第 37 卷 第 19 期
Vol.37 No.19
计 算 机 工 程
Computer Engineering
2011 年 10 月
October 2011
• 软件技术与数据库 •
文章编号：1000—3428(2011)19—0076—03 文献标识码：A 中图分类号：TP301.6

移动对象最近邻查询实时更新算法

周宗毅，杜忠军
(四川大学计算机学院，成都 610065)

摘 要：在移动对象最近邻查询实时更新算法中，安全区一般是难以求解的不规则凸多边形。针对该问题，将安全区设计为圆形，在此基础上提出一种效率更高的移动对象最近邻查询更新算法。将算法分别运行于基站上的最近邻管理系统和移动对象上的 Client 管理系统，实验结果表明，该算法可以降低系统的存储代价，提高其数据处理性能。

关键词：安全区；最近邻；移动对象；最近邻管理系统；实时更新

Real-time Update Algorithm for Nearest Neighbor Query of Moving Object

ZHOU Zong-yi, DU Zhong-jun
(College of Computer Science, Sichuan University, Chengdu 610065, China)

【Abstract】 In real-time update algorithm of nearest neighbor query of moving objects, safe area is always an irregular convex polygon which is difficult to solve. To address the problem, this paper designs safe area as a circular area, proposes a real-time update algorithm for nearest neighbor query of moving object, and takes it run on Nearest Neighbor Management System(NNMS) which works on fixed base station and Client Management System(ClientMS) which works on moving object. Experimental results show that the algorithm reduces storage costs of the system and improves data processing performance of the system.

【Key words】 safe area; nearest neighbor; moving object; Nearest Neighbor Management System(NNMS); real-time update

DOI: 10.3969/j.issn.1000-3428.2011.19.024

1 概述

移动对象最近邻查询实时更新问题是时空数据的研究重点之一，其查询更新效率对实际应用有十分重要的影响。最近邻查询实时更新算法可以分为 2 类：

(1)基于历史轨迹预测下一时刻移动对象位置的最近邻实时更新算法[1]。该算法对预测函数有较强的依赖性，不同的移动对象通常需要不同的预测函数。其优点是更新效率高、无延迟，缺点在于移动对象的工作负担大，系统不稳定。

(2)基于索引的最近邻实时更新算法[2]，为最近邻查询算法要用到的数据建立索引。该算法的优点是与移动对象的轨迹无关，具有稳定的查询结果，查准率较高，移动对象工作负担小。缺点是有延迟，存在较强的通信依赖性。

本文通过解决以下 2 个核心问题：(1)间隔Δt 时间后，移动对象如何实时更新最近邻；(2)如何设定Δt 的长短，充分利用 2 种算法的优点同时避免其缺点，设计了最近邻管理系统(Nearest Neighbor Management System, NNMS)和 Client 管理系统(Client Management System, ClientMS)一起实现最近邻查询功能。

2 相关概念

定义 1(欧几里德距离[3]) 欧几里德距离 $dist(M_i, M_j)$表示空间中点 $M_i(x_i, y_i)$ 与 $M_j(x_j, y_j)$ 之间的距离：

$$dist(M_i,M_j)=\sqrt{(x_i-x_j)^2+(y_i-y_j)^2}$$

定义 2(最近邻 NN)[4] 查询对象 q 在查询数据集 N 中的最近邻定义为 N_i，N_i 在时刻 t 满足如下条件：

$$NN(q)=\{N_i, dist(q, N_i)\leqslant dist(q, N_j), N_i\in N, N_j\in N\backslash\{N_i\}\}$$

定义 3(安全区 SA[5]) 移动对象 q 的安全区是指 q 在区域内移动时，关于 q 的 NN 不会改变。

定义 4(圆形安全区 CSA) N 是移动对象 q 的查询集，A 为能够包含查询集 N 的分布范围和移动对象可能的活动范围的最小矩形，则存在以 q 为圆心，d 为半径的圆形安全区，表示为 $CSA(q, d)$，$CSA(q,d)\subseteq A$。

定理 1 平面上任意 2 点 a、b 的垂直平分线为 L，任意查询点 q 与点 b 的连线与 L 相交，则 $dist(q,b)>dist(q,a)$；任意查询点 q'与点 b 的连线与 L 不相交，则 $dist(q',b)<dist(q',a)$。

证明：如图 1 所示，如果任意一点 q 与 b 的连线和直线 L 相交，则 q 在直线 L 靠 a 的一边，由于 L 为 a、b 的垂直平分线，因此 $dist(q,b)>dist(q,a)$。如果任意一点 q'与 a 的连线 HE 直线 L 不相交，则 q'在直线 L 靠 b 的一边，由于 L 为 a、b 的垂直平分线，因此 $dist(q',b)<dist(q',a)$。

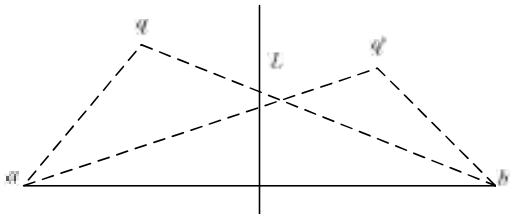

图 1 垂线距离示意图

定理 2 设空间中两点 a、b 的垂直平分线为 L，查询点 q 的初始位置为 $q_0(x_0, y_0)$，经过Δt 时间后 q 的位置为 $q_1(x_1, y_1)$，

作者简介：周宗毅(1986－)，男，硕士，主研方向：数据库技术，ERP 系统；杜忠军，副教授

收稿日期：2011-03-13 **E-mail：**308834809@qq.com

如果 $Segment(q_0,q_1)$ 与 L 相交，则 q 分别到 a、b 两点的最近点发生变化；否则，q 分别到 a、b 两点的最近点不变。

证明：定理 2 的证明与定理 1 类似。

定理 3 移动对象 q 必然存在最大圆形安全区 $maxCSA(q, d)$。$maxCSA(q, d)$满足以下 2 个条件：

(1)$maxCSA(q, d)$是移动对象 q 的圆形安全区。

(2)对任意 $d'>d$，存在点 $n \in Segment(q,d')$，当移动对象 q 移动到点 n 时，最近邻发生改变。

证明：如图 2 所示，不失一般性，设矩形活动区域 A 为移动对象的移动范围，A 包含 q 的所有查询集 N 和移动对象的活动范围。作最近邻居与其他所有点连线上的垂直平分线 L_1、L_2、L_3、L_4，则必然存在最大半径 d，d 是移动对象 q 到 n 条实线以及矩形区域 A 的 4 条边的距离中的最小值。设 q 与最近垂线的垂点为 c，根据定理 2，在以最近邻点 NN 为圆心、d 为半径的圆内，q 没有穿过任何一条垂直平分线，因此，最近邻没有发生改变。若任意 $d'>d$ 在从点 q 到点 c 的延长线方向上与 c 点距离为 $d'-d$，则必然存在点 f。因为 $d'-d>0$，所以 f 必然在相应垂线的另外一侧，同样由定理 2 可知，在 f 点 q 的最近邻发生变化。因此，d 为 q 的最大圆形安全区的最大半径，表示为 $maxCSA(q, d)$。

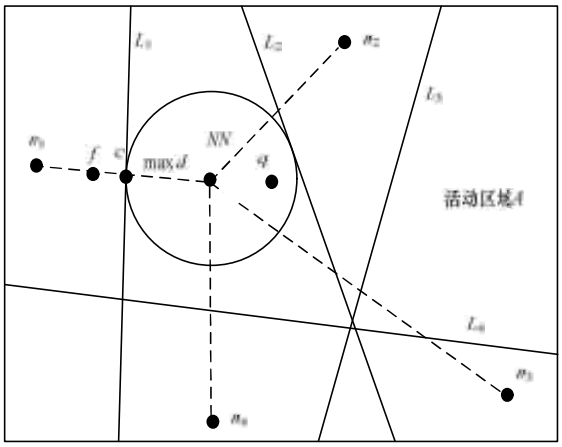

图 2　最大圆形安全区示意图

3　移动事物模型

本文针对 NNMS 设计了移动对象最近邻查询实时更新模型，如图 3 所示。

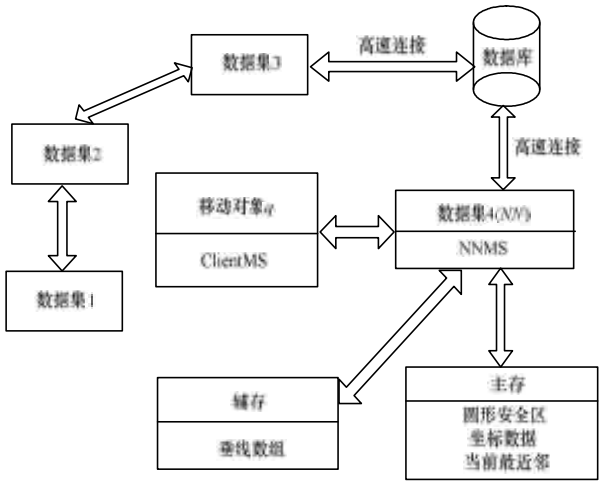

图 3　移动对象最近邻查询实时更新模型

数据库中存放了所有 Dataset 的位置等相关信息。移动对象具有简单的任务处理能力及很小的存储空间。NN 实质上也是一个 Dataset(与 n_1、n_2、n_3、n_4 相同)，当其与移动对象最近时，载入之前 NN 中关于该移动对象的相关信息。移动对象 q 是在空间中移动需要查询最近 Dataset 的对象，在空间中任意移动的同时保持固定时间间隔后与最近邻建立通信(或以触发安全边界的方式触发通信)。移动对象经过固定时间间隔后，调用 ClientMS 自检，判断当前位置是否处于安全区域之外：

(1)如果不在安全区或者在安全区边界上，则向最近邻发送 Message(新坐标)，等待接收更新后的最近邻以及安全区。需要注意的是，最近邻改变后，与 NNMS 相关的存放在之前最近邻数据集上的主存中的以及辅存中的内容一并复制到新的最近邻中。

(2)如果在安全区，同样向最近邻发送 Message(新坐标)，此时不需要等待更新移动对象就可以继续在空间中移动，当 NNMS 处理完毕后再发送更新后的安全区至移动对象即可。在此 NN 未改变，所以，NNMS 不需要复制。

4　移动对象最近邻查询实时更新算法

Moving Object 端的主要功能是接收来自固定终端的处理结果。运行在移动对象上的 ClientMS 其主要功能是根据检索点的最新位置坐标判断是否超出安全区。若移动对象没有超出其安全区，则向 NNMS 发送新坐标信息；若移动对象超出安全区，则向 NNMS 发送申请更新 NN 以及安全区发请求。

Dataset 端能够高速访问数据库中存储的所有终端的位置信息，当前最近邻端(Dataset)运行 NNMS 以响应 ClientMS 发送的请求。NNMS 中的核心算法为 NNUpdate，其功能是根据 q 的当前坐标以及最近一次更新位置的坐标来查询 NN。NNUpdate 首先根据 ClientMS 发送的不同请求分别进行处理：如果是请求更新 CSA(第 3~第 6 行)，调用 GetCSA 算法获得圆形安全区，将更新后的 CSA 发送至 ClientMS 后返回；否则，更新 NN(第 8~第 15 行)，Pointk 数组中读入除最近邻以外的所有 Dataset 信息。Linek 数组中读入最近邻与 Pointk 中所有点的垂线。TemPointk 用于存放所有 q 点越界的对应点。根据定理 2，只有 TemPointk 数组中的点才有可能是新的最近邻。For 循环(第 11~第 13 行)调用 JudgeISL 函数判断 q 点的轨迹线段 $Segment\ s=(q_0,q_1)$是否与 Linek 数组中的垂线相交，如果相交，那么与这条垂线相关的 Dataset 点可能为最近邻，否则，与这条垂线相关的 Dataset 点不可能为最近邻。此时得到一个包含较少 Dataset 的数组 TemPointk，调用距离查询函数 QNN()找出存在于 TemPointk 数组中的最近邻。

NNMS 系统中的核心算法如下：

```
Algorithm NNUpdate(point q1, int k)
1   Segment s =(q0,q1);
2   If(Moving Object Apply for new CSA)
3   //移动对象更新 CSA
4    {   SA sa; TemPointk[k]=emptyInt j=0;
5        Sa=GetCSA(q1, line[k]);
6        Send message(Sa);   }
7   Else
8   //移动对象更新 NN
9    {   Pointk[k]= read the array of k point except nn;
10       Linek[k] =read the array of k Perpendicular;
11       For(i=0;i<k;i++)
12  //q0 为移动对象初始位置，q1 为移动后的位置
13       If(JudgeISL (s(q0,q1),linek[i]) )
14            {   TemPointk[j++]=Pointk[i];}
```

```
15          nn=QNN(q,TemPointk[],j);
16          Send message(nn);            }
17       Return;
Algorithm    int GetCSA(point q1,line linek[k])
18     Return the minimum distance from q1 to each line in array linek
Algorithm Boolean JudgeISL(segment s,line    TemLine)
19    If segment s and line TemLine intersected
20       Return TRUE
21    Else
22     Return FALSE
Algorithm point QNN (point q,point TemPointk[],int j);
23      //Return the point which is the closest point to q in //TemPointk;
24    Int MiniDist=Maxdist;    Point temp;
25     For each p in TemPointk
26      If dist(q, p)< MiniDist
27              { MiniDist = dist(q, p); Temp=p;}
28    Next      Return temp;
```

NNUpdate 算法要求内存中时刻存放圆形安全区、位置坐标和最近邻这 3 个数据，并且要在辅助存储器中保存最近邻与其余 Dataset 的垂线集，设每条线需要的存储空间为 d，则其空间复杂度为 $O(kd)$，时间复杂度为 $O(k)$。由于算法运行在固定终端上，因此以上时空复杂度对硬件的要求非常低。

5 实验分析与比较

实验的软硬件平台：IntelCorei7 四核处理器，4 GB 内存，500 GB 硬盘，Win7 操作系统，Visual Studio 2010 编程环境，C#编程语言，SQL Server 2008 数据库系统。使用 C#开发的 NNMS 和 ClientMS 系统分别运行在多台电脑上模拟查询环境。ClientMS 系统在经过初步的数据处理后向 NNMS 系统发送的请求分为 2 类：一类要求更新最近邻，另一类则不需要。当最近邻更新事件发生时，NNMS 开始查询最近邻，在此过程中处理器对存储器的访问主要包括对查询集的访问和垂线集索引的访问，实验结果如图 4 所示。

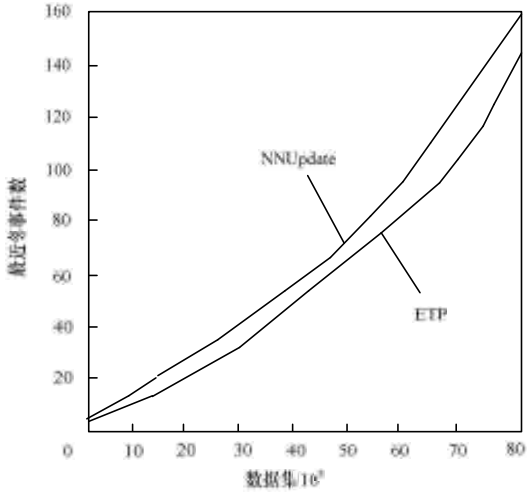

图 4 数据集与最近邻事件数的关系

NNUpdate 算法由于查询集不同，其固定时间内最近邻事件数变化较为明显，图 5 给出最近邻事件数从 10 变化到 80 时磁盘访问量的变化情况。并将 NNMS 的 NNUpdate 算法与文献[6]中的 ETP 算法进行了比较。

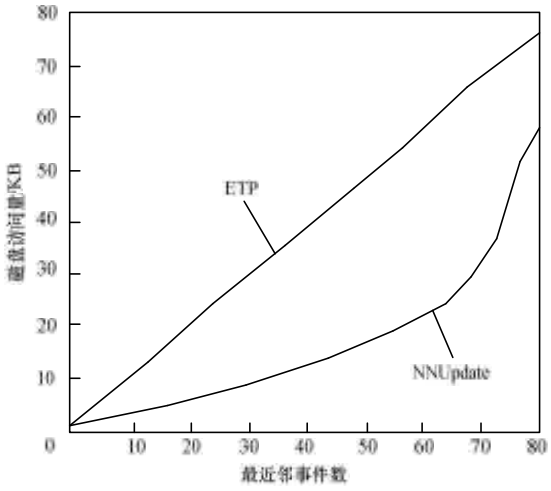

图 5 最近邻事件数与访问量的关系

图 4、图 5 的实验结果表明，NNUpdate 算法在相同数据集的前提下具有更低的事件触发频率，这就意味着更新的有效频率更高，且 NNUpdate 算法对于相同的最近邻更新处理，所需的磁盘访问请求更少，由此提高了系统的处理性能。

6 结束语

本文设计了易于求解的圆形安全区替代传统的不规则凸多边形安全区，从而降低了存储代价及通信数据量。将其用于 NNMS 系统和 ClientMS 系统，获得了不错的实验结果。虽然本文算法的设计是建立在二维平面空间上，但是算法可以推广到 N 维空间。比如推广到三维空间，那么圆形安全区就变成一个球形安全区，对跨区的判断就变成判断是否越过球面。另外还可以对 K 最近邻查询进行改进，通过同时对 K 个最近邻建立垂线索引集，分别判断 K 个最近邻是否越区，同样可以取得较好的效率。

参考文献

[1] Luo Yanmin, Chen Hanxiong. Efficient Methods in Finding Aggregate Nearest Neighbor by Projection-based Filtering[C]// Proc. of the International Conference on Computational Science and Its Applications. Kuala Lumpur, Malaysia: [s. n.], 2007.

[2] Kwon D, Lee S, Choi W, et al. An Adaptive Hashing Technique for Indexing Moving Objects[J]. Data & Knowledge Engineering, 2006, 56(3): 287-303.

[3] 欧几里得距离[EB/OL]. [2010-11-07]. http://baike.baidu.com/view/2946885.htm.

[4] 孙冬璞, 郝忠孝. 移动对象历史轨迹的连续最近邻查询算法[J]. 计算机工程, 2009, 35(1): 52-54.

[5] Hasan M, Cheema M A, Lin Xuemin, et al. Efficient Construction of Safe Regions for Moving KNN Queries over Dynamic Datasets[C]//Proceedings of SSTD'09. Heidelberg, Germany: [s. n.], 2009: 373-379.

[6] Iwerks G S, Samet H. Maintenance of Spatial Semi-join Queries on Moving Points[C]//Proceedings of the 30th International Conference on Very Large Data Bases. McLean, Virginia, USA: [s. n.], 2004.

编辑 张 帆