# 基于 TPRA-tree 面向预言查询的时空索引

何茂顺，董一鸿*，付世昌
（宁波大学 计算机科学技术研究所，浙江 宁波 315211）

摘要：基于 TPR-tree 典型的移动对象索引方法，提出了 TPRA-tree. TPRA-tree 从频繁更新的移动对象运动方向的角度进行构造，减少了结点面积，减小了结点间重叠. 实验结果表明: TPRA-tree 的更新和与查询性能优于 TPR-tree.

关键词：移动对象; TPR-tree; TPRA-tree; 时空索引
中图分类号：TP311 文献标识码：A 文章编号：1001-5132（2011）01-0034-04

随着信息技术的发展和普及，如何提供各种高效移动对象的处理技术成了当前涉及移动对象时空数据库研究领域的重点与热点之一，具有重要的研究意义. 在移动对象数据库中，为提高搜索效率，就必须对移动对象进行时空索引. 当前所研究的时空索引主要有移动对象历史索引和移动对象当前和未来索引两种. 相对于移动对象的查询，则可分为面向历史的查询和面向预言的查询. 笔者主要研究后一种索引和后一种查询.

针对移动对象当前和未来位置索引方法的研究是目前时空数据库研究的前沿领域. 近年来研究者提出了许多索引技术来对移动对象当前和未来位置进行存储和检索，这些方法主要是基于参数化和二元变换方法的，如 TPR-tree[1]及其变种 TPR*-tree[2]、VCI R-tree[3]、S-V 模型[4]、STRIPES 索引[5]、Hough 变换[6]、HVTPR-tree[7]等.

笔者也是考虑到移动对象在空间域和时间域上的特殊性，为了减少结点面积及结点重叠，以利于更新和查询，基于 TPR-tree 进行改进，提出了基于移动对象运动方向角度的 TPRA-tree(TPR-tree based on the Angle).

## 1 相关工作

当前，在人们所提出的移动对象面向预言的索引方法中，最典型的是 Saltenis 等人提出的 TPR-tree[1]. TPR-tree 是建立在 R*-tree 基础上的一种索引结构，能够存储和访问移动对象，可以对移动对象的将来位置进行预测，支持移动对象时间片查询、窗口查询以及移动查询. TPR-tree 的结构与普通 R*-tree 的最大区别是其对外包矩形的描述，它使用以时间为参数的边界矩形(Time-Parameterized Bounding Rectangle, TPBR)取代原来的最小边界矩形(Minimum Bounding Rectangle, MBR). TPBR 存储移动对象在 $t_{\text{ref}}$ (最近更新)时刻的移动对象状态信息{<MBR>,<VBR>}. 以下均以二维空间为例进行阐述.

MBR 结构为 $\{(x_{\min}, y_{\min}),(x_{\max}, y_{\max})\}$，其中，$x_{\min}, y_{\min}$ 表示该结点所包含移动对象在每一维上的位置下界，$x_{\max}, y_{\max}$ 表示该结点所包含移动对象在每一维上的位置上界. VBR 是移动对象的速度矢量，结构为 $\{(v_{x\min}, v_{y\min}),(v_{x\max}, v_{y\max})\}$，其中，$v_{x\min}$，$v_{y\min}$ 表示该结点所包含的移动对象在每一维上的速度下界，$v_{x\max}$，$v_{y\max}$ 表示该结点所包含的移动对象在每一维上的速度上界.

MBR 是关于时间的函数，可利用 VBR 来描述 MBR 在每一维上的速度矢量. 这样移动点在 $t$ 时刻位置就可通过时间函数 $x(t) = x(t_{ref}) + v(t - t_{ref})$ 来描述. 具体来讲，在每一维上 MBR 上界(下界)是其所包含的所有对象或子节点 MBR 速度的最大速度(最小速度)，即不论移动对象位置如何变化，

收稿日期：2009-08-02. 宁波大学学报（理工版）网址: http://3xb.nbu.edu.cn
基金项目：浙江省自然科学基金（Y1080490）.
第一作者：何茂顺（1983－），男，河南范县人，在读硕士研究生，主要研究方向：数据挖掘. E-mail: hemaoshun@163.com
*通讯作者：董一鸿（1969－），男，浙江宁波人，教授，主要研究方向：数据挖掘、人工智能和软计算. E-mail: dongyihong@nbu.edu.cn

TPR-tree 所有节点 MBR 始终包含其子节点或移动对象 MBR. 但随着时间的推移，移动对象结点间重叠严重，查询性能急剧下降，这时就需要通过紧致 MBR 来提高查询性能.

TPR-tree 动态更新算法通常采用删除-重插机制. 即当更新 1 个结点时，首先自根结点向叶子结点方向搜索到该结点，进而删除该结点，最后重新插入更新后的该结点. 与传统插入类似，对索引项的插入采用最小面积增长原则，使 TPBR 在$[t_1, t_1 + H]$时段内，任何时刻的面积增长能尽可能保持较小. 为控制 TPBR 增长速度，采用积分值最小原则. 假设 $A(t)$ 是 TPBR 或重叠区域，则应使得待插入项插入后的区域面积(体积) $S = \int_{t_1}^{t_1+H} A(t)\mathrm{d}t$ 最小.

TPR-tree 的删除算法与 R*-tree 的删除方式一致. 即当结点中的索引项个数小于阈值时，则把此结点中索引项插入到其他结点，然后删除该结点.

Tao 等[3]基于 TPR-tree 提出了 TPR*-tree，其主要改进在于更新操作，它考虑到在每个树结点中的局部最优可能导致结构性能的严重退化，设计了基于全局最优机制的更新原则. 其思想是维持一个优先队列，在队列中存放各层所有节点因 MBR 和／或 VBR 扩展所导致的性能下降代价估计，并按照节点性能下降代价大小进行排序. 在插入新的移动对象记录时，路径选择递归算法从优先队列中选择 1 个具有最小性能下降代价的节点，直至找到底层叶节点终止. 由于优先队列是全局的，因此最终插入路径也是全局最优的. TPR*-tree 中还设计了一个概率估价模型，用于验证时空索引的性能，并且用这个模型对于任意的数据项分布的索引进行了最优性能的分析.

## 2　TPRA-tree 索引

### 2.1　问题分析

在 TPR-tree 中，假设有 4 个移动对象，在 0 时刻移动对象的分布如图 1 所示. 途中箭头表示该移动对象的速度方向；矩形框表示当前时刻 TPR-tree 中某个中间结点，它包含 3 个移动对象 1、2、3. 而当到了 1 时刻，移动对象的状态变成了图 2 所示的状态. 实矩形框表示 TPR-tree 中在当前时刻的某个中间结点. 观察此时的情形，发现由于移动对象 2 与移动对象 1、3 速度方向的差异，导致结点的面积增大，这样则不利于查询操作. 2 个虚矩形框则表示另外一种结点分布，这种结点分布充分考虑了移动对象运动的方向性，减少了结点面积，使查询更加便利.

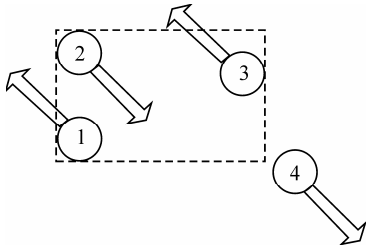


图 1　0 时刻的移动对象

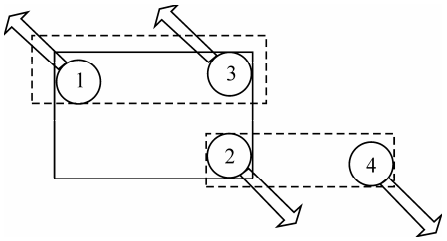


图 2　1 时刻的移动对象

### 2.2　TPRA-tree 索引结构

考虑移动对象的方向，笔者引入移动对象运动的方向角度，建立 TPRA-tree. TPRA-tree 沿用 TPR-tree 的索引结构，以二维平面对象为例，将平面 360°的范围分为某些数量的角度区间，将移动对象按速度方向的角度映射到对应的区间上，这些映射到同个区间上的移动对象构成了移动对象聚类. 然后在每个区间上建立相应的 TPR-tree. 为便于索引每个 TPR-tree，在内存中设立 1 个基于内存的线性链表结构用来存放每个 TPR-tree 的记录信息. 记录形式为四元组{TPR, AR, MBR, VBR}，TPR 指向对应 TPR-tree; AR 即为 Angle Range，为该 TPR-tree 的角度范围; MBR、VBR 分别指向该 TPR-tree 的空间域范围、速度域范围.

### 2.3　TPRA-tree 的索引创建、更新算法

TPRA-tree 索引创建算法首先是要确定角度区间的数目，划分好区间后，采用 R 树批装载技术以减少磁盘访问代价. 关于角度区间的数目多少是个要解决的问题，过少的数目无法充分利用移动对象在运动速度角度上的分布达到 TPRA-tree 查询和动态维护性能的最优；过多的数目则会造成移动对象运动方向角度更新频繁时动态维护性能的下降，且在查询过程中会使得查询的中间索引结

点过多，以致于增加 I/O 的次数.

TPRA-tree 索引的插入算法是首先扫描内存中的 TPRA-tree 线性链表中的 AR，以确定该移动对象方向角度落在那个 TPR-tree 中，并通过线性链表中 TPR 指针指向的根地址找到要插入到的 TPR-tree，具体算法可参照 TPR-tree 算法进行插入.

当删除某个对象时，同样扫描 TPRA-tree 线性链表，找到所在 TPR-tree，参照 TPR-tree 算法进行删除. TPRA-tree 的更新算法也采用先删除再插入的策略. 给定 TPRA-tree 的线性链表为 L，移动对象标识 oid，引入中间结点变量 tpr 指向相应的 TPR-tree，线性链表变量为a. 其具体的更新算法描述如下:

Input: oid, newangle, newMBR, newVBR, oldangle, oldMBR, oldVBR

Output: updated TPRA-tree

BEGIN

1. For all a∈L Do
2. If oldangle∈a.AR Then
3. tpr←a.TPR; break;
4. End If
5. End For
6. DeleteNode(tpr, oid, oldMBR ,oldVBR);
7. For all a∈L Do
8. If newangle∈a.AR Then
9. tpr←a.TPR; break;
10. End If
11. End For
12. InsertNode(tpr, oid, newMBR, newVBR);

END

上述算法中，1~5 为从内存中的线性链表中搜索到要更新的结点在那个对应的 TPR-tree 中；6 即是对该结点的删除；7~11 为从内存中的线性链表中搜索到要更新后的结点在那个对应的 TPR-tree 中; 12 即在对应的 TPR-tree 中插入.

### 2.4 更新算法性能分析

在该更新算法中，我们假设共有 $N$ 个移动对象，$n$ 个角度范围，移动对象的方向角度在[0,360)内均匀分布，则在每个 TPR-tree 中平均有 $N/n$ 个移动对象. 在删除操作时，首先通过该移动对象上一时刻的方向角度直接定位到对应的 TPR-tree，然后只在该 TPR-tree 上的 $N/n$ 个移动对象中查询到该移动对象，并进行删除操作，由于考虑到该 TPR-tree 中的移动对象方向角度在一定范围内，运动方向大体一致，则结点面积以及结点重叠面积大大减少，从而避免了一些不必要的结点查询，就嫩迅速找到该结点，进行删除操作. 最后通过需要更新移动对象新的方向角度定位到要插入的对应 TPR-tree 中, 同理插入过程中也只对 $N/n$ 个移动对象进行操作. 这样使得其索引性能大为提高. 当然 $n$ 值不能太大，否则 TPRA-tree 的查询性能将会由于查询过多中间索引结点而下降，将不会充分利用到由于考虑了方向角度而使得结点面积和结点重叠面积大大减少的优势.

## 3 实验

以 TPR-tree 源代码[8]为基础，并对其进行了适当修改，进而实现了 TPRA-tree. 实验数据通过上述 TPR-tree 的数据生成器来模拟移动对象的运动. 通过该生成器，模拟在范围 $10^4\times10^4$ km$^2$ 交通路网中模拟 100 K 个移动对象的运动. 移动对象速度在[15,100]之间均匀分布. 移动对象更新最大时间间隔为 100 s，即每个时间单元中约有 1000 个移动对象更新. 通过移动对象更新操作和查询操作所需要的平均 I/O 次数来衡量 TPRA-tree 和 TPR-tree 的性能. 在更新操作性能测试中, 采用每隔 10 K 次更新操作所需要的平均 I/O 次数来评估. 而在查询性能测试中, 同样采取 5 K 次更新操作后执行 20 次窗口查询的平均均 I/O 次数来评估，查询 q 空间窗口大小 qRlen 分别固定为 100×100, 200×200, 400×400, 800×800, 1200×1200, 1600×1600.

索引结构页面大小设置为 4 K 字节，并使用最近最少使用(Least Recently Used, LRU)缓存替代策略. 实验硬件环境是: Intel Pentium Dual E2200 的 CPU，内存为 1G; Windows XP 操作系统和 VS2005 集成开发环境. 笔者设定角度区间的数目为 8 进行实验.

图 3 显示了 TPRA-tree 与 TPR-tree 分别在更新操作中平均的 I/O 次数比较. TPRA-tree 更新性能没有随着时间变化而明显下降，表现出了很好的稳定性. TPRA-tree 考虑了移动对象运动的方向角度，其索引结点 MBR 扩张程度远小于 TPR-tree，减少

了结点重叠，从而使得更新性能更加稳定. 从图 4 可以看出，TPRA-tree 比 TPR-tree 查询性能要好. 虽然 TPRA-tree 在窗口查询中要在每个 TPR-tree 中去搜索，但由于减少了 MBR 之间的重叠，使得查询时可以减少搜索一些不必要的结点，从而提高了查询性能.

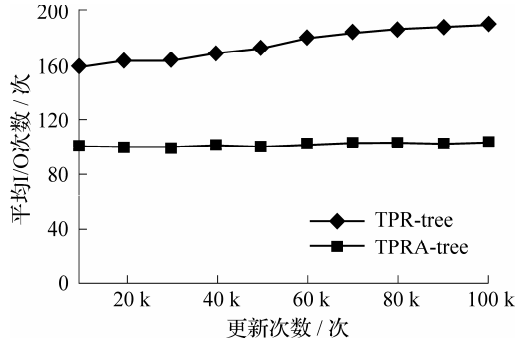


图 3　TPRA-tree 与 TPR-tree 更新性能比较

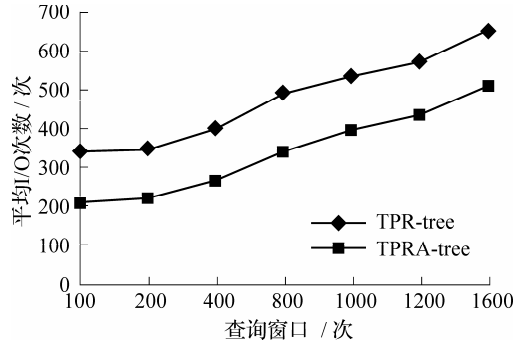


图 4　TPRA-tree 与 TPR-tree 查询性能比较

## 4　结论

出于对于 TPR-tree 存在的随着时间的推移性能恶化的考虑，基于移动对象速度方向的角度来划分区间进行移动对象的聚类，设计了 TPRA-tree，它减少了结点的面积和结点间的重叠，并且提高了更新和查询性能. 实验结果证明此点.

参考文献:

[1] Saltenis S, Jensen C, Leutenegger S, et al. Indexing the positions of continuously moving objects[C]//Proc. of the ACM SIGMOD International Conference on Management of Data, Dallas, 2000:331-342.

[2] Tao Yufei, Dimitris P, Sun Jimeng. The TPR*-tree: An optimized spatio-temporal access method for predictive queries[C]//Proceedings of the 29th VLDB Conference, Berlin, 2003:790-801.

[3] Sunil P, Xia Yuni, Dmitri K V, et al . Query indexing and velocity constrained indexing: Scalable techniques for continuous queries on moving objects[J]. IEEE Transactions on Computers, 2002, 51(10):1124-1140.

[4] Chon H, Agrawal D, Abbadi A. Storage and retrieval of moving objects[C]//Proceedings of the International Conference on Mobile Data Management, Hongkong, 2001: 173-183.

[5] Patel J, Chen Y, Chakka V. STRIPES: An efficient index for predicted trajectories[C]//Processdings of the International Conference on Management of Data, Paris, 2004: 635-646.

[6] Sioutas S, Tsakalidis K, Tsichlas K, et al. A new approach on indexing mobile objects on the plane[J]. Data & Knowledge Engineering, 2008, 67(3):362-380.

[7] 廖巍，唐桂芬，景宁，等. 基于速度分布的移动对象混合索引方法[J]. 计算机学报, 2007, 30(4):661-671.

[8] Marios H. Spatial index library[EB/OL]. [2008-05-22]. http://trac.gispython.org/spatialindex/wiki/Releases.

# TPRA-tree: An Improved Spatial-temporal Index for Predictive Queries

HE Mao-shun, DONG Yi-hong*, FU Shi-chang

( Institute of Computer Science and Technology, Ningbo University, Ningbo 315211, China )

**Abstract:** TPR-tree is a typical method of indexing moving objects whose performance continue to deteriorate as the time goes on. In this paper the TPRA-tree is proposed based on the TPR-tree. TPRA-tree, which is derived from the angle of the velocity, is presented for moving objects with frequent updates. It reduces the area and overlap of nodes. The experimental results show that TPRA-tree's update and query performance outperform the TPR-tree.

**Key words:** moving object; TPR-tree; TPRA-tree; spatial-temporal index

（责任编辑　章践立）