Indexing of Moving Objects on Road Network Using Composite Structure

Jun Feng[1], Jiamin Lu[1], Yuelong Zhu[1],
Naoto Mukai[2], and Toyohide Watanabe[2]

[1] Hohai University, Nanjing, Jiangsu 210098 China
{fengjun,welmanwenzi}@hhu.edu.cn
[2] Nagoya University, Nagoya, Aichi 464-8603 Japan

Abstract. Composite structures are proposed to index moving objects in road network. However, there are many problems for indexing moving objects for current and forecasting queries on road network, efficiently. This paper proposes methods for improving such kinds of structure, and gives a new structure R-TPR$^{\pm}$ Tree. Evaluation shows the new structure outperforms that of R-TPR-tree in query validity and disk access.

1 Introduction

With the improvements of geographic positioning technology and the popularity of wireless communication, location-based service (LBS) is proposed to provide the user with a service that is dependent on positional information associated with the user [1]. Prominent examples of LBS concern vehicle navigation, tracking and monitoring, where the positions of air, sea, or land-based equipment such as airplanes, boats and cars are of interest.

This paper is concerned with the indexing of moving objects on road network for real time and forecasting purposes, for example, transportation control and real time transportation guidance. It is quiet different to the indexing methods for objects moving in 2D free space, because not only the moving directions but also the distance among moving objects are constrained by the road network. Therefore the neighboring relations among moving objects should take the underlying road network into consideration. Past works on indexing of spatiotemporal data concerns either past data or present and future data. The former direction includes works in 2D space [2,3,4] or in spatial networks [5]. The latter direction includes works on indexing the present positions of moving points [3,6,7,8,9,10], all these approaches deal with spatial data in 2D space and cannot be applied to the moving objects in spatial network.

A straightforward method for indexing the static road network and the moving objects at the same time is the Composite Structure, shown in Fig. 1. The Composite Structure is made up of two parts: one is a Static Tree managing the information of road network, its leaf node such like the R1 in Fig. 1 denotes a region of the road network, points to a dynamic tree indexing moving objects move in this region, and all these dynamic trees compose the Dynamic Index

B. Apolloni et al. (Eds.): KES 2007/ WIRN 2007, Part II, LNAI 4693, pp. 1097–1104, 2007.

under the Static Tree. Using Composite Structure, we can insert and query spatial locations efficiently with the Static Tree, and update the positions of moving objects quicker as we divide them into small subsets and index them separately.

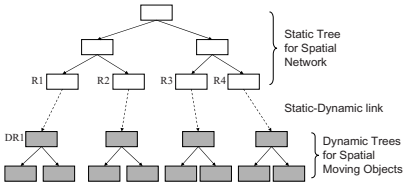


Fig. 1. Image of a Composite Tree

However, to assure the effective of the composite structure for indexing current positions of moving objects on road network, there are two issues need to be solved:

- **Update cost.** The update cost includes the cost for one operation and the times of operations. In current composite structures, index element of the Static Tree is road segment (in other words, the forecasting region is limited by the segment), and results in frequent updates when the objects move into other segments. Therefore, the issue of making the index element longer, enlarge the forecasting region and decrease the update times is the key to a "really" efficient composite tree.
- **Query cost.** The queries on moving objects includes the region query, nearest neighbor query, aggregate query and so on. In applications, aggregate queries are important as they can answer such kinds of questions: a specific road segment is fluent or not, or the average speed on a road. Therefore, how to make an index structure support kinds of queries should be considered.

In this paper, we propose methods to getting an efficient composite structure for indexing moving objects on road network. We bring in the idea of road connection algorithms [11] for dividing the approximate regions of Static Tree, which use the road network within which the objects are assumed to move for predicting their future positions, and can increase the forecasting period of moving object and decrease the update times of the database. Furthermore, we propose new spatiotemporal index structure for moving objects for every road region, which arranges the objects by their moving directions as well as their locations. This structure supports the aggregate and spatiotemporal queries more efficiently than other structures.

In the followings of this paper, Section 2 gives the road connection methods and the analysis based on experiments; Section 3 depicts our new spatiotemporal structure and Section 4 is experiments and analysis; conclusions are given in Section 5.

2 Road Connection Algorithms and Analysis

There is reality meaning in connecting several road segment as an index element, as on one hand, regarding the connected segments as a group will decrease the element number of the Static Tree, and increase the query efficiency; on another hand, the relativity of road segment provides the important cause for segment selection for a moving object when it faces several segment at a cross node. The better the connection algorithm is, the more precision the forecasting is. However, how to decide the size of forecasting region or the length of forecasting path is a problem in monitoring moving objects on road network. When the length of forecasting path is too long, the forecasting precision cannot be approved; while it is too short, the gathering of sample locations will be a heavy burden.

To find out a better way for road connection, we apply LSC algorithm (Length and Side-Based Segment Connection Algorithm) and ASC algorithm (Angle-based Segment Connection Algorithm) [11] to the real road networks (California Road Network from Geographic Data Technology Inc, 1999). Experiments shows that the numbers of road segments are decreased more than 59 percent, the length of segment and the nodes number are increased. And the road networks processed by ASC is better for representing the real road network for car tracking, because the number of main roads is less than those non-main roads, and the nodes included in the main roads are more, the length is longer than those of the non-main roads. These properties are just similar to the driving properties of cars in road networks.

In our composite tree for the moving objects in road networks, the problem of creating a "good" Static Tree for road network is how to decrease the update times occurred at every changing segment time of moving objects, the segment connection algorithms is just bring a practical way for solving this problem. In the followings, we use the road segment processed by ASC algorithm as the basic index element in our Static Tree.

3 R-TPR$^{\pm}$ Tree

R-TPR$^{\pm}$ Tree is a composite index structure used for managing the real-time data of moving objects on the road network. It is combined by two parts: a static 2D R^*-tree indexing the road network on the top of a forest of dynamic 1D TPR$^{\pm}$-trees indexing the moving objects. Objects on every road segment are supposed to be moving at a specific speed just the same as that at the sampling time. Though the directions of them may change all the time along the road segments, if they are still in one segment, it is thought to be keeping the same direction. Therefore, in a road segment, the moving directions of moving objects are just two: from start node s to end node e(denoted as + direction), or from e to s(denoted as - direction). Furthermore, in the real road networks, the cost or the weight of a segment in two directions are not always as the same, we propose a TPR$^{\pm}$-tree for indexing the spatio-temporal information in one road segment.

TPR$^{\pm}$-tree is a tree with a root node and two subtrees. In every subtree, there are only objects in the same direction are managed, in other words, the

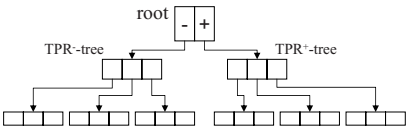


Fig. 2. Image of TPR$^{\pm}$ Tree

two subtrees are managing the objects in different moving directions. In the root node, there are two pointers point to these two subtrees, one is +, another is -. Fig. 2 depicts this situation. In a more formal speaking, there are two elements of R-TPR$^{\pm}$ Tree's data set, R and O, R is the road network and O is the moving objects set. After dividing R into segments by ASC algorithm, there will be a set of independent segments, R=$\{S_1, S_2, , S_n\}$, where $S_i (1 \leq i \leq n)$ means a segment of the road network, and n is the number of the segments in R. While R is divided, the set of moving objects O is also divided into n subsets, represented as O = $\{O_1, O_2, \cdots, O_n\}$, and $O_i = \{mo_p, mo_{p+1}, \cdots, mo_q\} (1 \leq i \leq n\ )$, where mo_j $(p \leq j \leq q\ )$denotes a moving object that moves in the segment S_i, and we will get: NUM(O) = $\sum_{i=1}^{n} NUM(O_i)$, Where NUM(O_i) means the amount of moving objects in the subset O_i. As we said before, in each segment of the road network, a moving object moves from the start point to the end, or opposite. We use TPR^{+}-tree(TPR^{-}-tree) to index the data of the set O_i in +(-) direction. The leaf node of the TPR^{+}-tree(TPR^{-}-tree) is denoted as: (j-Identifier, P_j, V_j), where j-Identifier refers to the object O_j. $P_j = \frac{Length(mo_j,A)}{Length(A,B)}$ is the relative position of the segment its on, where A is the start point of the segment, and B is the end point, Length(A, B) means the length between A and B. P_j approaches to 0 means O_j is as close to the start point A as much, and if O_j is more close to the end point B, P_j will approaches to 1 as much. At the same time, V_j means the velocity of O_j, when O_j moves from A to B, then $V_j > 0$, and $V_j < 0$ when it moves opposite. The non-leaf node of this TPR^{+}-tree(TPR^{-}-tree) is denoted as: (P_s, P_e, V_s, V_e, child-point). Where child-point is the address of a lower node, $|P_s, P_e|$ covers the position interval of its lower node's entries, and $|V_s, V_e|$ covers the velocity interval. As there are n subsets of O, so there will be n TPR$^{\pm}$-trees, and we can use a forest to indexing the moving objects in the road network.

Segments R are indexed by a 2D R^*-tree. The leaf node of the index is denoted as: (i-Identifier, MBR_i, TPR$^{\pm}$-point, P_i). Where i-Identifier refers to the segment S_i, MBR_i is a minimal bounding rectangle of the segment, TPR$^{\pm}$-point the pointer to the TPR$^{\pm}$-tree indexing the objects of O_i, and P_i is something other properties of the segment, such like length. The non-leaf node of this R^*-tree is denoted as: (child-MBR, child-pointer), where child-pointer points to the a lower node, and child-MBR covers all MBRs of its lower nodes.

4 Forecasting Query Method

In application, it is more important to get aggregate information than calculating exact positions of cars moving on the road network. A forecasting aggregate

query Q is denoted as: (R, t), where R describes the forecasting range, and t refers to the forecasting period. Using the 2D R^*-tree, we can find out the segments which we want to calculate inside R, and those TPR$^{\pm}$ Trees indexing moving objects in these segments. And we will get how many cars will be still in this segment after t seconds later according to their current speeds and positions with this forecasting method.

The non leaf node of the TPR$^{\pm}$ Tree is denoted as: (P_s, P_e, V_s, V_e, child-point), as the moving directions of those objects indexed by the same node are the same, and we assume their speeds keep constant, so we can get:

$$NP_s = P_s + V_s * t$$

$$NP_e = P_e + V_e * t$$

Here, $|NP_s, NP_e|$covers the position interval which those objects will not overstep after t seconds. So if NP_e is less than 1, which means all those objects indexed by this node will not exceed this segment t seconds later, and if NP_s is larger than 1, which means all those objects will leave this segment t seconds later. When NP_s is less than 1 and NP_e is larger than 1, we need recurs to its sub nodes and repeat this algorithm until arriving to the leaf nodes.

Those objects leave the segment after t seconds will select proper segments, insert into those other TPR$^{\pm}$ Trees , and deleted after the algorithm end. Those objects still in will be added into the traffic flow of current segment.

Algorithm Forecasting Traffic Flow FTF(R, t)
/*The number of segments of this road network is n. There is a global queue *TransArray* to save nodes will be inserted into other trees and a global array *FLOW[]* for recording the result. **Input**: R is the predict Range, and t is the forecasting period. **Output**: FLOW[] contains the traffic flows every segment will have after t seconds. */

```
1.   int FLOW[n]
2.   initialize a node array TransArray
3.   TPR±-tree trees[]    //find those TPR-trees inside R
4.   trees = 2D R*-tree.RangeQuery(R)
5.   for each tree in trees
6.      FLOW[tree.ID] = Single Segment Forcast(tree.+node, t)
7.      FLOW[tree.ID] = Single Segment Forcast(tree.−node, t)
8.   while TransArray is not empty
9.      tree = transArray.headNode.selectRoute
10.     tree.Insert(transArray.headNode)
11.     FLOW[tree.ID] = Single Segment Forcast(tree.+node, t)
12.     FLOW[tree.ID] = Single Segment Forcast(tree.−node, t)
13.     tree.delete(transArray.headNode)
14.     transArray.removeHead
```

End Algorithm Forecasting Traffic Flow FTF

Algorithm Single Segment Forecast SSF(N, t)
/∗ **Input**: N is a node of TPR tree, at the beginning of this algorithm, it means the root node of TPR$^+$- tree(TPR$^-$-tree); t is the forecasting period. **Output**: FLOW is the number of cars which will be still in the segment t seconds later indexed by N ∗/

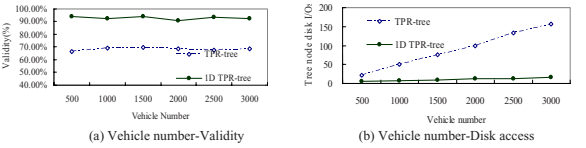
```
1.   int FLOW;
2.   if ( NPe ≤1 )
3.      /∗all cars indexed by N will not exceed the segment t seconds later ∗/
4.      FLOW += N.carNum;
5.    else if (NPs > 1)
6.      /∗all cars indexed by N will exceed the segment t seconds later ∗/
7.      For each subNode Ni in N
8.         TransArray.add(Ni);
9.    else if (N.level == 0)
10.     /∗part of cars will be in the segment and N is a leaf node ∗/
11.     For each car Moi in N
12.        If (Moi.position + Moi.velocity*t ≤1)
13.           FLOW++;
14.        Else
15.           transArray.Add(Moi);
16.   else
17.     /∗part of cars will be in the segment and N is an non leaf node ∗/
18.     For each subNode Ni in N
19.        FLOW += flowCal(Ni);
20.   return FLOW;
```

End Single Segment Forecast SSF

5 Experiments and Analysis

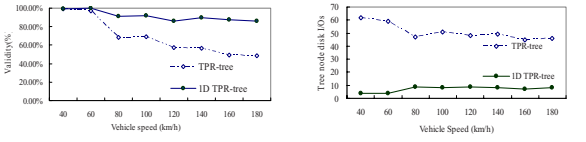
In this section, we evaluate the method proposed by this paper compared with that using TPR-tree [8] in the composite structure in terms of forecasting query validity and disk accesses (here refers to tree node access times). For our experiments, we used a PC of Pentium IV CPU with 3.19 GHz and 512M bytes of main memory. The Trees and the algorithms were implemented in C++ and compiled using Visual C++ version 2003.

Our simulating environment for experiments are: the moving objects in a road network are indexed by R-TPR$^\pm$ Tree and R-TPR-tree (the Static Tree is R*-tree and the Dynamic Tree is TPR-tree). The validity tests and tree node disk access tests are carried out for the vehicles moving on a 1 km road segment. For the queries, we varied vehicle numbers or forecasting period or vehicle speed. For every query, we performed it 3 times and the values shown in this section are the average per query. Fig. 3 depicts the results varied with different vehicle numbers when vehicle speed is 100 km/h and forecasting period is 5 seconds.



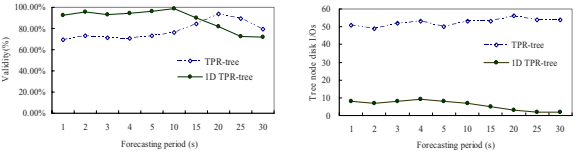
(a) Vehicle number-Validity (b) Vehicle number-Disk access

Fig. 3. Situations of vehicle speed is 100km/h and forecasting period is 5s



(a) Vehicle speed-Validity (b) Vehicle speed-Disk access

Fig. 4. Situations of vehicle number is 1000 and forecasting period is 5s



(a) Forecasting period-Validity (b) Forecasting period-Disk access

Fig. 5. Situations of vehicle number is 1000 and speed is 100km/h

Fig. 4 depicts the results varied with different vehicle speed when there are 1000 vehicles moving on that segment in two directions, and the forecasting period is 5 seconds. Fig. 5 depicts the results varied with different forecasting period when there are 1000 vehicles moving on that segment in two directions, and vehicle speed is 100 km/h.

By observing the results, there are: first, as expected, the number of disk accesses of our method is quite fewer than that of TPR-tree, and keeps a stable line. Second, it is clear that the validity of our method is better than that of TPR-tree while the forecasting period is shorter.

6 Conclusion

In this paper, we proposed R-TPR$^{\pm}$ Tree for indexing moving objects in road network for current and forecasting query. It is a composite structure consists of a R*-tree and a forest of TPR$^{\pm}$ Tree proposed by this paper. Evaluation shows that our method outperforms that of R-TPR-Tree in query validity and disk access. In our future work, the more forecasting queries under more complex situations will be tested, such as forecasting aggregate queries for several connected road segments and so on.

Acknowledgements

This research is supported by NSFC(granted number 60673141, Research on the Index Structure of Spatial Network-based Moving Objects).

References

1. Chung, J.D., Peak, O.H., Lee, J.W., Ryu, K.H.: Temporal pattern mining of moving objects for location-based services. In: Proc. Int'l Conf. Database and Expert Systems Applications (2002)
2. Hadjieleftheriou, M., Kollios, G., Tsotras, V., Gunopulos, D.: Efficient indexing of spatiotemporal objects. In: Proc. of the 8th Int'l Conference on Extending Database Technology, pp. 251–268 (2002)
3. Kollios, G., Gunopulos, D., Tsotras, V., Delis, A., Hadjieleftheriou, M.: Indexing animated objects using spatiotemporal access methods. IEEE TKDE 13(5), 758–777 (2001)
4. Lazaridis, I., Porkaew, K., Mehrotra, S.: Dynamic queries over mobile objects. In: Proc. of ACM-SIGMOD Conference on the Management of Data, pp. 269–286. ACM Press, New York (2002)
5. Pfoser, D., Jensen, C.S.: Indexing of network constrained moving objects. In: Proc. of GIS 2003, pp. 25–32 (2003)
6. Agarwal, P.K., Arge, L., Erickson, J.: Indexing moving points. In: Proc. of the 19th ACM Symposium on Principals of Database Systems, pp. 175–186. ACM Press, New York (2000)
7. Pfoser, D.: Indexing the trajectories of moving objects. IEEE Data Engineering Bulletin (2), 3–9 (2002)
8. Saltenis, S., Jensen, C.S., Leutenegger, S., Lopez, M.: Indexing the positions of continuously moving objects. In: Proc. of ACM-SIGMOD Conference on Management of Data, pp. 331–342. ACM Press, New York (2000)
9. Saltenis, S., Jensen, C.S.: Indexing of moving objects for location-based services. In: Proc. of the 18th Int'l Conference on Data Engineering, pp. 463–472 (2002)
10. Sistla, A.P., Wolfson, O., Chamberlain, S., Dao, S.: Modeling and querying moving objects. In: IEEE Int'l Conf. on Data Engineering, pp. 422–432. IEEE Computer Society Press, Los Alamitos (1997)
11. Civilis, A., Jensen, C.S., Pakalnis, S.: Techniques for efficient road-network-based tracking of moving objects. IEEE Trans. on Knowledeg and Data Engineering 17(5), 698–712 (2005)