GRkNN:空间数据库中组反 k 最近邻查询

宋晓宇 于程程 孙焕良 许景科
(沈阳建筑大学信息与控制工程学 沈阳 110168)

摘 要 反 k 最近邻(Reverse k-Nearest-Neighbor,RkNN)查询是在 k 最近邻(k-Nearest-Neighbor,kNN)查询问题的基础上产生的,获得将查询对象作为 kNN 的数据对象集合,RkNN 可以用于评价查询对象的影响力.根据实际应用中需要查询一组对象的 RkNN,如评价连锁店或商业区的影响.文中提出了针对空间数据库的组反 k 最近邻(Group RkNN,GRkNN)的概念,并设计了相关算法.查询点集合是一组邻近的空间对象,计算查询对象的最小覆盖圆,将最小覆盖圆中的对象作为一个整体进行过滤,设计了基于 R 树的剪枝方法,通过提炼获取了最终的 GRkNN 结果.针对真实数据集进行的大量实验表明,提出的 GRkNN 算法的效率明显优于目前最好的 RkNN 算法.

关键词 反最近邻;GRkNN 查询;R 树;最小覆盖圆
中图法分类号 TP311 **DOI 号**: 10.3724/SP.J.1016.2010.02229

GRkNN:Group Reverse k-Nearest-Neighbor Query in Spatial Databases

SONG Xiao-Yu YU Cheng-Cheng SUN Huan-Liang XU Jing-Ke
(*Department of Information and Control Engineering, Shenyang Jianzhu University, Shenyang* 110168)

Abstract Reverse k-Nearest-Neighbor(RkNN) query is proposed based on the k-Nearest-Neighbor (kNN) query, which finds objects that take the query object as one of their kNN. RkNN query can be used to evaluate the influence of the query objects. According to the practical application that RkNN of a group of objects are queried, such as evaluating the influence of chains or shopping centers, the authors propose the concept of Group RkNN(GRkNN) in spatial database and design the corresponding algorithms. Given a group of nearby space objects as the query input, the authors compute the smallest circle enclosing the query objects in the first, and consider the objects in the circle as a whole. Then, they design a filter method based on R-tree, and a refinement process is subsequently used to get the query's final GRkNN query results. A large of experiments according to real data sets, show that the efficiency of the proposed method for GRkNN outperforms the state-of-the-art algorithm.

Keywords reverse k-Nearest-Neighbor; group reverse k-Nearest-Neighbor; R-tree; the smallest circle

1 引 言

随着位置传感装置的广泛发展(如 GPS 接收),基于位置的服务得到了广泛的应用[1],RkNN 查询起到了重要作用.给定一数据集 P 和一个查询点 q,q 的 RkNN 是数据集 P 中的将 q 作为 kNN 的数据对象.RkNN 查询通常用来发现查询对象的影响集.例如,一家超市连锁店要在某地新开一家超市(假设认为人们一般选择就近购物),该超市所吸引的顾客

收稿日期:2010-06-11;最终修改稿收到日期:2010-11-03.本课题得到国家自然科学基金(61070024)、国家"十一五"科技支撑计划(2008BAJ08B08-04,2006BAJ11B07-01)、辽宁省自然科学基金(20071004)资助.**宋晓宇**,男,1963 年生,博士,教授,主要研究领域为地理信息系统和智能优化算法.E-mail:sxy@sjzu.edu.cn.**于程程**,女,1986 年生,硕士研究生,主要研究方向为空间数据库.**孙焕良**,男,1969 年生,博士,副教授,主要研究方向为空间数据库和数据挖掘.**许景科**,男,1976 年生,讲师,主要研究方向为地理信息系统.

群就是 RkNN 查询所要找的影响集. RkNN 查询可以分为单色 RkNN(monochromatic RkNN)和双色 RkNN(bichromatic RkNN),单色 RkNN 就是查询点所在数据集和查询结果所在数据集是一个数据集,而双色 RkNN 处理两个数据集. 现有的对于 RkNN 查询的解决方案[2-7]均是针对单个查询点进行处理. 考虑一个实际情况,查询一个商业区(由一组空间对象组成)的 RkNN 结果,或查询一个连锁店的 RkNN 结果. 针对这一应用需求,本文提出一种新的针对空间数据库的 RkNN 查询——GRkNN 查询(Group RkNN,GRkNN),该查询可以用于评价一组对象的影响. 解决该问题的直接方法是计算单个查询对象的 RkNN,再求出所有查询结果的并集,即为查询点集合的 RkNN 结果. 显然,当组内对象较多时,由于未能利用组内对象的紧邻关系进行查询,查询效率会较差.

本文采用 R 树[8]索引空间对象,采用过滤-提炼的框架模型. 在过滤阶段,计算包含所有查询点的最小覆盖圆,以该圆为整体结合一组数据集子集 S 来计算该组查询点 RkNN 的搜索区域,在这个查询区域的所有对象都是该 GRkNN 查询的候选集,而计算这个搜索区域的算法是 CINCH;在提炼阶段,对 Range-k 验证方法[2]进行改进应用到 GRkNN 查询中,去除非 GRkNN 查询结果的候选者,从而最终得到查询结果. 实验结果表明,提出的 GRkNN 算法的效率明显优于目前最好的 RkNN 算法.

本文第 2 节介绍相关工作;第 3 节是问题定义;第 4 节具体介绍 GRkNN 查询算法;第 5 节介绍双色 GRkNN 算法实现;第 6 节介绍本文所提出的算法的性能测试;第 7 节总结全文.

2 相关工作

Korn 和 Muthukrishnan[3]提出了 RNN 查询的概念,即(RkNN,k=1),并给出了求解 RNN 的查询方法. 设计了一个新索引结构 RNN 树以求得查询对象 q 的反最近邻. 为了解决两个索引结构的问题,Yang 和 Lin[4]提出了将这两个索引结构合并在同一个称为 RdNN 树的索引结构中,其叶节点存储了数据点的邻接圆,中间节点存储了子树中的点到其最近邻的距离的最大值. 以上方法依赖于预计算,近年来有许多学者已经提出了一些解决方法[5-7],均使用了过滤-提炼的模型框架,过滤阶段剪枝确定不包含候选者的部分,提炼阶段去除错误项.

2.1 过滤方法

2.1.1 60 度剪枝法

由 Stanoi[5]等人提出的 60 度剪枝法,其主要思想是:将围绕一个 RNN 查询点 q 的空间等分为 6 个区域(S_1,S_2,…,S_6),然后在每个区域中,只有 q 的最邻近位置才可能是 q 的反邻近位置. 由此,q 的 RNN 候选者被限制在 6 个区域中 q 的最邻近位置上. 该方法不需要特殊的索引结构,避免了数据点更新时频繁的计算和修改索引结构,相对于文献[3-4]效率较高. 同时,可以处理移动对象的 RkNN,也可以扩展到 $k>1$ 的情况. 但是随着空间维数的增加,需要搜索的区域空间会成指数增长,甚至在 3 维情况下,其效率会变得很低.

2.1.2 TPL 剪枝法

Tao[6]等人提出的 TPL 剪枝法,其基本思想如图 1 所示:查询点 q 与 P 中任意点 p 做垂直平分线,分空间为 q-half-plane 与 p-half-plane(也称为 $\bar{q}$-half-plane),在 p-half-plane 内的点不可能是 q 的 RNN. TPL 过滤过程如下:用广度优先的方法遍历 R 树,再以节点到查询点 q 的距离的升序堆中检索潜在候选者,因为 RNN(q)很可能在 q 的附近,半平面的概念用来剪枝不存在候选者的 *MBRs*. 每次被剪枝的条目被加入到提炼集 S_{rfn} 中,在提存阶段,S_{rfn} 中的条目被用来排除错误项. TPL 剪枝法比 60 度剪枝法有更强的剪枝能力[6]. 同时,TPL 剪枝技术也可扩展到 $k>1$ 的情况. 与前面介绍的方法相比,该方法效率提高了,但是该方法只适用于静态查询点的情况. 此外,TPL 方案花费了相当多的时间去寻找可用于剪枝的对象.

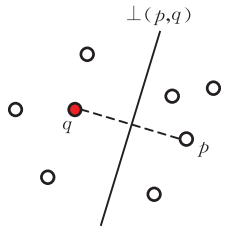

图 1 TPL 剪枝法

2.1.3 INCH 空间缩减法

Tan[7]等人也采用过滤-提炼的模型框架,在 TPL 的提示下提出了 INCH 空间缩减法算法,它的基本思想是:计算出一个 RkNN 查询的搜索范围(从查询结果候选者中确定),这个搜索区域约束了搜索空间,并且每当一个新的结果候选者被找到的时候这个搜索区域都会被缩小,最终确定出候选者集合. 具体 INCH 缩减过程如下:首先找到一个数

据集的子集 S，每个子集与查询点做垂直平分线从而将空间划分成多个多边形，然后根据多边形所在的 $\bar{q}$-half-planes 的个数来计算每个多边形的水平值，将所有多边形水平值小于 k 的连接起来形成搜索区域. INCH 空间缩减法利用 TPL 剪枝的基本原理，将单个用于剪枝的对象扩展到一个对象集，减小了搜索区域，提高了查询效率. 根据实验和分析的结果显示出，INCH 空间缩减法比 TPL 剪枝法有更好的空间剪枝能力[7]. 但是该方法也只针对单一查询点的剪枝法.

2.2　提炼方法

提炼方法包括两种：(1) 基础 kNN 法，主要是在文献[5]中使用，计算每个候选点的 kNN，并查看结果是否为查询点，如果有，则该候选点为给查询点的 RkNN 的查询结果，反之，则不是. (2) Range-k 验证法，对每个候选点进行核对，判断核对的候选点到查询点的距离比到其它候选点的距离小的个数是否小于 k，如果是，则该候选点为该查询点的 RkNN 的查询结果，反之，则不是. 这个方法被提出在文献[2]中，虽然它是针对解决连续 RkNN 查询的，但是 Range-k 验证法比基础 kNN 方法更有效.

现有的解决方法均是针对单个查询对象的 RkNN，本文采用过滤-提炼的框架模型设计了新的算法实现了 GRkNN 查询. 过滤阶段利用了 INCH 空间缩减法的基本原理，但不同的是将查询点集合的最小覆盖圆作为整体来确定搜索区域，从而避免了当查询点集是一组邻近的数据对象时搜索区域的频繁重叠情况，提高了查询效率. 提炼阶段则采用 Range 验证法来排除错误项.

3　问题定义

本节给出 kNN、RkNN 等查询的基本概念，从而引出 GRkNN 查询的定义. 给定二维空间数据集 P 和 R，其中包含形如 $p(x,y)$ 的数据点，$dist(p,q)$ 代表两个点 p、q 之间的距离.

定义 1.　kNN(q,k,P)[9] (k 最近邻查询). 给定数据集 P，当且仅当一个对象 $p\in P$ 满足 $\{o\in P|dist(o,q)<dist(p,q)\}<k$ 时，p 是 kNN(q,k,P) 的一个结果.

kNN 查询实际就是查找离查询对象最近的 k 个对象，它的查询结果一定是 k 个.

定义 2.　单色 RkNN(q,k,P)[3]. 给定一数据集 P 和一个查询点 q，若对象 p 的 kNN 包含 q，则对象 p 是 RkNN(q) 的查询结果. 具体定义形式为

$$\text{R}k\text{NN}(q,k,P)=\{p\in P|q\in k\text{NN}(p,k,P)\}.$$

单色 RkNN 查询是获得给定的数据集中将查询点作为 k 最近邻的一个结果的点的集合. 查询对象 q 可以是数据集 P 中的点，也可以不是.

定义 3.　双色 RkNN(q,k,P,R)[2]. 它包含两种数据集 P 和 R，P 是查询点 q 所在的数据集，R 是查询结果所在的数据集，对于每个 $r(r\in R)$ 的 kNN(r,k,P) 中，如果包含 q，则此 r 即为双色 RkNN(q,k,P,R) 的一个结果. 具体定义形式为

$$\text{BR}k\text{NN}(q,k,P,R)=\{r\in R|q\in k\text{NN}(r,k,P)\}.$$

双色 RkNN 查询是获得查询数据集中将查询点作为 k 最近邻的结果集中点的集合.

定义 4.　Range-$k(p,k,r)$[2]. 判断 p 点周围的点到 p 的距离小于 r 的点的个数是否小于 k，如果 $\{o|dist(p,o)<r\}<k$ 成立，则返回 true，否则返回 false.

本文在过滤阶段使用 Range-k 验证法来排除错误项.

定义 5.　GRkNN(Group RkNN). 给定一组查询点集 $Q=\{q_1,q_2,\cdots,q_n\}$ 和一个数据集 P，当一个对象 $p\in P$ 在 P 上的 kNN 的结果包括 Q 中的任意一个时，p 就是 GRkNN(Q,k) 的一个结果，具体定义形式为

$$\text{GR}k\text{NN}(Q,k)=\{\exists q\in Q,\exists p\in P|q\in k\text{NN}(p,k,P)\}.$$

对于 GRkNN 的查询，不论在单色或是双色的情况下，结果就是查询点集中每个查询点的 RkNN 或 BRkNN 结果的并集. 查询点集 Q 中的查询点可以是数据集 P 中的点，也可以不是.

4　GRkNN 查询算法

本文采用 R 树作为索引结构，受 TPL 空间剪枝法以及 INCH 空间缩减法的启发，本文 GRkNN 查询采用过滤-提炼的框架模型，过滤阶段以查询点集合的最小覆盖圆为整体确定最小覆盖圆的 RRkNN 的搜索区域从而得到 GRkNN 的候选集合，提炼阶段采用 Range-k 验证法排除错误项，从而得到最终结果集.

4.1　最小覆盖圆

给定点集的最小覆盖圆，即包含给定点集且半径最小的圆，如图 2 所示，圆 O 就是点集 $\{q_1,q_2,q_3,q_4\}$ 的最小覆盖圆. 本文采用汪卫和王文平在文献[10]中所提出的计算给定点集的最小覆盖圆的方法

来计算查询点集的最小覆盖圆.

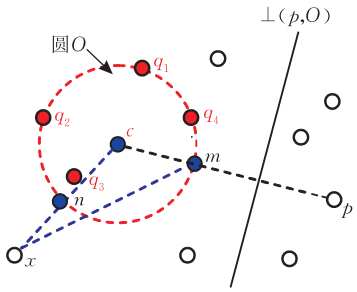

图 2 点与圆的关系

下面我们给出圆的一些相关定义、定理及其证明,从而作为以查询点集合的最小覆盖圆为整体来进行过滤剪枝的理论依据.

定义 6. 点与圆的垂直平分线. 某点 p 与一个以点 c 为中心 r 为半径的圆 O 的垂直平分线,就是直线 $l(p,c)$ 与圆的交点 m 与 p 的垂直平分线,用 $\perp(p,O)$ 表示. $l(p,c)$ 表示 p 与 c 两点形成的直线. 如图 2 所示,圆 O 与 p 的垂直平分线就是 $\perp(p,O)$.

一个点 p 与一个以点 c 为中心 r 为半径的圆 O 间做垂直平分线将空间分成两部分,包含点 p 的半平面叫作点 p-half-plane(也统称为 $\bar{O}$-half-plane),包含圆 O 的半平面叫作圆 O-half-plane.

定义 7. 点到圆的距离. 某点 p 到一个以点 c 为中心 r 为半径的圆 O 的距离记为 $d(p,O)$,当 $dist(p,c)>r$ 时,$d(p,O)=dist(p,c)-r$;当 $dist(p,c)\leq r$ 时,$d(p,O)=0$. 如图 2 所示,以点 c 为中心 r 为半径的圆 O 外有一点 x,点 x 到圆 O 的距离 $d(x,O)=dist(x,n)$,n 是直线 $l(x,c)$ 与圆 O 的交点.

定理 1. 给定点 p 和以点 c 为中心 r 为半径的圆 O,在 p-half-plane 内的点(不包括 p)不可能是圆 O 内任意点的 R1NN.

证明. 本文通过反证法来证明定理 1. 如图 2 所示,圆 O 与点 p 做垂直平分线,根据定义 2,实际上是直线 $l(p,c)$ 与圆的交点 m 与 p 的垂直平分线,假设 O-half-plane 内存在一点 x 使得 $dist(x,p)<d(x,O)$,其中 x 存在两种情况,一是 x 在圆 O 内或边界上时,$d(x,O)=0$,可推出 $dist(x,p)<0$,这与事实 $dist(x,p)\geq 0$ 相矛盾,因此假设不成立;另一种情况是 x 在圆 O 外时,圆心 c、点 m 和点 x 形成一个三角形,而 $dist(c,x)=dist(n,x)+dist(n,c)=d(O,x)+r$,$dist(c,m)=r$,其中 $d(O,x)>0$,根据两边之差大于第三边的几何原理,则有 $dist(c,x)-dist(c,m)=d(O,x)<dist(x,m)$,而根据假设 $dist(x,p)<d(x,O)$,则有 $dist(x,p)<dist(x,m)$,由于假设与事实 $dist(x,p)>dist(x,m)$ 相矛盾,假设不成立. 即得证. 证毕.

根据定理 1 可知,在 p-half-plane 内的点不包括 p,不可能是圆 O 的 R1NN 的结果,因为相对于圆 O 来说 p-half-plane 内的点离 p 更近.

4.2 过滤过程

GRkNN 查询采用过滤-提炼框架模型,过滤阶段需要快速剪枝一些对象而得到一组候选集合. 如果将所有查询点分开考虑会将问题复杂化,因此将所有查询点的最小覆盖圆作为整体考虑,然后进行剪枝.

首先计算 GRkNN 查询点集 Q 的最小覆盖圆 O,将圆 O 作为一个整体,利用给定的空间数据集子集 $S\subseteq P$,做 S 中每个点与圆 O 的垂直平分线,将空间分割成多个多边形,如图 3(a)所示,其中 $S=\{p_1,p_2,p_3,p_4,p_5\}$. 然后再计算每个多边形 *level* 值,多边形的 *level* 值就是该多边形所在的 $\bar{O}$-half-planes 个数,如图 3(b)所示,如果多边形的 *level* 值小于 k(GRkNN 查询的 k 值)就说明该多边形至多在 $k-1$ 个 $\bar{O}$-half-planes 内,而这样的多边形就有可能包含该圆内所有点的 RkNN 结果,将这样的多边形放在一起就形成了圆 O 内所有点的 RkNN 的候选区域,如图 3(c)所示的粗黑色边框区域. 由于候选区域包含的是圆 O 内所有点的 RkNN 结果,而 GRkNN 查询的查询点集 Q 都在圆 O 内或圆 O 上,因此该区域也是查询点集 Q 的 GRkNN 的结果候选区域,而这也解释了为什么将查询点集合的最小覆盖圆而不是任意一个包含所有查询的圆作为一个整体来考虑解

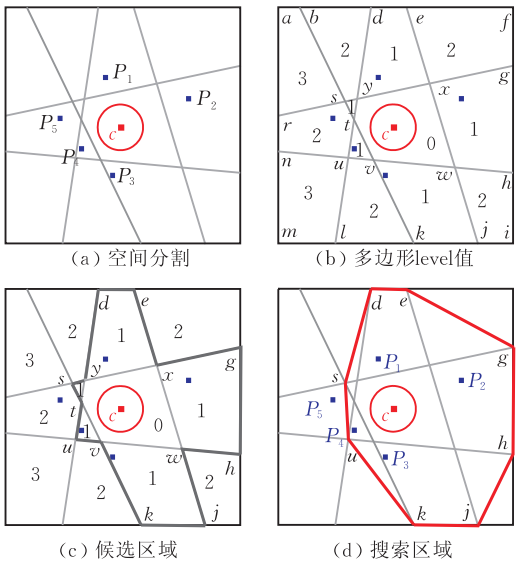

图 3 CINCH 算法

决 GRkNN 查询问题的原因，由于包含所有查询点的圆越大，计算所得的搜索区域就可能越大，候选集合就越大. 由于由多个多边形组成的候选区域不利于存储利用，因此形成如图 3(d)所示红色边框的凸外包多边形，这个凸外包多边形就是 GRkNN 的搜索区域. 在搜索区域内的所有数据对象和 S 都是查询的候选者.

4.3　CINCH：以圆为整体计算 GRkNN 的搜索区域

本文基于过滤-提炼框架模型实现 GRkNN 查询. 过滤阶段确定的搜索区域，实际上是最小覆盖圆内所有点的 RkNN 查询的搜索区域，由于 GRkNN 的查询点集包含在圆内，因此该搜索区域内的数据点也是 GRkNN 查询的候选者，所以该搜索区域也是 GRkNN 查询的搜索区域. 最后再在提炼阶段将非 GRkNN 查询结果的错误项排除.

本文提出的 CINCH (Circle INtersections' Convex Hull)算法就是将所有查询点的最小覆盖圆作为整体来计算 GRkNN 的搜索区域的算法. CINCH 算法的基本思想是：给定一个对象集 $S\subseteq P$，P 是整个数据集，对于 S 中所有对象分别做它们与该 GRkNN 最小覆盖圆 O 的垂直平分线，从而将空间分为多个多边形，计算每个多边形所在的 $\overline{O}$-half-planes 个数，数值小于 k 的多边形组成的外包凸多边形即为该 GRkNN 的搜索区域. 搜索区域内的对象和 S 的并集就是 GRkNN 的候选集. CINCH 算法的伪代码如算法 1 所示.

算法 1.　CINCH(O,k,S).

输入：整个数据集合 P 的子集 S，$S\subseteq P$

　　GRkNN 查询的 k 值

　　包含查询点集的最小覆盖圆 O

输出：以凸多边形的形式返回查询的搜索区域

1. 计算 S 中每个对象与 O 的垂直平分线；
2. 计算交点；
3. 计算每个交点的 *level* 值；
4. 把 *level* 值小于 k 的交点放入集合 I 中；
5. 计算返回 I 内所有点形成的凸外包多边形区域.

第 1 步. 分别做 S 中所有点与 O 的垂直平分线，将整个空间分为多个多边形，如图 3(a)所示.

第 2 步. 计算垂直平分线间的交点和垂直平分线与空间边界的交点以及空间边界间的交点，如图 3(b)所示交点$\{a,b,d,\cdots,n,r,\cdots,x\}$.

第 3 步. 计算每个交点的 *level* 值，某交点的 *level* 值等于该点所在的 $\overline{O}$-half-planes 的个数，不包括形成这个点的垂直平分线相应的 $\overline{O}$-half-planes. 例如，在图 3(b)中，$level(v)=0$；$level(d)=1$.

第 4 步. 把 *level* 值小于 k 的交点放入集合 I 中.

第 5 步. 利用集合 I 内的点形成的多边形就是候选区域，如图 3(c)所示的$\{d,e,x,g,h,w,j,k,v,u,t,s,y\}$区域，但由于候选区域是由多个多边形组成，在计算候选时不便于存储利用，因此形成如图 3(d)所示的粗边框$\{d,e,g,h,j,k,u,s\}$凸外包多边形，这个凸外包多边形就是搜索区域. 如果 S 中对象的个数小于 k，则搜索区域就是整个空间.

定理 2.　CINCH 算法的复杂度是 $O(m^2\log(m))$，其中 m 是 S 中对象的个数.

证明.　第 1 步计算垂直平分线的代价是 $O(m)$. 第 2 步计算交点的代价是 $O(m^2)$. 第 3 步计算交点的 *level* 值是 $O(m^2)$，因为 S 中的对象是逐个加入的，对于计算交点 *level* 值是随交点增加而计算和更新的. 第 4 步集合 I 的代价大小至少是 $O(m^2)$. 计算凸多边形的复杂度是 $O(n\log(n))$，n 是形成凸多边形点的个数；因此第 5 步的代价至少是 $O(m^2\log(m))$.

证毕.

定理 3.　CINCH 算法不会误拒结果对象.

证明.　只需证明 CINCH 算法所确定的搜索区域之外不包含任何结果. 根据文献[7]提出的定理 2，给定一个数据集 P、一个 RkNN 查询 RkNN(q,k)及一个数据对象子集 $S\subseteq P$，设 C 为由算法 INCH(q,k,S)所计算形成的搜索区域，对于在 $P-S$ 内的任意点 p，当且仅当 p 在 C 内或边界上时，有 $level(p)<k$. 本文提出的定理 1 说明 CINCH 算法与文献[7]提出 INCH 算法依据的原理相同，由此该定理也适用于 CINCH 算法. 既当 C 为算法 CINCH(O,k,S)计算形成的搜索区域时，该定理同样成立.

根据文献[7]提出的定理 2，对于在 $P-S$ 内的任意点，当 p 在 C 外时 p 的 *level* 值一定不小于 k，即 p 所在的 $\overline{O}$-half-planes 的个数大于等于 k，p 就不可能是圆 O 内任一点的 RkNN 结果；换句话说，在搜索区域 C 外一定不包含结果对象. 因此该定理也证明了 CINCH 算法的正确性，这里不再赘述.

4.4　过滤算法

过滤阶段，根据给定的 GRkNN(Q,k)和一组数据对象 $S\subseteq P$，首先计算查询集 Q 的最小覆盖圆 O，然后利用 CINCH 算法计算查询的搜索区域 SR. 基于 S 和 SR，查询候选集合就是 $S\cup S'$，其中 S' 是 $SR\cap(P-S)$.

S 的选择能影响查询候选集的大小，还能影响接下来从 $S\cup S'$ 中排除错误候选者的提炼阶段的效率. 而对于利用 CINCH 算法计算而得的搜索区域

有以下性质:(1)随着 S 中对象的增加,搜索区域会越来越缩减.(2)将距离圆 O 更近的对象加入到 S 中,搜索区域会更小.

根据以上两个性质采用迭代的方法,首先令 S 为空,然后初步依次增加圆 O 外距离圆 O 最近的数据对象到 S 中,从而缩减搜索空间.我们利用 R 树实现过滤算法,对于扩展 S 的方法是给圆 O 更近的点更高的权限来加入 S.使用 R 树作为数据的索引结构,广度优先遍历 R 树,数据以升序排列,在圆 O 外的数据对象依次加入 S 中,这个具体的以 R 树作为索引结构的 Filter 算法的伪代码如算法 2 所示.

算法 2. FCINCH-Filter(O,k).

输入:包含查询点集的最小覆盖圆 O
GRkNN 查询的 k 值
输出:包含查询结果的候选集合
1. $S:=\varnothing$; $S_{cnd}:=\varnothing$; $SR:=$整个数据空间;
2. H 为(e,key)的最小堆;
3. 插入(R 树根节点,0)到 H 中;
4. while(H 不为空){
5. 从 H 中提取出一个(p,key);
6. if ($p.MBR$ 与 SR 相交) then
7. if (p 是一个索引节点) then
8. for (p 中每个子节点 m)
9. if ($m.MBR$ 与 SR 相交) then
10. 插入(m,$min\text{-}dist(m,c)$)到 H 中;
11. else if (p 是叶子节点) then
12. for (p 中每一数据对象 n)
13. if (n 与 SR 相交) then
14. 插入(n,$dist(n,c)$)到 H 中;
15. else{ $S_{cnd}:=S_{cnd}\cup\{p\}$;
16. if ($key>0$) then{
17. $S:=S\cup\{p\}$;
18. $SR:=$CINCH(O,k,S);} //end if }}
19. return S_{cnd},S.

算法 FCINCH-Filter 为了广度优先遍历 R 树,设置了一个形为(p,key)的最小堆 H,其中 p 既有可能是 R 树的一个索引节点,也可能是 R 树的叶子节点,或是一个数据对象;key 则是 p 与圆 O 的圆心 c 的最小距离,r 是圆 O 的半径.S 初始化为空,用来存放形成搜索区域 SR 所用到的数据集,S_{cnd} 初始化为空,用来存放候选者的集合(第 1 步).将 R 树根节点插入 H 中对 H 进行初始化(第 2 步),遍历 R 树的过程中,通常从 H 中选择离圆心 c 最近的对象或是项目(第 5 步),每当从 H 中选择出的一个数据对象 p 时就将 p 加入到 S_{cnd} 中(第 15 步),如果 p 对应的 key 值大于 0(即 p 点在圆 O 外),则将 p 加入到 S 中(第 17 步),因为圆 O 内所有数据对象都是一个整体,找圆 O 外的数据对象加入 S 中才有意义($key=0$ 表示在圆内或在圆上),同时 CINCH 算法将搜索区域缩减(第 18 步).也就是说候选者集合 S_{cnd} 是 S 和圆 O 内所有数据点的集合.

由于搜索空间是随着数据对象不断地加入到 S 中而迭代缩减的,由此 CINCH 算法中每当有新的数据对象加入到 S 中时,就需要更新维护搜索区域和相关信息(例如,垂直平分线、交点和它们的 $level$ 值),并且优化计算量,也就是将第 17 步进行优化.

4.5 提炼阶段

提炼阶段,排除在过滤阶段得到的 GRkNN 的结果候选集 S_{cnd} 的错误项,提炼算法伪代码如算法 3 所示.

算法 3. FCINCH-Refine(Q,O,k,S,S_{cnd}).

输入:查询点集 Q
包含查询点集的最小覆盖圆 O
GRkNN 查询的 k 值
形成搜索区域的数据集 S,$S\subseteq S_{cnd}$
候选结果集 S_{cnd}
输出:GRkNN 结果集
1. $SR:=$CINCH(O,$k+1$,S);
2. for (S_{cnd} 中每个数据对象 p)
3. if (p 不在 SR 内) then
4. $S_{cnd}:=S_{cnd}-\{P\}$;
5. for (S_{cnd} 中每个数据对象 p){
6. if ($p\in Q$) then
7. $r:=\min\{dist(p,\{Q-p\})\}$;
8. else
9. $r:=\min\{dist(p,\{Q\})\}$;
10. if (Range-k(p,k,r}) return false) then
11. $S_{cnd}:=S_{cnd}-\{P\}$}; //end for
12. return S_{cnd}.

候选者提炼分为两个阶段:第 1 个阶段是(第 1~4 步),用算法 CINCH(O,$k+1$,S)计算新的区域来排除错误项.当一点 p 被选择加入到 S 中,S_{new} 是 p 加入到 S 后 S 的数据集,S_{old} 是 p 没加入 S 之前 S 的数据集,它们的关系式 $S_{new}-S_{old}=\{p\}$,p 在以 S_{old} 为参数计算而来的搜索区域中,也就是说 p 是在至少 $k-1$ 个 $\bar{O}$-half-planes 内,随着不断地有新的数据对象加入到 S 而最终变成现在的 S,p 所在的 $\bar{O}$-half-planes的个数就很有可能大于 $k-1$ 个,而 p 又一定在它自己的 p-half-plane 内,如果 p 是错误项,则需要满足 p 至少在 $k+1$ 个 $\bar{O}$-half-planes 内的条件,因此我们计算 $k+1$ 条件下的搜索空间用来排除错误项.第 2 个阶段是(第 5~11 步),使用 Range-k 验证法分别验证每一个候选对象来进一步

排除错误项. 候选集合 S_{cnd} 中有可能包括查询点集 Q 中的一个或多个，因为 Q 中某一查询点可能是另一查询点的 RkNN，因此当候选者是查询点时(第 6 步)，函数 Range-k 的参数 r 为 $\min\{dist(p,\{Q-p\}\}$(第 7 步)，其中 $\min\{dist(p,\{Q-p\}\}$ 是 p 到数据集 $\{Q-p\}$ 中所有点的距离的最小值，因为 p 是查询点，它距离除它之外的哪个查询点最近，就可能是那个查询点的 RkNN；如果候选者不是查询点，则函数 Range-k 的参数 r 为 $\min\{dist(p,Q)\}$(第 9 步)，其中 $\min\{dist(p,Q)\}$ 是 p 到查询点集 Q 中所有查询点的距离的最小值，p 距离 Q 中哪个查询点更近，p 就可能是那个查询点的 RkNN.

5　双色 GRkNN 查询算法

本节介绍了如何将 FCINCH 算法扩展来解决双色 GRkNN 查询的问题. 双色 GRkNN 查询算法也是采用过滤-提炼框架.

相对于 GRkNN 查询，双色 GRkNN 查询有两个数据集 P 和 R，P 是包含查询点集合 Q 的数据集，R 是包含查询结果的数据集. 例如，$r\in R$，如果 Q 中任意一个查询点是 r 在 P 上的 kNN 中的一个，则 r 是双色 RkNN 的一个查询结果. 而对于双色 GRkNN 查询也是将 Q 的最小覆盖圆 O 作为整体考虑计算.

双色 GRkNN 查询算法如算法 4 所示. 在数据集 P 上应用 FCINCH-Filter 算法计算得到查询点集为 Q 的 GRkNN 所形成搜索区域的集合 S(第 1 步)，然后用 CINCH 算法计算搜索区域 SR(第 2 步)，从而也确定了在数据集 R 上寻找候选者的搜索区域 SR，第 3 步就找到候选者集合 R'，以上就是双色 GRkNN 的过滤阶段. 在数据集 P 上应用 Range-k 验证法判断每个候选者是否为查询结果，从而最终得到结果集 T，该过程是双色 GRkNN 的提炼阶段.

算法 4.　FCINCH-B(O,k,P,R).

输入：查询点集 Q 的最小覆盖圆 O
　　GRkNN 查询的 k 值
　　包含查询点 Q 的数据集 P
　　包含结果集的数据集 R

输出：组 BRkNN 查询的结果集

1. S':=数据集 P 上 FCINCH-Filter(O,k)的 S；
2. SR:=CINCH(O,k,S')；
3. R':=数据集 R 内找到包含在 SR 内的数据的集合；
4. $T:=\varnothing$；
5. for(R'中每个对象 r)
6. 　if(P 上的 Range-k($r,k,\min\{dist(r,Q)\}$) return true) then
7. $T:=T\cup\{r\}$；
8. return T.

6　算法性能测试

本节将测试所提出的计算 GRkNN 查询的 FCINCH 算法和计算双色 GRkNN 查询的 FCINCH-B 算法的性能. 测试在查询点集大小相同情况下，查询点集形成的最小覆盖圆大小对算法效率的影响；测试当 k 变化、查询点集合大小变化、R 树页容量变化的影响下对算法效率的影响，然后同 FINCH 算法相比较. 算法采用 Java 实现，实验在处理器为 P4 2.4GHz，内存为 1GB，操作系统为 WindowsXP 的微机上进行.

测试中使用到北美居民点 NAp、北美文化地标 NAp 这两个点的数据集[8]，将它们分别以 4096 个字节的页容量用 R 树建立其索引，数据集的详细信息如表 1 所示.

表 1　数据集

数据集	数据量	R 树大小/KB
NAp	24254	1636
NAc	9203	600

本文以实际运行时间作为性能指标. 运行时间包括 CPU 时间和 I/O 时间. 下面的图中 FINCH 的运行时间则都是每个查询点的运行时间之和，而候选结果也是每个查询点的候选结果之和.

6.1　GRkNN 查询结果

如图 4 所示，设定查询点数量是 5，$k=5$，而随着 5 个查询点形成的最小覆盖圆越来越大(用圆的半径来衡量，半径取其近似值)，FCINCH 算法也有其使用范围，如图当半径大于 0.8 时，FINCH 算法的性能就优于 FCINCH 算法，因为 FINCH 算法计算每个查询点的性能相似，有几个查询点就会使计算这组查询点的 RkNN 的运行时间成倍数增加，与这些查询点的位置无关. 因此当查询点个数相同时，运行时间基本相同. 但 FCINCH 却不同，它是与查询点的位置相关的，在查询点个数确定的情况下，查询点集所形成的最小覆盖圆越大，算法的性能就越低，因为该算法是以查询点集最小覆盖圆为整体进行剪枝的，同时整个圆也将划入搜索区域，圆越大搜索区域越大，候选结果就越多，如图 5 所示，FCINCH

算法随着圆越来越大，候选者也越多，而 FINCH 却变化不大.

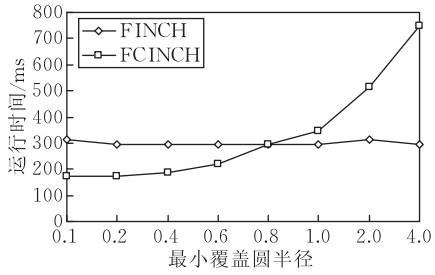

图 4 最小覆盖圆大小对算法性能的影响

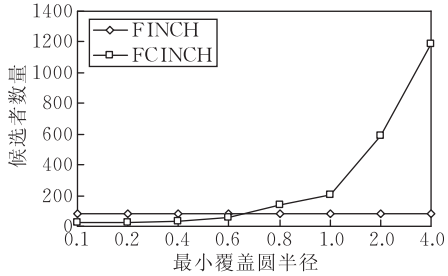

图 5 最小覆盖圆大小对候选者数量的影响

但是当查询点间相距较近的时候，FCINCH 算法就会比 FINCH 算法的性能更好. 如图 6 所示，在数据集 NAp 上，给定 5 个查询点，随着 k 值从 1～10，算法在 NAp 数据集上执行的性能对比. 图 7 是相同查询的候选者数量的比较，随着 k 的增大，FINCH 得到的候选集合比 FINCH 更多. 这两个图显示出在处理查询点间相距较近的 GRkNN 问题时 FCINCH 比 FINCH 更有效. 因为给定的 5 个查询点如果用 FINCH 算法思想解决，则是一个一个计算其搜索区域从而得到其候选集合，但当查询点较近时，会有搜索区域相重叠，而每个查询点的候选集合也会有很多重复，这样就降低了算法性能，而 FCINCH 算法避免了这样的问题发生，当查询点间的相近程度达到某种程度的时候，将其划定在它们的最小覆盖圆内，将这个圆作为整体考虑就避免了重复查找相同候选者的可能. 因此在这种情况下，处理 GRkNN 问题时 FCINCH 比 FINCH 更有效.

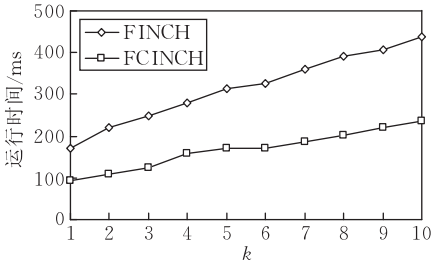

图 6 k 值对算法性能影响

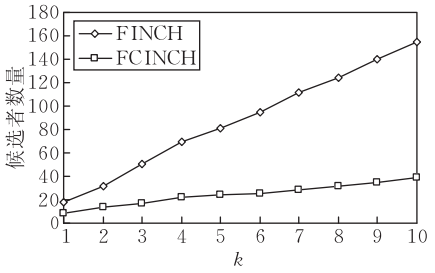

图 7 k 值对候选者数量的影响

如图 8 所示，在数据集 NAp 上，当 $k=5$、查询点数量不同时(所有查询点在某一较小范围内)，FCINCH 的性能优于 FINCH 的查询性能，因为在 FCINCH 算法中查询点都放在它们的最小覆盖圆内做整体考虑，所有查询点都在一个较小的范围内，每个查询点搜索区域交叠情况频繁，因此剪枝后形成的搜索区域基本相同，所以当查询数量增大时，并没有对性能造成影响，但 FINCH 却不同，每计算一个查询点的时间基本相同，而随着查询点增加时，其运行时间将成倍数增加.

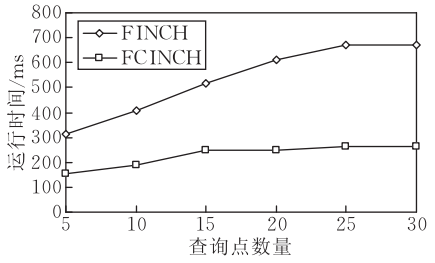

图 8 查询点数量对算法的性能影响

如图 9 所示，在数据集 NAp 上给定 5 个查询点(所有查询点在某一较小范围内)，查询的 k 值为 5，随着形成的 R 树的页容量不同，运行性能并没有相差很多，但仍可看出，FCINCH 性能优于 FINCH.

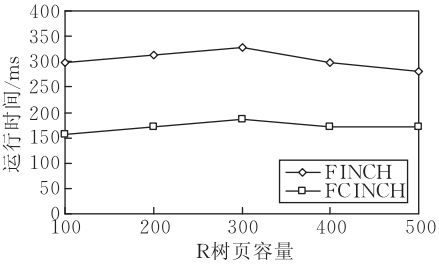

图 9 页容量对算法性能的影响

6.2 双色 GRkNN 查询结果

如图 10 所示，给定 5 个查询点，查询点所在集合为 NAc，结果集为 NAp，最小覆盖圆半径小于 0.2，随着 k 值增大，可看出 FCINCH-B 的性能优于 FINCH-B. k 值越大，FINCH-B 的每个查询点的搜索区域重叠的就越多，候选结果就越多，如图 11 所示.

而当查询点距离较近时，FCINCH-B 与 FCINCH 相同，都能发挥将查询点最小覆盖圆作为整体考虑减值的优势，因此相对于 FINCH-B 来说，FCINCH-B 的算法性能随着 k 值的增大却使得候选者数量较少.

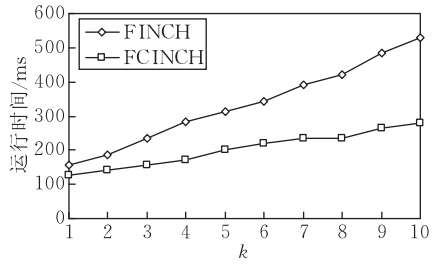

图 10　k 值对算法性能的影响(双色 GRkNN)

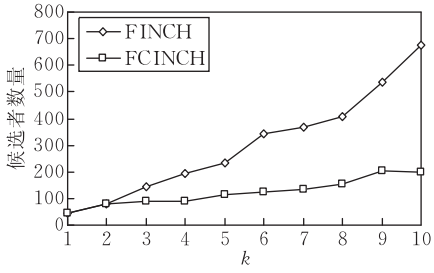

图 11　k 值对候选者数量的影响(双色 GRkNN)

7　结　论

根据实际应用需求提出了 GRkNN 查询的概念及相关算法. 采用 R 树[8]索引结构，利用过滤-提炼的模型结构来实现 GRkNN 查询. 在过滤阶段，将查询点集的最小覆盖圆作为整体考虑剪枝，利用 CINCH 算法确定搜索区域，最终确定了结果候选集合. 在提炼阶段，采用 Range-k 验证[2]，排除错误结果的数据点，最终得到结果集合. 并采用真实数据集对算法的性能进行了测试，实验表明所提出算法的效率在查询点集是一组邻近的空间对象时优于直接算法. GRkNN 查询具有重要的实际应用价值，可以用于评价集合对象的影响力，为决策支持提供了新的方法.

致　谢　感谢文献[7]的作者提供的 FINCH 的算法源码!

参　考　文　献

[1] Rao B, Minakakis L. Evolution of mobile location-based services. Association for Computing Machinery, 2003, 46(12): 61-65

[2] Wu Wei, Chee Fei Yang, Chan Yong, Tan Kian-Lee. Continuous reverse k-Nearest-Neighbor monitoring//Proceedings of the 9th International Conference on Mobile Data Management. Beijing, 2008: 132-139

[3] Korn F, Muthukrishnan S. Influence sets based on reverse nearest neighbor queries//Special Interest Group on Management of Data. Dallas, Texas, USA, 2000: 201-212

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

[5] Stanoi I, Agrawal D, El Abbadi A. Reverse nearest neighbor queries for dynamic databases//Special Interest Group on Management of Data Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD). Dallas, USA, 2000: 44-53

[6] Tao Y, Papadias D, Lian X. Reverse kNN search in arbitrary dimensionality//Proceedings of the Very Large Data Bases. Toronto, Canada, 2004: 744-755

[7] Wu Wei, Yang Fei, Chan Chee Yong, Tan Kian-Lee. FINCH: Evaluating reverse k-Nearest-Neighbor queries on location data//Proceedings of the Very Large Data Bases. Auckland, New Zealand, 2008: 1056-1067

[8] Guttman Q. R_Tree: A dynamic index structure for spatial searching//Association for Computing Machinery Special Interest Group on Management of Data Conference on Management of Data. Boston, MA, 1984: 47-57

[9] Roussopoulos N, Kelley S, Vincent F. Nearest neighbor queries//Association for Computing Machinery Special Interest Group on Management of Data. San Jose, CA, 1995: 71-79

[10] Wang Wei, Wang Wen-Ping. An algorithm for finding the smallest circle containing all points in a given point set. Journal of Software, 2000, 11(9): 1237-1240(in Chinese)
(汪卫，王文平. 求一个包含点集所有点的最小圆的算法. 软件学报，2000，9(11)：1237-1240)

SONG Xiao-Yu, born in 1963, Ph.D., professor. His research interests include intelligent optimization, GIS and data mining.

YU Cheng-Cheng, born in 1986, M. S. candidate. Her main research interests focus on spatial databases.

SUN Huan-Liang, born in 1969, Ph. D., associate professor. His research interests include spatial database and data mining.

XU Jing-Ke, born in 1976, lecturer. His research interests focus on GIS.

Background

With the wide deployment of location sensing devices (such as GPS receivers), location based services are getting popular. Location related queries play an important role in location based services. One such query type is Reverse-k-Nearest-Neighbor (RkNN). The problem of RkNN query is playing a significant role in many applications of spatial database.

Many researchers have proposed some solutions of RkNN query, but a common feature of these solutions is to query RkNN of a single object. According to the practical application that RkNN of a group of objects are queried, such as evaluating the influence of chains or shopping centers, the authors propose the concept of Group RkNN(GRkNN) query in spatial database. If the conventional method is used to tackle the problem of GRkNN query, the efficiency would be poorer when query objects are more due to ignoring the nearby relation of the query objects. Therefore, the authors design an algorithm of GRkNN query, and compute the smallest circle enclosing the query objects in the first, then the objects in the circle as a whole to filter. Then, a refinement process is subsequently used to get the query's final GRkNN query results. This algorithm of evaluating GRkNN queries can also be used to evaluate bichromatic RkNN queries.

The major aim of this paper is to find a flexible and effective algorithm of GRkNN query, more over, to establish a sound theoretical basis for the further research on query technology in spatial database. The work is supported by the National Natural Science Foundation of China (61070024), the National Technology R&D Program in the 11th Five year Plan of china (2008BAJ08B08-04, 2006BAJ11B07-01), the Natural Science Foundation of Liaoning Province (20071004).