LETTER

Indexing of Continuously Moving Objects on Road Networks

Kyoung Soo BOK[†], **Ho Won YOON**[††], **Dong Min SEO**[†††], **Myoung Ho KIM**[†], ***Nonmembers***, ***and*** **Jae Soo YOO**[†††a)], ***Member***

SUMMARY In this paper, a new access method is proposed for current positions of moving objects on road networks in order to efficiently update their positions. In the existing index structures, the connectivity of edges is lost because the intersection points in which three or more edges are split. The proposed index structure preserves the network connectivity, which uses intersection oriented network model by not splitting intersection nodes that three or more edges meet for preserving the connectivity of adjacent road segments. The data node stores not only the positions of moving object but also the connectivity of networks.

key words: *road network, connectivity, moving object, index structure*

1. Introduction

Location Based Service (LBS) rely on the tracking of the continuously changing positions of entire populations of service users, termed moving objects. This environment is characterized by large volumes of updates, for which reason moving objects with frequent updates on their positions require fast update mechanisms in spatiotemporal database systems. Existing index structures for moving objects can be divided into two categories: location-based index structures focusing on current/anticipated future locations of moving objects [1], [2] and trajectory-based ones for historical trajectories [3], [4].

In real world applications, movements of moving objects are constrained. That is, objects are moving in a constrained network space such as road networks. Recently, several index structures improve their performance by exploiting the properties of network [5]–[10]. The Fixed Network R-tree (FNR-tree) separates spatial and temporal components of the trajectories and indexes the time intervals that each moving object spends on a given network link [6]. The Moving Object in Networks tree (MON-tree) further improves the performance of the FNR-tree by representing each edge by multiple line segments instead of just one line segment [7].

In this paper, we focus on the current positions of moving objects on road networks. Indexing Moving Objects on Road Sectors (IMORS) indexes the poly lines of the road network into a spatial index structure for processing current positions of objects [8]. Then each object is associated with a poly line. PMR Quadtree, Quadtree and R*-tree (PQR-tree) proposes integrated tree for current and near future positions of moving objects in networks [10]. In PQR-tree, quisi-static objects are indexed to R*-tree and fast moving objects are related to PMR Quadtree. The disadvantage of the approaches doesn't support network connectivity information. Therefore, each object going out of the corresponding segment retrieves the index from the root to find out its next segment. This causes degradation of update performance as the number of objects increases.

In this paper, we propose a new index structure called IONR-tree, which is a new access method for efficiently updating current positions of moving objects on road network. The proposed indexing method preserves network connectivity by not splitting intersecting nodes that three or more edges meet always. Data node additionally stores adjacency list that represents the connectivity information.

The rest of the paper is organized as follows. Section 2 discusses the problem of related works. Section 3 presents a new index structure for moving object on road network. Section 4 contains experimental evaluation. Section 5 describes the conclusions and directions for future research.

2. Motivation

The existing index structures have a problem when a moving object traverses from one segment to adjacent connected segment. To find out the connected segment, the network R-tree (2DR-tree) must be searched from the root node [8], [9]. In [9], adjacency component that captures the network connectivity can be used to find out next lines or poly lines for escaping whole 2DR-tree searches. Using adjacency component can improve the query performance but is not a fundamental solution for updating the positions of moving objects. This can lead to serious update performance degradation as the number of objects increases.

Our indexing method aims to provide an efficient update of the positions of moving objects on road network. IMORS is proposed to improve update costs of moving objects on road network [8]. IMORS consists of two parts to reduce the part of data structure changing upon update requests. Since the information of the geometry and connectivity of road segments rarely changes, static part contains R*-tree indexing road networks. The leaf node of static part stores a set of road segments. The dynamic part stores

Manuscript received August 17, 2007.
Manuscript revised December 12, 2007.
[†]The authors are with Korea Advanced Institute of Science and Technology, Korea.
[††]The author is with Thinkware System Corporation, Korea.
[†††]The authors are with Chungbuk National University, Korea.
a) E-mail: yjs@chungbuk.ac.kr
DOI: 10.1093/ietisy/e91–d.7.2061

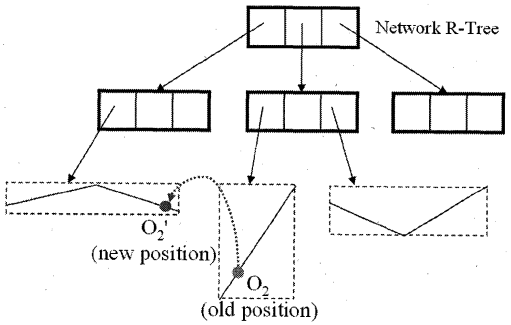

Fig. 1 The problem of existing network model based indexing.

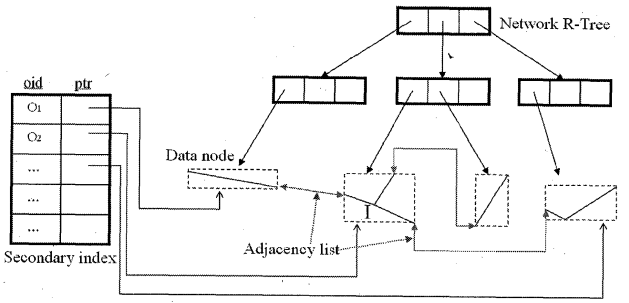

Fig. 2 The IONR-tree.

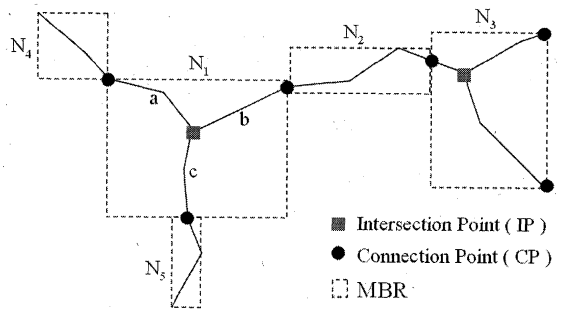

Fig. 3 Example of our road network.

the information of moving objects directly related with the update costs. The dynamic part has two structures; road sector blocks and data blocks of moving objects. Each road segment entry on leaf nodes of R*-tree points to a road sector block, which stores the identifiers of the moving objects on this road segment. The data block of moving objects stores the velocities and other related attributes of each moving object. Once the update request of moving objects is submitted, the information data of road sector blocks and data blocks of moving objects must be updated. This also leads to performance degradation as the number of objects increases because the update requests of the object are more likely random distributed and display less locality.

If there are millions of objects scattered on road networks and each object sends update requests to the server independently, we don't know where the next update request occurs on road networks. An object going out of the segment means additional disk accesses to the adjacency component. Figure 1 shows the problem of the existing index structures. When object O_2 moves to the new segment in real network, the network R-tree must be searched from the root to insert the object with a new segment. Moving objects going out MBR cause lots of performance degradation.

3. The IONR-Tree

3.1 Architecture

We propose the IONR (Intersection Oriented Network R)-tree for efficiently updating the current positions of moving objects on road networks. The IONR-tree is composed of secondary index and network R-tree. Figure 2 shows the architecture of our index structure. The secondary index is used for directly accessing the current positions of moving object. The network R-tree is different from existing index structures for current positions on road networks because an intersection point in which three or more edges meet is split to preserve network connectivity.

In the existing index structures, the intersection point in which three or more edges meet is split resulting in connectivity information to be lost. The IONR-tree preserves network connectivity through deliberately splitting road networks. We preserve network connectivity by not splitting intersecting nodes in which 3 or more edges meet. We define *IP* (Intersection Point) as a node in which 3 or more edges meet and *CP* (Connection Point) as a point split by *MBR*s. Figure 3 shows an example of our road network. There are two *IP*s included in two *MBR*s that are not split.

The main issue is how we determine the size of each MBR. The IMORS has proposed a cost model for determining the optimal size of a split and indexed poly line. Given a road network, the IMORS found the optimal number of entries n_{opt} that minimizes leaf node access cost from the root in the R*-tree. Let L_x, L_y, Q_x, Q_y be the sums of length of all road segments and the side lengths of query q along x and y-axis respectively. The optimal number of entries $n_{opt} = \sqrt{L_x L_y / Q_x Q_y}$. Let the total length of road networks be L, then the sum of lengths of road segments of data node ls is determined by $ls = L/n_{opt}$. The sum of lengths of the split road segments for a data node with an *IP* is obtained by length expansion from the *IP* to other *IP*s of every direction until it reaches ls. As shown in Fig. 3, the sum of lengths of road segments ls in the node N_1 is computed as the sum of road segment a, b, and c.

In a data node, adjacency list that represents the connectivity information is added additionally. Poly-line part stores the concrete shape between *IP* and *CP* or *CP* and *CP*. The moving objects are stored in the objects part of the node. Figure 4 shows the structure of a data node. There are two kinds of data nodes according to the number of *CP*s that the node has. Figure 4 (b) and (c) represent the adjacency lists for a data node with N *CP*s. $Offset_n$ points to Poly-line part to represent the real shape between *IP* and the n-th Connecting Point CP_n. P_n points to a data node which has a road segment connected to CP_n. We can directly access

the connected road segment by this pointer P_n when objects exit from CP_n.

3.2 The Operations of IONR-Tree

The Insertion of a new moving object is similar to IMORS except that the position of moving object is inserted to data node. First, a new moving object with object id is first registered in secondary hash index of the IONR-tree and then Network R-Tree is searched to find out a data node in which the object must be included. The search algorithm of Network R-tree is the same as that of IMORS except for the representation of a data node. The object is inserted to the data node with its id and coordinates. In general, moving objects have fast movement pattern along road segments. Therefore, a fast and simple overflow treatment method is required. If there is no room in data node for adding a new moving object, a new data node called overflow node is created. A new moving object is stored in a new data node. Finally, the object in secondary index points to the data node to be accessed directly next time. Deletion of an object is handled by deleting it from secondary index and the corresponding data node in network R-tree.

Figure 5 shows the update algorithm of the IONR-Tree. Updating the position of an object is processed in three steps. First, we find a data node that the object is stored in secondary hash index and check whether the object still lies on the poly lines of the data node. If so, we just overwrite position value (x, y) of the object and terminate the process. If not so, we delete the object from the data node and calculate the nearest CP to the object's new position in Euclidean distance. Then we access the data node that the CP points to. Finally, we insert the object into the data node and modify secondary index for directly accessing this node next time the object requests update.

IONR-tree supports range queries. Given a query region, our search algorithm is the same as the IMORS. As a result, the objects contained in the given query region are returned as a query result.

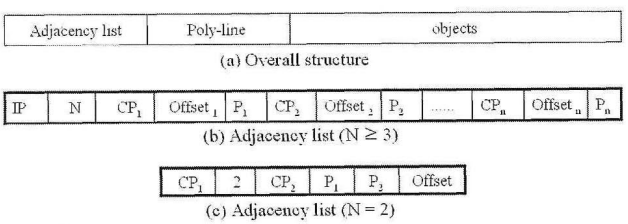

Fig. 4 Structure of data node.

```
Update( O, (x,y) )
O: moving object, (x,y): O's new position
Find O in secondary hash index and access a data node N
If (x,y) is in N then
    Update O's location in N
Else
    Delete O from N
    for each connecting point in N do
        Calculate Euclidian distance to (x,y)
    Find the nearest CP to (x, y)
    Access the node the CP points to
    If O is in the node then
        Insert O with (x,y) to the new node
    Else
        Search network R-tree and find the corresponding node
        Insert O with (x,y) to the node
    Update secondary index for object O
```

Fig. 5 Update algorithm of IONR-tree.

4. Performance Evaluation

We implemented both IONR-tree and IMORS in Java and carried out experiments on a Pentium 4, 2.8 GHz PC with 512 MB RAM running Windows XP. A network-based generator of moving objects has been used to create trajectories of moving objects on the real-world road network of Oldenburg [11]. In Fig. 7, the maximum speed division defines the speed of the moving object in a network-based generator. The larger this value is the slower moving object is. To evaluate the update performance, we measure the total number of disk accesses when the number of moving objects and the speed of moving objects vary.

Figures 6 and 7 show the results of update performance

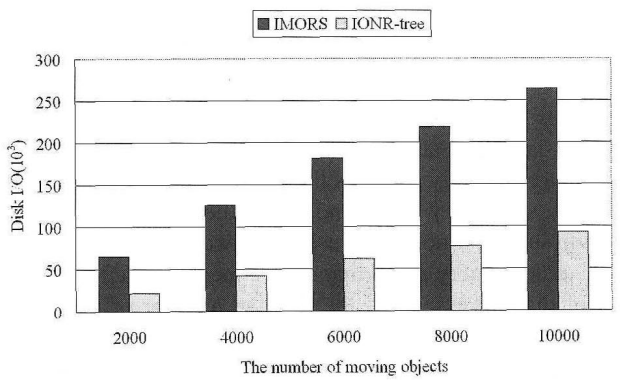

Fig. 6 The total number of disk access for updating when the number of moving objects varies.

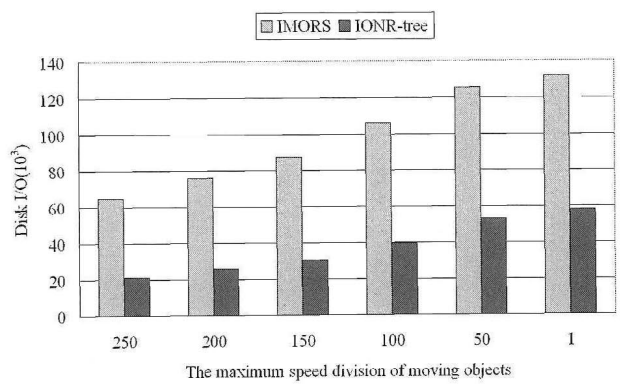

Fig. 7 The total number of disk access for updating when the speed of moving objects varies.

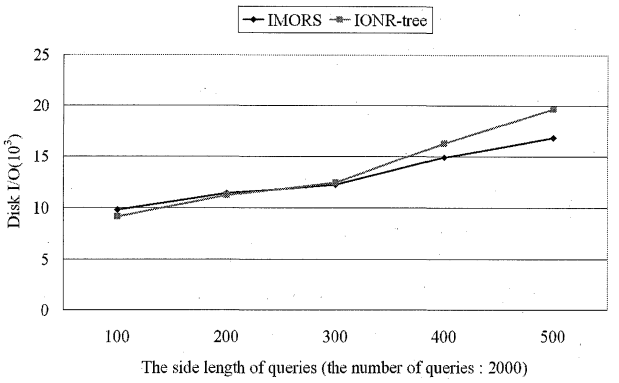

Fig. 8 The total number of disk accesses for query processing.

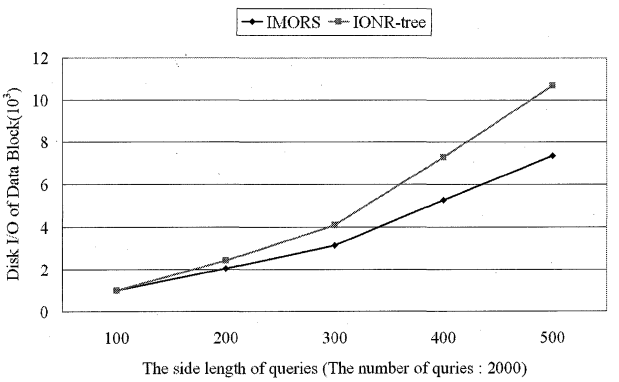

Fig. 9 The average number of data node accesses for query processing.

in terms of disk I/O. The IONR-tree has shown about 2 times faster update performance than IMORS because the IONR-tree directly accesses the old positions of moving objects by secondary index and stores the connectivity information in a data node to preserve network connectivity. To evaluate the query performance, we measure the total number of disk accesses and the average number of data node accesses when query size varies.

Figures 8 and 9 show the range query performance when the number of queries was 2000. The IONR-tree needs additional information such as adjacency list to improve the update efficiency and so degrades space utilization. In the worst case, the IONR-tree increases its depth for preserving network connectivity. As a result, the query performance of the IONR-tree is slightly deteriorated as the size of queries is increased.

5. Conclusions

We proposed the IONR-tree for efficiently updating the current positions of moving objects on road networks. IONR-tree exploits intersection-oriented network model that split networks deliberately to preserve network connectivity. IONR-tree has shown about 2 times better update performance and similar query performance compared to IMORS. Future work includes applying k-NN processing algorithms to the IONR-tree.

References

[1] Y. Tao, D. Papadias, and J. Sun, "The TPR*-tree: An optimized spatio-temporal access method for predictive query," Proc. VLDB, pp.790–801, 2003.
[2] X. Xiong and W.G. Aref, "R-trees with update memos," Proc. ICDE, p.22, 2006.
[3] Y. Tao and D. Papadias, "MV3R-tree: A spatio-temporal access method for timestamp and interval queries," Proc. VLDB, pp.431–440, 2001.
[4] D. Pfoser, C. Jensen, and Y. Tehodoridis, "Novel approaches to the indexing of moving object trajectories," Proc. VLDB, pp.395–406, 2000.
[5] C.S. Jensen and D. Pfoser, "Indexing of network constrained moving objects," Proc. ACM-GIS, pp.25–32, 2003.
[6] E. Frentzos, "Indexing objects moving on fixed networks," Proc. SSTD, pp.289–305, 2003.
[7] V.T. Almeida and R.H. Güting, "Indexing the trajectories of moving objects in networks," GeoInformatica, vol.9, no.1, pp.33–60, 2005.
[8] K.S. Kim, S. Kim, T. Kim, and K. Li, "Fast indexing and updating method for moving objects on road networks," Proc. WISEW, pp.34–42, 2003.
[9] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao, "Query processing in spatial network databases," Proc. VLDB, pp.802–813, 2003.
[10] J. Guo, W. Guo, and D. Zhou, "Indexing of constrained moving objects for current and near future positions in GIS," Proc. IMSCCS, pp.504–509, 2006.
[11] T. Brinkhoff, "A framework for generating network-based moving objects," GeoInformatica, vol.6, no.2, pp.153–180, 2002.