路网中基于Voronoi图的反向最近邻查询方法

张丽平 经海东 李 松 崔环宇
(哈尔滨理工大学计算机科学与技术学院 哈尔滨 150080)

摘 要 针对已有的在路网中的反向最近邻(Reverse Nearest Neighbor,RNN)查询方法存在的不足,提出了利用网络Voronoi图(Network Voronoi Diagram,NVD)的NVD-RNN算法,该算法具有较好的效果,它把路网划分成小的Voronoi区域,并且采用了两个过程:过滤过程和精炼过程。过滤过程主要是提前存储可能的查询结果。精炼过程主要是从可能的结果集合中找到查询结果。并且进一步给出了处理新增加点的ADDNVD-RNN算法和处理删除点的DENVD-RNN算法。实验表明,该算法在处理路网中的反向最近邻问题时有明显的优势。

关键词 网络Voronoi图,反向近邻查询,路网环境
中图法分类号 TP311.13 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.8.047

Reverse Nearest Neighbor Query Based on Voronoi Diagram for Road Network

ZHANG Li-ping JING Hai-dong LI Song CUI Huan-yu
(Department of Computer Science and Technology, Harbin University of Science and Technology, Harbin 150080, China)

Abstract In view of the shortage of exisiting reverse nearest neighbour(RNN) query method in road network, NVD-RNN algorithm making use of network Voronoi diagram(NVD) has good effect. The algorithm divides the road network into small Voronoi regions and adopts two processes: filtering and refining processes. The filtering process mainly stores possible query results in advance. The major task of refining process is to find query results from potential outcome sets, in addition it provides ADDNVD-RNN algorithm to dispose of newly increased points and DENVD-RNN algorithm to cope with deleted points any further. The experiment demonstrates that the algorithm has obvious advantages in dealing with reverse nearest neighbor in road network.

Keywords Network Voronoi diagram, Reverse nearest neighbor query, Road network environment

1 引言

空间数据查询技术在空间数据库、空间拓扑关系分析、空间数据挖掘、地理信息系统和智能交通等领域具有广泛的应用。目前为止,空间数据库中的相关查询从数据环境上可以分为两类:一类是基于欧氏空间的查询,另一类是基于道路网络(路网)上的查询。两种环境的区别在于,欧氏空间中两点之间的距离是直线距离,并且对象的位置可以是空间的任何位置;而在路网环境中,两点之间的距离是它们之间最短路径的距离和,且对象只能是在道路上,即只能在路网图中的边上。相比较而言,在路网中的查询会更有意义,因为这与人们的生活和实际应用更加贴近。常见的空间数据查询有最近邻(Nearest Neighbor,NN)查询及其变体。许多涉及拓扑关系和方向关系的查询技术往往与最近邻查询密切相关。最近邻查询已经成功应用于很多领域,比如地理信息系统[1]、点集分类[2]、智能交通[3]、球面上的查询[4]、离群检测[5]和受限区域内的查询[6]等。

反向最近邻查询是在最近邻查询的基础上,由Korn和Muthukrishnan首次提出的一种查询,他们使用基于RNN-树的方法解决该类查询[7],但是该方法需要建立第二棵索引树,会耗费时间。为此C. Yang和K. I. Lin提出了基于Rdnn-树的查询方法[8],其省掉了第二棵索引树,这样就较大程度节省了查询时间。在此之后研究者提出了很多方法来提高反向最近邻查询的效率,如C. Xia等人提出了基于估计的过滤方法[9],郝忠孝等人提出了基于SRdnn-树的查询方法[10],W. Wu等人提出了利用INCH裁剪查询区域的方法[11],李松等人提出了基于Voronoi图的方法[12]以及Tobias Emrich等人提出了最大化双色反向最近邻搜索的新方法[13]等。以上的研究成果主要是针对欧氏空间下给出的反向最近邻查询,没有考虑在路网中的限制。实际应用中,大量的查询往往不是针对欧氏空间的,而是经常在路网环境下。为了更加有效地解决路网中的反向最近邻查询问题,Sun Huan-liang等人提出了依靠启发式扩展的策略,采用PMR四叉树和Dijkstra算法相结合的RNN查询算法[14],该方法可以很好地处理路网中的反向最近邻查询,但是每次查询需要遍历PMR四叉树中的所有节点,导致查询时间较长。齐峰等人提出了在路网

到稿日期:2014-08-07 返修日期:2015-01-30 本文受黑龙江省教育厅科学技术研究项目(12541128)资助。
张丽平(1976-),女,硕士,副教授,主要研究方向为数据结构和算法设计、数据库,E-mail:zhangliping0730@163.com;经海东(1990-),男,硕士生,主要研究方向为空间数据查询与处理、算法设计;李 松(1977-),男,博士,副教授,主要研究方向为数据库理论与技术、算法设计;崔环宇(1990-),男,硕士生,主要研究方向为数据查询与分析。

中基于M树索引的近似反向最近邻查询[15]，该方法虽有一定的性能优势，但M树的网格距离的计算代价影响了查询效率。朱彩云等人提出了在路网中的反向最近邻查询方法[16]，该方法虽有一定的适用性，但是在处理动态增加或减少数据点时耗时较长。Muonammad Aramir Cheema 等人提出了在欧氏空间和网络空间的连续反 k 近邻查询[17]，该算法的优点是简单，查询速度快，但是随着数据点的增加，查询的精炼性有明显的降低。

针对以上方法的不足，本文提出了在路网中的基于网络Voronoi图的反向最近邻查询方法，该方法包含过滤和精炼两个过程。在过滤过程中会生成候选集合，用来存储有可能成为结果的点，方法是把所有与此查询结果相邻的NVP加入到候选集合中。在精炼过程中，通过计算查询点到新加入到候选集合中的NVP的距离，来更新查询点到边界点的最短距离，其中更新距离的思想类似于Dijkstra算法。对于每一次查询都要递归调用过滤过程和精炼过程。

2 基本定义

定义1(反向最近邻[7]) 设 $d(p,q)$ 为两点 p 和 q 之间的距离，有一 d 维数据集 S 和一查询点 q，RNN查询就是找出 S 的子集 $RNNs(q)$ 即：$RNNs(q)=\{r\in S \mid \forall p\in S: d(q,r)<d(r,p)\}$。

定义2(Voronoi图[18]) 给定一组数据点的集合 $Q=\{p_1,\cdots,p_n\}\subset R^2$，其中 $2<n<\infty$，当 $i\neq j$ 时，$p_i\neq p_j$。Voronoi区域由以下公式给出：$VH(p_i)=\{q \mid d(q,p_i)\leqslant d(q,p_j)\}$。$d(q,p_i)$ 为 q 与 p_i 之间的最小距离。p_i 称为Voronoi生成点，由 p_i 所决定的Voronoi区域 $VH(p_i)$ 称为Voronoi多边形，Voronoi多边形的棱记为 $VL(p_i)$。由 $V(H)=\{VH(p_1),\cdots,VH(p_n)\}$ 所定义的图形被称为Voronoi图。与 $VH(p_i)$ 共享相同的棱的Voronoi多边形被称为 $VH(p_i)$ 的邻接格，它们的生成点被称为 q_i 的一级邻接生成点，简记为 $LJG(p_i)$。依次可定义多级邻接生成点。

性质1[18] 距离 p_i 最近的生成点一定在与 $VH(p_i)$ 有共同Voronoi边的Voronoi多边形中。

网络Voronoi图(Network Voronoi Diagram，NVD)[19]与Voronoi图的区别在于NVD采用的是路网距离，而并非欧氏距离。如图1所示，在NVD中，由路网中的边和虚线所围成的区域称为NVP(Network Voronoi Polygon)。由 p_i 生成的NVP记为 $NVP(p_i)$。例如，$NVP(p_1)$ 记为线段 (n_1,p_1)，(n_5,p_1)，(n_1,b_1)，(b_1,b_2)，(b_2,v)，(v,b_5)，(n_5,b_5) 所围成的区域。若给定生成点集合 $P=\{p_1,p_2,\cdots,p_n\}$，边的集合 $E=\{e_1,e_2,\cdots,e_k\}$，p_j 为不同于 p_i 的生成点，则 $NVP(p_i)$ 可以形式化定义为：

$$NVP(p_i)=\{p \mid p\in\bigcup_{i=1}^{k}e_i, d(p,p_i)\leqslant d(p,p_j)\}$$

算法1[19]描述了NVD生成的伪代码。第1行—第3行为每一个节点构造标签；第4行—第7行扫描每一条边，计算出所有边界点的位置。

算法1 NVD生成

输入：N，P

输出：NVP

1. for each v∈V
2. 计算节点v的最近生成点P及距离d(v)；
3. 构造标签(d(v)，P(v))；
4. for each(u，v)∈E
5. if(P(u)∩P(v)=∅)
6. 计算边界点b的位置；
7. 保存边界点(u，v|d(u)−d(v)−w(u，v)/2)；
8. end if
9. end for
10. 返回NVD

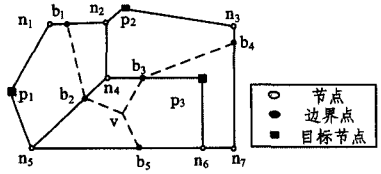

图1 网络Voronoi图

3 路网中基于Voronoi图的反向最近邻查询

3.1 过滤过程

该过程主要是生成下一个可能的RNN候选集合。在计算这个候选集合时，需要用到网络Voronoi图中的相邻NVP。之后把已经得到的查询结果的相邻NVP放入候选集合，因此下一个反向最近邻查询结果就在候选集合中。性质2和性质3证明了这个结论。

性质2 根据性质1，对于生成点 p_i 而言，$NVP(p_i)$ 内任何一个查询点的下一个RNN一定是在与 $NVP(p_i)$ 相邻的NVP中。

证明：此性质用反证法来证明。如图2所示，查询点 q 的第一个RNN是 p_2，因为 q 在NVP(p_2)中。假设 q 的第二个RNN是 p_3，p_3 与 p_2 是不相邻的NVP。这里需要 q 到 p_3 的最短路径 $d(q,p_3)$。$NVP(p_3)$ 与 $NVP(p_9)$ 在点 b_{12} 处相交。注意在NVD中，$d(q,p_3)$ 或许不是一条直线。由Voronoi图的性质可知 $d(b_{12},p_3)=d(b_{12},p_9)$。已知在路网或是欧氏空间中都服从三角不等式，即 $d(b_6,b_{12})+d(b_{12},p_9)>d(b_6,p_9)$。从而可以得出结论：$d(b_6,b_{12})+d(b_{12},p_3)>d(b_6,p_9)$，并通过增加 $d(q,b_6)$ 到不等式两侧，就可以得到最终的结论 $d(q,b_6)+d(b_6,b_{12})+d(b_{12},p_3)>d(q,b_4)+d(b_4,p_9)$，即 $d(q,p_3)>d(q,p_9)$。与假设 q 的第二个RNN是 p_3 矛盾，故原性质成立。证毕。

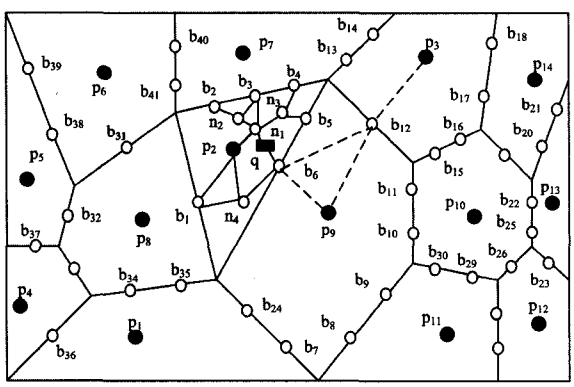

图2 网络Voronoi图的示例

性质3 假设 $P'=\{p_1,p_2,\cdots,p_k\}\in P$ 为 $NVP(p_l)$ 中查询点 q 的前 k 个RNN。令 $P''=\{p \mid p\in P, p$ 相邻于 p_i 且 $p\notin P', 1\leqslant i\leqslant k\}$。则查询点 q 的下一个RNN肯定在集合 P'' 中。

证明：此证明与性质2证明类似，也同样使用反证法来进行证明。令从 q 到 p_k 的最短路径是 $L(q,p_k)$，则它必须与 $NVP(p_k)$的一个边界相交。假设其为 E_k，$L(q,p_k)$与 E_k 在点 b_k处相交，并且 E_k 是 $NVP(p_k)$与 $NVP(x)$之间的公共边界。假设 $x\notin P''$，可以得到 $L(q,p_k)$必须与 $NVP(x)$相交至少一个点。利用性质2可以得到结论：$d(b_x,x)<d(b_x,b_k)+d(b_k,p_k)$。则 x 比 p_k 更接近 q。因此 $x\in P'$，与结论矛盾。证毕。

性质2和性质3保证了下一个查询点肯定在候选集合 P''中，这其实是保证了过滤过程的正确性。图2为一个详细的网络 Voronoi 图。对于查询点 q，显然它的 1-RNN 为它所在网络 Voronoi 图的生成点，即 p_2。根据性质2，它的 2-RNN 肯定在候选集合$\{p_7,p_8,p_9\}$中。假设它的 2-RNN 为 p_7，那么根据性质3，它的 3-RNN 肯定在候选集合$\{p_8,p_9,p_3,p_6\}$中。

3.2 精炼过程

精炼过程主要对过滤过程中得到的候选集合进行操作，其主要方法是计算查询点到新加入到候选集合中的生成点的最短距离。查询点到候选集合中其他生成点的距离不用计算，因为在它们作为生成点加入到候选集合时已经计算过了。为了计算查询点到新生成点之间的最短距离，进一步给出性质4和性质5。

性质4 在一个 NVD 中，如果$\{p_1,p_2\}$为查询点 q 的前两个 RNN，那么从 q 到 p_2 的最短路径只能经过$\{NVP(p_1),NVP(p_2)\}$，即只能经过 $NVP(p_1)$与 $NVP(p_2)$的共同边。

证明：使用反证法来证明。如果 q 到 p_2 的最短路径通过 $NVP(p_k)$，假设 $p_k\notin\{p_1,p_2\}$，则在 $NVP(p_k)$里面的那部分最短路径比 p_2 更接近 p_k。这里的 p_2 是紧接着 p_1 的反向最近邻。如图2所示，假设$\{p_2,p_9\}$是 q 的前两个反向最近邻，令从 q 到 p_9 的最短路径是 $L=\langle q,n_1,n_3,b_4,b_{13},b_{12},p_9\rangle$，它与 $NVP(p_7)$相交于点 b_4 和 b_{13}。因此在线段上的任意点都比 p_9 更接近 p_7。这里就需要 p_7 比 p_9 更接近于 q 的反向最近邻。这与假设相矛盾，原性质成立。证毕。

性质5 在一个 NVD 中，如果$\{p_1,p_2,\cdots,p_k\}$为查询点 q 的前 k 个 RNN，那么从 q 到 p_2 的最短路径只能经过$\{NVP(p_1),NVP(p_2),\cdots,NVP(p_k)\}$，即只能经过$\{NVP(p_1),NVP(p_2),\cdots,NVP(p_k)\}$的共同边的组合。

证明：此性质是性质4的一般化。同样采用反证法。如果 q 到 p_k 的最短路径通过 $NVP(p_m)$，假设 $p_m\notin\{p_1,\cdots,p_k\}$，则在 $NVP(p_m)$里面的那部分最短路径比 p_k 更接近 p_m。这与假设$\{p_1,p_2,\cdots,p_k\}$为查询点 q 的前 k 个 RNN 相矛盾，原性质成立。证毕。

性质6 在一个 NVD 中，如果查询点 q 到 p_k 的最短路径经过了 $NVP(p_i)$，那么 p_i 比 p_k 更接近 q。

证明：由性质5可知，从 q 到 p_k 的最短路径只能是$\{NVP(p_1),NVP(p_2),\cdots,NVP(p_k)\}$的合集。因此，$p_i$ 必须是$\{NVP(p_1),NVP(p_2),\cdots,NVP(p_{k-1})\}$中的一个。即 p_i 比 p_k 更接近 q。证毕。

根据以上的性质，可以得出计算查询点到新加入生成点的距离的方法。首先要计算出查询点 q 到它所在的 $NVP(p_1)$的所有边界点的最短距离。然后在计算 2-RNN 时，根据过滤过程，把与 p_1 相邻的生成点加入到候选集合中，再根据性质4，查询点 q 到这些新加入的生成点的最短距离必定会经过它们的共同的边界点，因此最短距离包括两个部分：查询点到边界点的距离和边界点到生成点的距离。即最短距离＝Min{查询点到边界点的距离＋边界点到生成点的距离}。按照 Dijkstra 算法的思想维护查询点 q 到已经计算出的前 k 个 RNN 所在的 NVP 共同的边界点的最短距离，那么在计算下一个 RNN 时，只要选取使得查询点到边界点的距离与边界点到生成点的距离的和最小的生成点作为下一个 RNN。

如图2所示，查询点 q 在 $NVP(p_2)$内部，所以它的 1-RNN为 p_2。过滤过程得到的候选集合为$\{p_7,p_8,p_9\}$，精炼过程要计算查询点 q 到 p_7、p_8、p_9 的最短距离。这里用 $d(u,v)$表示节点 u 到 v 的最短路网距离。根据精炼过程中计算最短距离的方法，有

$$d(q,p_7)=\min\{d(q,b_2)+d(b_2,p_7),d(q,b_3)+d(b_3,p_7),d(q,b_2)+d(b_3,p_7)\}$$

$$d(q,p_8)=\min\{d(q,b_1)+d(b_1,p_8)\}$$

$$d(q,p_9)=\min\{d(q,b_5)+d(b_5,p_9),d(q,b_6)+d(b_6,p_9)\}$$

假设 p_9 是 2-RNN，那么在计算 3-RNN 时，候选集合为$\{p_7,p_8,p_1,p_{10},p_{11}\}$，即在精炼阶段需要计算查询点 q 到$\{p_1,p_3,p_{10},p_{11}\}$的距离。以计算查询点 q 到 p_{10}的距离为例，有

$$d(q,p_{10})=\min\{d(q,b_{10})+d(b_{10},p_{10}),d(q,b_{11})+d(b_{11},p_{10})\}$$

而且

$$d(q,b_{10})=\min\{d(q,b_5)+d(b_5,p_{10}),d(q,b_6)+d(b_6,b_{10})\}$$

$$d(q,b_{11})=\min\{d(q,b_5)+d(b_5,b_{11}),d(q,b_6)+d(b_6,p_{11})\}$$

3.3 基于 NVD 的 RNN 算法描述

算法2利用具有良好的过滤功能的网络 Voronoi 图来处理路网中反向最近邻查询问题。该算法分成过滤过程和精炼过程。通过过滤过程过滤出所有可能的结果，得到候选集合。在精炼过程中主要是从过滤集合中找出最终的查询结果。

算法2 NVD-RNN(q,S)

输入：查询点 q，数据集 $S=\{p_1,p_2,\cdots,p_n\}$

输出：点 q 的 RNN

```
begin
1. ResultSet=∅, CandidateSet=∅; //初始化
2. determine NVP(q); //确定 q 所在的网络 Voronoi 区域
3. add p_i into ResultSet; //将第一个 NVP 的生成点 p_i 加入到结果的集合中
4. while not_empty(S) do
5.       determine Adjacent_NVP(p_i, p_j); //确定与 p_i 相邻的 NVP 生成点 p_j
6.       add p_j into CandidateSet; //将 p_j 加入到候选集合中
7. end while
8. while not_empty(CandidateSet) do
9.   if p_x 与 p_y 是相邻的 then
10.    d(p_x,p_y)=d(p_x,b_z)+d(b_z,p_y); //计算 p_x 与 p_y 之间的距离，其中 b_z 是 p_x 与 p_y 之间的点
11.  end if
12.  if p_x 与 p_y 的距离大于 q 与 1RNN(q)的距离 then
13.    Remove p_x and p_y from CandidateSet; //将 p_x 与 p_y 从候选集
```

合中移除
14. end if
15. end while
16. while not_empty(CandidateSet) do
17. $d(p_x,q)=d(p_x,b_z)+d(b_z,q)$;//计算 p_x 与 q 之间的距离,其中 b_z 是 p_x 与 q 之间的点
18. compute $\min\{d(p_x,q)\}$;//计算 p_x 与 q 之间的最小距离
19. add p_x into ResultSet;//将 p_x 加入到结果集合
20. end while
21. return ResultSet;//返回结果的集合
end

算法 2 首先建立两个空队列 ResultSet 和 CandidateSet,分别用于存储结果集合和候选集合。其次确定查询点 q 所在的 NVP,该 NVP 的生成点 p_i 记为 q 的第一个 RNN,并将其加入结果集。再次确定与 $NVP(p_i)$ 相邻的 $NVP(p_j)$,将 p_j 加入候选集合中。计算候选集中相邻的生成点之间的距离,比较相邻的生成点之间的距离与生成点到反向最近邻之间的距离,如果前者大,则将相邻的两个生成点从候选集中删除。最后计算查询点 q 到新加入到候选集合的 p_x 的距离,找到最短距离 $d(p_x,q)$,然后将 p_x 加入到 ResultSet 中,返回 ResultSet。

假设查询点为 q,令算法 2 返回点 o,且 o 不是点 q 的反向最近邻对象。令 $q_k(o)$ 为点 o 的反向最近邻对象,$d(o,q_k(o))$ 指 o 的反向最近邻距离。如果 o 不是查询点 q 的反向最近邻对象,则必有 $d(o,q_k(o))<d(o,q)$。然而,算法 2 只在精炼过程中返回候选集中的生成点到查询点 q 的最短路径。因为 $d(o,q_k(o))<d(o,q)$,即查询点 q 不是 o 的 k 最近邻对象之一,o 点不可能被返回。因此,通过算法 2 返回的点 o 必定是 q 的反向最近邻对象。故算法 2 是正确的。

算法 2 分为过滤和精炼两个过程,在过滤过程中,有两个 while 循环,第一个 while 循环的终止条件是在数据集 S 中的全部数据被循环一次。因为数据集 S 的个数是有限的,所以第一个 while 循环可在有限时间内结束,故不是死循环。第二个 while 循环的中止条件是候选集合 CandidateSet 中的全部数据被循环一次。因为数据集 CandidateSet 的个数也是有限的,所以第二个 while 循环可在有限时间内结束,故也不是死循环。在精炼过程中,只有一个 while 循环,在这个 while 循环体中,候选集中所有生成点的元素个数是有限的,当队列中数据被循环一遍后,就会停止循环,所以该 while 循环可在有限时间内结束,故也不是死循环。综上分析可知,算法 2 是可中止的。

3.4 路网中新增点对反向最近邻查询的影响

若数据集 S 增加数据点,那么给定查询点的反向最近邻查询将会受到一定的影响。为了方便研究,给出了处理数据集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 NVD 进行局部重构。若新增加的点在根据算法所画的圆内,则重新调用算法 2;若不在,则返回原有的查询结果。算法 3 是新增点对路网中反向最近邻查询的影响算法。

算法 3 ADDNVD-RNN(S,q,w)
输入:查询点 q,数据集 S=$\{p_1,p_2,\cdots,p_n\}$,增加点 w
输出:点 q 的新的 RNN
begin
1. $S'=\{p_1,p_2,\cdots,p_n,w\}$;//将新增加的点 w 加入到数据据集 S 中得到新的数据集 S'
2. RNN[m]=∅,NewRNN[n]=∅;Dist[o]=∅;//初始化
3. call NVD-RNN(q,S);//调用 NVD-RNN 算法得到 q 的 RNN 为 $p_1,p_2,\cdots,p_i$
4. add $p_1,p_2,\cdots,p_i$ into RNN[m];//将 $p_1,p_2,\cdots,p_i$ 加入到 RNN[m]队列中
5. $d(p_x,q)=d(p_x,b_z)+d(b_z,q)$;//计算 RNN[m]队列中任意的 p_x 与 q 之间的距离,其中 b_z 是 p_x 与 q 之间的点
6. add $d(p_x,q)$ into Dist[o];//将 p_x 加入到 Dist[o]队列
7. make_Circle(p_x,$d(p_x,q)$);//以点 p_x 为圆心、$d(p_x,q)$为半径做圆
8. if 新增点 w 在圆内 then
9. call NVD-RNN(q,S');//重新调用 NVD-RNN 算法得到 q 的新的 RNN 为 $p_1,p_2,\cdots,p_j$
10. add $p_1,p_2,\cdots,p_j$ into NewRNN[n];//将 $p_1,p_2,\cdots,p_j$ 加入到 NewRNN[n]队列中
11. return NewRNN[n];//返回 NewRNN[n]
12. end if
13. else return RNN[m];//如果不在圆内,返回 RNN[m]
end

算法 3 首先将新增加的点加入到数据集 S' 中,然后建立 3 个空队列 RNN[m]、NewRNN[n]和 Dist[o]分别用于存储查询点的查询结果、存储查询点新的反向最近邻点和存储查询点与反向最近邻点的距离。接下来调用 NVD-RNN(q,S)算法,将所得到的反向最近邻点加入 RNN[m]队列中。然后计算 q 到 RNN[m]队列内的任意 p_x 的最短距离,将 q 到 p_x 的最短距离加入 Dist[o]队列中。以 p_x 为圆心、$d(p_x,q)$为半径做圆。如果新增点 w 位于某一个或某几个圆内,重新调用 NVD-RNN(q,S)算法,将反向最近邻点存储于 NewRNN[n]中,然后返回 NewRNN[n];否则返回原来的 RNN[m]。

3.5 路网中删除点对反向最近邻查询的影响

若从数据集 S 中删除数据点,则对查询点的反向最近邻查询会有一定的影响,若删除点在结果的集合中,则重新调用 NVD-RNN 算法;若不在,则返回原来的结果。算法 4 是删除点后对路网中 RNN 查询的影响算法。

算法 4 DENVD-RNN(S,q,d)
输入:查询点 q,数据集 S=$\{p_1,p_2,\cdots,p_n\}$,删除点 d(d∈S)
输出:点 q 的新的 RNN
begin
1. $S'=\{p_1,p_2,\cdots,p_m\}$;//数据集 S 中删除 d 得到数据集 S'
2. RNN[m]=∅,NewRNN[n]=∅;//初始化
3. call NVD-RNN(q,S);//调用 NVD-RNN 算法得到 q 的 RNN 为 $p_1,p_2,\cdots,p_i$
4. add $p_1,p_2,\cdots,p_i$ into RNN[m];//将 $p_1,p_2,\cdots,p_i$ 加入到 RNN[m]队列中
5. if 删除点 d 在 RNN[m]队列中 then
6. call NVD-RNN(q,S');//重新调用 NVD-RNN 算法得到新的 RNN 为 $p_1,p_2,\cdots,p_j$
7. add $p_1,p_2,\cdots,p_j$ into NewRNN[n];//将 $p_1,p_2,\cdots,p_j$ 加入到 RNN[n]队列中
8. return NewRNN[n];//返回 NewRNN[n]
9. end if
10. else return RNN[m];//如果不在,返回 RNN[m]
end

算法 4 首先在数据集 S 中删除任意点得到数据集 S',然后建立两个空队列 RNN[m]和 NewRNN[n]分别用于存储查

询点的查询结果和查询点新的反向最近邻点。接下来调用NVD-RNN(q, S)算法将所得到的反向最近邻点加入RNN[m]队列中。如果删除点位于RNN[m]中，重新调用NVD-RNN(q,S)算法，将反向最近邻点存储于NewRNN[n]中，返回NewRNN[n]；如果不在，返回原来的RNN[m]。

4 实验与分析

本文采用的数据集是道路网络——美国旧金山和加州的路网[20]。对目标节点和分布进行了适度调整。参数 k 表示要得到的目标节点个数，参数 D 表示数据点个数。集合 Q 中的查询点均匀分布在路网的子区域中。用 A 表示子区域占整个路网的百分比。此外，用参数 F 表示边的权值与实际长度的偏离程度，F=权值/欧氏距离。在默认情况下，每次查询设20个查询点，分布在 A=5%的路网图中，目标节点的密度 P=0.05，参数 F=2。对于每次实验，实验结果均为执行30次的平均效果。实验运行环境为：2.0GHz CPU、4GB内存、500GB硬盘、Windows 7操作系统。

首先，给出了 k 值变化对CPU时间的影响。图3显示NVD-RNN算法与ARkNN算法[15]和网络扩展(EP)算法[21]的对比。从图中可以看出，随着 k 的增大，3种算法的CPU时间都在增大。但是由于NVD-RNN算法具有过滤阶段，因此不需要遍历所有节点，而ARkNN算法和EP算法都需要遍历整个节点，故后两者的CPU时间增长较大。实验结果表明随着 k 的增大，NVD-RNN算法优于另外两种算法。

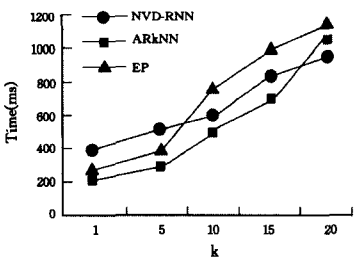

图3 k 对CPU时间的影响

其次，验证了不同的 D 值对CPU时间的影响。由图4可知，随着 D 的增大，NVD-RNN算法、EP算法和ARkNN算法的CPU时间都在增大，因为 D 值增大，遍历的节点个数增多，从而增加了节点距离的计算，而NVD-RNN算法不需要查询所有的节点，这是因为利用过滤阶段可以有效地减小查询节点的个数，提高了查询效率。实验结果表明本文提出的NVD-RNN算法优于另外两种算法。

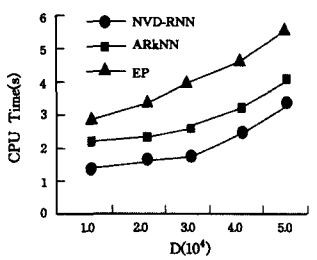

图4 D 对CPU时间的影响

接下来从执行时间、页面访问量这两个方面来评价NVD-RNN算法和对比算法。表1显示了NVD-RNN算法与对比算法在计算RNN结果时的性能对比。实验数据为美国旧金山路网和加州路网信息。为便于实验，数据信息进行了适当的预处理。实验环境都为默认情况，即查询点个数为1，A=5%，P=0.05，F=2。通过比较表中的数据，可以发现NVD-RNN算法的执行时间和页面的访问量要优于ARkNN算法和EP算法。

表1 NVD-RNN与ARkNN和EP的性能对比

旧金山数据	执行时间(s)	数据集	加州数据	执行时间(s)	页面访问量
NVD-RNN	0.923	8900	NVD-RNN	0.875	2540
ARkNN	2.465	18900	ARkNN	1.976	4300
EP	3.236	19570	EP	2.847	5880

最后，评价ADDNVD-RNN算法和DENVD-RNN算法的性能。表2显示了在数据集合中加入任意点后的ADDNVD-RNN算法与对比算法在计算RNN结果时的性能对比，执行时间是取多次测试的平均值。通过对比数据可以发现，ADDNVD-RNN算法在处理数据集中增加点后的反向最近邻查询时时要优于ARkNN算法和EP算法。

表2 ADDNVD-RNN与ARkNN和EP的性能对比

旧金山数据	执行时间(s)	数据集	旧金山数据	执行时间(s)	数据集
NVD-RNN	1.913	22000	ADDNVD-RNN	2.106	22001
ARkNN	2.032	22000	ARkNN	3.479	22001
EP	3.636	22000	EP	4.147	22001

表3显示了在数据集合中删除任意点后的DENVD-RNN算法与对比算法在计算RNN结果时的性能对比，执行时间同样是取多次测试的平均值。对比数据可以发现，DENVD-RNN算法在处理数据集中删除点后的反向最近邻查询时要优于EP算法和ARkNN算法。

表3 DENVD-RNN与ARkNN和EP的性能对比

旧金山数据	执行时间(s)	数据集	旧金山数据	执行时间(s)	数据集
NVD-RNN	1.295	10500	DENVD-RNN	1.326	10049
ARkNN	2.269	10500	ARkNN	3.371	10049
EP	2.834	10500	EP	3.948	10049

由分析可知，路网信息的复杂程度、数据集的大小和数据点的分布对算法性能的影响较大。在处理查询点较少时，NVD-RNN算法的优势并不是很明显，即该算法不适合处理查询点较少的反向最近邻查询，因为NVD-RNN算法开始时要构建NVD图，会耗费时间。但是随着查询点的增多，NVD-RNN算法的过滤优势开始显现。

结束语 随着无线通信技术、全球定位系统、地理信息系统以及移动计算等现代化技术的迅速发展，空间关系的近邻查询和分析技术具有重要的作用。本文分析了现有路网环境下反向最近邻查询处理方法存在的问题，在此基础上提出了路网的反向最近邻查询方法，解决方案包含过滤和精炼两个过程：在过滤过程中得到一个候选结果的集合；而精炼过程的主要工作是对候选结果的集合进行查找，找出最终结果并将其加入到结果集合中。实验证明，在一定情况下NVD-RNN算法提高了路网环境下的反向最近邻查询效率。本文同时也给出了处理新增加点的ADDNVD-RNN算法和处理删除点的DENVD-RNN算法，实验表明这两种算法较适合处理数据点更新后的反向最近邻查询。未来的研究重点主要集中在以下两点：

(下转第258页)

[15] 郭莹,张长胜,张斌. 一种求解 SAT 问题的人工蜂群算法[J]. 东北大学学报(自然科学版),2014,35(1):29-32
Guo Ying, Zhang Chang-sheng, Zhang Bin. An Artificial Bee Colony Algorithm for Solving SAT Problem[J]. Journal of Northeastern University(Natural Science),2014,35(1):29-32

[16] 秦全德,程适,李丽,等. 人工蜂群算法研究综述[J]. 智能系统学报,2014,9(2):127-135
Qin Quan-de, Cheng Shi, Li Li, et al. Artificial bee colony algorithm: a survey[J]. CAAI Transactions on Intelligent Systems, 2014,9(2):127-135

[17] Wolpert D H, Macready W G. No free lunch theorems for optimization[J]. IEEE Transactions on Evolutionary Computation, 1997,1(1):67-82

[18] 杨小芹,黎明,周琳霞. 基于熵的双群体遗传算法研究[J]. 模式识别与人工智能,2005,18(3):286-289
Yang Xiao-qin, Li Ming, Zhou Lin-xia. Entropy Based Genetic Algorithm with Dual Subpopulations[J]. Pattern Recognition and Artificial Intelligence,2005,18(3):286-289

[19] Chun J S, Kim M K, Jung H K. Shape Optimization of Electromagnetic Devices Using Immune Algorithm[J]. IEEE Trans on Magnetics,1997,33(2):1876-1879

[20] Bessaou M, Siarry P. A Genetic Algorithm with Real-Value Coding to Optimize Multimodal Continuous Functions[J]. Structure Multidiscipline Optimization,2001,23(1):63-74

(上接第 235 页)

(1)基于 Voronoi 图的不确定拓扑关系和不确定反向最近邻查询方法。

(2)障碍空间中基于 Voronoi 图的最近邻查询方法及几种变体结构。

参 考 文 献

[1] Shekhar S, Chawla S. 空间网络数据库[M]. 北京:机械工业出版社,2004
Shekhar S, Chawla S. Space Network Database[M]. Beijing: Machinery Industry Press,2004

[2] 马希荣,王嵘. 一种基于最近邻决策的点集分类方法的确定与实现[J]. 计算机科学,2007,34(12):182-183
Ma Xi-rong, Wang Rong. A New Method to Class a Point Set Based on Nearest-Neighbour Decision Boundaries[J]. Computer Science,2007,34(12):182-183

[3] 郭薇,郭菁,胡志勇. 空间数据库索引技术[M]. 上海:上海交通大学出版社,2006:1-7
Guo Wei, Guo Jing, Hu Zhi-yong. The technology of spatial database indexing[M]. Shanghai: Shanghai Jiaotong University Press,2006:1-7

[4] 张丽平,李松,郝忠孝. 球面上的最近邻查询方法研究[J]. 计算机工程与应用,2011,47(5):126-129
Zhang Li-ping, LI Song, Hao Zhong-xiao. Research of methods for nearest neighbor query on spherical surface[J]. Computer Engineering and Applications,2011,47(5):126-129

[5] 朱庆生,唐汇,冯骥. 一种基于自然最近邻的离群检测算法[J]. 计算机科学,2014,41(3):276-278,305
Zhu Qing-sheng, Tang Hui, Feng Ji. Outlier Detection Algorithm Based on Natural Nearest Neighbor[J]. Computer Science,2014,41(3):276-278,305

[6] 李松,张丽平,朱德龙,等. 动态受限区域内的单纯型连续近邻链查询方法[J]. 计算机科学,2014,41(6):136-141
Li Song, Zhang Li-ping, Zhu De-long, et al. Simple Continues Near Neighbor Chain Query in Dynamic Constrained Regions [J]. Computer Science,2014,41(6):136-141

[7] Korn F, Muthukrishnan S. Influence sets based on reverse nearest neighbor queries [C]// Proceedings of 2000 ACM SIGMOD International Conference on Management of Data. Dallas, Texas, USA,2000:201-212

[8] Yang C, Lin K. An index structure for efficient reverse nearest neighbor queries [C]// Proceedings of 17th International Conference on Data Engineering. Heidelberg, Germany,2001:485-492

[9] Xia C, Hsu W, Lee M L. ERkNN: Efficient Reverse k-Nearest Neighbors Retrieval with Local kNN-Distance Estimation[C]// Proceedings of CIKM 2005. Bremen, Germany. 2005:533-540

[10] 郝忠孝,刘永山. 空间对象的反向最近邻查询[J]. 计算机科学,2005,32(11):115-118
Hao Zhong-xiao, Liu Yong-shan. Reverse Nearest Neighbor Search in Spatial Database[J]. Computer Science,2005,32(11): 115-118

[11] Wu W, Yang F, Chan C, et al. FINCH: Evaluating Reverse k-Nearest-Neighbor Queries on Location Data[J]. VLDB Endowment,2008,1(1):1056-1067

[12] 李松,郝忠孝. 基于 Voronoi 图的反向最近邻查询方法研究[J]. 哈尔滨工程大学学报,2008,29(3):261-265
Li Song, Hao Zhong-xiao. Reverse nearest neighbor queries using Voronoi diagrams[J]. Journal of Harbin Engineering University,2008,29(3):261-265

[13] Emrich T, Kriegel H, Mamoulis N, et al. Reverse-Nearest Neighbor Queries on Uncertain Moving Object Trajectories[M]// Database Systems for Advanced Applications. Springer, 2014: 92-107

[14] Sun H L, Jiang C, Liu J L, et al. Continuous Reverse Nearest Neighbor Queries on Moving Objects in Road Networks[C]// The Ninth International Conference on Web-Age Information Management(WAIM 08). 2008:238-245

[15] 齐峰,金顺福,刘国华,等. 道路网络环境下的近似反 k 最近邻查询算法[J]. 小型微型计算机系统,2010,8(8):1599-1603
Qi Feng, Jin Shun-fu, Liu Guo-hua, et al. Approximation Reverse k-nearest Neighbor Queries in Road Network[J]. Journal of Chinese Computer Systems,2010,8(8):1599-1603

[16] 朱彩云,刘国华,宋金玲,等. 空间网络数据库中反 k 最近邻查询算法[J]. 小型微型计算机系统,2009,9(9):1781-1786
Zhu Cai-yun, Liu Guo-hua, Song Jin-ling, et al. Algorithm for Reverse k-Nearest Neighbor Queries in Spatial Network Databases[J]. Journal of Chinese Computer Systems, 2009, 9(9): 1781-1786

[17] Cheema M A, Zhang W, Lin X, et al. Continuous reverse k nearest neighbors queries in Euclidean space and in spatial networks [J]. The VLDB Journal—The International Journal on Very Large Data Bases,2012,21(1):69-95

[18] Okabe A, Boots B, Sugihara K, et al. Spatial Tessellations, Concepts and Applications of Voronoi Diagrams(2nd ed)[M]. John Wiley and Sons Ltd,2000

[19] Kolahdouzan M R, Shahabi C. Voronoi-Based K Nearest Neighbor Search for Spatial Network Databases[C]// Proceedings of the 30th International Conference on very Large Data Bases (VLDB). 2004:840-851

[20] Data set [EB/OL]. http://www.cs.fsu.edu/lifeifei/SpatialDataset.htm,2013

[21] Yin M L, Papadias D, Mamoulis N. Reverse nearest neighbors in large graphs[C]// The 21st International Conference on Data Engineering. Tokyo, Japan,2005:301-304