Optimization of history tree in 3DR-tree index structure

Zhang Zhi-tong
Faculty of Technology, Harbin University
Harbin,china
sjzzt@hrbu.edu.cn

***Abstract*—Many optimizations have been done to 3DR-tree index structure and many opinions have been proposed. Modification by splitting mechanism is one of them. There are two index trees in 3DR-tree index structure after modification: one is a history tree for past data storage and the other is an active tree for current data storage. In this article, optimization of history tree is firstly done and is proved theoretically. Then a correspondent insert algorithm is designed.**

Keywords-3DR-tree; index structure; R*-Tree structure model; history data insert algorithm

I. .FACTORS FOR INSPECTION

Based on structure algorithm[1] of R*-Tree[2], while inserting a new record, the optimization of 3DR-tres[3] index structure is investigated from the following aspects: area and circumference of MBR, coverage area between MBR within a same node, as well as the distance from center of MBR to center of node which contains the MBR.

II. STRUCTURE MODEL

In history tree, existence of large amount of blank space influences searching efficiency, e.g. searching section may intersect with node's MBR but without actual object exists in the intersection, or visiting unwanted nodes[4]. In the case of spatial-temporal data processed in 3DR-tree, the time dimension continuously expends upwardly along with increasing update time t in spatial-temporal object data, which induces a large proportion of MBR is occupied by blank space. Therefore, it is considered to take blank space as an inserted cost function in order to control the increasing blank space of nodes in 3DR-tree structure algorithm. This modification has the advantages of more compacted nodes, a reducing possibility of intersection between search space and blank space of node MBR, and an improved searching efficiency.

- Definition 1: Blank space. In 3DR tree, the blank space in a node is defined as the area of the space which is not covered by real object, expressed as:

Dead_Space(Nodeo)=Coverage(o)-Area(o) （1）

- Definition 2: Insert cost function. By setting leave node layer level=0，root_level represents the layers of root nodes, and then the cost function of history tree during inserting is defined as below:

$$Cost = \sum_{\Delta} Area(o) \quad \text{1<=node level<=root\_level}$$

$$Cost = \sum_{\Delta} dead\_space \quad \text{node level = 0} \qquad (2)$$

Where, Area is node area, and Coverage is the actual covered area. At the same time, while inserting a new record, in non-leave node layer, the node with minimum sum area increment is selected by history tree to be the insert node for next step. A smaller node area will be selected if area increments are equal. The efficiency of this method is lowered when several nodes have an identical or similar area increment. As shown in Figure 1(in this case, a 2D data is used for demonstration. 3D has a similar form to 2D), h, i, and j are child nodes; a, b…g are both child nodes corresponding to h, i and j and the leave nodes of the upper level nodes. The layer of node is indicated by level. The leave node level=0 is thus defined in history tree. Record p is assumed to be inserted to the position as shown in Figure 1. According to algorithm of 3DR-tree, inserting p into neither node h nor node j will enlarge the node area. Under this circumstance, a node with a smaller area is selected, which is node h. Then, node h chooses a node with minimum sum area expansion caused by insertion as the node for next insertion step, which is node e. However, the best choice to insert record p is node a because node a causes its area expansion is far smaller than node e. In this case, the each step of insert algorithm used in 3DR-tress could only obtain an optimum insert choice for next step, instead of an optimum result for the overall system.

Published by Atlantis Press, Paris, France.

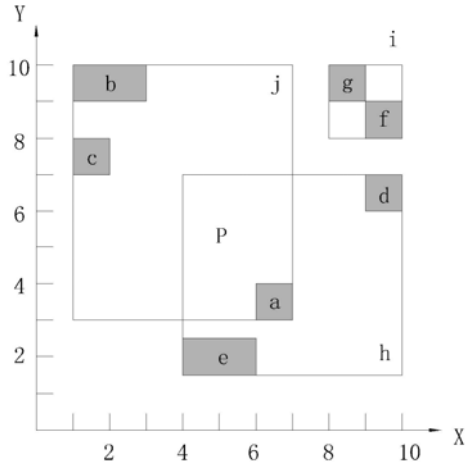

Fig.1 Inserting cost of 3DR-tree

Thus it is considered to use minimal cost priority algorithm while inserting a new record into history tree. The cost parameter is set as the summation of area expansion of all the nodes after insertion in the insert path from root node to node level=1.

According to inset cost function, an overall optimum insert path is selected as the path from root node to node level=1, during inserting a new record. This path satisfies the minimal sum area increments of all nodes, after the record is inserted along the path. Then, according to this path, the leave node with minimum dead_space increment after record insertion, is selected from child nodes of node level=1 to finish insertion process.

III. HISTORY DATA INSERT ALGORITHM

The partially long lasting [5] (partial persistent) index data has all history versions for data storage. However, only the current version is updatable. There is no physical deletion in Spatio-temporal index. Updating data only changes the now to a specific time value t2 in time interval [t1，now] for object's current data item. The previous data item becomes(o_id，sj，[t1，t2]) as read-only history data and meanwhile new data item (o_id，sj，[t2，now]) is inserted into index structure. The index data update algorithm given below contains two parts: update current time data item in R1; insert history data into R2.

While inserting a new mobile object data to history tree, unlike, differing from inserting by layers according to greedy policy in 3DR-tree, history tree first get overall optimum insert path according to choose-path[6] algorithm and then insert record by this path.

A. The Basic Logic of Algorithm

The basic logic of algorithm is to expand a node, selected from all the expandable nodes, having a minimal parameter. This enables an overall viewpoint. Take Figure 1 as an example, setting r as root node, the procedure of searching optimum insert path by Choose Path algorithm can be described as:

- Initially, priority queue PQ is {<(r)，0>}. Insert into PQ the path obtained by expanding the child node of r, and get{<(h，r)，0>，<(j，r)，0>，<(i，r)，16>} after delete the first record <(r)，0> in PQ to. Each value represents the value of cost parameter after p is inserted into this path. Insertion of p into node h or node j will not increase node area, therefore the cost parameter is 0 but the area is increased by 16 after node i is inserted.
- Visit node h in the first item of PQ, insert another two path (e，h，r) and (d，h，r) to PQ and then delete <(h，r)，0> from PQ. PQ={<(j，r)，0>，<(e，h，r)，14>，<(d，h，r)，13>，<(i，r)，16}. Path (e，h，r) and path (d，h，r)has reached the node of le, and will not continue.
- Expand path (j，r) in the first item, and insert path (a，j，r)，(b，j，r)and (c，j，r) into priority queue; PQ={<(a，j，r)，12>，<(e，h，r)，14>，<(d，h，r)，13>，<(c，j，r)，13>，<(i，r)，16>，<(b，j，r)，14>}. At this moment, a path (a，g，r)leads to node of le by minimum cost is found and(a，g，r) is the path of final selection.
- While selecting leave node, the node with minimum dead_space increment after inserting p is selected from child nodes of node a for nest step insertion, which ensures the all insert path from root node to leave node. During application, utilizing a sample of a heap path_heap is applied to realize priority queue. Each heap entry of path_heap contains properties: son, path and cost. The son indicates the id of the node for visit in next step. Cost=∑ervery nodeo△Area(o).The path records the path being visited currently, which is also the sum area increment in the path after new record is contained in every node. The o is the node in path.

B. Definition

- Definition 3: Status. In history tree, status is defined as the saved path from root node to middle node items in priority queue PQ. Its form is represented as <N1，N2，…Nk>, where N1 is root node and Ni+1 is the child node of Ni(1<=i<=k-1). 1<=Nk.level<=root_level.
- Definition 4: Descendant and direct descendant. A status descendant is defined the descendant path generated by this status. Its form is defined as: assuming <N1，N2，…Nk> is a status and then its descendant is <N1，N2，…Nk，Nk+1，…Nh >; Ni+1 is the child node (k+1<=i<=h-1) of Ni and 1<N<=h.level<=root_level. In the case h=k+1, the descendant is called direct descendant.

C. Provement

Theorem: Choose Path algorithm can be stopped within a limited steps and find a optimum path satisfying minimum cost parameter Cost=∑ervery nodeo△Area(o).

The provement is done as follows：

- Firstly, there must be an optimum path from initial node (root node) to target node (node level=1). Due to the limited height of history tree, the path from initial node to target node has a finite number, among which one or more path must exist to enable the minimum cost parameter.
- Secondly, before algorithm terminates, each optimum path has a status in priority queue PQ.
- In factor, every path from initial node to target node has a status in priority queue PQ. This is because that starting from initial node, every descendent generated by current status will be inserted into PQ. The current status will not be deleted from PQ unless all the descendents are generated. Now, assuming the status N=<N1，N2，…Nk> is one status in optimum path, there are finite status before N in PQ at any designated moment. The finite status is called first generation status where the minimum cost parameter is a1.
- The direct descendent of first generation status is called second generation status, where the minimum cost parameter is a2. Assuming the cost parameter increased by e after each descendent is generated from a status, then a2>=al+e, and aj>=a1+(j-1)e in general. Assuming the cost parameter of optimum path is c, then aj>=c when j is large enough. This shows that the descendents generated from the first generation status has only finite generations before N. It will be deleted from PQ because every status has finite direct descendents. Therefore, S will become the first status in PQ after finite steps. Again, algorithm inevitably reaches target status T and stop after finite steps because of the finite status in optimum path. This is the optimum solution.
- If algorithm stops before reaching T, this means another target status or another path is found. This is also an optimum solution because the status in PQ is queued by cost parameter value.

Comparing to 3DR-tree algorithm, more node visits are required for Choose Path to find optimum insert path by minimum cost priority algorithm at node level>0. However, Choose Path algorithm generates better tree structure and improves search efficiency finally.

IV. CONCLUSION

Based on structure algorithm of R*-Tree, out work introduces dead_space as cost parameter which enables an overall optimum solution search and improves the modified 3DR-tree. Moreover, according to feathers of current model and combining general indexing method of R-tress structure, an insert algorithm is proposed to realize a 3DR-tree index structure. Then insert algorithm is demonstrated from the aspect of history date.

REFERENCE

[1] PATEL J M, CHEN Y, CHALLA V P. STRIPES: An Efficient Index for Predicted Trajectories[C]. In Proc.of the ACM SIGMOD Intl.Conf.on Management of Data, Paris, France, 2007: 637-646.

[2] GUTTMAN A. R-trees: A Dynamic Index Structure for Spatial Searching[C]. In: Proe. Of the ACMSIGMOD, Boston, MA ACMPress, 1984: 47-57.

[3] TAO Y, PAPADIAS D. MV3R-tree: A Spatio-Temporal Access Method for Timestamp and Interval Queries[C]. Proceedings of the 27th International Conference on Very Large Databases, San Francisco, 2001: 431-440.

[4] PATEL J M, CHEN Y, CHALLA V P. STRIPES: An Efficient Index for Predicted Trajectories[C]. In Proc.of the ACM SIGMOD Intl.Conf.on Management of Data, Paris, France, 2007: 637-646.

[5] PFOSER D, JENSEN C, THEODORIDIS Y. Novel Approaches in Query Processing for Moving Object Trajectories[C]. In Proceedings of 26th International Conference on Very Large Data Bases, Cairo, Egypt. 2000: 395-406.

[6] TAO Y, PAPADIAS D, SUN J. The TPR*-Tree: An Optimized Spatio-Temporal Access Method for Predictive Queries[C]. In Proc.Of the Intl.Conf. On Vary Large Data Bases, VLDB, Rome, Italy, 2006: 431-440.