第35卷 第19期 计 算 机 工 程 2009年10月
Vol.35 No.19 Computer Engineering October 2009
·软件技术与数据库· 文章编号：1000—3428(2009)19—0081—02 文献标识码：A 中图分类号：TP311.13

基于 Voronoi 图的反向最近邻查询

刘润涛[1]，张佳佳[2]
(1. 哈尔滨理工大学信息与科学计算技术研究所，哈尔滨 150080；2. 哈尔滨理工大学应用科学学院，哈尔滨 150080)

摘 要：为了解决反向最近邻查询问题，利用 Voronoi 图及数据集中点的凸包进行反向最近邻查询，通过判断查询点与凸包的位置关系，可去除大量的数据点，并且给出在数据点被加入或删除后，对查询点的反向最近邻变化情况的判断方法与算法。为了便于查询，设计相应的空间存储数据结构。比较分析表明，该方法在处理多个查询点的反向最近邻时有一定的优势。

关键词：反向最近邻；Voronoi 图；凸包

Reverse Nearest Neighbor Search Based on Voronoi Diagram

LIU Run-tao[1], ZHANG Jia-jia[2]
(1. Institute of Information and Scientific Computing Technology, Harbin University of Science and Technology, Harbin 150080;
2. College of Applied Science, Harbin University of Science and Technology, Harbin 150080)

【Abstract】 In order to solve Reverse Nearest Neighbor(RNN) search in a dataset, Voronoi diagram and the hulls of dataset are used for RNN search. By determinating the location relation between query point and convex hulls, many points can be cut. And an algorithm and judgment method for the change of the RNN of query point are given when data points are added or deleted. To be convenient to search, corresponding spatial storage structures are designed. Comparative analysis shows that this method has evident advantages when many query points are dealt with.

【Key words】 Reverse Nearest Neighbor(RNN); Voronoi diagram; convex hull

1 概述

反向最近邻(Reverse Nearest Neighbor, RNN)查询是在数据集中找到以查询点为最近邻的对象点，它可被应用到知识发现、决策支持、设施定位、地理信息系统和多媒体数据库等多种领域。反向最近邻查询问题在国内外是一个较新的研究课题，已有的方法主要是利用 R 树及其改进树建立索引结构进行处理[1-2]，然而受 R 树结构的限制，在维数增加时，其性能急剧恶化，这些方法一般需要复杂的索引结构且不适用于处理曲面等非欧空间上的数据对象点。

为了解决上述问题，本文利用 Voronoi 图及凸包的性质进行反向最近邻查询。

2 相关定义及定理

定义 1(反向最近邻查询)[1] 设 $D(q,p)$ 为 2 点 p 和 q 之间的距离，假设有 1 个数据集 S 和查询点，数据集中 RNN 查询就是找出 S 的子集 $RNNs(q)$，即

$$RNNs(q)=\{r\in S \mid \forall p\in S: D(q,r)\le D(r,p)\}$$

定义 2(Voronoi 图)[3] 给定平面上 n 个点的点集 S，$S=\{p_1,p_2,\cdots,p_n\}$。定义 $V(p_i)=\bigcap_{i\neq j}H(p_i,p_j)$，即 $V(p_i)$ 表示比其他点更接近 p_i 的点的轨迹是 $n-1$ 个半平面的交，它是一个不多于 $n-1$ 条边的凸多边形域，称为关联于 p_i 的 Voronoi 多边形或关联于 p_i 的 Voronoi 域。对 S 中的每个点都可做一个 Voronoi 多边形，这样 n 个 Voronoi 多边形组成的图称为 Voronoi 图，记为 $Vor(S)$，如图 1 所示。且称共享一条棱的 2 个 Voronoi 多边形为邻接 Voronoi 多边形。

定理 1 若查询点 q 位于以 p_i 为中心的 Voronoi 多边形内，则离 p_i 最近的点为查询点 q。

推论 1 查询点 q 的反向最近邻必包括 p_i，且查询点 q 的反向最近邻的候选集只有与以 p_i 为中心的 Voronoi 多边形邻接的 Voronoi 多边形的中心。

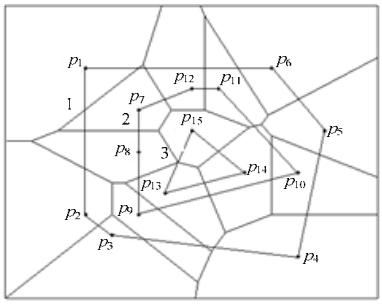

图 1 Voronoi 图、邻接多边形及凸包

定理 2 若查询点 q 位于以 p_i 为中心的 Voronoi 多边形内，且查询点 q 的反向最近邻的候选集为 $\{p_1',p_2',\cdots,p_m'\}$，做 qp_i 的垂直平分线 l，将平面分为 l_q,l_{p_i} 2 个部分，则 q 的反向最近邻的候选集为 $\{p_1',p_2',\cdots,p_m'\}$ 中落在 l_q 一侧的点。

定理 3 对于数据集 S 中的点做全局 Voronoi 图，并将 S 中的点从外到内逐层构造凸壳，将凸壳从内到外顺序进行编号 1,2,…,m，则找 S 中某一点 p_i 的 Voronoi 多边形的邻接多边形(设 p_i 位于第 l_i 层凸壳上)，只需查找第 l_{i-1},l_i,l_{i+1} 层凸包中的点的 Voronoi 多边形是否与 p_i 的 Voronoi 多边形共享同一条 Voronoi 边即可。

基金项目：国家自然科学基金资助项目(10571037)；黑龙江省教育厅基金资助项目(11511027)
作者简介：刘润涛(1961－)，男，教授、博士研究生，主研方向：空间数据库技术；张佳佳，硕士研究生
收稿日期：2009-06-24 **E-mail：**suxiaoyan003@yahoo.com.cn

3 基于 Voronoi 图的 RNN 查询

随着对 Voronoi 图研究的深入，Voronoi 图的应用已经越来越广泛，其中最近邻查询就是 Voronoi 图的一个非常重要的应用，本文在分析基于 Voronoi 图的最近邻查询的基础上，将 Voronoi 图应用于反向最近邻查询，从一个新角度去解决反向最近邻查询问题。

3.1 点的 RNN 查询

算法 Point RNN, PRNN(q,S)

输入 查询点 $q(x,y)$，数据集 $S=\{p_1(x_1,y_1),p_2(x_2,y_2),\cdots,p_n(x_n,y_n)\}$

输出 点 q 的 $RNN=\{p_1',p_2',\cdots,p_l'\}$

Step1 进行预处理。

Step1.1 将数据集 S 中的点从外到内逐层构造凸壳，直至凸壳内部不含有点或只含有一个点或只含有一条线段，并将每层的凸壳顶点以 x 坐标最小的点位起点按照逆时针排序，将第 i 层凸壳记为 $p_{i,1}p_{i,2}\cdots p_{i,li}$ (其中 li 表示每层凸壳中顶点的个数)，并将凸壳按照从内到外的顺序进行编号为 $1,2,\cdots,m$，建立如图 2 所示的索引结构(对应于图 1 中数据)。其中，$0,\infty$ 分别代表最内层凸包的内部和最外层凸包的外部。

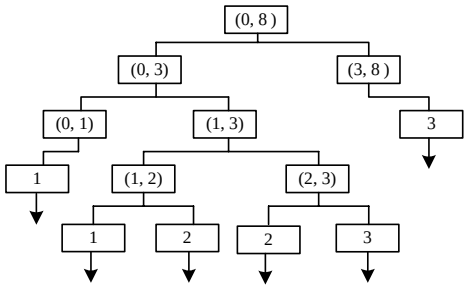

图 2 索引结构图

Step1.2 构造由数据集 S 中的点生成的 Voronoi 图并且将 Voronoi 图中的 Voronoi 顶点、数据集 S 中各个点、各个点的最近邻及数据集 S 中各个点与其最近邻的距离，存储一个栈中，图 2 中叶子节点的指针指向该栈中对应的地址，如图 2 中的 1 指向 $(p_i,Vor(p_i),NN(p_i),dNN(p_i))\ (13\le i\le 15)$ 的地址，其中，$Vor(p_i)$ 为以 p_i 为中心 Voronoi 多边形的顶点；$dNN(p_i)$ 为 p_i 与其最近邻间的距离。

Step2 利用图 2 的索引结构来确定查询点 q 的位置：

若 q 位于凸壳 $p_{1,1}p_{1,2}\cdots p_{1,l1}$ 的外侧或位于凸壳 $p_{n,1}p_{1,2}\cdots p_{n,ln}$ 内侧或位于凸壳 $p_{i,1}p_{i,2}\cdots p_{i,li}$ 上，则转 Step3；

若 q 位于凸壳 $p_{i,1}p_{i,2}\cdots p_{i,li}$ 内侧且位于凸壳 $p_{i+1,1}p_{i+1,2}\cdots p_{i+1,li+1}$ 的外侧，则转 Step4。

Step3 判断点 q 是否位于 $p_{i,1},p_{i,2},\cdots,p_{i,li}$ 中点的 Voronoi 图内。

Step3.1 点 q 位于 $p_{i,1},p_{i,2},\cdots,p_{i,li}$ 中点的 Voronoi 图内：若 q 位于某条 Voronoi 边上，则转 Step5；若 q 与某 Voronoi 中心重合，则转 Step6；若 q 位于 $p_{ij}(1\le j\le li)$ 的 Voronoi 图内，令 $p_{ij}=p_1'$，下转 Step7。

Step3.2 点 q 不位于 $p_{i,1},p_{i,2},\cdots,p_{i,li}$ 中点的 Voronoi 图内，则判断点 q 是否位于 $p_{i,1},p_{i,2},\cdots,p_{i,li}$ 内层或外层凸壳的顶点 $p_{i-1,1},p_{i-1,2},\cdots,p_{i-1,li-1},p_{i+1,1},p_{i+1,2},\cdots,p_{i+1,li+1}$ 中点的 Voronoi 图内，然后重复 Step3.1。

Step4 判断 q 是否位于 $p_{i,1},p_{i,2},\cdots,p_{i,li},p_{i+1,1},p_{i+1,2},\cdots,p_{i+1,li+1}$ 中点的 Voronoi 图内。

Step4.1 点 q 位于 $p_{i,1},p_{i,2},\cdots,p_{i,li},p_{i+1,1},p_{i+1,2},\cdots,p_{i+1,li+1}$ 中点的 Voronoi 图内：若 q 位于某条 Voronoi 边上，则转 Step5；若 q 与某 Voronoi 中心重合，则转 Step6；若 q 位于 $p_{ij}(1\le j\le li)$ 的 Voronoi 图内，则转 Step7。

Step4.2 点 q 不位于 $p_{i,1},p_{i,2},\cdots,p_{i,li},\ p_{i+1,1},p_{i+1,2},\cdots,p_{i+1,li+1}$ 中点的 Voronoi 图内，则判断点 q 是否位于 $p_{i,1},p_{i,2},\cdots,p_{i,li}$ 内层凸壳的顶点 $p_{i-1,1},p_{i-1,2},\cdots,p_{i-1,li-1}$ 或 $p_{i+1,1},p_{i+1,2},\cdots,p_{i+1,li+1}$ 外层凸壳 $p_{i+2,1},p_{i+2,2},\cdots,p_{i+2,li+2}$ 的顶点中点的 Voronoi 图内，然后重复 Step4.1。

Step5 设 q 所在的 Voronoi 边被以 p_2' 为中心的 Voronoi 多边形和以 p_3' 为中心的 Voronoi 多边形共享，找出所有与以 p_2'，p_3' 为中心的 Voronoi 多边形邻接的 Voronoi 多边形的中心 $\{p_{l1}',p_{l2}',\cdots,p_{li1}'\}$，做 qp_2' 和 qp_3' 的垂直平分线 l_1,l_2，落入 l_{1q} 一侧且落入 l_{2q} 一侧的点 $\{p_{l1}'',p_{l2}'',\cdots,p_{lj1}''\}(j1<i1)$ 是 q 的反向最近邻的候选集，计算出 $|qp_i''|(4\le i\le lj1)$，下转 Step8。

Step6 设 q 与 Voronoi 中心 p_i 重合，找出所有以 p_i 为最近邻的点 $\{p_{m1}',p_{m2}',\cdots,p_{mi2}'\}$，即为 q 的反向最近邻，将 $\{p_{m1}',p_{m2}',\cdots,p_{mi2}'\}$ 存入 $RNN[l]$ 中。

Step7 找出所有与以 $p_{ij}(1\le j\le li)$ 为中心的的 Voronoi 多边形邻接的 Voronoi 多边形，设其 Voronoi 中心为 $\{p_{n1}',p_{n2}',\cdots,p_{ni3}'\}$，做 qp_{ij} 的垂直平分线 l_3，落入 l_{3q} 一侧的点 $\{p_{n1}'',p_{n2}'',\cdots,p_{nj3}''\}(j3<i3)$ 是 q 反向最近邻的候选集，计算出 $|qp_i''|(4\le i\le nj3)$，下转 Step8。

Step8 令 $l=0$，若 $|qp_i''|\le NN(p_i'')$，则将 p_i' 存入数组 $RNN[l]$ 中；$l++$。

Step9 输出 q 的反向最近邻的集合 $RNN[l]$。

3.2 新增点对 RNN 查询的影响

对数据集 S 进行更新，那么给定查询点的反向最近邻将会受到一定的影响，本节给出了处理数据集中一次增加一个数据点的方法，当增加多个数据点时，可根据需要对 Voronoi 图进行局部重构。

算法 Add Point RNN, ADDPRNN(S,q,w)

输入 数据点集 S，查询点 q，增加点 w

输出 新的反向最近邻集

Step1 调用算法 PRNN(q,S)，得到 q 的反向最近邻集 $RNN[l]$。

Step2 以 $RNN[l]$ 中的点为圆心，该点与 q 之间的距离为半径做圆，若新增点 w 位于某一个或某几个圆内，则转 Step3；若新增点 w 不位于任何圆内，则转 Step4。

Step3 设新增点 w 位于以 p_i ($p_i\in RNN[l]$)为圆心，$|qp_i|$ 为半径的圆内，则做线段 qp_i 的垂直平分线 l_{qp_i}，若新增点 w 位于 l_{qp_i} 的 p_i 侧，则在 $RNN[l]$ 中删除点 p_i；若新增点 w 位于 l_{qp_i} 的 q 侧，则在 $RNN[l]$ 中删除点 p_i 且在 $RNN[l]$ 中增加点 w。

Step4 输出 q 的新的反向最近邻集 $RNN[l]$。

（下转第 85 页）