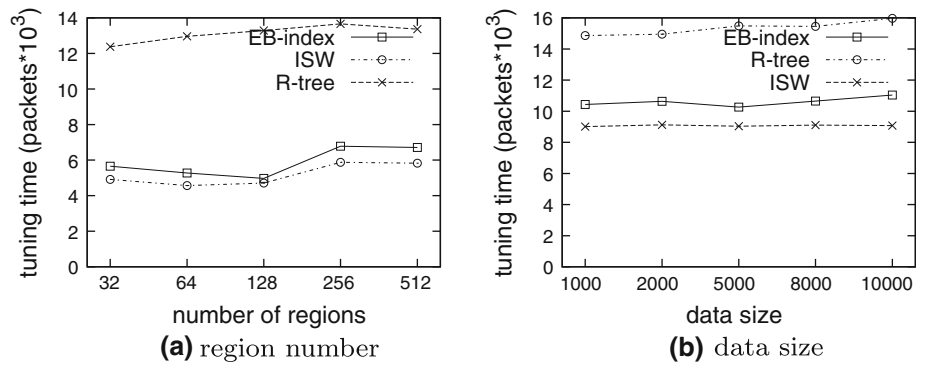


Fig. 24 Effect of region number on access latency (RNN query)

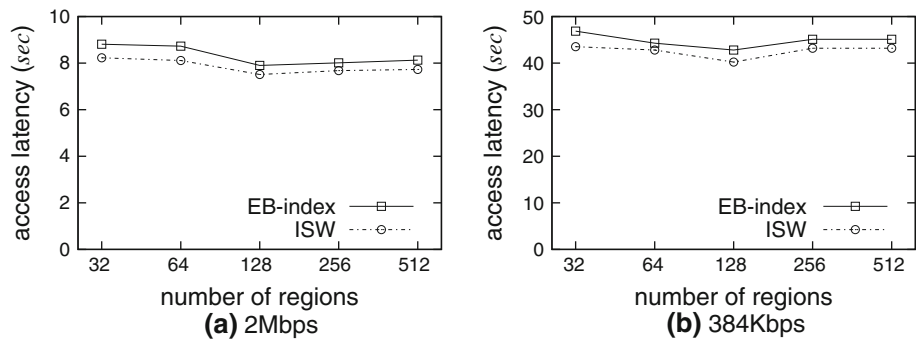
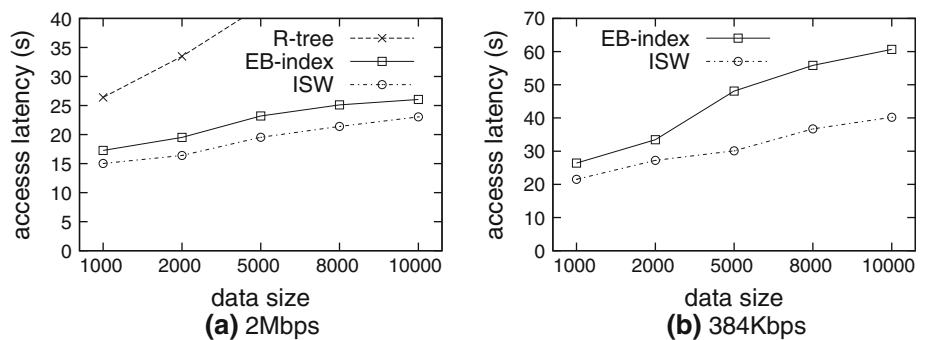


Fig. 25 Effect of data size on access latency (RNN query)



effect of the region number on the latency for the k NN query and range query.

8 Conclusions

This paper addresses the problem of answering k NN, range and RNN queries in road networks via broadcast channels. A new index structure based on kd -tree, namely ISW -index, is proposed to support the above spatial queries. For the query point q and any cell, ISW provides a pair of distance bounds, which is effective and powerful for pruning the search space. In this way, ISW can present a basic and general framework for processing spatial queries, and algorithms for k NN, range and RNN queries, based on ISW , are thus separately developed. Finally, we demonstrate the effectiveness and efficiency of our index technique and spatial query methods through the theoretical analysis and experiments.

In the future, we will design optimized mechanism for efficiently executing continuous spatial queries in spatial networks via broadcast channels.

Acknowledgments This work was supported in part by the the National Natural Science Foundation of China under grants Nos. 60933001, 61003058, and the Fundamental Research Funds for the Central Universities under grants No.N100604014 and No. N110604001.

References

- Zheng, B., Xu, J., Lee, W.-C., & Lee, L. (2006). Grid-partition index: A hybrid method for nearest-neighbor queries in wireless location-based services. *Journal on Very Large Data Bases*, 15, 21–39.
- Park, K., Choo, H., & Valduriez, P. (2010). A scalable energy-efficient continuous nearest neighbor search in wireless broadcast systems. *Wireless Networks*, 16, 1011–1031.
- Lin, L.-F., Chen, C.-C., & Lee, C. (2010). Benefit-oriented data retrieval in data broadcast environments. *Wireless Networks*, 16(1):1–15.
- Gedik, B., Singh, A., & Liu, L. (2004). Energy efficient exact knn search in wireless broadcast environments. In *GIS* (pp. 137–146).
- Zheng, B., Lee, W.-C., Lee, K., Lee, D. L., & Shao, M. (2009). A distributed spatial index for error-prone wireless data broadcast. *The VLDB Journal*, 18, 959–986.
- Ambient information network. <http://www.ambientdevices.com/cat/index.html>.
- Zheng, B., Lee, W.-C., Lee, & D. L. (2004). Spatial queries in wireless broadcast systems. *Wireless Networks*, 10, 723–736.
- Zhang, X., Lee, W.-C., Mitra, P., & Zheng, B. (2008). Processing transitive nearest-neighbor queries in multi-channel access environments. In *Proceedings of EDBT* (pp. 452–463).
- Zheng, B., Lee, W.-C., Lee, & D. L. (2007). On searching continuous k nearest neighbors in wireless data broadcast systems. *IEEE Transactions on Mobile Computing*, 6, 748–761.
- Shahabi, C., Kolahdouzan, M. R., & Sharifzadeh, M. (2002). A road network embedding technique for k -nearest neighbor search in moving object databases. In *GIS* (pp. 94–100).
- Papadias, D., Zhang, J., Mamoulis, N., & Tao, Y. (2003). Query processing in spatial network databases. In *VLDB* (pp. 802–813).
- Kolahdouzan, M., & Shahabi, C. (2004). Voronoi-based k nearest neighbor search for spatial network databases. In *VLDB* (pp. 840–851).
- Samet, H., Sankaranarayanan, J., & Alborzi, H. (2008). Scalable network distance browsing in spatial databases. In *SIGMOD* (pp. 43–54).
- Kellaris, G., & Mouratidis, K. (2010). Shortest path computation on air indexes. In *VLDB* (pp. 747–757).
- Anticaglia, S., Barsi, F., Bertossi, A., Lamele, L., & Pinotti, M. (2008). Efficient heuristics for data broadcasting on multiple channels. *Wireless Networks*, 14(2), 219–231.
- Imielinski, T., Viswanathan, S., & Badrinath, B. (1997). Data on air: Organization and access. *IEEE Transactions on Knowledge and Data Engineering*, 9, 353–372.
- Mouratidis, K., Bakiras, S., & Papadias, D. (2009). Continuous monitoring of spatial queries in wireless broadcast environments. *IEEE Transactions on Mobile Computing*, 8, 1297–1311.
- Lee, W.-C., & Zheng, B. (2005). Dsi: A fully distributed spatial index for location-based wireless broadcast services. In *ICDCS* (pp. 349–358).
- Xu, J., Lee, W.-C., & Tang, X. (2004). Exponential index: a parameterized distributed indexing scheme for data on air. In *Proceedings of MobiSys* (pp. 153–164).
- Shen, J.-H., & Chang, Y.-I. (2008). An efficient nonuniform index in the wireless broadcast environments. *Journal of Systems and Software*, 81, 2091–2103.
- Zhong, J., Wu, W., Shi, Y., & Gao, X. (2011). Energy-efficient tree-based indexing schemes for information retrieval in wireless data broadcast. In *Proceedings of DASFAA* (pp. 335–351).
- Yiu, M. L., Papadias, D., Mamoulis, N., & Tao, Y. (2006). Reverse nearest neighbors in large graphs. *IEEE Transactions on Knowledge and Data Engineering*, 18(4), 540–553.
- Möhring, R., Schilling, H., Schütz, B., Wagner, D., & Willhalm, T. (2006). Partitioning graphs to speedup dijkstra's algorithm. *Journal of Experimental Algorithmics*, 11, 1–29.
- Xiao, X., Yao, B., & Li, F. (2011). Optimal location queries in road network databases. In *ICDE* (pp. 804–815).

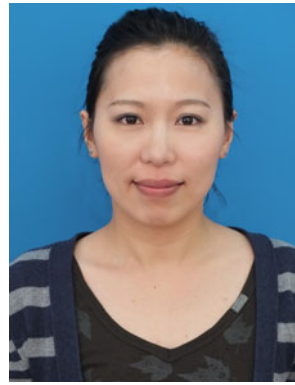
Author Biographies



Yanqiu Wang received the B.E. degree and M.E. degree in Computer science from Northeastern University of China in 2007, 2009 respectively. Currently, She is a Ph.D. candidate in Northeastern University of China. Her research interests include moving objects database, spatial database management.



Chuanfei Xu received the B.E. degree in 2007 from Shenyang University of Technology, the M.E. degree in 2009 from Computer Science, Northeastern University. Currently, he is a Ph.D. candidate in Computer Software and Theory, Northeastern University, China. His research interests include spatial database management and uncertain data management.



Mo Chen received her Ph.D. degree in the College of Information Science and Engineering, Northeastern University of China at 2012. She is a lecturer of Northeastern University, China. She is a member of CCF and ACM. Her major research interests include spatial data management, uncertain database.



Yu Gu received his B.E., M.E., and Ph.D. degree in Computer Software and Theory from Northeastern University of China, in 2004, 2007 and 2010, respectively. Currently, he is an associate professor in Computer Software and Theory, Northeastern University, China. His current area of research is spatial database management and graph data management.



Ge Yu received his B.E. degree and M.E. degree in Computer Science from Northeastern University of China in 1982 and 1986, respectively, Ph.D. degree in Computer Science from Kyushu University of Japan in 1996. He has been a professor at Northeastern University of China since 1996. His research interests include data base theory and technology, distributed and parallel systems, embedded software. He is a senior member of the IEEE.