


RESEARCH

Open Access



DIM: a distributed air index based on MapReduce for spatial query processing in road networks

Ran Jin^{1,2*} , Gang Chen¹, Anthony K. H. Tung³, Lidan Shou¹, Beng Chin Ooi³ and Yuting Gu²

Abstract

In this era of mobile application and socialization, *location-based services* (LBSs) have become unprecedentedly popular and emphasized. By submitting its location to the service provider, the mobile terminal can obtain useful service. As an important technology that can provide location-based services, spatial query processing has become a research hotspot. According to the problems existing in current road network partitioning, first of all, this paper proposes the partition density formula and the optimal road network partitioning method according to the partition density. Based on that, we propose the concepts of partition self-attraction, mutual attraction, and merger factor of partition to effectively merge the partitions with low densities, which can further reduce the number of unnecessary broadcast frames and optimize the index method. Then, we propose the real distributed air index method DIM and *k*NN spatial query processing algorithm based on the MapReduce platform. According to the DIM index method, the Name Node only stores the prime index, while not storing the data, and each frame consists of defined index and the data in a partition that need to be broadcasted. The mobile user can quickly localize the frame data that they need to obtain according to the main index or the index of current frame, in this way to reduce the tuning time and access latency, which can significantly optimize the query efficiency. Massive experiments have proved the stability and effectiveness of the proposed algorithm.

Keywords: Spatial query, Air index, Wireless broadcast, Road networks, *k*NN

1 Introduction

With the rapid rise of *location-based services* (LBSs), there has been an increasing interest in wireless data services from both industrial and academic communities in recent years. To provide location-based services, spatial query processing is an important enabling technology and acquires a great attention. As the core of data mining applications and decision support applications, there is a broad application prospect for the research of location-dependent query processing technology in wireless data broadcast.

In general, there are mainly two basic approaches for information access through wireless data broadcast technology: point-to-point access (on-demand access) and periodic broadcast. For point-to-point access, a mobile client issues a query to the server, which is responsible for processing query in road networks and returns the query result to the mobile client through a point-to-point channel. Point-to-point access has the following shortcomings: first, it might cause network overloading when many people issue the same query to the server in the same area. Second, it might give away the location privacy because a mobile client needs to send his current location to the server. For periodic broadcast, which can be viewed as “storage on the air” and save power on the client side by avoiding power con-

* Correspondence: ranjin@163.com

¹College of Computer Science and Technology, Zhejiang University, No.38, Zheda Road, Hangzhou, China

²College of Electronics and Computer Science, Zhejiang Wanli University, No.8, South Qianhu Road, Ningbo, China

Full list of author information is available at the end of the article

suming uplink transmission [1], the server periodically broadcasts the data with index via a wireless channel, while the mobile client tunes in the channel to retrieve interesting information. In this mode, the server only needs to monitor the information of the data objects, but it is unaware of the mobile clients and their queries [2]. So it can satisfy an arbitrary number of mobile clients at the same time. Moreover, the location privacy of clients is well protected. However, there are some disadvantages, e.g., data access in the wireless broadcast is limited to be sequential.

Spatial queries are database queries supported by geodatabases and differ from traditional queries. As an auxiliary spatial data structure, index can improve greatly the performance of spatial query processing. There are two types of spatial index: disk-based index and air index. A disk-based index is used in point-to-point methods, which can speed up spatial query processing in road networks, and has six characteristics, but is not suitable in wireless data broadcast [1]. An air index is used in periodic broadcast methods, which is usually in front of the data in a broadcast cycle. Mobile clients rely on the air index to find the demanded data. The purpose of using indexes is to reduce access latency and tuning time, and there are two performance metrics used to measure access efficiency and energy [3].

There are many research achievements on designing broadcast-based spatial query processing methods. They can be roughly divided into three categories: earlier approaches only consider the spatial queries in a Euclidean space, e.g., Zheng et al. [4] present a distributed spatial index (called DSI); DSI is highly efficient because it has a linear yet fully distributed structure that naturally shares links in different search paths. However, the objects' movement is constrained in a road network in real applications [2]. In recent years, many scholars have begun to process spatial queries in a road network and propose many efficient algorithms, e.g., Sun et al. [2] present an air index called Network Partition Index (NPI) to support efficient spatial query processing in road networks via wireless broadcast and propose multiple client-side algorithms to facilitate the processing of different spatial queries such as k NN query, range query, and CNN query. However, a mobile client needs to wait for the arrival of the index before starting a query processing, which can lead to long access latency [5]. Moreover, a client must wait for the next broadcast cycle to arrive if he misses some data. In order to guarantee the efficiency and scalability of query processing, some distributed query methods are proposed in a road network to issue the above problems, e.g., Li et al. [5] explore the problem of spatial query

processing in road sensor networks by means of wireless data broadcast and present an efficient method to partition the record-keeping information about the underlying road sensor network and its associated objects, by which a fully distributed air index, called integrated exponential index, is developed, based on an extended version of the Hilbert curve. However, if the grid cells are increased quickly, there must be multiple handovers between the indexes of frames to query the required data. So both the access latency and tuning time are long.

In this paper, we propose a real distributed air index based on MapReduce and k NN query processing algorithm to overcome the existing problems and limitations. The contributions of this paper can be summarized as follows:

1. Based on research of the working mechanism of wireless data broadcast, according to the problems currently existing in the road network partitioning method, we propose the partition density formula and further propose the optimized road network partitioning method based on the partition density. This method calculates the density of each partition and determines whether secondary partitioning is required according to the density value, which has addressed the defect of big broadcast cycle caused by simple even partitioning.
2. Based on the road network partitioning, we propose the concepts of partition self-attraction, mutual attraction, and merger factor of partition to effectively merge the partitions with low densities, which can further reduce the number of unnecessary broadcast frames and optimize the index method.
3. We propose the real distributed air index method DIM based on the MapReduce platform. This index method evenly stores all broadcast cycle frames in various Data Nodes according to the number of Data Nodes on the cloud platform. The Name Node only stores the prime index, while not storing the data, and each frame consists of defined index and the data in a partition that need to be broadcasted. The mobile user can quickly localize the frame data that they need to obtain according to the main index or the index of current frame, in this way to reduce the tuning time and access latency, which can significantly optimize the query efficiency.
4. Based on the DIM index method, we propose the k NN spatial query processing algorithm.
5. We conducted many experiments to verify the performance of the proposed method. First of all, we use the access latency and tuning time as the measurement indicators, and according to

different k values that need to be queried, we conducted experiments of four index methods on four actual city road network datasets to prove the stability and effectiveness of the DIM index method proposed in this paper. Then, we compare the performance of four index methods according to different densities of targets in the road network.

The rest of this paper is organized as follows. Section 2 briefly introduces the wireless data broadcast and the query processing technologies in road networks and so on. Section 3 reviews the related work. The proposed real distributed air index is discussed in Section 4. Then, we describe our distributed k NN query algorithms based on MapReduce in Section 5. In Section 6, we report the results of experimental evaluation. Finally, we conclude this paper in Section 7.

2 Preliminaries

In the wireless data broadcast system, the server repeatedly broadcasts data to the client, while the mobile client tunes the broadcast channel to search for the air data, process, and query the data on the local terminal. The broadcast access efficiency and energy consumption are two critical parameters to measure the performance of the client. In order to reduce energy consumption, the mobile device generally has two working modes—i.e., the doze mode and the active mode—while the energy consumption in doze mode is significantly lower than that in active mode. When the device does not need to received data and is in the doze mode, it can significantly reduce the energy consumption. The access latency and tuning time are the main performance indicators of wireless broadcast system [6, 7].

Define 1 Tuning time The time period a mobile client stays active to receive the requested data.

Define 2 Access latency The time elapsed from the moment a query is invoked to the moment the answers are received.

$$\begin{aligned}
 \text{Indexsize} &= \left(\sum_{p^0=0}^{k^0} pID^0 + \sum_{p^0=0}^{k'_0} \psi_{n0} \right) + \left(\sum_{p^1=0}^{k^1} pID^1 + \sum_{p^1=0}^{k'_1} \psi_{n1} \right) \\
 &+ \left(\sum_{p^2=0}^{k^2} pID^2 + \sum_{p^2=0}^{k'_2} \psi_{n2} \right) + \left(\sum_{p^3=0}^{k^3} pID^3 + \sum_{p^3=0}^{k'_3} \psi_{n3} \right) \\
 &= \sum_{R_n=0}^3 \left(\sum_{p^n=0}^{k^n} pID + \sum_{p^n=0}^{k'_n} \psi \right) \tag{1}
 \end{aligned}$$

Air indexing techniques are often used for conserving the energy of mobile clients [2]. Generally, an air index can be allocated to data items on the wireless broadcast and divided into two types: $(1, m)$ index scheme and distributed index scheme. The $(1, m)$ index scheme makes the index for all of the data items allocated m times preceding each $1/m$ fraction of data items on the wireless channel [8, 9, 25] as shown in Fig. 1. This scheme can take a long time to meet the index after tuning into the wireless channel because the repeated index information extends the broadcast cycle. For the distributed index scheme, all data items are divided into several fractions. Each fraction is allocated behind responding index. This scheme has a shorter broadcast cycle and can take less time to meet the index after tuning into the channel than under the $(1, m)$ index scheme. Thus, the distributed scheme has the advantage of improving the access time that is affected by the length of the broadcast cycle [3, 28]. As analyzed in [1], the size of HGI spatial index can be computed by Eq. 1, and the optimal balance between the tuning time and access latency can be reached when $m = \sqrt{\frac{\text{size of data}}{\text{Index size}}}$ as analyzed in [9].

Define 3 Road network A road network is a two-dimensional space, which can be denoted as $G = (V,$

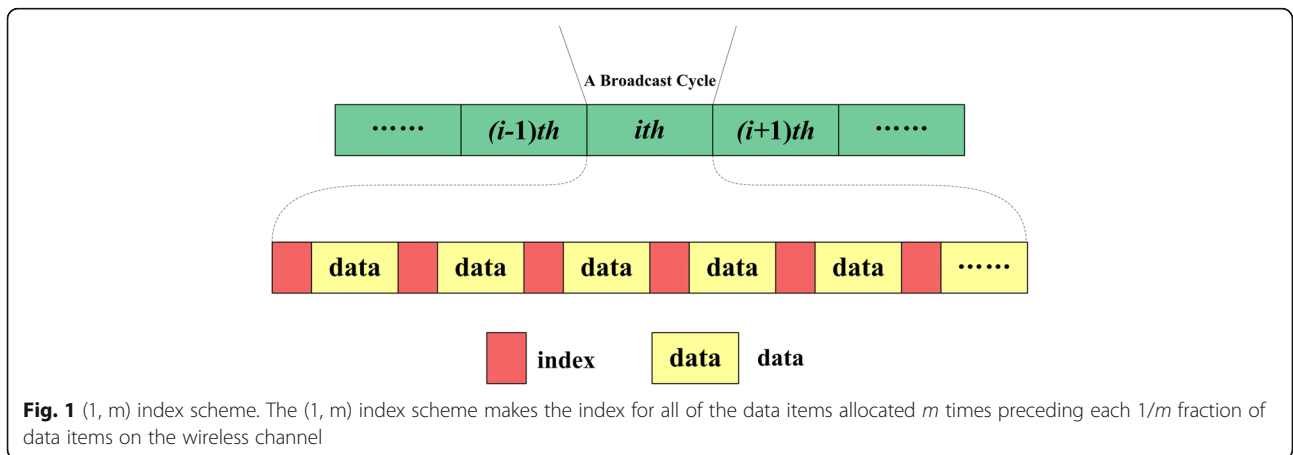


Fig. 1 $(1, m)$ index scheme. The $(1, m)$ index scheme makes the index for all of the data items allocated m times preceding each $1/m$ fraction of data items on the wireless channel

E), in which $V = \{v_1, v_2, v_3, \dots, v_{|V|}\}$ is a set of nodes in a road network. Each $v \in V$ consists of a location $v.x, v.y$, where $v.x$ is the x -coordinate and $v.y$ is the y -coordinate of the node. So a node is denoted by $v = (v.x, v.y)$. $E = \{e_1, e_2, e_3, \dots, e_{|E|}\}$ is a set of edges between nodes. In a similar way, for each $e \in E$, it consists of two nodes, one is the start node v_s , and another is the end node v_e . There is a weight w , denoting the length of the edge. Therefore, an edge is denoted by $e = (v_s, v_e, w)$. The nodes and edges are the basic elements of a road network. In addition, we denote the $\text{dist}(v_i, v_j)$ as the shortest path of two nodes in a road network. Figure 2 is a sample roadmap.

Define 4 Cost of partition While partitioning the set of nodes V in a road network G into different partitions $\Phi = \{\Phi_1, \Phi_2, \Phi_3, \dots, \Phi_k\}$, $1 < k <= |\Phi|$, the cost of partitioning is defined as the aggregation of affinity values of all possible node pairs (v_i, v_j) for which v_i and v_j lie in different partition in the final result [11].

We need to partition a two-dimensional road network into a linear sequence for broadcasting data. Generally, the partitioning methods include kd tree, quad-tree, grid

partitioning, and so on. The different partitions exhibit different cost.

3 Related works

3.1 Index for spatial query processing in wireless data broadcast

Gedik [6] describes mechanisms to perform exact k NN search on conventional sequential-access R-trees and optimize established k NN search algorithms. The author proposes a novel use of histograms for guiding the search and derives analytical results on maximum queue size and node access count. In addition, the effects of different broadcast organizations on search performance and challenge of the traditional use of Depth-First (dfs) organization are discussed. Wireless data broadcast is a promising technique for information dissemination that leverages the computational capabilities of the mobile devices in order to enhance the scalability of the system. Previous work on spatial query processing for wireless broadcast systems has only considered snapshot queries over static data. In

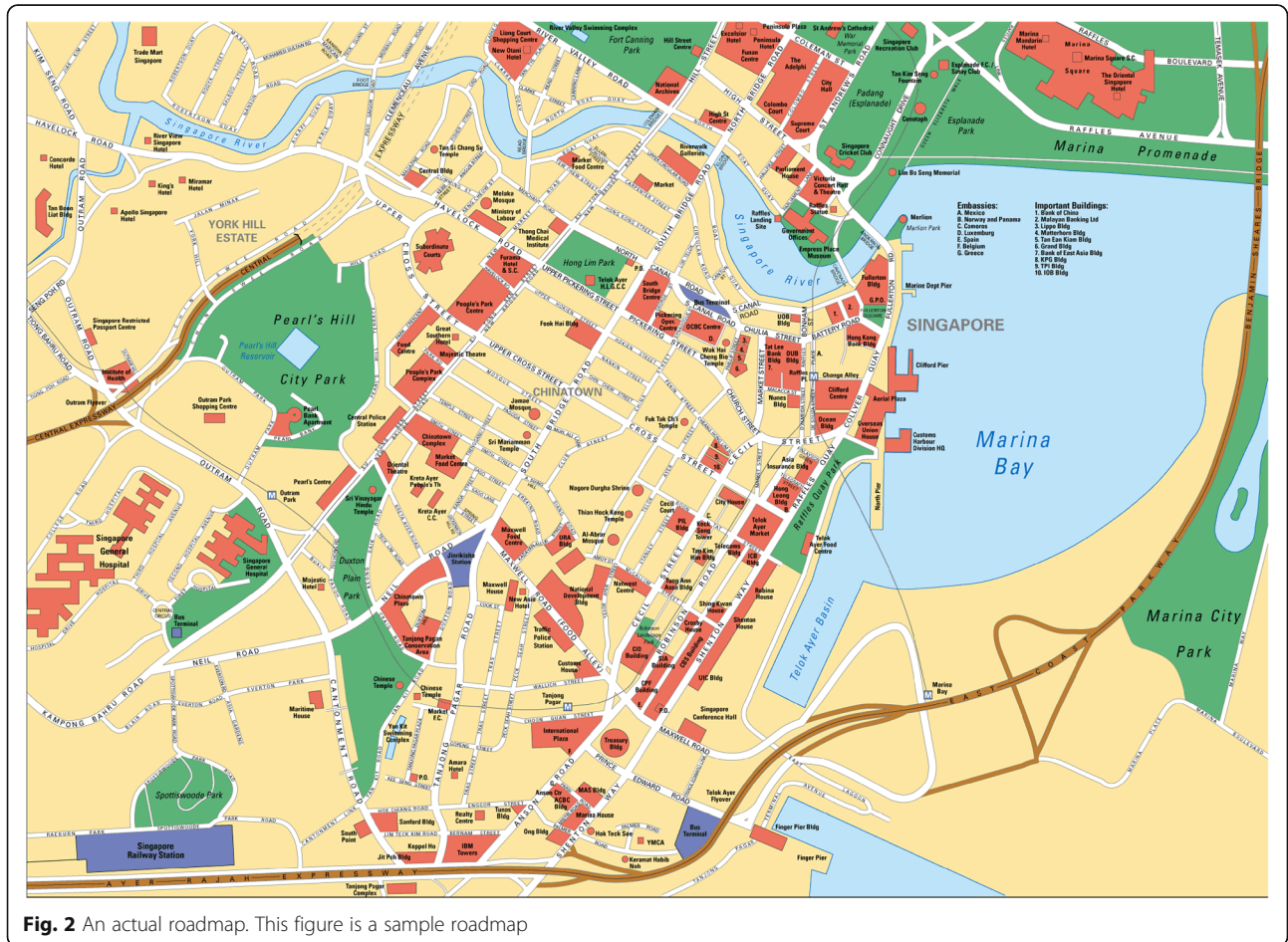


Fig. 2 An actual roadmap. This figure is a sample roadmap

[7], Mouratidis proposes the Broadcast Grid Index (BGI) method, which is suitable for both snapshot and continuous queries. Furthermore, BGI extends to the case that the data are also dynamic. Park [8] proposes new spatial query processing algorithms to support Mobile Continuous Nearest Neighbor Query (MCNNQ) in wireless broadcast environments. The solution provides a general client-server architecture for answering MCNNQ on objects with unknown, and possibly variable, movement types and enables the application of spatio-temporal access methods specifically designed for a particular type, to arbitrary movements without any false misses. The proposed algorithm does not require any conventional spatial index for MCNNQ processing. It can be adapted to static or moving objects and does not require additional knowledge (e.g., direction of moving objects) beyond the maximum speed and the location of each object. Zheng [10] introduces a new index method, called the grid-partition index, to support NN search in both on-demand access and periodic broadcast modes of mobile computing. The grid-partition index is constructed based on the Voronoi diagram. It has two distinctive characteristics. First, it divides the solution space into grid cells such that a query point can be efficiently mapped into a grid cell around which the nearest object is located. This significantly reduces the search space. Second, the grid-partition index stores the objects that are potential NNs of any query falling within the cell. The storage of objects, instead of the Voronoi cells, makes the grid-partition index a hybrid of the solution-based and object-based approaches. As a result, it achieves a much more compact representation than the pure solution-based approach and avoids backtracked traversals required in the typical object-based approach, thus realizing the advantages of both approaches. Several works on similarity queries are proposed in [11–15].

However, these methods operate on the Euclidean space. Thus, they are unable to provide the high quality of service to the query clients because they cannot consider the road networks where the query clients actually move along.

3.2 Index for spatial queries in road networks

Most of the existing broadcast-based LBQ methods are aimed at Euclidean spaces and cannot be readily extended to road networks. Recently, many researchers have presented several methods to process spatial queries in road networks [2, 16–24]. Sun et al. [2] present an air index called Network Partition Index (NPI) to support efficient spatial query processing in road networks via wireless broadcast. The main idea is to partition the road network into a number of regions and then build the index to carry

some pre-computation information of each region. Wang et al. [22] propose a novel index for spatial queries in wireless broadcast environments (ISW). With the reasonable organization and the effectively pre-computation 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. In [23], Jing proposes an energy-efficient scheme for on air shortest path query processing on road networks, which leverages an elaborate air index called BagIndex based upon the novel Hilbert-based heuristic tree decomposition for the road networks. Experimental results show that the proposed approach incurs less energy consumption on both communication and computation than the previous schemes. Motivated by scalability challenges faced in the mobile network industry, Kellaris [18] proposes adopting the wireless broadcast model for such location-dependent applications. In this model, the data are continuously transmitted on the air, while clients listen to the broadcast and process their queries locally. Although spatial problems have been considered in this environment, there exists no study on shortest path queries in road networks. The author develops the first framework to compute the shortest paths on the air and demonstrates the practicality and efficiency of our techniques through experiments with real road networks and actual device specifications. Li [19] presents an efficient method to partition the network Voronoi diagram (NVD) structure of the underlying road networks into a set of grid cells and number the grid cells obtained, based on which further proposes an NVD-based distributed air index (NVD-DI) to support CN3B query processing. However, the above methods are subjected to conditional constraints when they are executed.

3.3 Distributed index for spatial query processing in road networks

In order to improve query efficiency, a number of researchers have begun to design distributed air index to optimize spatial query processing [5, 25–35]. In [5], Li presents an efficient method to partition the record-keeping information about the underlying road sensor network and its associated objects, by which a fully distributed air index is developed, called integrated exponential index, based on an extended version of the Hilbert curve. Moreover, efficient client-side algorithms to facilitate the processing of several kinds of spatial queries are proposed, including k NN query, Ck NN query, and range query. Seokjin [25] proposes a distributed air index based on a maximum boundary rectangle (MaxBR) over grid-cells (abbreviated DAIM), which uses MaxBRs for filtering out hot data items on the wireless channel. Unlike the existing index that repeats regular data items in close proximity to hot items at the same frequency as hot data items in a broadcast cycle, DAIM makes it possible to

repeat only hot data items in a cycle and reduces the length of the broadcast cycle. In [26], Seokjin proposes a data scheduling scheme letting the popular items appear more frequently on the channel and grid-based distributed index for non-flat broadcast (GDIN) for window query processing. The proposed GDIN allows quick and energy-efficient processing of window query, matching the clients' linear channel access pattern and letting the clients access only the queried data items.

However, the above methods exist defects, e.g., when the grid cells are large, the number of frames of a broadcast cycle will be large, so the IEI proposed by Li et al. [5] have to transfer many times for retrieving needed data.

In this paper, we focus on the problem of spatial query processing in a road network on wireless data broadcast and design a real distributed air index based on MapReduce (DIM). Furthermore, we propose an efficient k NN query algorithm using distributed air index.

4 Distributed air index on MapReduce

In this section, we describe how to design a distributed air index on MapReduce in a road network, namely, DIM index. In fact, DIM is a real distributed index using the Hadoop parallel idea, which can provide an efficient guideline for spatial querying without switching frequently between multiple frames. In Section 4.1, we propose an efficient partitioning method for a road network; then, DIM index is described in Section 4.2.

4.1 Partitioning the road network

As mentioned in Section 2, we need to partition a two-dimensional road network into a linear sequence for broadcasting data and adopt space-filling curve to partition the entire road network. In [5], authors partition it according to the $N_{\text{weighted-entity}}$ value, but the value is not good to represent the congestion of each grid cell. In order to issue this problem, inspired by [11], we propose a partitioning method based on cell density.

Define 5 Cell density Given a set of data objects $\Theta_a = \{o_{1a}, o_{2a}, \dots, o_{n|\Theta_a}\}$ in cell a and a set of general node $V_a = \{v_{1a}, v_{2a}, v_{3a}, \dots, v_{n|V_a}\}$ in cell a , each object is located on one edge, and the density $\rho(o_{ia})$ of each data object is defined as the number of objects closer than a predefined distance threshold $\alpha^{|\Theta_a|}$ to o_{ia} ; it is formulated in Eq. (2). Vice versa, the density $\rho(v_{ia})$ of each node is defined as the number of nodes closer than a predefined distance threshold $\beta^{|V_a|}$ to v_{ia} . So the cell density $\rho(\Theta_a)$ is acquired by adding the density of all data objects in this cell as shown in Eq. (4). Equation (5) represents the density of all nodes in this cell. Finally, the cell density $\rho(\phi_a)$ is computed by Eq. (6), in which ϕ_a represents a cell.

$$\rho(o_{ia}) = \sum_j \Gamma(\text{dist}(o_{ia}, o_{ja}) - \alpha^{|\Theta_a|}) \quad (2)$$

$$\Gamma(x) = \begin{cases} 1 & x < 0 \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

$$\rho(\Theta_a) = \sum_i \rho(o_{ia}) \quad (4)$$

$$\rho(V_a) = \sum_i \rho(v_{ia}) \quad (5)$$

$$\rho(\phi_a) = \sum_i \rho(o_{ia}) + \sum_i \rho(v_{ia}) \quad (6)$$

During the partitioning process, first of all, regard the whole road network as an entity and divide it into 2×2 cells with side length of w ; then, calculate the density $\rho(\phi_f)$ ($f = 1, 2, 3, 4$) of each partition, if the density of partition is higher than the threshold value λ , further divide this partition to 2×2 smaller partitions until the density of each partition is smaller than the threshold value λ . Figure 3a shows a sample of road network, and the road network diagram after partitioning is as shown in Fig. 3b. In Fig. 3a, the round peak refers to the node, which can be regarded as a crossroad in the road network, and the connecting line between two nodes can be regarded as a section. The square peak represents the object, and they are all located in a certain section. The five-pointed star refers to a query point.

According to the Define 3 and 5, the network distance between two objects is determined by the length of the shortest path connecting two objects in road network, which is denoted as $\text{dist}(o_i, o_j)$ between v_i and v_j . If there is an object o_i in the edge e_{ij} , then $\text{dist}(o_i, q) = \min\{\text{dist}(o_i, v_i) + \text{dist}(q, v_i), \text{dist}(o_i, v_j) + \text{dist}(q, v_j)\}$, where $\text{dist}(o_i, v_i)$ ($\text{dist}(o_i, v_j)$) denotes the network distance from o_i to v_i (v_j). In the wireless data broadcast mode, the data and the index are organized in a one-dimensional frame form, one frame corresponding to a partition in the road network. In order to effectively organize the broadcast data, we use a space-filling curve to number the cells to form an ordered sequence so that neighboring cells are close to each other in the sequence. In this paper, we adopt a combination of several different orders of the Hilbert curve. A Hilbert curve (also known as a Hilbert space-filling curve) is a continuous fractal space-filling curve first described by the German mathematician David Hilbert in 1891, as a variant of the space-filling curves discovered by Giuseppe Peano in 1890, which has four basic subdivision modes as shown in Fig. 4a. Each subdivision mode includes two basic parts: *Cups* and

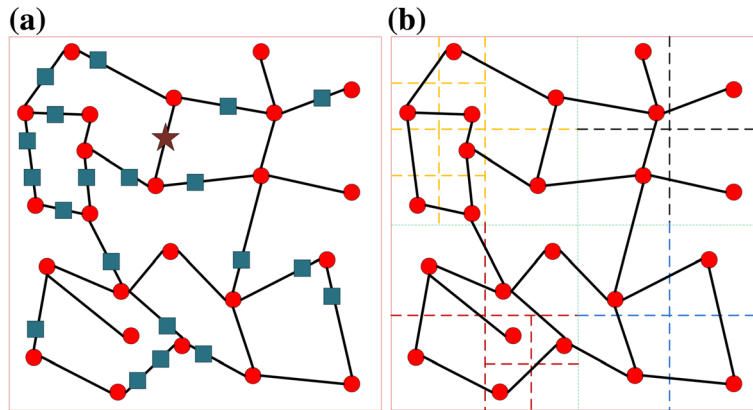


Fig. 3 Our partitioning method for a road network. **a** A sample of road network. **b** Partitioned road network. In **a**, solid dots represent nodes, which can be viewed as intersections in the road map, and the edges between the two nodes are treated as road sections

Joins. A Cup is a square with one open side, and a Join is a vector that joins two cups.

In the wireless data broadcast model, the data and indices are organized in the form of one-dimensional frames, and one frame corresponds to a partition in the road network. In order to effectively organize the broadcast data, as described in [4, 5], after dividing the whole road network into 2×2 cells, each cell is represented by a red dot; the dots are connected with lines, and they form a cup with opening downward. This is the first layer of Hilbert curve partitioning, as shown in Fig. 4a. Calculate the densities of four partitions on the first layer. According to Eq. (6), conduct secondary partitioning of partitions with densities higher than the threshold value, as shown in Fig. 4b. The final partitioning result of Hilbert curve and the data broadcast cycle route of road network are as shown in Fig. 4c. The final Hilbert curve result and a road network data broadcast cycle are shown in Fig. 4d.

We number each partition in the road network according to the direction of Hilbert curve partitioning and the partitioning level. If the subscript consists of three digits, it indicates that this partition is obtained through three times of partitioning, as shown in Fig. 4e. For example, after the first partitioning of the whole space, the partitions are numbered as $\phi_0, \phi_1, \phi_2, \phi_3$; after the second partitioning, the partitions are numbered as $\phi_{00}, \phi_{01}, \phi_{02}, \phi_{03}$ and so on. In order to convert the road network space into the one-dimensional space, we provide each numbered partition with a triple tag (Partition ID, Start, End) to indicate related information of this partition, in which the Partition ID refers to the

partition number after one partitioning which a certain partition belongs to. For example, the seven partitions such as ϕ_{02}, ϕ_{013} , and ϕ_{00} belong to partition ϕ_0 after one partitioning, and we denote it as 0; the 10 partitions such as ϕ_{13}, ϕ_{111} , and ϕ_{100} belong to partition ϕ_1 after one partitioning. “Start” represents the order of a partition in the same Partition ID, while “End” is expressed as “Start+1.” In the one-dimensional space, initialize the “Start” of first partition to 0, and the “Start” of first partition equals to the “End” value of last partition. For example, the ϕ_{00} partition can be represented as a triple (0, 0, 1), and the ϕ_{010} partition can be expressed as (0, 1, 2). According to this sorting method, we can convert the two-dimensional space to one-dimensional space, as shown in Fig. 5a.

Figure 5a shows the one-dimensional order of road network space. However, by combining Fig. 3 and Fig. 5a, we find that many partitions actually contain very few nodes and objects, there are natural “mutual attraction” and “intention” for merging between these partitions, and they are actually passively divided. Therefore, in order to integrate the partitions, we propose the concept of partition merging.

Define 6 Self-attraction of partition Assume ϕ_a is a partition in road network G ; ϕ_{a-in} represents the number of connecting lines within ϕ_a ; ϕ_{a-out} represents the number of connecting lines between ϕ_a and other partitions. Then, the self-attraction of ϕ_a partition can be defined as:

$$\mathfrak{R}_a = \frac{(\phi_{a-in} - \phi_{a-out})}{\phi_{a-in}} \tag{7}$$

Define 7 Mutual attraction between partitions Assume ϕ_a and ϕ_b are two adjacent partitions in road

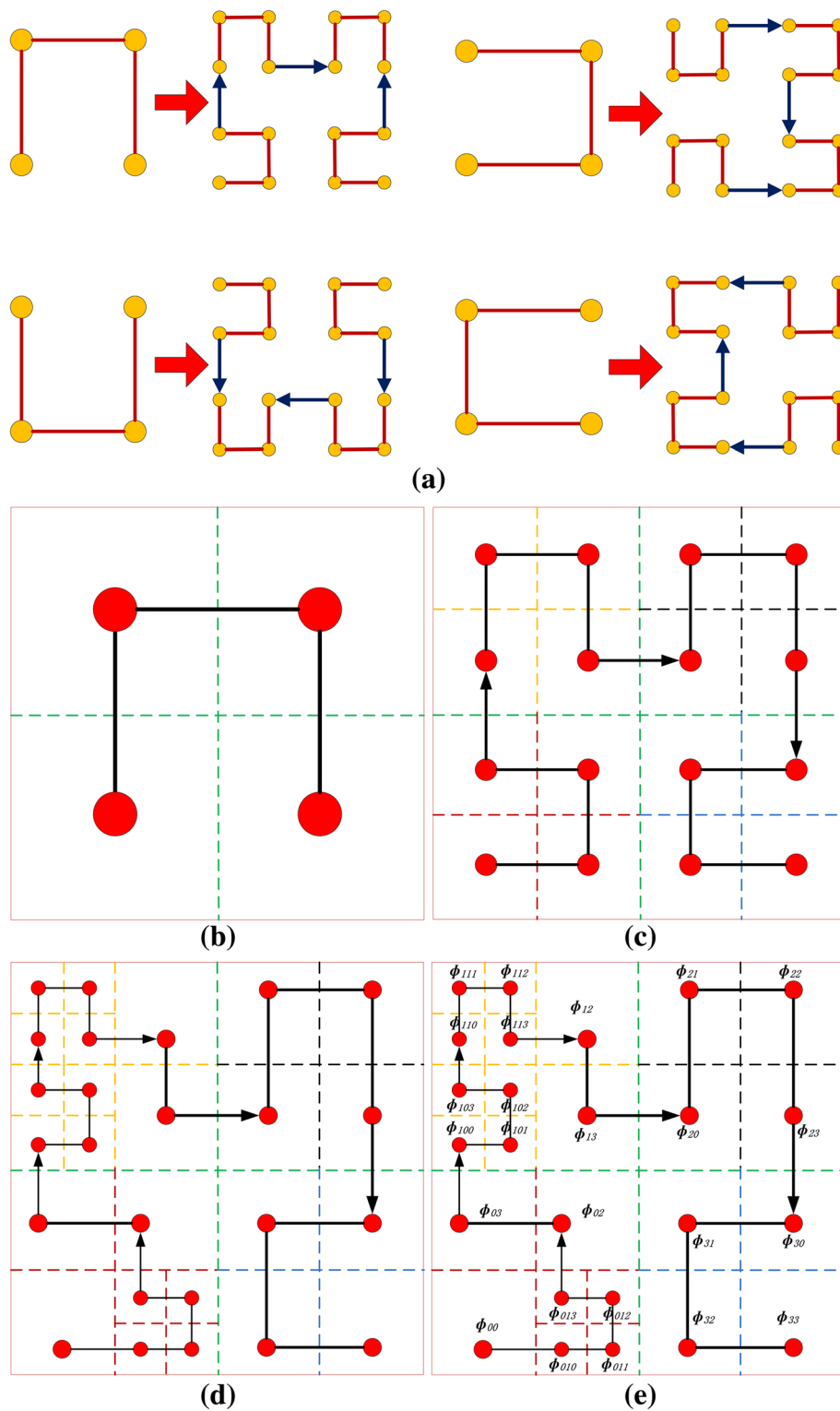
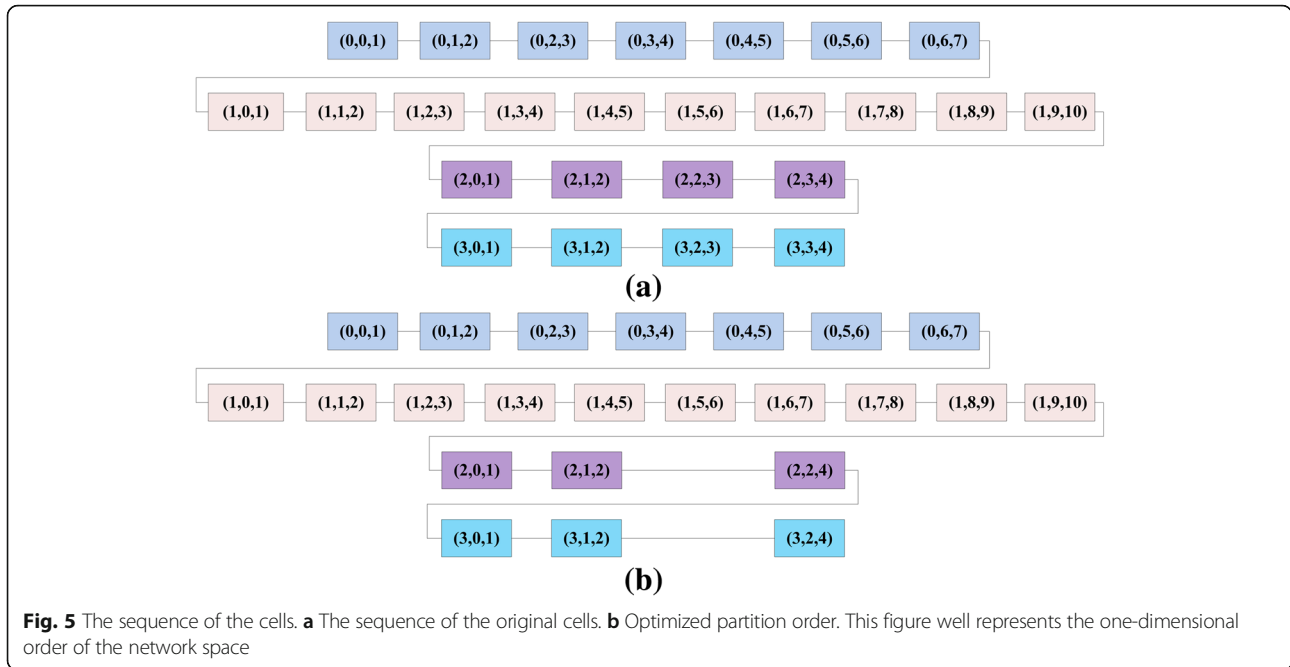


Fig. 4 Hilbert curve and number. **a** Four basic subdivision modes of Hilbert curve. **b** The first level of Hilbert curve. **c** The second level of Hilbert curve. **d** The third level of Hilbert curve. **e** Number for each grid cell. **a** Each subdivision mode includes two basic parts: cups and joins. A cup is a square with one open side, and a join is a vector that joins two cups. **b** Each grid is represented by a red dot, and the dots are connected by lines to form a cup with bottom opening, which is the first layer of the Hilbert curve. **c** According to the Eq. (6), the partitions exceeding the density threshold are divided twice. The final Hilbert curve result and a road network data broadcast cycle are shown in **d**. According to the direction of Hilbert curve and division level, we numbered each grid cell. If a subscript value consists of three bits, that means the cell is acquired by being divided by three times, as shown in **e**



network G after partitioning; ϕ_{a-in} represents the number of connecting lines within ϕ_a ; ϕ_{ab} represents the number of connecting lines between partitions ϕ_a and ϕ_b . Then, the mutual attraction between partitions ϕ_a and ϕ_b can be defined as:

$$\mathfrak{R}_{ab} = \frac{(\phi_{a-in} - \phi_{ab})}{\phi_{a-in}} \quad (8)$$

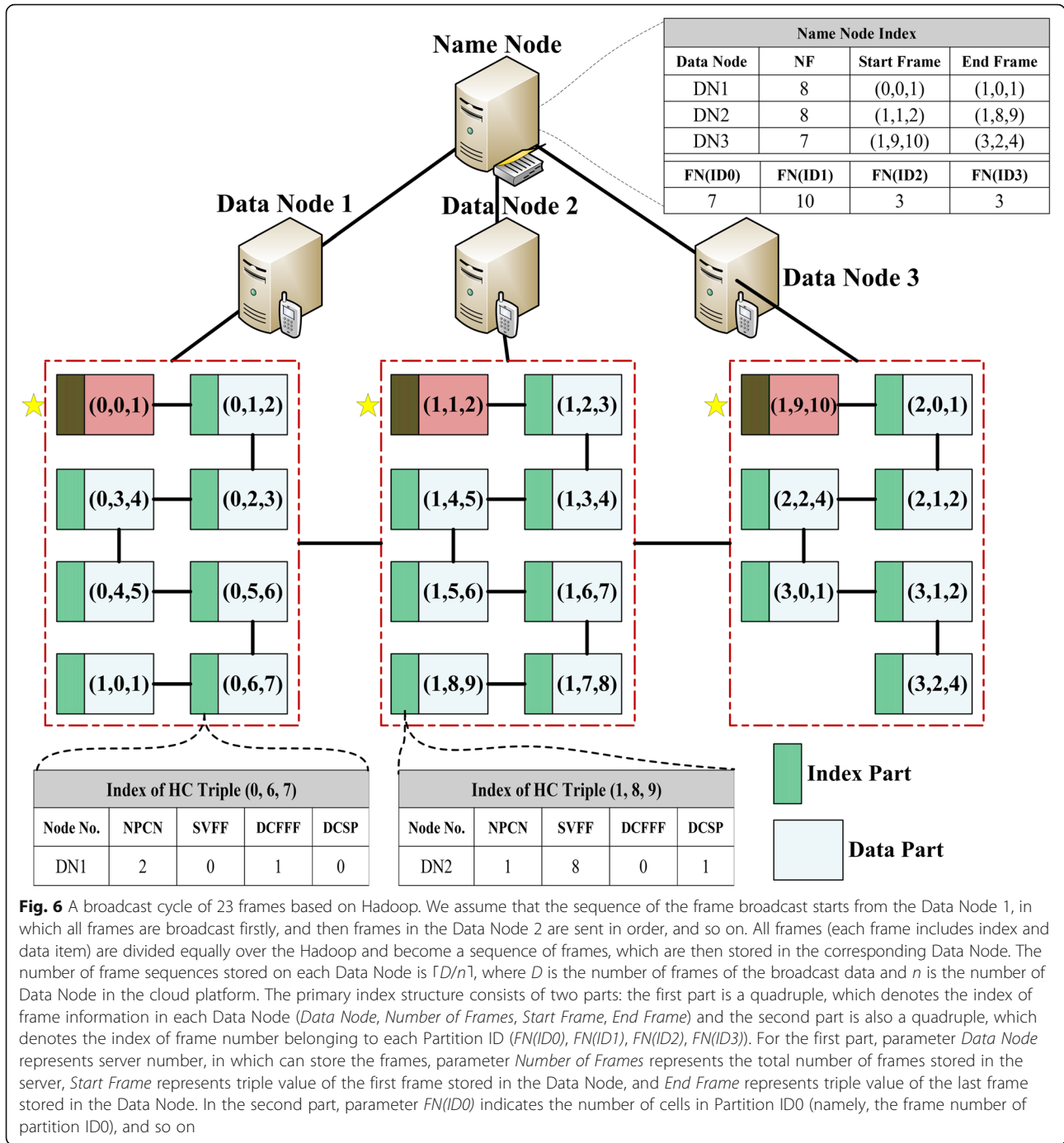
Define 8 *Partition merging factor* If the mutual attraction between two adjacent partitions is higher than a certain threshold value σ , then the two partitions can be merged into a new partition; otherwise, do not merge them. Then, we call this value σ the partition merging factor.

According to Definitions 6–8, under the precondition of merging threshold value σ , we merge the partitions with sparse nodes and objects. We merge the adjacent $_{22}$ and $_{23}$ as well as $_{32}$ and $_{33}$, i.e., (2, 2, 3) and (2, 3, 4) merged into (2, 2, 4), and (3, 2, 3, 3, 3, 4) are merged into (3, 2, 4). The final one-dimensional sorting result of partitioning is as shown in Fig. 5b.

4.2 Designing the air index for wireless broadcast

In this section, we illustrate the designing idea of DIM index based on Hadoop. All frames (indexes and data) are broadcast over the cloud platform. We assume that the sequence of the frame broadcast starts from the Data Node 1, in which all frames are broadcast firstly, and

then frames in the Data Node 2 are sent in order, and so on. All frames (each frame includes index and data item) are divided equally over the Hadoop and become a sequence of frames, which are then stored in the corresponding Data Node. The number of frame sequences stored on each Data Node is $\lceil D/n \rceil$, in which D refers to the frames of broadcast data and n is the number of Data Nodes on the cloud platform. For the 23 frames as shown in Figs. 5b, 8, and 7, frames are put into the Data Node of Fig. 6 respectively. The Name Node does not store any data but only contains a prime index (Name Node Index). The structure of this prime index mainly consists of two parts: the first part is about the index that stores the frame information in each Data Node (Data Node, Number of Frames, Start Frame, End Frame) and the second part is about the index of frame number contained in each Partition ID (FN(ID0), FN(ID1), FN(ID2), FN(ID3)), in which the “Data Node” represents the number of server that stores frames, the “Number of Frames” represents the total number of frames stored in the server, the “Start Frame” represents the triple group value of first frame in this Data Node, the “End Frame” refers to the triple group value of the last frame stored in this Data Node, the “FN(ID0)” represents the partition number in Partition ID0 (namely, the frame number of partition ID0), and so on. In the Data Node, the frames in each Data Node are in linear order series, and the frame beside the five-pointed star represents the first frame. In the



Data Node, the index part is represented in the form of (Node No., NPCN, SVFF, DCFFF, DCSP), and the five parameters have the following definitions:

- Node No.*: the number of Data Node where this frame is located;
- NPCN*: number of Partition ID in current Data Node;
- SVFF*: start value of final frame in Data Node;

- DCFFF*: distance from current frame to final frame in current Data Node;
- DCSP*: distance from current frame to final frame of the same Partition ID.

The idea of DIM index is as shown in Fig. 6. In order to better illustrate the concept of DIM index, we will provide an example. In the first case, assume we

need to query the 16th frame (1, 8, 9). First of all, we search for the prime index in Name Node, obtain the 16th frame, and store it in DN2, so the mobile client switches to the doze mode until the broadcast data is sent to the 8th frame in the frame sequence of DN2 to start tuning in. In the second case, assume the current frame is the 5th frame in DN1 (0, 4, 5), and we need to query the 16th frame (1, 8, 9). Then, by querying the index of (0, 4, 5) frame, we obtain the quintuple group parameters: the Node No. is DN1, the NPCN is 2, the SVFF is 0, the DCFPP is 3, and the DCSP is 2. Then, we can find out that the frame that needs to be queried can only be obtained after 11 frames since the current frame, and then the mobile client will enter the doze mode until the frame of (1, 8, 9). In this distributed index structure, we do not need to frequently switch between multiple frames, the query performance is improved, and it can also provide minimum access latency and tuning time. In this example, in the first case, we can immediately obtain the sequence of target frame through the query index, so the tuning time is 1 frame (not including the query prime index), and the access time is 16 frames; in the second case, the tuning time is 2 frames, and the access time is 12 frames.

5 Client-site query processing

When the querying client issues a k NN query, it will tune to the broadcasting channel to receive the index information and selectively receive the data information required by query processing under the guidance of index information, in this way to reduce the time and energy consumption required by query processing. Finally, the client independently conducts query processing according to the obtained information and obtains the query result. The k NN query can be regarded as (q, k, S) , in which q is the given query location, S is a dataset in the road network, and a k NN query refers to real-time search in S and return of k target points with minimum network distances from k . Here, the target can be a store, gas station, or restaurant that the user needs to query. Our k NN spatial query processing method has very efficient procedure. First of all, according to the proposed DIM index method, the map() function evenly divides all broadcast cycle data and stores them in corresponding Data Nodes. Each frame of data is represented in the triple group form of triple_group. In the reduce() function, the prime index and each frame of index are used to determine the location of queried frame. Then, after obtaining the edge of related partition, object on the edge and fixed-point information, "list" neighboring nodes closest to the query point q are obtained, and if $k < \text{list}$, the query algorithm will return k objects with smallest network distances.

Algorithm : k NN spatial query processing

Input: a query point q , number of queried object k , a dataset S

Output: k objects in S closest to q

Procedure:

```

1: HC_triple= $\emptyset$ , count=0,  $j=1$ ,  $k$ NN_list= $\emptyset$ , result= $\emptyset$ ;
2: locate the grid cell triple_group[ $i$ ]  $q$  lies;
3: insert triple_group $_q$  into HC_triple;
4: Execute(MapReduce)
5:   create main_index as (Data_No, Frame_num, Start_frame);
6:   partition the number of grid cells as  $D/n$ ;
7:   label each partition as (Partition ID $_i$ , Start $_i$ , End $_i$ ) of triple_group[ $i$ ];
8:    $j=1$ ;
9:   (1) $q$ th frame to be queried
10:  switch( $q$ ):
11:     $q \leq (0+D/n)$ : the queried frame is located in Data_No1; break;
12:     $q \leq 2*D/n$ : the queried frame is located in Data_No2; break;
13:    ...
14:    return to doze mode and wait for the  $q$ th frame following;
15:  (2)the frame (Partition ID $_q$ , Start $_q$ , End $_q$ ) to be queried
16:    if ID $_q$ =ID $_{current}$  then
17:      count= Start $_q$ - Start $_{current}$ ;
18:    else if ID $_q$ =ID $_{current}+1$ ;
19:      a=Start $_{final\_same\_ID}$ -Start $_{current}$ ;
20:      b= Start $_{final\_current\_node}$ - Start $_{current}$ ;
21:      z=b-a;
22:    if ID $_q$ =ID $_{final\_current\_node}$  then
23:      c= Start $_q$ - Start $_{final\_current\_node}$ ;
24:      count=a+c;
25:    return to doze mode and wait for the following (count)th frame;
26:    .....
27: locate the queried cell and obtain  $e.start.k$ NN_list,  $e.end.k$ NN_list and  $e.object\_list$ ;
28: End Execute(MapReduce)
29: while result $\leq k$  do
30:   for  $o_i$  in  $e.start.k$ NN_list do
31:     dist( $o_i, q$ )=dist( $o_i, e.start$ )+dist( $e.start, q$ );
32:     if dist( $o_i, q$ ) be satisfied condition then
33:       result=result+{ $o_i$ };
34:     else break;
35:   for  $o_i$  in  $e.end.k$ NN_list do
36:     dist( $o_i, q$ )=dist( $o_i, e.end$ )+dist( $e.end, q$ );
37:     if dist( $o_i, q$ ) be satisfied condition then
38:       result=result+{ $o_i$ };
39:     else break;
40: return result;

```

In order to increase the readability of pseudocode, we introduce several important steps in detail. In the 9th to 15th lines of code, the query process starts from the prime index of broadcast cycle, and by determining the frames that need to be queried, the sequence of Data Node can be directly determined in the prime index list. The 16th and 25th lines of code introduce the process to

start search from a random frame, in which $Start_q$ refers to the start value of query point, $Start_{current}$ represents the start value of the currently searched frame, and $Start_{final_same_ID}$ refers to the start value of the last frame with the same partition ID of the current frame. The 29th to 40th lines describe the process in which after the user obtains the frame of query point, it will query the k nearest neighboring objects in this frame.

6 Performance evaluation

In order to better reflect the superiority of the method proposed by us, this paper compares the DIM index method with the IEI index method [5], NPI index method [2], and ISW index method [16], and these three index methods have the following characteristics:

IEI Index which uses Hilbert curve (HC) to partition a road sensor network into numbered grid cells and integrates the information of objects and the road sensor network into the design of the index.

NPI Index which broadcasts grid partition information and some precomputed distance information in its index. As its index size is determined by the number of grid cells, the author had tested the performance of NPI with 16, 64, and 256 grid cells, denoted as NPI-16, NPI-64, and NPI-256. Because NPI-64 generally showed the best performance among different partitions, we choose NPI-64 as our competitor.

ISW Index which 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.

6.1 Experimental setup

For our experiments, we build a cloud platform with four computers, installed JDK1.8.0, Spark-1.5.2; the specific configuration information is shown in (Table 1):

On the cloud platform consisting of four computers, one computer is used as the Name Node, and the other three are used as Data Nodes to save all frames (including the index part and data part).

The simulation environment of our experiment consists of the base station, query user group, and a broadcast channel, in which the bandwidth of the broadcast channel is set at 2 Mbps. The dataset used in the

experiment consists of four actual city road networks, as shown below:

The City of Oldenburg Road Network (OL), which contains 6105 nodes and 7035 edges.

The California Road Network (CAL) [36], which contains 21,048 nodes and 21,693 edges.

The San Rafael Road Network (SR), which contains 16,130 nodes and 20,178 edges.

The Florida Road Network (FL), which contains 1,068,615 nodes and 1,357,204 edges.

In k NN, k takes 1, 5, 10, 15, and 20. We adopt both the access latency and tuning time as the main performance metrics. For each road network, a set of objects are randomly generated and uniformly distributed over the network. The number of objects is set as follows: 1k, 4k, 7k, 10k, and 15k for OL dataset; 5k, 10k, 20k, 30k, and 40k for SR dataset and CAL dataset; and 50k, 100k, 200k, 300k, and 400k for FL dataset. For each dataset, we operate ten times and then obtain the average.

6.2 Experimental results

In the experiment section, we evaluate the performance of k NN query processing method based on the DIM index, including the access latency, tuning time, and $OD(object)$.

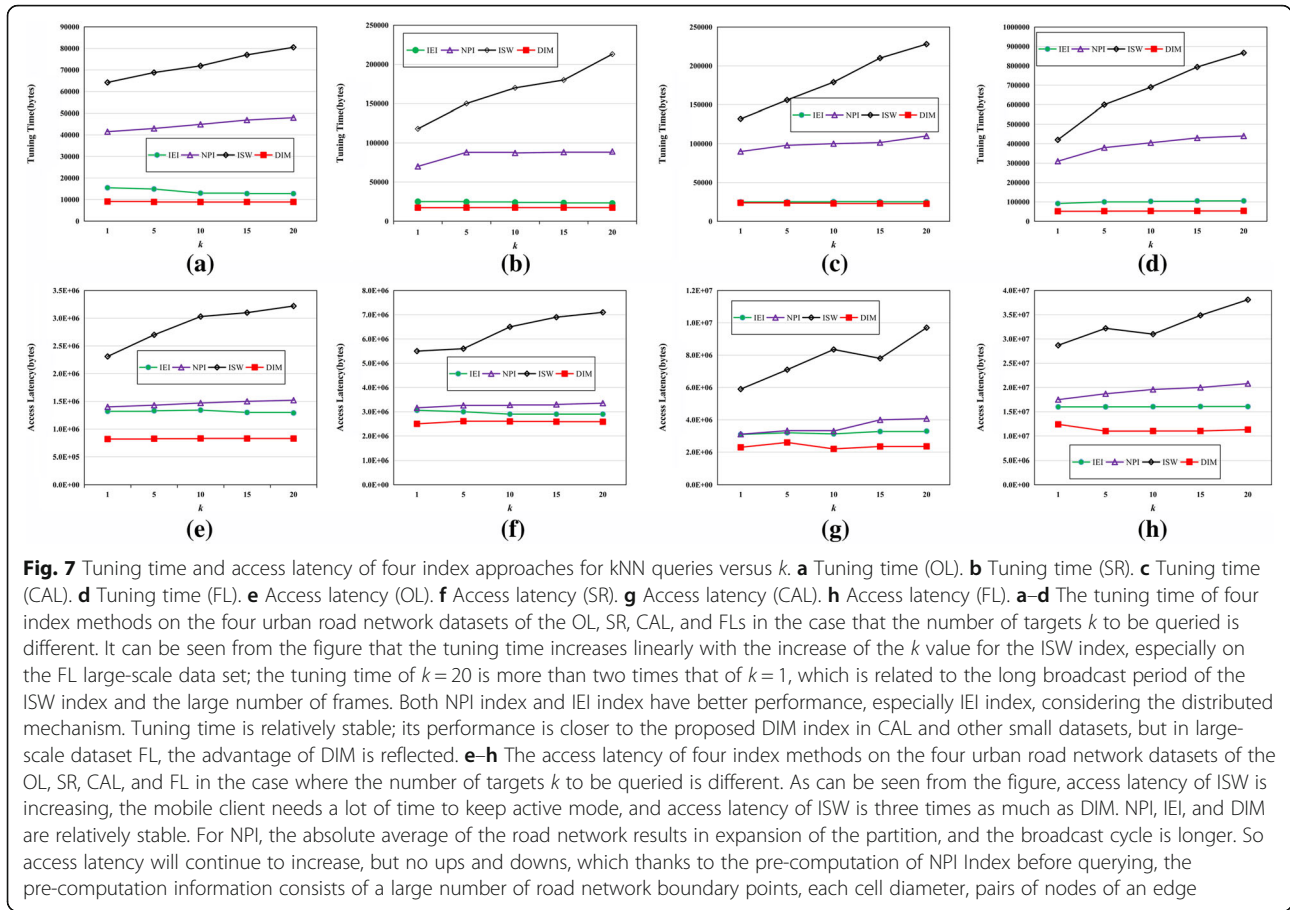
First, we evaluate the performance of four index approaches for access latency and tuning time under various k . The results are shown in Fig. 7.

Figure 7a–d describes the tuning time of four index methods in the four city road network datasets of OL, SR, CAL, and FL under different number k of targets that need to be queried. According to Fig. 7a–d, we can see that with the increase of k value, the tuning time of ISW index method also rises linearly, especially in the large-scale dataset of FL; the tuning time when $k = 20$ is more than twice the tuning time when $k = 1$, and this is due to the long broadcast cycle and the large number of frames in ISW index method. The NPI and IEI index methods have great performance, especially the IEI index method. Considering the distributed mechanism, the tuning time is stable, and especially for small dataset such as CAL, its performance is closer to that of the DIM index method proposed in this paper. However, for large dataset such as FL, the advantages of DIM can be reflected, because the IEI method requires frequent switches between frames to obtain the query result. On the FL dataset, the tuning time of DIM method is approximately 35% shorter than that of the IEI method.

Figure 7e–h describes the access latency of four index methods in the four city road network datasets of OL, SR, CAL, and FL under different number k of objects that need to be queried. According to these figures, we

Table 1 Cloud platform configuration

Item	Personal computer
Memory (GB)	8
Hard disk (GB)	512
Processor (GHz)	Intel Core(TM) i7-5500U 2.4GHz
Core number	4
OS	CenOS6.6



can see that the access latency of ISW method continuously increases, the mobile user needs to keep an active mode for a long time, and the access latency of ISW is three times of that of DIM. The NPI, IEI, and DIM methods have stable access latency. For NPI, absolute even division of road network has resulted in partition expansion and extension of broadcast cycle, so the access latency will continuously increase, but without significant fluctuation, because in the NPI index method, massive information regarding the boundary points, partition diameter, and edge terminal nodes of road network are obtained through pre-computation before the query starts, while in the IEI index method, the data of various frames after current frame are obtained through pre-computation, and the user can quickly obtain the query result.

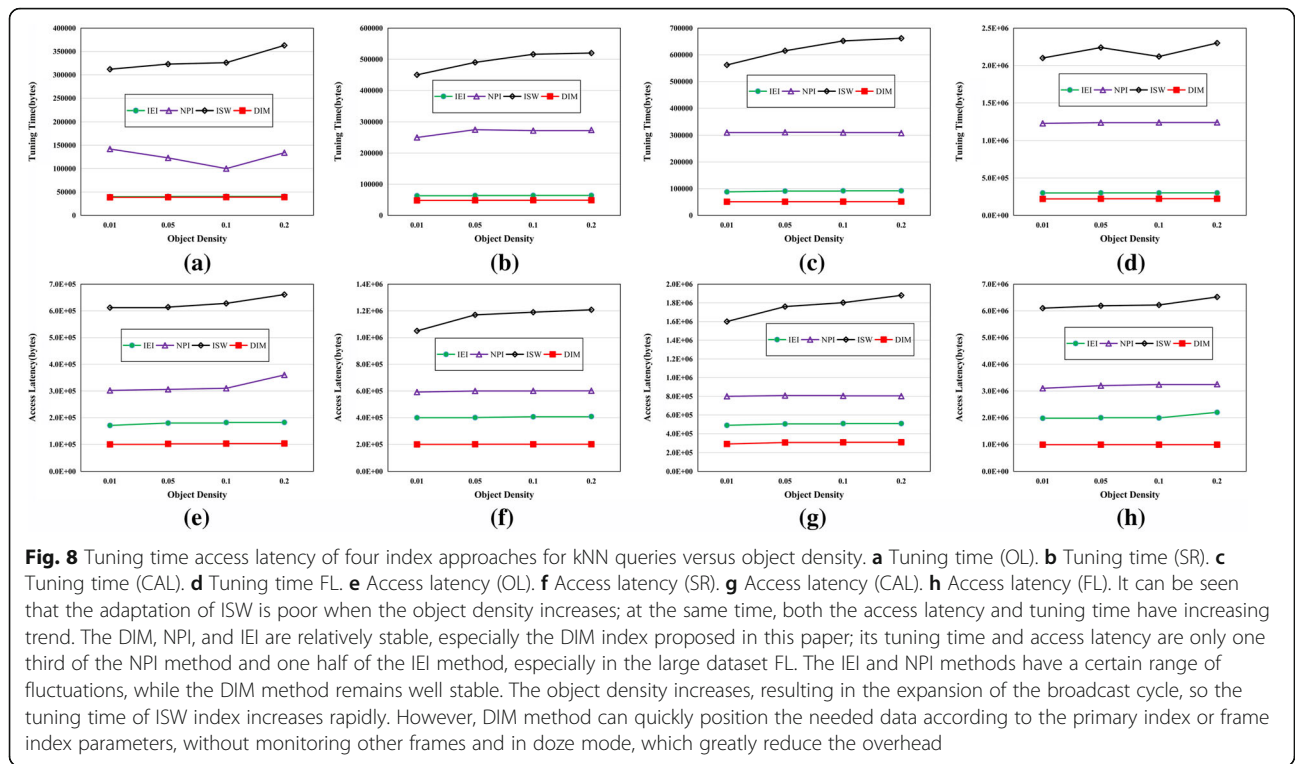
Secondly, under different object densities, we evaluate the performance of four index methods on four road network datasets, as shown in Fig. 8.

According to Fig. 8a–h, we can see that with the increase of target density, ISW has poor fitness and the access latency and tuning time present the trend of increase. The DIM, NPI, and IEI have stable performance, especially the DIM index method proposed in this

paper, and its tuning time and access latency are only 1/3 of that of NPI method and 1/2 of that of IEI method. Especially on the large dataset FL, both the IEI and NPI methods present certain fluctuations, while the DIM method maintains great stability. The increase of object density has resulted in the expansion of broadcast cycle, so the tuning time of ISW index method would increase rapidly. Our DIM method can quickly localize the data that needs to be queried based on the prime index or the parameter in frame index; it does not need to monitor other frames, and these frames are in the doze mode, which can significantly reduce the overhead.

7 Conclusions

With the popularization of intelligent mobile devices and rapid increase in demand to obtain information everywhere, researches on wireless data services have become a focus of common concern in the industrial and academic fields. According to the problems existing in current road network partitioning methods, this paper proposes the partition density formula and further proposes the optimized road network partitioning method based on the partition density. This method calculates the density of each partition and determines whether



secondary partitioning is required according to the density value, which has addressed the defect of big broadcast cycle caused by simple even partitioning. In the meantime, we propose the concepts of partition self-attraction, mutual attraction, and merger factor of partition to effectively merge and optimize sparse partitions, which can reduce energy consumption and monitoring overhead. Based on that, we propose the distributed air index method DIM and kNN spatial query algorithm and verify the effectiveness and stability of the method through many experiments, which can provide certain reference value to researches on the distributed air index technology.

Abbreviations

BGI: Broadcast Grid Index; DCFF: Distance from current frame to final frame in current Data Node; DCSP: Distance from current frame to final frame of the same Partition ID; DIM: Distributed air index based on MapReduce; DSI: Distributed spatial index; GDIN: Grid-based distributed index for non-flat broadcast; LBSS: Location-based services; MaxBR: Maximum boundary rectangle; kNN: *k*-nearest neighbor; MCNNQ: Mobile Continuous Nearest Neighbor Query; NPCN: Number of Partition ID in current Data Node; NPI: Network Partition Index; NVD: Network Voronoi diagram; NVD-DI: NVD-based distributed air index; SVFF: Start value of final frame in Data Node

Acknowledgments

The authors would like to thank the editor and the anonymous reviewers for their helpful comments and suggestions in improving the quality of this paper. This work was supported by the Postdoctoral Research Project of Zhejiang Province, by the National Natural Science Foundation of China under Grant No. 61472348 and 61672455, by the Humanities and Social Science Fund of the Ministry of Education of China under Grant No. 17YJCZH076, by Zhejiang Science and Technology Project under Grant No.

LGF18F020001, and by the Ningbo Natural Science Foundation under Grant No.2017A610111.

Funding

This work was supported by the Postdoctoral Research Project of Zhejiang Province, by the National Natural Science Foundation of China under Grant No. 61472348 and 61672455, by the Humanities and Social Science Fund of the Ministry of Education of China under Grant No. 17YJCZH076, by Zhejiang Science and Technology Project under Grant No. LGF18F020001, and by the Ningbo Natural Science Foundation under Grant No.2017A610111.

Availability of data and materials

The datasets supporting the conclusions of this article are included within the article.

Authors' contributions

RJ and GC contributed to the conception and algorithm design of the study. LS and AT took charge of the implementation of the experiment. BCO and YG verified all the results. RJ drafted and revised the manuscript. All authors read and approved the final manuscript.

Authors' information

Ran Jin received the PhD degree from Donghua University, China, in 2015. He is currently a postdoctoral fellow in the College of Computer Science and Technology, Zhejiang University, China. Gang Chen received the BSc, MSc, and PhD degrees in computer science and engineering from Zhejiang University in 1993, 1995, and 1998, respectively. He is currently a professor at the College of Computer Science, Zhejiang University. Anthony K.H. Tung received the BSc (second class honor) and MSc degrees in computer science from the National University of Singapore (NUS), in 1997 and 1998, respectively, and the PhD degree in computer science from Simon Fraser University, in 2001. He is currently an associate professor in the Department of Computer Science, NUS. Lidan Shou received the PhD degree in computer science from the National University of Singapore. He is a professor in the College of Computer Science, Zhejiang University, China. Prior to joining the faculty, he worked in the software industry for more than 2 years.

Beng Chin Ooi is currently a distinguished professor of computer science at the National University of Singapore. His research interests include database system architectures, performance issues, indexing techniques, and query processing, in the context of multimedia, spatiotemporal, distributed, parallel, peer-to-peer, in-memory, and cloud database systems. He has served as a PC member for a number of international conferences (including SIGMOD, VLDB, ICDE, WWW, EDBT, DASFAA, GIS, KDD, CIKM, and SSD). He was an editor of the VLDB Journal and the IEEE Transactions on Knowledge and Data Engineering, editor-in-chief of the IEEE Transactions on Knowledge and Data Engineering (2009–2012), and a co-chair of the ACM SIGMOD Jim Gray Best Thesis Award committee. He is serving as a trustee board member and the president of the VLDB Endowment. He is a fellow of the IEEE and ACM.

Competing interests

The authors declare that they have no competing interests.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Author details

¹College of Computer Science and Technology, Zhejiang University, No.38, Zheda Road, Hangzhou, China. ²College of Electronics and Computer Science, Zhejiang Wanli University, No.8, South Qianhu Road, Ningbo, China. ³School of Computing, National University of Singapore, 21 Lower Kent Ridge Road, Singapore, Singapore.

Received: 8 March 2018 Accepted: 7 November 2018

Published online: 05 December 2018

References

1. K Park, P Valduriez, A hierarchical grid index (HGI), spatial queries in wireless data broadcasting. *Distributed Parallel Database*. 31(3), 413–446(2013)
2. S. Weiwei, C. Chunan, Z. Baihua, et al., An air index for spatial query processing in road networks. *IEEE Transactions on Knowledge and Data Engineering* 27(2), 382–395 (2015)
3. Lee Wangchien, Zheng Baihua: DSI: a fully distributed spatial index for location-based wireless broadcast services, *International Conference Distributed Computing Systems*, (Columbus, USA), 2005, pp. 349–358
4. B. Zheng, W.-C. Lee, K.C. Lee, et al., A distributed spatial index for error-prone wireless data broadcast. *The VLDB Journal* 18(4), 959–986 (2009)
5. L. Yanhong, S. LihChyun, R. Zhu, et al., A novel distributed air index for efficient spatial query processing in road sensor networks on the air. *International Journal of Communication System* 30(5), 1–23 (2017)
6. J. Xu, X. Tang, W.C. Lee, Time-critical on-demand data broadcast: algorithms, analysis, and performance evaluation. *IEEE Transactions on Parallel and Distributed Systems* 17(1), 3–14 (2006)
7. A. Datta, D. VanderMeer, A. Celik, V. Kumar, Broadcast protocols to support efficient retrieval from databases by mobile users. *ACM Transactions on Database Systems* 24(1), 1–79 (1999)
8. B. Zheng, W.C. Lee, D.L. Lee, Spatial queries in wireless broadcast systems. *Wireless Networks* 10(6), 723–736 (2004)
9. T. Imielinski, S. Viswanathan, B.R. Bardrinath, Data on air: organization and access. *IEEE Transactions on Knowledge and Data Engineering* 9(3), 353–372 (1997)
10. C.J. Su, L. Tassiulas, Broadcast scheduling for information distribution. *Wireless Networks* 5(2), 137–147 (1997)
11. A. Tarique, L. Chengfei, L. Vu Hai, et al., Partitioning road networks using density peak graphs: efficiency vs. accuracy. *Information Systems* 64(C), 22–40 (2016)
12. Gedik B, Singh A, Liu L. Energy Efficient Exact kNN Search in Wireless Broadcast Environments, the 12th Annual ACM International Workshop on Geographic Information Systems, (New York, USA, 2004), pp. 137–146
13. K. Mouratidis, S. Bakiras, D. Papadias, Continuous monitoring of spatial queries in wireless broadcast environments. *IEEE Transactions on Mobile Computing* 8(10), 1297–1311 (2009)
14. K. Park, H. Choo, P. Valduriez, A scalable energy efficient continuous nearest neighbor search in wireless broadcast systems. *Wireless Networks* 16(4), 1011–1031 (2010)
15. Xu J., Zheng B., Lee W., et al. Energy efficient index for querying location-dependent data in mobile broadcast environments. *International Conference on Data Engineering*, (Bangalore, India), 2003, pp. 239–250
16. B. Zheng, J. Xu, W. Lee, et al., Grid-partition index: a hybrid method for nearest-neighbor queries in wireless location-based services. *Vldb Journal* 15(1), 21–39 (2006)
17. Chow C, Mokbel M F, Naps J, Nath S. Approximate evaluation of range nearest neighbor queries with quality guarantee, *International Symposium on Advances in Spatial and Temporal Databases*, (Aalborg, Denmark), 5644, 283–301 (2009)
18. P. Kalnis, G. Ghinita, K. Mouratidis, D. Papadias, Preventing location-based identity inference in anonymous spatial queries. *IEEE Transactions on Knowledge and Data Engineering* 19(12), 1719–1733 (2007)
19. J. Um, Y. Kim, H. Lee, M. Jang, J. Chang, k-nearest neighbor query processing algorithm for cloaking regions towards user privacy protection in location-based services. *Journal of Systems Architecture* 58(9), 354–371 (2012)
20. K. Park, H. Choo, Energy-efficient data dissemination schemes for nearest neighbor query processing. *IEEE Transactions on Computers* 56(6), 754–768 (2007)
21. C.M. Liu, S.Y. Fu, Effective protocols for kNN search on broadcast multi-dimensional index trees. *Information Systems* 33(1), 18–35 (2008)
22. W. Yanqiu, X. Chuanfei, G. Yu, Spatial query processing in road networks for wireless data broadcast. *Wireless Network* 19(4), 477–494 (2013)
23. Jing Y., Chen C., Sun W., et al. Energy-efficient shortest path query processing on air, *International Conference on Advances Geographic Information Systems*, (Chicago, USA), 2011, pp. 393–396
24. G. Kellaris, K. Mouratidis, Shortest path computation on air indexes. *Proceedings of the VLDB Endowment* 3(3), 747–757 (2010)
25. Y. Li, J. Li, L. Shu, et al., Searching continuous nearest neighbors in road networks on the air. *Information Systems* 42(3), 177–194 (2014)
26. H. Wang, R. Zimmermann, Processing of continuous location-based range queries on moving objects in road networks. *IEEE Transactions Knowledge & Data Engineering* 23(7), 1065–1078 (2011)
27. Pushpam PMM, Felix Enigo VS. Energy-efficient and fault tolerant spatial query processing in wireless sensor networks, *International Conference on Advanced Communication Control and Computing Technologies*, (Ramanathapuram, India), 2014, pp. 790–794
28. C. Zhu, L.T. Yang, L. Shu, et al., Insights of top-query in duty-cycled wireless sensor networks. *IEEE Transactions on Industrial Electronics* 62(2), 1317–1328 (2015)
29. S. Cheng, Z. Cai, J. Li, Curve query processing in wireless sensor networks. *IEEE Transactions on Vehicular Technology* 64(11), 5198–5209 (2015)
30. O. Diallo, J.J.P.C. Rodrigues, M. Sene, X. Feng, Real-time query processing optimization for wireless sensor networks. *International Journal of Sensor Networks* 18(1/2), 49–61 (2015)
31. S. Im, J.T. Choi, A distributed air index based on maximum boundary rectangle over grid-cells for wireless non-flat spatial data broadcast. *Sensors* 14(6), 10619–10643 (2014)
32. S. Im, H. Youn, J. Choi, et al., A novel air indexing scheme for window query in non-flat wireless spatial data broadcast. *Journal of Communications & Networks* 13(4), 400–407 (2011)
33. Im SJ, Song MB, Kim J. Cell-based distributed index for range query processing in wireless data broadcast systems, *International file Conference on Knowledge-based Intelligent Information & Engineering Systems*, (Bournemouth, UK), 4251, 1139–1146 (2006)
34. Im SJ, Song MB, Kim J, et al. An Error-resilient cell-based distributed index for location-based wireless broadcast services, *International Workshop on Data Engineering for Wireless and Mobile Access*, (Chicago, USA), 2006, pp. 59–66
35. Xu J., Lee W.C., Tang X. Exponential index: a parameterized distributed indexing scheme for data on air, *International Conference On Mobile System, Applications, and Service*, (Boston, USA), 2004, pp. 153–164
36. F. Li, D. Cheng, M. Hadjieleftheriou, et al. On trip planning queries in spatial databases, *International Conference on Advances in Spatial and Temporal Databases*, (Angra dos Reis, Brazil), 2005, pp. 923–923