障碍空间中基于Voronoi图的k最近邻查询

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 为了提升障碍空间中k最近邻查询的效率,研究了障碍空间中基于Voronoi图的k最近邻查询方法,提出了在障碍空间基于Voronoi图的kNN-Obs算法。该算法采用了两个过程:过滤过程和精炼过程。过滤过程主要是利用Voronoi图的过滤功能,较大程度地减少了被查询点的个数。精炼过程主要根据障碍距离和邻接生成点对候选集内对象进行第二次筛选。进一步给出了处理新增加点的ADDkNN-Obs算法和处理删除点的DENkNN-Obs算法。实验表明该算法在处理障碍空间中的k最近邻问题时具有优势。

关键词 Voronoi图,k最近邻查询(kNN),障碍空间,障碍距离
中图法分类号 TP311.13 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.5.032

k Nearest Neighbor Query Based on Voronoi Diagram for Obstructed Spaces

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

Abstract In order to enhance the efficiency of k nearest neighbor in obstructed spaces, k nearest neighbor query method based on Voronoi diagram in obstructed spaces was studied, and kNN-Obs algorithm based on Voronoi diagram in obstructed spaces was proposed. This algorithm adopts two processes: filtration and refinement. Filtration mainly uses the filter function of Voronoi diagram and largely reduces the number of query points. Refinement mainly screens objects within the candidate set according to barrier distance and Voronoi neighbor. And further ADDkNN-Obs algorithm for newly-added points and DENkNN-Obs algorithm for deleted points were given. The experiment shows that the algorithm has advantages in dealing with the k nearest neighbor problem in obstructed spaces.

Keywords Voronoi diagram, k nearest neighbor query, Obstructed spaces, Obstructed distance

1 引言

Voronoi图是计算几何领域中的一个重要分支,Voronoi图的一些重要性质如近邻性质、最大空圆性质、控制范围性质等,使其在实际生产和科学研究领域受到了较多的关注。针对复杂的数据集,Voronoi图在查找最近点、求最大空圆、求n个点的凸包、求最小树等问题方面具有重要作用。此外,Voronoi图与其对偶图Delaunay三角剖分已被广泛地应用到几何形体重构、空间数据聚类、空间数据挖掘、图像处理、移动通信、模式识别、机器人运动规划、方向关系分析等领域[1]。在GIS、卫星定位与遥感领域,Voronoi图可表达空间数据信息中的侧向邻近关系,并兼具矢量和连续铺盖数据模型的基本特点,可以很好地管理空间数据,因此成为GIS等领域近年的研究热点之一[2,3]。

随着移动设备和无线网络的广泛应用,移动对象的动态查询迅速发展。其中,最近邻查询成为空间数据库中的研究重点。最近邻查询的研究中大部分都是关于kNN查询以及连续kNN查询的算法[4-7]。国内在空间与时空对象最近邻查询处理技术方面的研究工作也取得了较为丰硕的成果。张丽平等人提出了Voronoi图的构建与受限区域内的最近邻查询方法[8];袁培森、沙朝锋等人研究了一种基于学习的高维数据c-近似最近邻查询算法[9];卢炎生等人提出了一种改进的基于BF的NN算法[10];李松等人提出了动态受限区域内的单纯型连续近邻链查询方法[11]。

但是以上算法都是基于理想的欧氏空间的。在障碍空间中,kNN查询能获取在障碍距离上k个距离查询点q最近的空间数据点。Gao等人[12]通过有效地分割对象行进的路径,提出了障碍空间内连续kNN查询优化方法,该算法的优点是查询速度较快,但是随着数据点的增加其查询的准确性会降低。Sarana Nutanong等[13]利用增量算法来处理障碍空间的最近邻问题,该算法的准确性高,但是在海量数据查询方面尚需进一步提升。Gao等[14]提出了一种基于R-tree的障碍空间反最近邻(RNN)查询处理方法,该方法不需要预计算,通过引入边界区域的概念对查询结果进行高效的剪枝。Li等

到稿日期:2015-03-18 返修日期:2015-07-28 本文受国家自然科学基金资助项目(61370084),黑龙江省自然科学基金资助项目(F201302),黑龙江省教育厅科学技术研究项目(12541128,12531z004)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。

人[15]提出了一种安全的基于区域的方法来进行空间移动的kNN查询，该方法在一些情况下计算量较大。Wang等人[16]提出了移动对象的持续可见k近邻查询。

为了进一步提升查询性能，提出了一种改进的障碍空间中基于Voronoi图的kNN算法，主要是利用Voronoi图做预计算。解决方案包含两个过程：过滤过程和精炼过程。在过滤过程中会生成一个候选集合，在该集合中包含了所有的下一个可能的查询结果。而精炼过程的主要工作是利用障碍距离和邻接生成点对候选集合进行剪枝操作。

2 基本定义

定义1（Voronoi图[3]） 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：$VH(p_i)=\{q|d(q,p_i)\leqslant d(q,p_j)\}$。$d(q,p_i)$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。图1展示了部分Voronoi图。

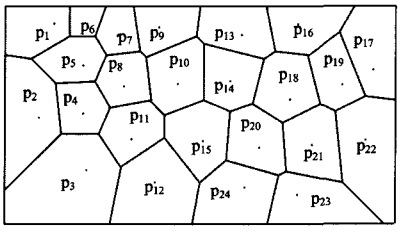

图1 Voronoi图

性质1[3] 如果点q在Voronoi多边形$VH(p_i)$中，则q到生成点p_i的距离小于q到其他生成点的距离。

性质2[3] 2个相邻Voronoi多边形公共边上任一点到这2个多边形的生成点的距离相等。

性质3[3] Voronoi图的每个Voronoi多边形平均有6条边，即每个生成点平均有6个邻接生成点。

定义2（障碍物） 在本文中，空间内的障碍物被抽象为线段形状，且障碍物不相交于数据点。为了便于研究，每个邻接格中只存在一个障碍物。

定义3（障碍距离[17]） 在有障碍物的空间中，欧氏距离记为d_{euc}，若两点p和q之间无障碍物，则障碍距离即两点间的欧氏距离，即$d_{obs}(p,q)=d_{euc}(p,q)$；若$p$和$q$之间有障碍物$O_k$，则障碍距离为这两点绕过障碍物的最短距离，即

$$d_{obs}(p,q)=\min\{d_{euc}(q,O_{k1})+d_{euc}(O_{k1},p_2),d_{euc}(q,O_{k2})+d_{euc}(O_{k2},p_2)\}$$

定义4（最近邻查询[2]） 假设有一n维空间的点集S和一个查询点q，最近邻查询就是找出子集$NN(q,S)$：$NN(q,S)=\{p\in S|\forall s\in S:d(q,p)\leqslant d(q,s)\}$。

定义5（k最近邻查询[2]） 给定n维空间中的一个点集S以及一个查询点q，则q的k最近邻查询的定义可以描述为：$kNN(q)=\{p_1,p_2,\cdots,p_k\}$，其中$p_i\in S$且$i=1,2,\cdots,k$，则对于$\forall p\in S$且$p\notin kNN(q)$，有$D(q,p_i)\leqslant D(q,p)$。

3 障碍空间中基于Voronoi图的kNN查询

3.1 过滤过程

该过程主要是生成下一个可能的kNN候选集合。在计算这个候选集合时，需要用到邻接生成点。之后把查询点到生成点个数小于或等于k的对象加入候选集合，则kNN的查询结果是在候选集合中。性质1证明了这个结论。

性质4 只有查询点q到生成点个数小于或等于k的那些生成点才能放入候选集合中。

证明：假设$count(q,p)$为查询点q到对象p需要经过的最少的生成点个数。当$k=1$时，即查询点q的1NN，而那些能成为q的1NN的数据点p一定是q的邻接生成点，即点p满足$count(q,p)\leqslant 1$；当$k>1$时，假设$count(q,p_x)\leqslant x$，$0\leqslant x\leqslant k$，那么$count(q,p_{k+1})\leqslant k+1$。又由Voronoi图的定义可知，$p_{k+1}$是$p_x\in\{p_1,\cdots,p_k\}$的一个邻接生成点。因此，可以得到$count(q,p_{k+1})\leqslant\max(count(q,p_x))+1\leqslant k+1$，性质4得证，证毕。

根据性质4得到算法1，算法1首先初始化3个队列Computedist、Accessed和CandidateSet，分别用于存放将要计算的数据点及其到查询点的距离、已被访问的节点和候选集合。由$VN(q)$求出查询点q的邻接生成点集合，计算这些生成点到q点的障碍距离，将这些生成点放入Computedist队列中。然后从Computedist队列中取出一个元素i，若它到q点经过的数据点个数不大于k，则将它加入到候选集合CandidateSet中。然后求出i的生成点集合，若这些生成点没有被访问过，则将其记住，放入Computedist和Accessed队列中。最后返回候选集合。

算法1 filter(P,q,k)

```
输入：查询点 q，查询对象集 P，kNN 查询的 k 值
输出：候选集合 CandidateSet
begin
1. Computedist=∅, Accessed=∅, CandidateSet=∅; //初始化 3 个队列
2. while p_x in VN(q) do
3.    add( p_x,1) into Computedist; //将 p_x 及其到查询点 q 的距离加入 Computedist 中
4.    add p_x into Accessed; //标记 p_x 被访问过
5. end while
6. while not_empty(Computedist) do
7.    Pop an element i; //从队列中取出元素 i
8.    if count(q,i)≤k then //判断 q 到点 i 经过的数据点个数是否大于 k
9.       add(i,d(q,i)) into CandidateSet; //小于等于 k，将 i 及其到达查询点 q 的距离放入 CandidateSet 中
10.      while p_x' in VN(i) do
11.         if not_Visited(p_x') then//如果 p_x'没被访问
12.            count(q,p_x')=count(q,i)+1; //count(q,i)加 1
13.         end if
14.         add(count(q,p_x'),p_x') into Computedist; //将查询点 q 及其到达 p_x'的最少数据点个数加入 Computedist 中
15.         add p_x' into Accessed; //标记 p_x'被访问
16.      end while
17.   end if
18.   else continue;
```

```
19. end while
20. return CandidateSet;//返回候选集合
end
```

3.2 精炼过程

在过滤过程完成之后，得到了一个候选集合，在精炼过程中要对该集合进行剪枝操作。利用性质 2、性质 3 对候选集合内的对象进行精炼，对不符合条件的对象进行剪枝。

性质 5 查询点 q 与对象 p 之间经过若干对象如 p_x，连接 p 与 p_x，若无障碍物，则 $count(q,p)$ 增加 1。如果 $count(q,p)\geqslant k$，则对象 p 被剪枝。

证明：查询点 q 与对象 p_x 之间若没有障碍物，则说明查询点 q 到对象 p_x 的欧氏距离一定比到对象 p 的欧氏距离近，若 q 到 p 之间存在障碍物，那么查询点 q 到 p_x 的障碍距离也一定比到 p 的近。当 $k=1$ 时，这样的 p_x 就有可能是 q 的 1NN，而 q 一定不是 p 的 1NN；当 $k\neq 1$ 时，只要 p_x 不小于 k，那么查询点 q 的 kNN 一定不包括 p，即 $count(p,q)\geqslant k$ 时，p 一定不是候选集里的对象，证毕。

根据性质 5 得到算法 2，算法 2 首先查看候选集中的每一个对象 p 与其查询点 q，若 p 与它经过的对象之间有 k 个以上无障碍物的对象，则 p 被剪枝。对于候选集中的每个对象 p，以查询点 q 为圆心，以 $|pq|$ 为半径做圆，若在此候选集中的对象 e 也在这个圆内，判断 p 与 e 之间是否存在障碍物，若对于一个 p，有多于 k 个障碍物，则 p 就从候选集合中移除。

算法 2 prune_by_obs(*CandidateSet*,*q*,*O*,*k*)

```
输入：候选集合 CandidateSet，查询点 q，障碍物集 O，kNN 查询的 k 值
输出：筛选后的候选集合 CandidateSet
begin
1. while p in CandidateSet do
2.   make_Circle(q,d(q,p));//以点 p 为圆心、以 d(q,p)为半径做圆 round(q)
3.   while e in round(q)∩CandidateSet do //e 同时在 round(q)内和候选集合中
4.     if not_obstacle(p,e) then//p 和 e 之间没有障碍物
5.       count++;//计数器加 1
6.     end if
7.     else continue;
8.   end while
9.     if count≥k then//计数器大于等于 k
10.      Remove p from CandidateSet;//将对象 p 从候选集合中移除
11.    end if
12.    else continue;
13. end while
14. return CandidateSet;//返回候选集合
end
```

性质 6 若候选集合中点 p 的邻接生成点都被剪枝了，则点 p 也被剪枝。

证明：假设点 p 在候选集合中，且点 p 的邻接生成点都被剪枝。由 Voronoi 图的性质知，q 到 p 一定经过 p 的邻接生成点，而 $count(q,p)\geqslant k$，从而得到 $count(q,p)\geqslant k+1\geqslant k$，由性质 5 知 p 被剪枝，证毕。

根据性质 6 得到算法 3，算法 3 设置了一个变量来标记对象是否需要被剪枝。若候选集合中任意对象 p 的全部邻接生成点都被剪枝了，则对象 p 也被剪枝。

算法 3 prune_by_neigh(*CandidateSet*,*q*)

```
输入：候选集合 CandidateSet，查询点 q
输出：新的候选集合 CandidateSet
begin
1. flag=0;
2. while p in CandidateSet do
3.   while e∈VN(p) do
4.     if e∈CandidateSet then
5.       flag=1;
6.     end if
7.     else continue;
8.   end while
9.   if flag=0 then
10.    remove p from CandidateSet;//从候选集合 CandidateSet 中移除 p
11.  end if
12.  else continue;
13. end while
14. return CandidateSet;
end
```

3.3 障碍空间中基于 Voronoi 的 kNN 算法描述

下面介绍在障碍空间中，给出查询点 q、数据点集 P、障碍物集 O 的 kNN 查询。

算法 4 首先根据空间的数据点和障碍物画出 Voronoi 图。然后开始进行过滤过程，在过滤过程中只取那些到查询点 q 经过的数据点个数不大于 k 的数据点放入候选集合。接下来是精炼过程，在精炼过程中查看候选集中的点 p 与它到 q 之前的所有点之间的距离，若有不少于 k 个点无障碍物，则这样的点 p_x 就可以从候选集合中取出，若候选集合中存在点 p_x，它的所有邻接生成点都被移出候选集，则此点 p 也被移出候选集。最后对候选集合中的每个对象 p 计算它的第 k 个近邻点 p_k，若查询点 q 到 p 的障碍距离 $d(p,q)<d(p,p_k)$，则 q 的 kNN 里没有 p，因此从候选集中移除 p。返回最终结果。

算法 4 kNN-Obs(*q*,*k*,*P*,*O*)

```
输入：查询点 q，数据集 P，障碍物集合 O，kNN 查询的 k 值
输出：点 q 的 kNN 的结果集合
begin
1.   DrawVoronoi(P,O);//生成 Voronoi 图
2.   while p in CandidateSet do
3.     call filter(P,q,k);//调用算法 1
4.     call prune_by_obs(p,q,O,k);//调用算法 2
5.     call prune_by_neigh(p,q);//调用算法 3
6.     while each p in CandidateSet do
7.       if d(p,q)>d(p,pk) then//pk 是 q 的 k 个最近邻
8.       remove p from CandidateSet;//从候选集合 CandidateSet 中移除 p
9.       end if
10.    end while
11.  end while
12. return CandidateSet;
end
```

3.4 障碍空间中新增点对 kNN 的影响

若数据集 P 增加数据点，那么给定查询点的 kNN 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Voronoi图进行局部重构。若新增加的点在根据算法所画的圆内，重新调用算法4，若不在，则返回原有的查询结果。算法5是新增点对障碍空间中kNN查询的影响算法。

算法5首先将新增加的点加入到数据集 P' 中，然后建立3个空队列 $kNN[m]$、$NewkNN[n]$ 和 $Dist[o]$ 分别用于存储查询点的查询结果、查询点新的 k 最近邻点和查询点与 k 最近邻点的障碍距离。接下来调用kNN-Obs算法，将所得到的 k 最近邻点加入到 $kNN[m]$ 队列中。然后计算 q 到 $kNN[m]$ 队列内的任意 p_x 的最短障碍距离，将 q 到 p_x 的最短障碍距离加入 $Dist[o]$ 队列中。以 q 为圆心、$d(q,p_x)$ 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kNN-Obs算法，将 k 最近邻点存储于 $NewkNN[n]$ 中，然后返回 $NewkNN[n]$。否则返回原来的 $kNN[m]$。

算法5 ADDkNN-Obs(q,k,P,O,w)

输入：查询点q，kNN查询的k值，数据集P，障碍物集合O，新增点w

输出：点q的新的kNN

begin

1. P′=add w into P;//将新增加的点w加入到数据集P中得到新的数据集P′
2. kNN[m]=∅,NewkNN[n]=∅;Dist[o]=∅;//初始化3个队列
3. call kNN-Obs(q,k,P,O);//调用kNN-Obs算法得到q的kNN为 $p_1,p_2,\cdots,p_k$
4. add $p_1,p_2,\cdots,p_k$ into kNN[m];//将 $p_1,p_2,\cdots,p_k$ 加入到kNN[m]队列中
5. $d(p_x,q)=d(p_x,b_z)+d(b_z,q)$;//计算kNN[m]队列中任意的 p_x 与q之间的障碍距离，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(q,d(q,p_x));//以点q为圆心，d(q,p_x)为半径做圆round(q)
8. if w∈round(q) then
9. call kNN-Obs(q,k,P′,O);//重新调用kNN-Obs算法得到q的新的kNN为 $p_1,p_2,\cdots,p_k$
10. add $p_1,p_2,\cdots,p_k$ into NewkNN[n];//将 $p_1,p_2,\cdots,p_k$ 加入到NewkNN[n]队列中
11. return NewkNN[n];//返回NewkNN[n]
12. end if
13. else return kNN[m];//如果新增加点w不在圆内，返回kNN[m]

end

3.5 障碍空间中删除点对kNN的影响

若从数据集 P 中删除数据点，则对查询点的kNN有一定的影响，若删除点在结果的集合中，则重新调用kNN-Obs算法；若不在，则返回原来的结果。根据以上的分析可得到解决删除点后对障碍空间中kNN的影响问题的算法。算法6是删除点后对障碍空间中kNN查询的影响算法。

算法6首先在数据集 P 中删除任意点得到数据集 P'，然后建立两个空队列 $kNN[m]$ 和 $NewkNN[n]$ 分别用于存储查询点的查询结果和查询点新的 k 最近邻点。接下来调用kNN-Obs算法，将所得到的 k 最近邻点加入 $kNN[m]$ 队列中。如果删除点位于 $kNN[m]$ 中，重新调用kNN-Obs算法，将 k 最近邻点存储于 $NewkNN[n]$ 中，返回 $NewkNN[n]$；否则，返回原来的 $kNN[m]$。

算法6 DENkNN-Obs(q,k,P,O,d)

输入：查询点q，数据集P，障碍物集合O，kNN查询的k值，删除点d(d∈P)

输出：查询点q新的kNN

begin

1. P′=remove d from P;//数据集P中删除d得到数据集P′
2. kNN[m]=∅,NewkNN[n]=∅;//初始化两个队列
3. call kNN-Obs(q,k,P,O);//调用kNN-Obs算法得到点q的kNN为 $p_1,p_2,\cdots,p_k$
4. add $p_1,p_2,\cdots,p_i$ into kNN[m];//将 $p_1,p_2,\cdots,p_i$ 加入到kNN[m]队列中
5. if d∈kNN[m] then
6. call kNN-Obs(q,k,P′,O);//重新调用kNN-Obs算法得到新的kNN
7. add $p_1,p_2,\cdots,p_k$ into NewkNN[n];//将 $p_1,p_2,\cdots,p_k$ 加入到NewkNN[n]队列中
8. return NewkNN[n];//返回NewkNN[n]
9. end if
10. else return kNN[m];//如果删除点不在结果集合中，返回kNN[m]

end

4 实验与分析

本节将所提出的kNN-Obs算法、ADDkNN-Obs算法、DENkNN-Obs算法与文献[13]中的算法(PostPruning算法)进行了比较分析，为便于比较，对实验算法细节进行了局部调整。每次实验的结果为执行100次的平均效果。实验运行环境为：2.0GHz CPU、4GB内存、Windows 7操作系统。测试数据集 P、障碍集 O 和查询点 q 是由数据生成器产生的随机分布的空间数据，其中数据集是点数据构成的，障碍集是由线段构成的。

首先是 k 值的影响。图2和图3分别给出了 k 值的变化对CPU时间和磁盘I/O代价的影响。实验将 P 的数据量设为20000个，O 的数据量设为10000个。

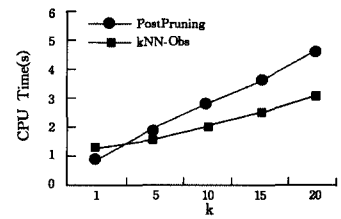

图2 k 值对CPU运行时间的影响

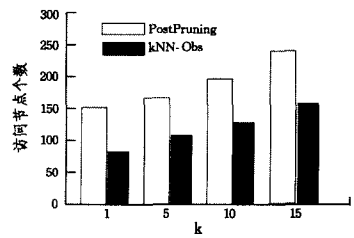

图3 k 值对磁盘I/O代价的影响

由图2给出了 k 值的变化对CPU时间的影响。k 值从1变化到20的过程中，CPU时间随着 k 值的增大而逐渐变大。kNN-Obs算法需要构造Voronoi图，所以当 k 较小时，CPU时间比PostPruning算法长，但随着 k 值的增加，kNN-Obs算

法的优势就逐渐显现出来。

图 3 给出了 k 值的变化对磁盘 I/O 代价的影响。两种算法的磁盘 I/O 都是随着 k 的增加而增加。这是因为 k 值越大，需要访问的数据点越多，磁盘 I/O 代价越大。但是由于 kNN-Obs 算法只访问候选集中的数据点，因此在一些情况下，其磁盘 I/O 代价相对会小。

其次是障碍物个数的影响，图 4 和图 5 分别给出了障碍物个数的变化对 CPU 时间和磁盘 I/O 代价的影响。实验将 P 的数据量设为 30000 个，k 值设为 10。

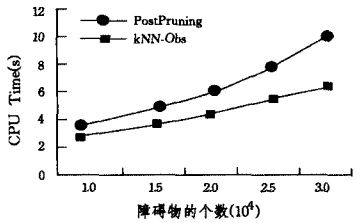

图 4　障碍物的个数对于 CPU 时间的影响

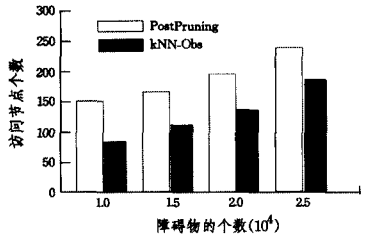

图 5　障碍物个数对磁盘 I/O 代价的影响

由图 4 可知，障碍物较少时，kNN-Obs 算法的优势并不明显，但是随着障碍物个数的增加，kNN-Obs 算法的优势才显现出来。因为障碍物的个数越多，需要计算的障碍距离越多，CPU 时间越长。而计算障碍距离要比计算欧氏距离用的时间更多，PostPruning 算法需要计算一些数据点的障碍距离，这比 kNN-Obs 算法计算的障碍距离要多，所以 kNN-Obs 算法在一些情况下略好于 PostPruning 算法。

图 5 给出了不同数量的障碍物对磁盘 I/O 代价的影响。随着障碍物个数的增加，两种方法的 I/O 代价都随之增加，但是 kNN-Obs 算法由于有过滤过程和精炼过程，有效地减少了障碍物的访问数量，因此磁盘 I/O 代价在一些情况下比 PostPruning 算法的代价小。

最后，评价 ADDkNN-Obs 算法和 DENkNN-Obs 算法的性能。实验将 k 取 20，P 的数据量取 10000 个。每次实验都是随机添加或删除一点。

表 1 显示了在数据集合中加入任意点后的 ADDkNN-Obs 算法与对比算法在计算 kNN 结果时的性能对比，执行时间是取 100 次测试的平均值。对比数据可以发现 ADDkNN-Obs 算法处理在数据集中增加点后的 k 最近邻查询时优于 PostPruning 算法。

表 1　ADDkNN-Obs 与 PostPruning 性能对比

算法	执行时间(s)	P
kNN-Obs	2.062	10000
PostPruning	3.032	10000
ADDkNN-Obs	2.526	10001
PostPruning	5.077	10001

表 2 显示在数据集合中删除点后的 DENkNN-Obs 算法与对比算法计算 kNN 结果时的性能对比，执行时间同样是取 100 次测试的平均值。由数据分析可知，DENkNN-Obs 算法在处理数据集中删除点后的 k 最近邻查询时优于 Post-Pruning 算法。

表 2　DENkNN-Obs 与 PostPruning 性能对比

算法	执行时间(s)	P
kNN-Obs	3.096	20000
PostPruning	4.779	20000
DENkNN-Obs	3.959	19999
PostPruning	6.569	19999

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，基于位置的服务(Location-Based-Service，LBS)得到广泛的应用。而 kNN 是 LBS 中最常用的查询类型，具有广泛的应用前景。本文分析了现有 kNN 查询方法存在的不足，在此基础上对基于障碍空间的 kNN 查询方法进行了研究。现实生活中的大多数查询请求都不是在理想的欧氏空间中，并且现有算法随着数据点的增加其查询的时间会增加较大。为了有效处理障碍空间中的 kNN 问题，本文提出基于 Voronoi 图的 kNN-Obs 算法，解决方案包含两个过程：过滤过程和精炼过程。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进一步的剪枝。实验证明在一定情况下，kNN-Obs 算法提高了障碍空间下的 kNN 查询效率。未来的研究重点主要集中在以下两点：

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. 路网中基于 Voronoi 图的 NN 方法及几种变体结构。

参 考 文 献

[1] 郝忠孝. 时空数据库查询与推理[M]. 北京：科学出版社，2010

[2] 李松，张丽平，孙冬璞. 空间关系查询与分析[M]. 哈尔滨：哈尔滨工业大学出版社，2011

[3] Sacl J R, Urrutia J. Vorouoi diagrams. handbook ou computatioual geometry[M]. Ottawa: Elsevier Science, 2000: 201-290

[4] Jegou H, Inria R, Schmid C. Product Quantization for Nearest Neighbor Search[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2011, 33(1): 117-128

[5] Geng Zhao, Xuan K F, Johanna R, et al. VIC. Voronoi-Based Continuous k Nearest Neighbor Search in Mobile Navigation [J]. IEEE Transactions on Industrial Electronics, 2011, 58(6): 2247-2257

[6] Esmaeili M M, Ward R K, Fatourechi M. A Fast Approximate Nearest Neighbor Search Algorithm in the Hamming Space[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2012: 2481-2488

[7] Zheng Li-gang, Qiu Guo-ping, Huang Ji-wu, et al. Fast and accurate Nearest Neighbor search in the manifolds of symmetric positive definite matrices[C]// Acoustics, Speech and Signal Processing. 2014: 3804-3808

[8] Zhang Li-ping, Zhao Ji-qiao, Li Song, et al. Research on Methods of Construction of Voronoi Diagram and Nearest Neighbor Query in Constrained Regions[J]. Computer Science, 2014, 41(9): 220-224, 247(in Chinese)
张丽平，赵纪桥，李松，等. Voronoi 图的构建与受限区域内的最近邻查询方法研究[J]. 计算机科学，2014，41(9)：220-224，247

(下转第 187 页)

ta. SIGMOD,2009:193-206

[2] Antunes C, Oliveira A L. Generalization of pattern growth methods for sequential pattern mining with gap constraints[M]. Machine Learning and Data Mining in Pattern Recognition. 2003:239-251

[3] Pei J, Han J, Mortazavi B, et al. Prefixspan: mining sequential patterns efficiently by prefix-projected pattern growth[C]// Proceedings of the 17th International Conference on Data Engineering(ICDE). 2001:215-226

[4] Wu E, Diao Y, Rizvi S. High performance complex event processing over streams[C]//Proceedings of the 32th SIGMOD International Conference on Management of Data. SIGMOD, 2006:407-418

[5] Alex D, Robert R, Subrahmanian V S. Probabilistic temporal databases[J]. ACM Transaction on Database Systems, 2001, 26(1):41-95

[6] Barga R S, Goldstein J, Ali M, et al. Consistent streaming through time: a vision for event streamprocessing[C]// The 3rd Biennial Conference on Innovative Data Systems Research. IDAR, 2007

[7] Wu S, Chen Y, Mining nonambiguous temporal patterns for interval-based events[J]. IEEE Transactions on Knowledge and Data Engineering, 2007, 19(6):742-758

[8] Patel D, Hsu W, Lee M L. Mining relationships among interval-based events for classification[C]//Proceedings of the 34th SIGMOD International Conference on Management of Data. SIGMOD, 2008:393-404

[9] Hammad M A, et al. Scheduling for shared window joins over data streams[C]// The 29th International Conference on Very Large Data Bases. 2003, 29:297-308

[10] Liu M, Li M, Golovnya D, et al. Sequence pattern query processing over out-of-order event streams[C]// Proceedings of the 25th International Conference on Data Engineering (ICDE). 2009:274-295

[11] Eyerman S, Eeckhout L, Karkhanis T, et al. A mechanistic performance model for superscalar out-of-order processors[J]. ACM Transactions on Computer Systems(TOCS), 2009, 27(2): 824-833

[12] Wang F, Liu S, Liu P. Complex RFID event processing[J]. The International Journal on Very Large Data Bases(VLDBJ), 2009, 18(4):913-931

[13] Papapetrou P, Kollios G, Sclaroff S, et al. Mining frequent arrangements of temporal intervals[J]. Knowledge & Information Systems, 2009, 21(2):133-171

[14] Ding L, Mehta N, Rundensteiner E A, et al. Joining punctuated streams[M] // Advances in Database Technology (EDBT). 2004:587-604

[15] Kam P S, Fu A W. Discovering temporal patterns for interval-based events[C]// Proceedings of the 2nd International Conference on Data Warehousing and Knowledge Discovery. 2000: 317-326

[16] Barga R S, Goldstein J, Ali M, et al. Consistent streaming through time: a vision for event stream processing[C]// The 3rd Biennial Conference on Innovative Data Systems Research. 2006:363-374

[17] Zhou C J, Meng X F. A framework of complex event detection and operation in pervasive computing[C]//The Third SIGMOD PhD Workshop on Innovative Database Research. 2009

(上接第 178 页)

[9] Yuan P S, Sha C F, Wang X L, et al. c-Approximate nearest neighbor query algorithm based on learning for high-dimensional data[J]. Journal of Software, 2012, 23(8): 2018-2031(in Chinese)
袁培森,沙朝锋,王晓玲,等. 一种基于学习的高维数据 c-近似最近邻查询算法[J]. 软件学报,2012,23(8):2018-2031

[10] Lu Yan-sheng, He Ya-jun, Pan Peng. Improvement of the EINN Algorithm for NN Query Traversing[J]. Computer Engineering and Science, 2005, 27(7):62-64(in Chinese)
卢炎生,何亚军,潘鹏. EINN 最近邻居查询索引遍历算法改进[J]. 计算机工程与科学,2005,27(7):62-64

[11] Li Song, Zhang Li-ping. Simple Continues Near Neighbor Chain Query in Dynamic Constrained Regions[J]. Computer Science, 2014, 41(6):136-141(in Chinese)
李松,张丽平. 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

[12] Gao Y, Zheng B. Continuous obstructed nearest neighbor queries in spatial databases[C]// Proceedings of the 28th ACM SIGMOD International Conference of Management of Data(Sigmod'09). New York, USA, 2009:577-590

[13] Nutanong S, Tanin E, Zhang Rui. Incremental evaluation of visible nearest neighbor queries[J]. IEEE Transactions on Knowledge Engineering, 2010, 22(5):665-681

[14] Gao Y, Yang J C, Chen G, et al. On efficient obstructed reverse nearest neighbor query processing[C]// Proc. of the 19th Int'l Conf. on Advances in Geographic Information Systems. Chicago: ACM Press, 2011:191-120

[15] Li Chuan-wen, Gu Yu, Qi Jian-zhong, et al. A safe region based approach to moving KNN queries in obstructed space[J]. Knowledge and Information Systems, 2014, 24(3):179-195

[16] Wang Yan-qiu, Zhang Rui, Xu Chuan-fei, et al. Continuous visible k nearest neighbor query on moving objects[J]. Information Systems, 2014, 44(8):1-21

[17] Yu Xiao-nan, Gu Yu, Zhang Tian-cheng, et al. A Method for Reverse k-Nearest-Neighbor Queries in Obstructed Spaces[J]. Chinese Journal of Computers, 2011, 34(10): 1917-1925(in Chinese)
于晓楠,谷峪,张天成,等. 一种障碍空间中的反 k 最近邻查询方法[J]. 计算机学报,2011,34(10):1917-1925