보존 경계 사각형을 이용한 이동객체의 현재와 미래 위치 색인†

황 도 퉁°, 정 영 진, 이 응 재, 류 근 호
충북대학교 데이터베이스 연구실
{tunghdt°, yjjeong, eungjae, khryu}@dblab.chungbuk.ac.kr

Indexing for current and future positions of moving objects using new conservative bounding rectangle

Hoang Do Thanh Tung°, Young-Jin Jung, Eung-Jae Lee, Keun-Ho Ryu
Database laboratory, Chungbuk National University

Abstract

Nowadays, with numerous emerging applications (e.g., traffic control, meteorology monitoring, mobile computing, etc.), access methods to process current and future queries for moving objects are becoming increasingly important. Among these methods, the time-parameterized R-tree (TPR-tree) seems likely the most flexible method in one, two, or three-dimensional space. A key point of TPR-tree is that the (conservative) bounding rectangles are expressed by functions of time. In this paper, we propose a new method, which takes into account positions of its moving objects against the rectangle's bounds. In proposed method, the size of bounding rectangle is significantly smaller than the traditional bounding rectangle in many cases. By this approach, we believe that the TPR-tree can improve query performance considerably.

1. Introduction

Spatio-temporal database that manages huge of dynamic objects are becoming increasingly important due to numerous emerging applications (e.g., traffic control, meteorology monitoring, mobile computing, etc.). The access methods for such database can be classified in two major categories depending on whether they deal with past information retrieval, or current and future positions of moving objects. In this work, we focus on second category.

Among many current access methods processing current and future queries, the time-parameterized R-tree (TPR-tree) proposed by Simonas Saltenis [1] seems likely the most flexible method. The technique naturally extends the R*-tree. Instead of using true, always minimum bounding rectangles, the TPR-tree employs rectangles termed *conservative* bounding rectangles (**Figure 1**). They are minimum at some time point, but most likely not at later times. The lower bound of a conservative bounding rectangle is set to move with the minimum speed of the enclosed points, while the upper bound is set to move with the maximum speed of the enclosed points. This ensures that conservative bounding intervals are indeed bounding for all times considered.

In this paper, we propose a new type of the conservative bounding rectangle. In many cases, the new type is significantly smaller than the traditional bounding rectangle. With new approach, we can reduce much dead space as well as improves its query performance compared with TPR-tree.

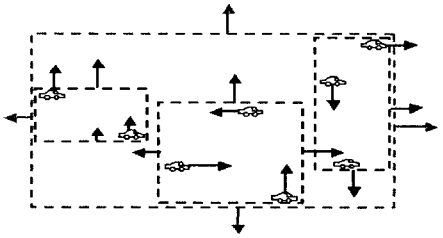


Figure 1. Conservative bounding rectangles and formula

The rest of the paper is organized as follows. Section 2 reviews development of the TPR-tree. Section 3 analyzes conservative bounding rectangle problems. Section 4 proposes new type of conservative bounding rectangle. Section 5 concludes the paper with the future work.

2. Related work

In 2000, Saltenis et al. [1] proposed the TPR-tree, which adapts the R*-tree construction algorithms to moving objects. The main idea is to make the bounding rectangles functions of time so that the enclosed moving objects will be in the same rectangles. This approach seems more practical because it overcomes drawbacks of previous approaches. Moreover, there

† This work was supported by KOSEF RRC project (Cheongju Univ. ICRC) in Korea

were two remarkably improvements. First, the R^{EXP}-tree [2] is an extension of the TPR-tree to handle moving objects with expiration times. Secondly, Yufei Tao [3] proposed TPR*-tree, which integrates novel insertion/deletion algorithms to enhance performance of TPR-tree. TPR*-tree consider swept regions for insertion/deletion algorithm instead of integral metric

3. Problems of Conservative bounding rectangle

Until now, the TPR-tree still use the maximum speeds for the upper bounds as well as the minimum speeds for the lower bounds of bounding rectangles in every direction. The follows is an example and analysis to clarify problems.

3.1 Example of growing nodes

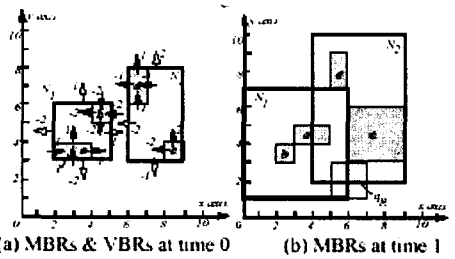


Figure 2. Entries in a TPR-tree

Figure 2 shows an example of growing of two nodes N_1, N_2 from time 0 to time 1. N_1, N_2 follow movements of enclosed objects a, b, c, d (notice that each edge moves according to its velocity). The MBR of a non-leaf entry always encloses those of the objects in its subtree, but it is not necessarily tight. For example, N_1 (N_2) at timestamp 1 is much larger than the tightest bounding rectangle for *a, b* (*c, d*). This example pointed out that the conservative bounding rectangles not tight, which generate much dead space overtime.

3.2 Analyzing Problem

Now, we turn into to analyze the reason of much dead space, which the conservative bounding rectangles generate.

Without losing generality, we analyze examples in one axis. There are two objects A and B, of which V_A (velocity of A) is the minimum and V_B (velocity of B) is the maximum. Thus, their conservative bounding rectangle has the lower bound velocity V_L equal to V_A and the upper bound velocity V_U equal to V_B. We assume that,

At t=0, Distance (lower bound, upper bound) = D_{LU}= 10, and

At t=0, Distance (A, B) = D_{AB} =D_{LU}.

Now, we exam two cases following

Case 1: when A is before B. **Figure 3** shows that the bounding interval is always tight

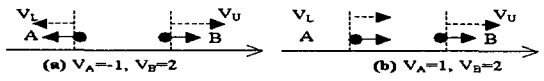


Figure 3.A is before B (a) moving in the opposite direction (b) moving in the same direction

Case 2: when A is after B. **Figure 4, 5** show the bounding interval is not always tight except current time.

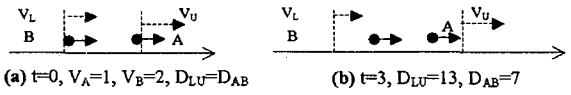


Figure 4.A is after B, they are moving in the same direction

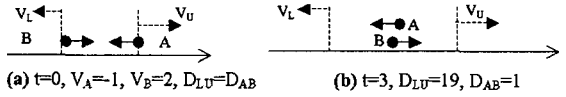


Figure 5.A after B, they are moving toward each another

From the above observation, we found that case 1 are optimal because the bounding interval is always minimum. It is easy to realize that, case 2 is not optimal because the bounding intervals are always not tight. Furthermore, when moving objects fall into case 2, the conservative bounding rectangle will generate much dead space and obviously degenerate query performance.

4. New approach for conservative bounding rectangle

To overcome the above problem, we thought a new type of conservative bounding rectangle, where the speed of the upper bound is reduced as much as possible and, analogously, the speed of the lower bound is increased as much as possible.

From observation that the TPR-tree is optimized for timestamp queries in interval $[T_C, T_C+H]$, where T_C is the current update time, and H is a tree parameter. By taking into account positions of its moving objects against the rectangle's bounds, we created a new bounding rectangle, which is nearly optimal, replaced the current one.

4.1 In case of one dimension

In **Figure 6**, we examine for one axis, assume that position P_u of the upper bound is farther than position P_2 of O_2, who moves farthest after H time. V_2 is O_2 speed. The distant between O_2 and the upper bound is d= P_u –P_2, d>0.

In H time, O_2 will move a distant S= V_2*H, and the upper bound will move a distant S'= V_2*H. At time H, to the O_2 catches up with the upper bound, S=S' + d, so that V_2*H = V_2*H + d. Given V_2, to O_2 always runs within that rectangle u need running at speed $V_u = \frac{V_2 \times H - d}{H}$, V_u <V_2 within H,. It is similar for the lower bound has speed $\left|V_l = \frac{V_1 \times H - d}{H}\right|$, d=$P_L$-$P_1$, L is position of the lower bound.

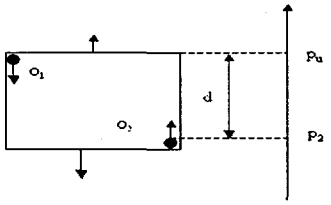


Figure 6. Example in one axis

4.2 In case of multi dimension

In generality, with moving objects in d-dimensional space and having more than two objects, we need to choose objects, which will go farthest after H time for every dimension. Therefore, the bounds will reach the positions of objects that move farthest for each axis.

We replace the traditional formula

$$v_i^{\min} = \min \{o.v_i \mid o \in node\}$$
$$v_i^{\max} = \max \{o.v_i \mid o \in node\}$$

By the new formula to calculate new bound speeds as follows

$$v_i^{\min} = \min\left\{\frac{o.v_i \times H - l_i - o.x_i}{H} \mid o \in node\right\}$$
$$v_i^{\max} = \max\left\{\frac{o.v_i \times H - u_i - o.x_i}{H} \mid o \in node\right\}$$

U_i is the upper bound's position of i^{th} dimension, l_i is the lower bound's position of i^{th} dimension.

By applying the above formula, we have new conservative bounding rectangle, which always has smaller speed in case 2 of section 3.2, so they spend less dead space.

See **Figure 7**, O_1 is running slowest and O_2 is running fastest. Notice that the new conservative bounding rectangle is minimum at time T_{ref}+H, it grows slower than load-time bounding rectangle, which is running with speed of O_1 and O_2. We really reduce much unnecessary space.

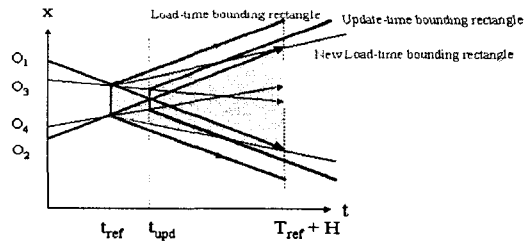


Figure 7. New conservative bounding rectangle

4.3 Example

An example is for comparison between two conservative bounding rectangles. We assume that there are 4 moving cars (km/minute): a(-1,1), b(2,-2), c(-1,0), d(1,2).

The traditional conservative (dashed) bounding rectangle, see **Figure 8**,

$v_x^{\max} = \max(a_x, b_x, c_x, d_x) = 2$

$v_x^{\min} = mix(a_x, b_x, c_x, d_x) = -1$

$v_y^{\max} = \max(a_y, b_y, c_y, d_y) = 2$

$v_y^{\min} = mix(a_y, b_y, c_y, d_y) = -2$

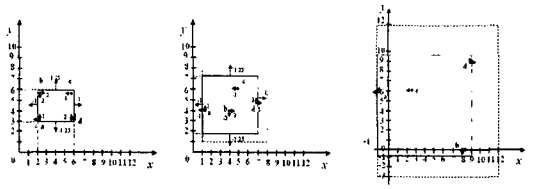
The new conservative (bold) bounding rectangle, see **Figure 8**, We assume that H= 4 and use the new formula to calculate

$v_x^{\max} = \max(-1,1,0,1) = 1$

$v_x^{\min} = \min(-1,0,0,1) = -1$

$v_y^{\max} = \max(0.25,-1.25,0,1.25) = 1.25$

$v_y^{\min} = \min(0,-1.25,0.75,0) = -1.25$



(a) t=0 (b) t=1 (c) t=3

Figure 8. Comparision between two kinds of rectangles

5. Conclusion

Motivated by improving efficiency for queries performance, in this paper, we proposed new approach for new type of conservative bounding rectangle, which is nearly optimal in many case. Due to much smaller bounding rectangles, we believe TPR-tree technique will gain overall good query performance. Moreover, if considering object positions in the insertion algorithm, we can find a better node for a new object inserted.

In the future work, we are going to develop an expansion TPR-tree, which uses new type of conservative bounding rectangle and improve insertion and deletion of the TPR-tree.

References

[1] S. Saltenis, C. S. Jensen, S. T. Leutenegger, and M. A. Lopez. "Indexing the Positions of Continuously Moving Objects". In Proceedings of SIGMOD, 2000.

[2] S. Saltenis and C. S. Jensen. "Indexing of Moving Objects for Location-Based Services". In Proc. of the Intl. Conf. On Data Engineering, ICDE, Feb. 2002.

[3] Tao, Y., Papadias, D., Sun, J. "The TPR*-Tree: An Optimized Spatio-Temporal Access Method for Predictive Queries". To appear in 29th Very Large Data Bases Conference (VLDB), 2003.

[4] H. D. Chon, D. Agrawal, and A. E. Abbadi. "Storage and Retrieval of Moving Objects". In *Mobile Data Management*, pages 173–184, Jan. 2001.

[5] C. M. Procopiuc, P. K. Agarwal, and S. Har-Peled. "STAR-Tree: An Efficient Self-Adjusting Index for Moving Objects". In *Proc. of the Workshop on Alg. Eng. and Experimentation, ALENEX*, pages 178–193, Jan. 2002.