第34卷 第10期 计 算 机 工 程 2008年5月
Vol.34 No.10 Computer Engineering May 2008
·软件技术与数据库· 文章编号：1000—3428(2008)10—0040—03 文献标识码：A 中图分类号：TP311.13

移动对象的动态反向最近邻查询技术

李 松[1]，郝忠孝[1,2,3]
(1. 哈尔滨理工大学计算机科学与技术学院，哈尔滨 150080；2. 齐齐哈尔大学计算机与控制工程学院，齐齐哈尔 161006；
3. 哈尔滨工业大学计算机科学与技术学院，哈尔滨 150001)

摘 要：为了处理移动对象的动态反向最近邻，对时空动态反向最近邻查询问题进行形式化的定义，利用时空距离函数及限界区域等概念给出计算移动对象的动态反向最近邻的定理与算法，提出移动查询点的动态最近邻的全域查询及局域查询的方法，利用动态检测圆及时空距离函数进行动态反向最近邻的查询判断，其计算量可减少40%~60%。构建新的时空索引结构——TP^{RDNN}树，给出操作TP^{RDNN}树的查询算法。

关键词：动态反向最近邻；六分区域；距离函数；TP^{RDNN}树

Query Technology of Dynamic Reverse Nearest Neighbors for Moving Object

LI Song[1], HAO Zhong-xiao[1,2,3]
(1. School of Computer Science and Technology, Harbin University of Science and Technology, Harbin 150080; 2. School of Computer and Control Engineering, Qiqihar University, Qiqihar 161006; 3. School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001)

【Abstract】 To solve the dynamic reverse nearest neighbor queries for the moving objects, the definition and the primary properties of the reverse nearest neighbors are proposed and the theorems and arithmetics for computing the dynamic reverse nearest neighbors using the conceptions of the distance-function and delimitation-regions are developed. To get the dynamic reverse nearest neighbors, the methods based on the distance-function and the dynamic inspect-circle are proposed. The amount of the calculation can be reduced by 40%~60%. To solve this question in the spatio-temporal database, a new spatio-temporal index tree——TP^{RDNN} tree is presented and the arithmetics for querying the new tree are introduced.

【Key words】 dynamic reverse nearest neighbors; six-dividing regions; distance-function; TP^{RDNN} tree

反向最近邻查询[1-4]已广泛应用于地理信息系统、移动计算和图像处理等领域，如何及时准确地回答时空移动对象发出的动态反向最近邻查询请求成为了新的研究热点和难点。

1 基本定义与定理

定义 1 设有一个d维动态数据集P和一移动查询点q，时空反向最近邻(RNN)查询就是在某时刻t_i或某一时间段$[ts,te]$内找出P的子集$RNNs(q)$，即$RNNs(q)=\{r\in P \mid \forall p\in P: D(q,r)\le D(r,p)$，$t=[ts,te]$ 或$t=t_i\}$。$D(q,r)$为q和r之间的最短距离。

定义 2 在二维空间里设有一查询点$q=(x_q,y_q)$，其中，x_q和y_q分别是其横坐标值和纵坐标值，3条直线相交于点q，且其中的一条直线平行于横轴，它们相交的角度均为60^o，则此3条直线将q点周围的空间分成6个区域$S_1,S_2,\cdots,S_6$。将这3条直线称为空间分割线，6个区域称为六分区域，如图1所示。

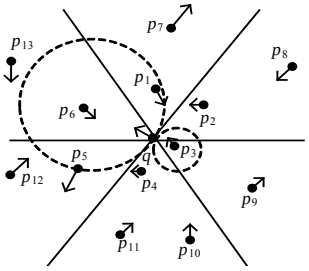

图1 限界区域及检测圆

定理 1[1] 给定任意一个二维数据空间的数据集，对任意一查询点q，$RNNs(q)$中最多包含6个数据点。

定理 2[1] (1)每个六分区域最多有2个反向最近邻。(2)如果某个六分区域正好有2个反向最近邻，则这2个反向最近邻必在该区域的边界线上。

定理 3[1] 假设p是查询点q的最近邻，且p在某个六分区域S_i中，如果p不在该区域的边界线上，则要么q也是p的最近邻(即p是q的反向最近邻)，要么q在该区域中没有反向最近邻。

2 利用时空距离函数及限界区域进行的计算

本节利用时空距离函数(见图 2)和限界区域处理移动对象的动态反向最近邻问题。

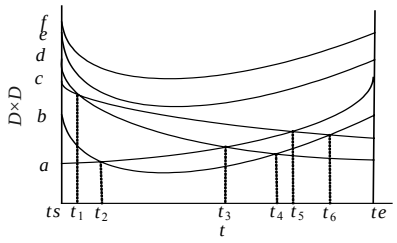

图2 时空距离函数

2.1 移动对象动态最近邻的计算

基金项目：黑龙江省自然科学基金资助项目(F200601)
作者简介：李 松(1977－)，男，博士研究生，主研方向：数据库理论及应用；郝忠孝，教授、博士生导师
收稿日期：2007-05-30 **E-mail：**lisongbeifen@163.com

对时空动态对象先进行全域查询，再根据全域查询的结果进行局域查询。两阶段查询的结果可联动性地用于动态反向最近邻的判断。

定义 3 以移动查询点或动态对象为圆心，预设的时空距离为半径的圆域称为限界查询区域。

定义 4 查询移动对象的 *RNN* 时，先在限界查询区域内查询 q 的μk 个最近邻称为全域查询。进行全域查询所得的 q 的最近邻称为全域最近邻。k 称为全域查询常数，根据定理 1 可设 k 为 6 的倍数，预先设定的整数μ称为全域查询因子。

定义 5 根据时间段内的全域查询情况，随时间变化更新限界查询区域再进行的查询称为变域查询。

定义 6 经过全域查询及变域查询后，对移动对象 q 进行某个六分区域内的时空最近邻查询称为局域查询。进行局域查询所得的 q 的最近邻称为局域最近邻。

定义 7 由全域查询及局域查询所得的 q 的动态最近邻所组成的集合称为 q 的动态反向最近邻的候选集。经过全域查询及局域查询后，q 的所有六分区域内的最近邻已全部确定，则由这些最近邻组成的候选集称为 q 的动态反向最近邻的完备候选集。

定义 8 由将来时间内最有可能成为 q 的动态反向最近邻的动态对象组成的集合称为临界集。

定义9 利用时空距离函数曲线计算移动对象的动态最近邻时，其k个最近邻次序发生变化的时间点称为时间分裂点，2个相邻时间分裂点之间的时间段称为时间分裂段。如图2所示，$t_1,t_2,\cdots,t_6$称为时间分裂点，时间段$[ts,t_1],[t_1,t_2],\cdots,[t_6,te]$称为时间分裂段。

由定理 1~定理 3 可得出筛选策略如下：

筛选策略 1 若 q 的某一个六分区域内仅含有 q 的 k 个动态最近邻中的一个，则将此动态最近邻直接加入 q 的动态反向最近邻的候选集。

筛选策略 2 若 q 的 k 个动态最近邻中有 $m(m\geq 2)$个动态对象在同一六分区域内，则将此 m 个动态对象中离 q 最近的点加入到 q 的动态反向最近邻的候选集中。其他的点加入到临界集。

筛选策略 3 若 q 的 k 个动态最近邻集中在 $s(s\leq 3)$个六分区域内，则利用时空距离函数曲线计算出第 k+1 到第 $2k$ 个最近邻，再依据筛选策略 1 和筛选策略 2 进行处理。依次进行，根据预设的查询因子μ确定需要计算的动态最近邻个数的最大值μk。

筛选策略 4 经过筛选策略 1~筛选策略 3 的判断，若仅剩 $p(p\leq 2)$个六分区域内还没确定 q 的动态最近邻，则进行局域查询。

由筛选策略可得出求移动对象 q 的六分区域内的动态最近邻算法如下：

算法 1 MOVEQ_NNs_six

输入 移动查询点 q，一组动态对象集 O，时间段[ts,te]

输出 q的动态最近邻集K_i，K_i形如$([t_1,t_2],p_1,p_2,\cdots,p_{uk})$；反向最近邻的候选集$T_i$, T_i形如$([t_1,t_2],p_1,p_2,\cdots,p_k, k\leq 6)$；动态对象的临界集C

```
begin
初始化：Ki=Φ, Ti=Φ, C=Φ, μ←1, i←1, j←1；
1：计算全域最近邻，将查询结果加入Ki集；
2: for oj ∈ Ki do
   if oj ∈ Ki and oj ∈ Sk and(Ki-oj) ∉ Sk then
      Ti←oj；Ki←Ki-oj；
      j←j＋1；k←k+1；转 2；
   elseif (min(dds(q, oj))=true and oj ∈ Sk
         and (om ⊂ (Ki –oj)) ∈ Sk  then
      C←om；Ki←Ki-oj -om；
      j←j+1+m；k←k+1；转 2；
   if 一个时间分裂段已经处理完 then
i←i＋1；转 2；
   if s≤3 then
      μ←μ＋1，转 1；
   elseif p≤2 then
计算q的动态局域最近邻，将结果加入Ki集；
    return Ki , Ti；
end
```

该算法无须对所有的动态对象进行计算，在查询时间段内根据筛选策略及全域查询因子对局部的动态对象进行筛选判断，可将少数的最有可能成为动态反向最近邻的候选对象找出来，剔除了大量无用对象。

2.2 查询移动对象的动态反向最近邻

2.2.1 利用动态检测圆进行判断

利用动态检测圆(如图 1 所示)查询移动对象 q 的动态反向最近邻的算法如下：

算法 2 inspect_RNNs

输入 动态反向最近邻的候选集 T，动态对象的临界集 C，时间段[ts,te]

输出 q 的动态反向最近邻集 RK 和更新后的临界集 C

```
begin
for Ti ∈ T do
   for pi ∈ Ti do
      call Circle(pi,| Dq,pi(t)|)
      if tx ∈ [ti,tj] and pj in Circle(pi,| Dq,pi(t)|)
         C←pi；
      else RK←pi；
      更新动态检测圆；
return RK；
end
```

2.2.2 利用时空距离函数进行判断

利用时空距离函数进行动态反向最近邻的二次判断算法如下：

算法 3 search_RNNs

输入 由 MOVEQ_NNs_six 算法所得的候选集 T；动态对象的临界集 C；查询时间点 ts 或时间段[ts,te]

输出 q的动态反向最近邻集RK, RK形如$[(t_1,t_2), p_1,p_2,\cdots, p_j, j\leq 6]$的一组数据

```
begin
1：for Ti ∈ T do
2：  for pj ∈ Ti do
3：     if ts ∈ [ti,tj] and te ∈ [ti,tj] then
          if q是pi的第 1 个最近邻then
             RK←pi；
          else
             C←pi；
        elseif ts ∈ [ti,tj] and te ∈ [tj,tx]
          在[ts,tj]时间段内，转 3；
          在[tj,te]时间段内，i←i+1，转 1；
return RK；
end
```

经过理论分析，利用时空距离函数及限界区域进行计算，可将计算量减少 40%~60%。

3 时空索引结构

为了进行时空索引，本节在RDNN树[3]基础上构建了新的时空查询树TP^{RDNN}树。和TPR树类似，它在节点中增加了时空距离信息以查询动态反向最近邻。TP^{RDNN}树的叶子节点包含形如(*ptid*, *dnn*, *moveptr*)的记录，其中，*ptid*表示数据集*S*中的一个*n*维数据点；*dnn*表示该点到其最近邻的距离值；*moveptr*指向所对应的动态对象的详细信息；非叶子节点包含形如(*pcaddr*, *mrect*, *max_dnn*)的一组记录，其中，*pcaddr*是孩子节点的地址；*mrect*是指向其孩子节点内所有最小外包矩形的最小外包矩形；*max_dnn*=max{*dnns*(*p*)}，*p*是以*pcaddr*指向的节点为根的子树内的点。

本节给出基于TP^{RDNN}树的算法来进行动态反向最近邻的查询，查询算法如下：

算法 4 RNNS_SEARCH

输入 TP^{RDNN}树；一个移动查询点q

输出 时间段里 q 的动态反向最近邻

```
begin
1：设定：动态反向最近邻候选集 O，动态反向最近邻集 O′；
2：if 节点是叶子 then
      if O 集不是完备候选集 then
          call MOVEQ_NNs_six
          更新候选集 O；
      (A)：call search_RNNs；
          更新动态反向最近邻集 O′；
      elseif O 集是完备候选集 then
              转 (A)；
3：elseif 节点是一个内部节点 then
      执行剪枝策略；
      依据最短距离排序节点项；
      递归调用 RNNS_SEARCH；
end
```

根据动态对象的运动情况，TP^{RDNN}树需进行周期性重建。利用TP^{RDNN}树进行查询的效率很大程度上取决于剪枝策略及MOVEQ_NNs_six和 search_RNNs算法的效率，其查询性能和动态对象的运动状况有关。

4 结束语

本文利用时空距离函数及限界区域的特点对移动对象的动态反向最近邻进行了系统分析。基于筛选策略和构建的新的时空查询树TP^{RDNN}树，本文的方法可有效提高时空数据库对动态反向最近邻查询的处理能力。下一步的研究重点主要集中在 2 个方面：(1)高维空间中时空移动对象的动态反向最近邻的研究；(2)时空移动对象的近似动态反向最近邻的查询算法研究。

参考文献

[1] Stanoi I, Agrawal D, Abbadi A E. Reverse Nearest Neighbor Queries for Dynamic Databases[C]//Proceedings of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery. Dallas, USA: [s. n.], 2000.

[2] Achtert E, Bohm C, Kroger P, et al. Efficient Reverse K-nearest Neighbor Search in Arbitrary Metric Spaces[C]//Proc. of International Conference on Management of Data. Chicago, USA: [s. n.], 2006.

[3] Yang Congjun, Lin K I. An Index Structure for Efficient Reverse Nearest Neighbor Queries[C]//Proceedings of the 17th International Conf. on Data Engineering. Heidelberg, Germany: IEEE Computer Society, 2001.

[4] Tao Yufei, Yiu M L, Mamoulis N. Reverse Nearest Neighbor Search in Metric Spaces[J]. IEEE Trans. on Knowledge Data Eng., 2006, 18(9): 1239-1252.

(上接第 39 页)

法快。由于 *D* 中满足最小支持度为 0.7 的频繁项集数量很少，因此 PFUP 算法和 FUP 算法的执行时间几乎相等。当支持度为 0.2 时，*D* 的频繁项集数目最大，PFUP 算法的执行时间与 FUP 算法时间之差最大。因此，PFUP 算法适用于稠密型数据库。在该类数据库的关联规则更新方面，PFUP 比 FUP 具有更小的时间复杂度。

图 2 是在最小支持度为 0.5 的基础上，通过测试数据库观察 PFUP 与 FUP 算法的执行时间情况。当测试数据库变大时，FUP 算法执行时间的增长速度远大于 PFUP 算法。因此，PFUP 比 FUP 更适用于大型数据库的关联规则更新。

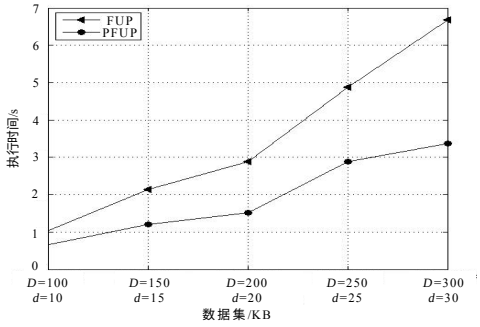

图 2 测试数据库中算法执行时间比较

5 结束语

本文探讨了关联规则的增量式更新，提出了一种改进的关联规则增量式算法 PFUP。与 FUP 算法相比，PFUP 减少了候选项集的数量及扫描数据库 *D* 或 *d* 的次数，从而减少了时间复杂度。理论和实验表明，PFUP 算法是有效和实用的。今后可以结合已有的关联规则优化算法将优化措施应用到 PFUP 算法中，产生更高性能的关联规则增量更新算法。

参考文献

[1] Cheung David, Han Jiawei, Vincent T N, et al. Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique[C]//Proc. of the 12th Int'l Conf. on Data Engineering. New Orleans, Louisiana, USA: [s. n.], 1996.

[2] 朱玉全, 孙志挥, 赵传申. 快速更新频繁项集[J]. 计算机研究与发展, 2003, 40(1): 94-99.

[3] Ayan N F. An Efficient Algorithm to Updating Large Itemsets with Early Pruning[C]//Proc. of the 5th Int'l Conf. on KDD'99. San Diego, California, USA: [s. n.], 1999.

[4] 朱红蕾, 李 明. 一种高效维护关联规则的增量算法[J]. 计算机应用研究, 2004, 21(9): 107-109.

[5] 付长贺, 赵传立, 唐恒永. 一种改进的关联规则增量式更新算法[J]. 沈阳师范大学学报, 2006, 24(1): 51-54.