软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2014,25(8):1806−1816 [doi: 10.13328/j.cnki.jos.004459]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

一种障碍空间数据库中的连续反 k 近邻查询方法*

谷 峪[1,2], 于晓楠[1,2], 于 戈[1,2]

[1](东北大学 信息科学与工程学院,辽宁 沈阳 110819)

[2](医学影像计算教育部重点实验室(东北大学),辽宁 沈阳 110819)

通讯作者: 谷峪, E-mail: guyu@ise.neu.edu.cn

摘 要: 随着智能移动设备和无线定位技术的飞速发展,使用基于位置服务应用的用户越来越多.特别地,不同于传统的针对固定位置的快照查询,移动的用户往往基于移动轨迹发出连续的查询.在真实和虚拟的空间环境中,障碍物的影响都是广泛存在的,障碍空间内的查询处理技术得到了越来越多的关注,其中,障碍空间内的连续反 k 近邻查询处理有着重要的应用.对障碍空间中的连续反 k 近邻查询问题进行了定义和系统的研究,通过定义控制点和分割点,提出了针对该问题的处理框架.进一步地,提出了一系列的过滤和求精算法,包括剪枝数据集、获取障碍物、剪枝和计算控制点和更新结果集等处理策略.基于多种数据集对所提出的算法进行了实验评估.与针对每个数据点进行 k 近邻计算的基本方法相比,这些方法可以大幅度提高查询处理的 CPU 和 I/O 效率.

关键词: 连续查询;反 k 近邻;障碍空间;查询优化;控制点

中图法分类号: TP311

中文引用格式: 谷峪,于晓楠,于戈.一种障碍空间数据库中的连续反 k 近邻查询方法.软件学报,2014,25(8):1806−1816. http://www.jos.org.cn/1000-9825/4459.htm

英文引用格式: Gu Y, Yu XN, Yu G. Method for continuous reverse k-nearest neighbor queries in obstructed spatial databases. Ruan Jian Xue Bao/Journal of Software, 2014,25(8):1806−1816 (in Chinese). http://www.jos.org.cn/1000-9825/4459.htm

Method for Continuous Reverse k-Nearest Neighbor Queries in Obstructed Spatial Databases

GU Yu[1,2], YU Xiao-Nan[1,2], YU Ge[1,2]

[1](School of Information Science and Engineering, Northeastern University, Shenyang 110819, China)

[2](Key Laboratory of Medical Image Computing of Ministry of Education (Northeastern University), Shenyang 110819, China)

Corresponding author: GU Yu, E-mail: guyu@ise.neu.edu.cn

Abstract: With the rapid development of smart mobile devices and wireless location techniques, more and more users tend to attempt location-based service. Specifically, mobile users usually request continuous queries based on moving trajectories instead of traditional snap-shot queries for fixed locations. As obstacles can be found everywhere in the real-world or virtual space, more and more attentions has been paid on query processing techniques in the obstructed space. Notably, continuous reverse k-nearest neighbor queries in obstructed space are widely used. This paper presents an in-depth study on the problem of moving reverse k-nearest neighbor queries in obstructed spatial databases. By defining control points and split points, the processing framework for this problem is constructed. Furthermore, several pruning and verification algorithms, including data points reduction, obstacles retrieving, control points calculating and results set updating, are proposed to improve the query efficiency. Extensive experimental evaluation is conducted based on various datasets. Compared with the basic method which computes the k-nearest neighbors for each data point, the proposed methods can significantly improve CPU and I/O efficiency.

Key words: continous query; reverse k-nearest neighbor; obstructed space; query optimization; control point

* 基金项目: 国家重点基础研究发展计划(973)(2012CB316201); 国家自然科学基金(61003058, 61033007, 61202086); 中央高校基本科研业务费专项资金(N130404010)

收稿时间: 2012-10-09; 修改时间: 2013-02-04; 定稿时间: 2013-07-09

近年来,随着智能移动设备的快速发展,基于位置服务技术得到了广泛的应用.空间数据库作为重要的支撑技术[1],为各类位置相关的查询请求,例如 k 近邻查询[2−4]、skyline 查询[5,6]和反 k 近邻查询[7,8]等提供了高效的数据获取服务.特别地,在这些查询中,反 k 近邻查询用来搜索那些 k 近邻里面包含查询点 q 的空间数据点,从而反映了查询点对哪些空间数据点影响较大,被证实在空间决策支持、资源分配和数据挖掘方面有着广泛的应用.然而,之前的大多数空间查询工作往往考虑理想的欧式空间和路网空间.实际上,地面、室内甚至虚拟空间内移动的物体一般都会受到地理条件的限制(例如建筑、湖泊等),因此,准确的距离计算需要考虑障碍物的因素.

在各类空间查询中,连续的反 k 近邻查询是一类相对复杂的查询,可以支持高级的分析和预测应用.例如,在地震的救灾重建工作中,需要在灾区设置医疗和餐饮安置点,如果用户驾驶一辆载满配送药物和食物的车,查询从出发点到终止点的路线上,可以配送给哪些安置点药物和食物.显然,这些安置点需要距离查询线段比较近,也就是在安置点的 k 近邻范围内(否则,可以从其他安置点搬运物品),从而为优化配送做决策支持.类似的查询可以支持冰山碰撞预测、野生动物领袖发现和虚拟游戏伙伴邀请等应用场景.而在这些空间中,障碍物都是广泛存在的,准确地计算需要考虑障碍物的影响.从上面的例子可以看出:给定一个轨迹作为查询区间,连续反 k 近邻问题即连续地给出查询点在区间内的反 k 近邻数据对象.因为移动对象在查询区间上的反 k 近邻是分段变化的,因此该问题即转化为对查询区间进行合理的分割,使得每个分割的区间具有相同的反 k 近邻查询结果.

本文针对障碍空间数据库中的连续反 k 近邻查询进行研究,引入了控制点和分割点的概念,提出了剪枝数据集、获取障碍物、剪枝和计算控制点和更新结果集的一套高效处理策略,从而给出了障碍空间数据库中的连续反 k 近邻查询的解决方法.

本文第 1 节介绍相关工作.第 2 节定义障碍空间的连续 RkNN 查询和相关概念.第 3 节详细描述障碍空间中连续 RkNN 查询的处理过程,阐述剪枝和求精的方法.第 4 节给出实验结果以及分析.第 5 节给出总结.

1 相关工作

对于无障碍空间 RkNN 查询,文献[7−9]提出了无需预计算的 RkNN 查询方法.Stanoi 等人[7]给出了在查询点 q 处将空间划分成 6 等份(每份为 60°)的方法,可以证明,在每一个区域离查询点 q 的第 k 个最近邻以外的区域都可以被剪枝,得到这些未被剪枝的区域里的数据点就成为 RkNN 查询的候选点.Tao 等人[8]提出了使用中垂线性质剪枝搜索空间的方法,有效地剪枝那些处在 k 个以上中垂线的半区间的数据点.Wu 等人[9]提出一种新的剪枝方法——FINCH 算法,该方法使用一个多边形来近似未被剪枝区域.尽管这些方法能够高效地执行 RkNN 查询,但它们都是在理想的欧氏空间,即无障碍物空间中查询.Cheema 等人[10]对欧式空间和路网空间内的连续反 k 近邻问题进行了系统的研究,提出了高效的空间剪枝方法.以上这些技术并不适合应用到障碍空间中.

近年来,障碍空间的 kNN 查询技术得到了广泛的研究.障碍空间中,kNN 查询是获取在障碍距离上 k 个距离查询点 q 最近的空间数据点.Zhang 等人[11]给出了在障碍空间中基于 R-tree 方法解决常见的空间查询,例如范围查询、最近邻查询、e-距离连接查询、最近对查询.Xia 等人[12]提出了障碍最近邻查询方法,用增量的方式处理只与查询有关的数据点和障碍物,因此过滤掉了大量数据点和障碍物.Gao 等人[13]通过有效的分割对象行进的路径,提出了障碍空间内连续 kNN 查询的优化方法.在文献[14]中,Gao 又对文献[13]中提出的算法进行了扩展研究和理论分析.由于 RkNN 比起 kNN 查询更加复杂,因此,kNN 查询的处理技术并不适合应用于 RkNN 查询.此外,有些工作关注了障碍空间内的可见 k 近邻搜索[4,15],提出了有效的优化方法.但是,由于可见近邻只考虑那些没被障碍物遮挡的数据对象,从问题定义到解决方案与本文关注的基于障碍距离的查询方法完全不同.还有一些工作对障碍空间的数据挖掘问题进行了研究,例如,文献[16]研究了障碍空间内不确定数据的聚类算法.

在最近的一项工作中,我们给出一种基于障碍 Voronoi 图的高效搜索固定点 RkNN 对象的方法[17].通过有效的剪枝规则,减少查询处理反 kNN 的代价.同时,Gao 等人提出了一种基于 R-tree 的障碍空间反最近邻(RNN)查询处理方法[18].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.这两项工作针对的是快照查询,与连续查询的侧重点不同.对于连续查询,如果周期性地采用快照查询,效率很低,需要提出高效的模型和算法来划分移动轨迹,从而提供实时的查询结果.

2 问题定义

本节首先给出一些重要的定义.

定义 1(可视区间(visible region)). 一个点 p 在查询区间 q 上的可视区间 $R_{p,q}$ 由 q 上所有与 p 的连线均不与障碍物相交的点构成.

如图 1 中在查询区间 q 上点 p 的可视区间 $R_{p,q}$ 为$[S,s_1]\cup[s_3,s_1]\cup[s_8,E]$,在这个区间上的每一个点 p'与 p 的连线均不会与空间内的任何障碍物相交.

定义 2(障碍控制点(obstructed control point)). 在障碍空间中,给定数据点 p,障碍物集合 O,在 p 的可视区间内,p 定义为控制点,可视区间定义为 p 的控制区间;在 p 的不可视区间内,障碍物端点 cp 定义为 p 在区间 R 上的一个障碍控制点,当且仅当点 p 到区间 R 的最短路径经过 cp,且 cp 对于区间 R 内的每个点是可见的,此时,R 定义为 cp 的控制区间.

如图 1 所示,障碍物 O_1 的左端点 u 就是控制区间$[s_1,s_2]$的控制点,记为$\langle u,[s_1,s_2]\rangle$,也就是说,对于区间$[s_1,s_2]$上的任意一点 p',从点 p 到 p'的最短路径一定经过点 u,且这段距离 $d_{obs}(p,p')$等于 $d_{obs}(p,u)$与 $d_{obs}(u,p')$之和.

定义 3(障碍分割点(obstructed split point)). 在障碍空间内,查询区间为 $q=[S,E]$,障碍物集合为 O,给定区间$[s_1,m]$和$[m,s_2](S\leqslant s_1\leqslant m\leqslant s_2\leqslant E)$,如果$[s_1,m]$所有查询点的障碍反 k 近邻结果一样,记为 P_1,$[m,s_2]$所有查询点的障碍反 k 近邻结果一样,记为 P_2,且 $P_1!=P_2$,则点 m 即为障碍空间上关于查询区间 q 的障碍反 k 近邻查询的一个分割点.所有的分割点将查询区间分割成多个分割区间.每个分割区间内所有查询点的反 k 近邻查询结果保持不变.

例如,图 2 中关于查询 q 的障碍分割点为$\{s_2\}$,即在区间$[S,s_2]$上,点 p 为 q 的反最近邻;在区间$[s_2,E]$上,点 p 不再是 q 的反最近邻,点 p'则成为 q 的反最近邻.因此在分割点 s_2 处,查询结果有所改变.

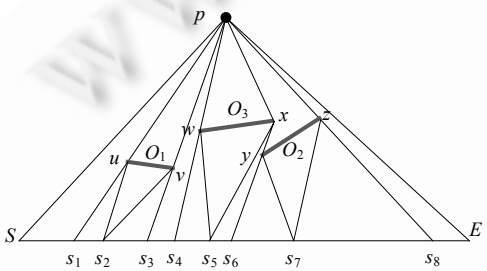

Fig.1 Illustration for obstructed control point

图 1 障碍控制点示意图

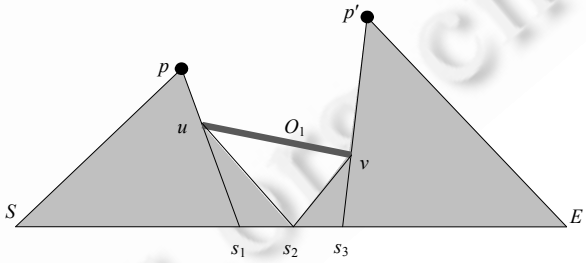

Fig.2 Illustration for obstructed split point

图 2 障碍分割点示意图

定义 4(障碍空间的连续反 k 近邻(continuous obstructed reverse k-nearest neighbor,简称 CORkNN)查询). 在障碍空间中,给定数据点集 P,障碍物集合 O,查询区间 $q=[S,E]$,CORkNN 即查询点在区间$[S,E]$上做连续查询,返回结果集为每个分割区间所对应的查询点的反 k 近邻数据对象.

3 CORkNN 查询处理过程

本节提出连续障碍反 k 近邻查询的处理方法.通过剪枝数据点,对每个数据点求其对应的控制点和分割点,并在计算控制点的过程中加入剪枝求精的方法,减少了计算控制点的时间和空间代价,最后对结果集进行更新,得到最终结果.

3.1 剪枝数据集

本节针对障碍空间中的反 k 近邻查询,将大量的空间数据点进行精简处理,从而减少数据点计算的个数,减少了后面小节中对每个数据点求解控制点的总计算代价.

定理 1. 若数据点到查询可视区间的最短距离大于此数据点到其障碍 k 最近邻(OkNN)点的距离,则此数据

点一定不是 q 的 ORkNN.

证明:假设此数据点(设为点 p)是 q 的 ORkNN,且点 p 到其障碍 k 近邻点的距离大于点 p 到查询可视区间的最短距离,即 $d_{obs}(p,OkNN(p))>d_{ver}(p,q)$.由垂线性质得到:$d_{ver}(p,q)$是 p 到查询 q 上的最短距离,即$\forall p_i\in q$, $d_{ver}(p,q)\geqslant d_{euc}(p,p_i)$,因此,$d_{obs}(p,OkNN(p))>d_{euc}(p,p_i)$.又因为 $d_{euc}(p,p_i)\leqslant d_{obs}(p,p_i)$,所以 $d_{obs}(p,OkNN(p))>d_{obs}(p,p_i)$,即查询$q$上的任意一点到$p$的距离都远于$p$的OkNN到$p$的距离.因此,查询$q$上的每个点都不能为点$p$的OkNN,所以点 p 也一定不会是 q 的 ORkNN,定理得证. □

如图 3 所示,以 p_2 为圆心、$d_{obs}(p_2,OkNN(p_2))$为半径的圆没有与查询 q=[S,E]相交,即 $d_{obs}(p_2,OkNN(p_2))<d_{ver}(p_2,q)$,那么点 p_2 的 OkNN 一定不会是查询 q 上的点,即点 p_2 一定不是 q 的 ORkNN,因此 p_2 可以被剪枝掉.

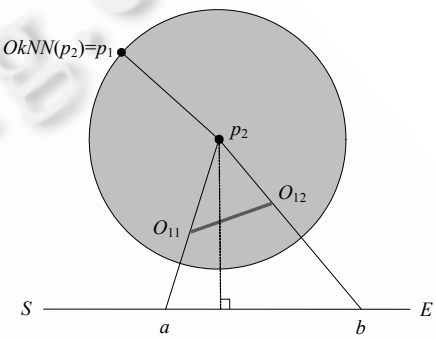

Fig.3 Illustration for pruning datasets

图 3 精简数据集示意图

基于定理 1,本文提出针对 CORkNN 查询的精简数据集算法 DPR.

DPR 首先构建障碍 Voronoi 图,接着获得查询线段上经过的 Voronoi 单元(cell),进而对每一个单元对应的数据点都作为一个快照查询点,针对每个快照查询点作为文献[17]的数据集剪枝算法的输入,以得到初步的精简数据集.这个过程的耗时几乎是整个 CORkNN 算法的耗时,这阶段所用时间复杂度为 $O(n'\log n)$,其中,n'为 ORkNN 算法通过剪枝后得到的数据点个数,n 为空间中数据点个数,且 $n'<<n$.然后,对精简后的数据集从边界距离上进行剪枝,即定理 1 中所述,从而最后得到精简数据集.这一阶段时间复杂度为 $O(c)$,c 为第 1 阶段 ORkNN 算法得到的候选集数据点个数,c 远小于空间中数据点个数.

3.2 获取障碍物

第 3.1 节中提出的精简数据集算法将障碍空间中的数据点集精简到成为结果集的最小范围,接着对每一个候选集中的每个数据点进行求解.本节解决求解的第 1 步——获取障碍物,即对每个数据点在查询范围 q 上的对计算有影响的障碍物,进而减少后面步骤涉及到障碍物的计算,极大地减少了计算量.

引理 1. 对一个查询区间 q=[S,E],一个数据点 p,由数据点 p 和查询 q 围成的区域记为ΔpSE,那么对 p 有影响的障碍物就是那些和ΔpSE 有交集的障碍物.

证明:对 p 有影响的障碍物即在后面计算控制点步骤时用的障碍物,在与ΔpSE 有交集的障碍物中,点 p 到查询 q 的路径都有可能经过这些障碍物;在ΔpSE 之外,即与之无交集的障碍物中,点 p 到查询 q 的路径都不可能经过这些障碍物.因此,仅仅是与ΔpSE 有交集的障碍物才能对数据点 p 有影响. □

由引理 1 可知,在考察数据点 p 及后面计算其对应的控制点及分割点时,仅需计算与ΔpSE 有交集的障碍物即可,这些障碍物影响控制点和分割点的计算.

定理 2. 假设数据点 p 到查询 q 的两个端点(S,E)的欧式距离为 $d_{euc}(p,S)$及 $d_{euc}(p,E)$,那么到 p 的最小距离大于 $\max\{d_{euc}(p,S),d_{euc}(p,E)\}$的 MBR 所包含的障碍物一定不是对 p 有影响的障碍物.

证明:若这样的障碍物是对 p 有影响的,由引理 1 可知,这些障碍物一定在ΔpSE 内或者与ΔpSE 有交集,如图 4 所示,设此 MBR 在ΔpSE 内的一个点为 p'.那么,点 p 到此 MBR 的最短距离 $d_{\min}(p,MBR)<d_{euc}(p,p')$成立,并且 $d_{euc}(p,p')<\max\{d_{euc}(p,S),d_{euc}(p,E)\}$.因此可以得到 $d_{\min}(p,MBR)<\max\{d_{euc}(p,S),d_{euc}(p,E)\}$,这与已知条件矛盾,因此假设不成立,从而定理得证. □

根据引理 1 和定理 2 对获取障碍物求解的方法,本文提出获取有效障碍物的算法 OSR.该算法通过剪枝的方法获取每个数据点在查询范围 q 上对计算有影响的障碍物,极大地减少了计算代价.该算法的基本想法是:用 R-tree 的数据结构将障碍物索引,以到数据点 p 的最小欧式距离递增的顺序访问障碍物.对于查询 q,数据点 p 的查询范围即为查询端点与 p 这 3 个点组成的三角形.凡是与这个三角形相交的障碍物,都是对 p 有影响的障碍

物.这里可以进行进一步剪枝,如果 p 到某个障碍物 MBR 的最短距离都大于这个三角形中以 p 为顶点的两条边的长度,则这个 MBR 被剪枝,即其不会影响到点 p;反之,还需要继续判断此障碍物是否与三角形相交,如果确实有交集,则将此障碍物放入对 p 有影响的障碍物集合 O_p 中,并将此障碍物顶点放入可视图 VG 中供后面使用.OSR 算法的时间复杂度为 $O(mn'')$,m 为空间中障碍物的个数,n''为 DPR 算法剪枝后候选集中数据点个数.

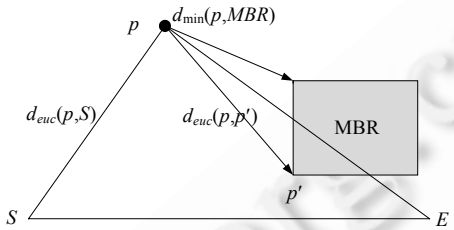

Fig.4 Illustration for proof of Theorem 2

图 4 定理 2 的证明示意图

例如,如图 5 所示,查询 q=[S,E],数据点为 p,障碍物由 R-tree 索引.对于点 p,其查询空间范围为ΔpSE,p 到 q 的两个端点的最大距离 $\max\{d_{euc}(p,S),d_{euc}(p,E)\}$作为后面障碍物 MBR 剪枝的界限.遍历障碍物 R-tree 从 N_1 和 N_2 开始,此时,最小堆 H_O 为$((N_2,d_{min}(p,N_2)),(N_1,d_{min}(p,N_1)))$.由于 p 到节点 N_2 的最短距离 $d_{min}(p,N_2)$小于 d_{max},因此将 N_2 从最小堆移除,将其孩子节点(N_5,N_6)及其到 p 的欧式距离作为 key 加入最小堆 H_O.从 H_O 移除堆顶 N_1,并将其孩子节点(N_3,N_4)加入 H_O.依此方法遍历 R-tree,得到对于 p 的障碍物结果集 O_p=(O_3,O_4,O_5,O_6).因为满足剪枝条件,原来 R-tree 里的叶子节点 O_1,O_2 组成的节点 N_3 被剪枝.此外,O_7,O_8 在ΔpSE 之外,所以这两个障碍物也不会产生影响.

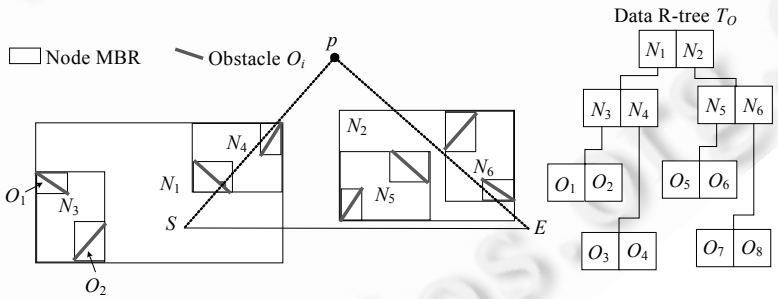

Fig.5 Illustration for retrieving obstacles O_p which have effects on p

图 5 获取对点 p 有影响的障碍物 O_p 示意图

3.3 控制点的计算

上面的方法能够获取一个数据点 p 的所有有效障碍物及其对应的可视图 VG,接下来需要找出点 p 在 q 上的障碍控制点集合.基本方法是:从 p 对 q 的查询范围内的可视图里的障碍物集合 O_p 端点中找出对查询 q 可见的点,如果这些点在 q 上的可见区间 $R_{p,q}$ 有交集,则需要更新.对于控制点的高效计算.本文首先使用了文献[13]提出的两种基本剪枝方法,在此不做详细介绍.在此基础上,本文提出了另一种剪枝方法.

定理 3. 若一个数据点对应的控制点到查询区间的最短距离大于此数据点到其 OkNN 点的距离与此数据点到此控制点之间的障碍距离的差值,则此控制点无效.

证明:假设此控制点是有效的,那么由障碍控制点定义可知,此控制点一定对应一段查询区间,此查询区间是由此控制点所控制.那么,当这段查询区间内的任意一点(如 p')为查询点时,此数据点(如 p)为查询点的ORkNN 结果的一部分,即点 p'到点 p 的距离一定小于或者等于点 p 到 p 的 OkNN 的距离,这与定理的假设矛盾,

从而定理得证. □

如图 6 所示,点 p 为空间中的一个数据点,点 p_1 为 p 的障碍第 k 近邻点,障碍物为 $O_1=O_{11}O_{12}$.

由于一个数据点 p 对应的控制点 O_{12} 到查询区间$[S,E]$的最短距离大于此数据点 p 到其 OkNN 点 p_1 的距离与点 p 到此控制点之间的障碍距离的差值,即 $d_{ver}(O_{12},q)\geqslant d_{obs}(p,OkNN(p))-d_{obs}(p,O_{12})$,那么由定理 3 可知,控制点 O_{12} 被移出控制点集.

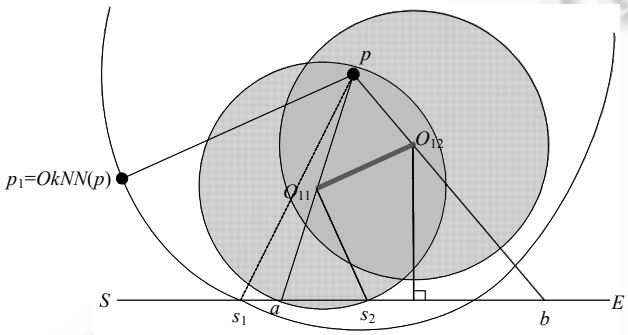

Fig.6 Illustration for control points pruning

图 6 控制点剪枝示意图

基于定理 3,本文提出计算某一个数据点在查询 q 下的控制点算法 CPLC.CPLC 首先求出此数据点 p 在查询 q 上的可视区间,记为 Vis.接着求出 p 为圆心、d 为半径的圆与区间 Vis 的相交区间.如果 p 的可视区间是整个查询区间,则此相交区间即为以 p 为控制点的控制区间.反之,先更新控制点集,然后对控制点集中的每个控制点应用定理 3 进行判断:如果该控制点到查询 q 的垂直距离比 p 与其障碍 k 近邻点的距离与 p 到此控制点的障碍距离的差值还大,那么从控制点集中移除此无效控制点;反之,求以控制点为圆心的圆与不可见区间的交点区间作为此控制点的控制区间,并加入结果集.CPLC 算法时间复杂度为 $O(mn'')$,m 为空间中障碍物的个数,n''为剪枝后候选集中数据点个数.

下面通过一个具体例子来解释算法 CPLC.

如图 7 所示,图中已知点 p_2,求 p_2 关于查询 q 的分割点及结果集.

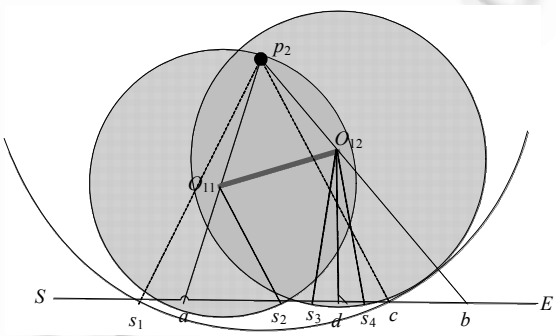

Fig.7 Illustration for control points computation

图 7 控制点计算示意图

首先,求出 p_2 的最近邻为 p_1,$d=d_{obs}(p_1,p_2)$.因为 $d_{ver}(p_2,q)<d_{obs}(p_1,p_2)$,所以点 p_2 可能会是结果.画出点 p_2 的可视区间为 $Vis=[S,a]\cup[b,E]$ 且 $a\neq b$;$Cir(p_2,d)$与可视区间有交点 s_1,则将$\langle p_2,p_2,[s_1,a]\rangle$加入结果集.控制点集合 $CPL=\{O_{11},O_{12}\}$.对于每个控制点进行考察:

- 对于点 O_{11},因为 $d_{ver}(O_{11},q)<d_{obs}(p_1,p_2)-d_{obs}(p_2,O_{11})$,因此计算 $Cir(O_{11},d_{obs}(p_1,p_2)-d_{obs}(p_2,O_{11}))$与不可视区

间 $q-Vis$ 的交点(记为 s_2),因此,将$\langle p_2,O_{11},[a,s_2]\rangle$加入结果集;

- 对于点 O_{12},因为 $d_{ver}(O_{12},q)<d_{obs}(p_1,p_2)-d_{obs}(p_2,O_{12})$,因此计算 $Cir(O_{11},d_{obs}(p_1,p_2)-d_{obs}(p_2,O_{12}))$与不可视区间 $q-Vis$ 的交点(记为 s_3,s_4),且 $s_2<s_3$,因此,将$\langle p_2,O_{12},[s_3,s_4]\rangle$加入结果集.

3.4 更新结果集

前面一节介绍了对于每个数据点的控制点的计算方法,这样,在查询 q 上对某个数据点的控制点集已经获得,下一步就是要评估此数据点对于当前结果集的影响.更新结果集算法 RLU 的基本想法就是在结果集上的查询区间与此数据点对应的控制区间合并.具体来说,RLU 将数据点 p 得到的结果集中的每个分割区间 R_{pi} 与 CORkNN 查询结果集的每个分割区间 R_i 进行逐个判断,如果 R_{pi} 为当前查询结果集的分割区间的子集,即可将此数据点 p 加入到结果集对应区间结果里;反之,求 R_{pi} 与查询结果集分割区间能够相交的区间 R_i 并求其交集,即将结果集分割区间细划分,交集部分对应的数据点集为 p 和 R_i 的数据点集的并集,其他两部分的结果分别为原查询分割区间对应结果点集和点 p.此算法的时间复杂度为 $O(n''n'')$,n''为剪枝后候选集中数据点个数.

3.5 CORkNN查询算法

前面 4 小节介绍了在求解障碍连续反 k 近邻查询中每个关键问题的求解算法.这些算法可以构成完整的 CORkNN 查询算法.CORkNN 查询算法首先根据数据点和障碍物的 R-tree 构造障碍 Voronoi 图;接着用前面所述的 DPR 算法精简数据点,对于得到的精简后的数据点中的每个点(如点 p),求其 OkNN,以 $d_{obs}(p,OkNN(p))$作为 key 值建立最小堆;对于堆中的每个数据点进行计算,使用 OSR 算法获取有效障碍物,接着使用 CPLC 算法计算此点 p 对应的控制点,得到点 p 的结果集;最后,根据算法 RLU 更新结果集.

4 实验结果与分析

在这一节里,我们进行详细的实验评估.所有的实验都是在 C++环境下实现的,实验运行环境为 Inter Core4 i7-2600 3.40GHz CPU,8.00GB 内存和运行 64 位 Windows 操作系统.

4.1 实验设置

实验分别基于真实数据和模拟数据的数据集.真实数据集使用两组数据,分别来自 hypsogr 和 utility[**].

Hypsogr 包含代表在德国 76999MBRs 的 2D 高度数据,utility 包含代表在德国 17790MBRs 的 2D 公用网络数据.模拟数据集的大小和真实数据集的大小相似,并服从正态分布(N)、Zipf 分布(Z)和均匀分布(U).

因为障碍空间的 RkNN 查询包括一个数据集 P 和一个障碍物集合 O,真实数据集分别表示这两种数据集,即(P,O)=(hypsogr,utility).此外，我们使用 6 种不同的分布组合(N,Z),(N,U),(Z,U),(Z,N),(U,N),(U,Z)来模拟数据集 P 和障碍物集合 O.

所有的数据点和障碍物都由一棵 R-tree 索引,节点大小设为 4KB,k 值缺省设为 5,查询长度设为数据集大小的百分比默认为 4%.为了详细测试算法的性能,选用了不同的测试参数,详见表 1.

Table1 Parameter range and default values

表 1 参数范围和默认值

参数	范围
查询长度(占空间数据点的百分比)	1.5, 3, 4.5, 6, 7.5
k 值	1, 3, 5, 7, 9
$\|P\|$ ($\times 10^4$)	1, 2, 3, 4, 5, 6, 7
$\|O\|$ ($\times 10^4$)	1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7
(P,O)	(N,Z), (N,U), (Z,U), (Z,N), (U,N), (U,Z)

** R-tree Portal-Spatial (geographical) Datasets in 2D space. http://www.rtreeportal.org/index.php?option=com_content&task=view&id=29&Itemid=42

4.2 实验分析

在这一节中,实验评估和分析本文所提出的算法 CORkNN 的 CPU 处理时间和 I/O 代价.由于本文算法首次针对障碍空间数据库中的连续反 k 近邻查询进行了研究,现有的相关算法(例如障碍空间连续 kNN 查询等)也无法通过修改来解决该问题,因此首先设计一种基本方法(Basic 方法)作为对比算法.Basic 算法连续查询障碍空间 RkNN 对象,即求出每个点的障碍 kNN,再找出这些 kNN 里与查询线段 q 有相交的区间的数据点进行比较.

首先评估 k 值的变化对 CPU 时间和磁盘 I/O 代价的影响.从图 8 可以看出:在 k 值从 1 变化到 9 的过程中,CPU 时间随着 k 值的增加而增加.提出的 CORkNN 算法的 CPU 占用时间明显比传统方法 Basic 要少,而且两者方法的 CPU 时间差距较大,因此用对时间取对数的方法表示.实验结果表明,本文所提出的 CORkNN 方法明显好于传统 Basic 方法.

图 9 给出了 k 值的增加对磁盘 I/O 代价的影响,纵坐标为需要访问的 R 树节点数目.可以发现:随着 k 值的增加,需要访问的数据点越多,CPU 处理占用的时间越多,磁盘 I/O 代价越大;而且由于本文的 CORkNN 方法仅仅需要访问离查询线段较近的数据点,因此其磁盘 I/O 代价要小很多.

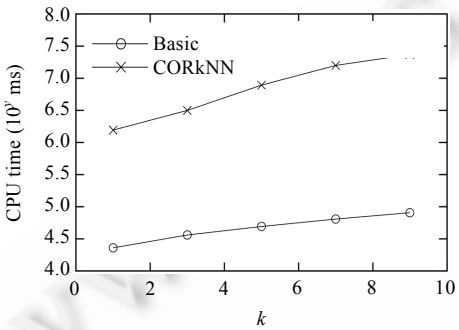

Fig.8 Effects of k on CPU time

图 8 k 值对 CPU 时间的影响

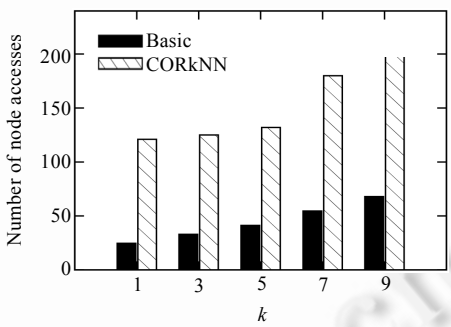

Fig.9 Effects of k on I/O costs

图 9 k 值对 I/O 代价的影响

下面评估数据点个数对 CPU 执行性能和 I/O 代价的影响.从图 10 可以发现:随着数据点数量的增加,CORkNN 方法的 CPU 用时下降,而传统的方法的 CPU 用时增加,而且 CORkNN 方法比传统方法的耗时要少.这是因为对于 CORkNN 方法,数据点越密集,查询点与其障碍 Voronoi 邻居越近,计算障碍距离的时候大多数只需要计算欧氏距离;而且获得最小候选集后,对每个数据点计算其有效障碍物时,由于数据点密度加大,需要处理的障碍物也相对较少,因此计算控制点的时间大幅度减少.而传统 Basic 方法需要计算每一个数据点的连续障碍反 kNN,因此对于 Basic 方法,数据点越多,CPU 用时越多.

图 11 反映了随着数据点数量的增加,传统 Basic 方法的磁盘 I/O 代价随之增加,而我们的 CORkNN 方法的 I/O 代价则降低.这是由于 Basic 方法需要计算每一个数据点,因而数据点越多,计算的数据点越多.而对于 CORkNN 方法,数据点越密集,需要访问的障碍物越少,因而在每个区间里剪枝掉的数据点越多,并且计算控制点时访问的节点个数也越少,所以 CORkNN 方法的磁盘 I/O 代价随着数据点个数的增加而减少.

我们来进一步地评价障碍物密度的变化对执行效率的影响.由图 12 可知:障碍物密度越大,CPU 执行时间越长,而且 CORkNN 方法明显好于传统方法.这是由于障碍物个数越多,需要计算的障碍距离越多,CPU 执行时间越长.而计算障碍距离比计算欧氏距离用时更多,传统 Basic 方法需要计算每一个数据点的障碍距离,比 CORkNN 方法需要计算的障碍距离要大,因此它所耗用的 CPU 时间更多.

图 13 给出了不同数据量的障碍物对磁盘 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 CORkNN 方法有效的剪枝策略极大地减少了障碍物的访问数量,因此磁盘 I/O 代价明显比传统的 Basic 方法的代价要小很多.

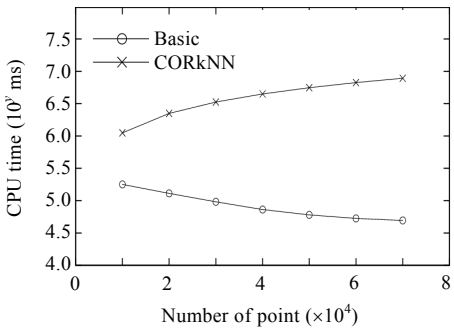

Fig.10 Effects of data point number on CPU time

图 10 数据点的个数对 CPU 时间的影响

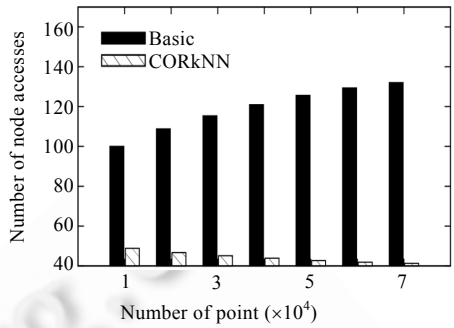

Fig.11 Effects of data point number on I/O costs

图 11 数据点的个数对 I/O 代价的影响

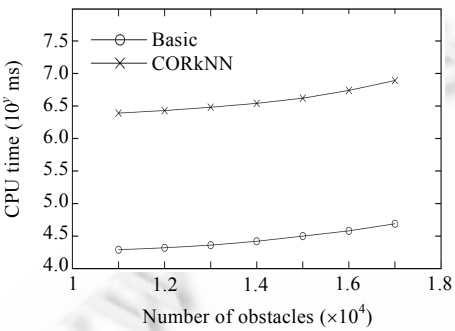

Fig.12 Effects of obstacle number on CPU time

图 12 障碍物的个数对 CPU 时间的影响

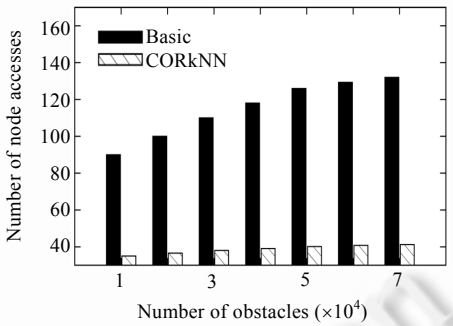

Fig.13 Effects of obstacle number on I/O costs

图 13 障碍物的个数对 I/O 代价的影响

图 14 和图 15 分别给出了两种算法不同数据分布的影响的实验结果.横坐标 1,2,…,6 分别代表数据点和障碍物分布(数据点分布,障碍物分布)的不同组合,分别对应为(*P*,*O*)=(N,Z),(N,U),(Z,U),(Z,N),(U,N),(U,Z).其中,*P*,*O* 分别服从正态分布(N)、Zipf 分布(Z)和均匀分布(U).例如,(N,Z)代表数据点服从均匀分布,障碍物服从 Zipf 分布.

图 14 给出了数据点和障碍物服从于不同的数据分布对 CPU 执行时间的影响情况.可以看出:数据分布的变化对 CPU 执行时间的影响不大,但是 CORkNN 方法由于访问数据比 Basic 方法要少,因此在 CPU 时间上优于传统方法.

图 15 给出了不同的数据分布组合对于数据点和障碍物在磁盘 I/O 代价上的影响.可以发现,数据分布的不同对磁盘 I/O 代价影响不大.CORkNN 方法由于包含高效的剪枝方法,减少了大量访问数据点和障碍物的个数,降低了磁盘 I/O 代价.

最后,用实验验证研究查询长度 *ql*(查询占数据空间的百分比)的影响.图 16 和图 17 显示了当 *ql* 作为变量,将 *k* 值固定在 *k*=5 时,CORkNN 算法在 CPU 占用时间和磁盘 I/O 代价上的有效性.

由图 16 可以观察到,CORkNN 算法的查询时间随着查询线段的长度的增加而增加.这是由于查询长度越大,需要获取的数据点越多、搜索障碍物范围越大、计算分割点的时间增加以及需要更新的结果集增大;而原始 Basic 方法由于对每个数据点都进行连续障碍 kNN 查询,与查询线段长度关系不大,因此其所占用的 CPU 时间不变.

图 17 给出了查询长度的变化对磁盘 I/O 代价的影响情况.可以看出:随着查询长度 *ql* 的增加,传统 Basic 方法对磁盘 I/O 代价的影响几乎为 0.这是由于传统方法对每个数据点都需要计算,查询长度的改变对其计算涉及到的数据点个数没有影响;而 CORkNN 算法由于在查询线段附近计算需要有更多的计算,因此对磁盘 I/O 代价

的影响随着查询线段长度的增加而增加.

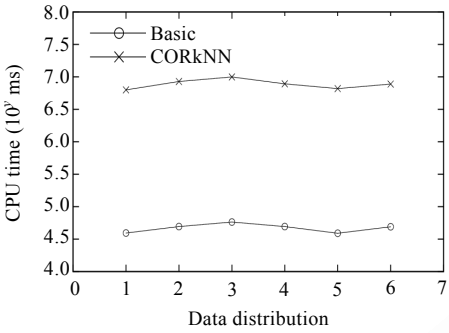

Fig.14 Effects of data distribution on CPU time

图 14 数据的分布对 CPU 时间的影响

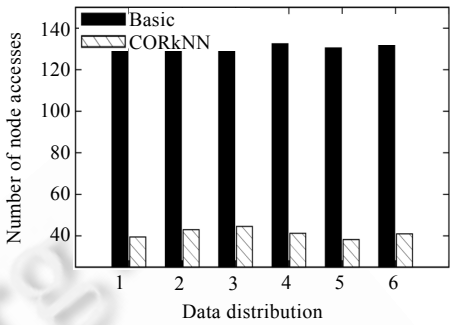

Fig.15 Effects of data distribution on I/O costs

图 15 数据的分布对 I/O 代价的影响

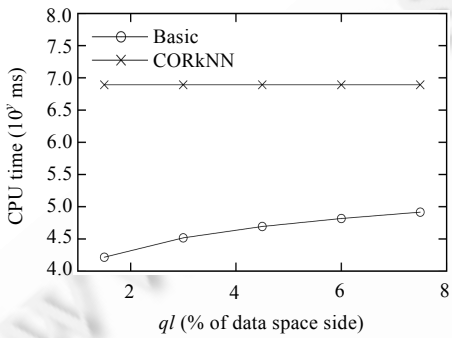

Fig.16 Effects of query length on CPU time

图 16 查询长度对 CPU 时间的影响

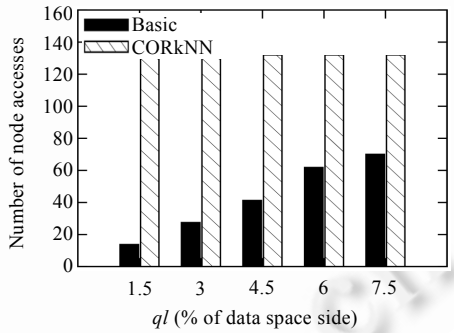

Fig.17 Effects of query length on I/O costs

图 17 查询长度对 I/O 代价的影响

5 结 论

本文对障碍空间数据库中的连续反 k 近邻查询问题进行定义,然后提出了解决 CORkNN 查询的所需要的剪枝数据集、获取障碍物、计算控制点和更新结果集各个步骤的优化技术和完整的 CORkNN 算法.通过多种数据集上的测试,验证了所提出的算法与基本方法相比,在处理效率上有大幅度的提高.

References:

[1] Zhang J, Zhu ML, Papadias D, Tao YF, Lee DL. Location-Based spatial queries. In: Proc. of the 2003 ACM SIGMOD Int'l Conf. on Management of Data. San Diego: ACM Press, 2003. 443−454. [doi:10.1145/872757.872812]

[2] Roussopoulos N, Kelley S, Vincent F. Nearest neighbor queries. In: Proc. of the '95 ACM SIGMOD Int'l Conf. on Management of Data. San Jose: ACM Press, 1995.71−79. [doi: 10.1145/223784.223794]

[3] Berchtold S, Ertl B, Keim DA, Kriegel HP, Seidl T. Fast nearest neighbor search in high-dimensional space. In: Proc. of the 14th Int'l Conf. on Data Engineering. Orlando: IEEE Computer Society, 1998. 209−218. [doi: 10.1109/ICDE.1998.655779]

[4] Gao YJ, Zheng BH, Chen GC, Li Q, Guo XF. Continuous visible nearest neighbor query processing in spatial databases. VLDB Journal, 2011,20(3):371−396. [doi: 10.1007/s00778-010-0200-z]

[5] Sharifzadeh M, Shahabi C. The spatial skyline queries. In: Proc. of the 32nd Int'l Conf. on Very Large Data Bases. Seoul: ACM Press, 2006. 751−762.

[6] Lee KCK, Lee WC, Zheng BH, Li HJ, Tian Y. Z-SKY: An efficient skyline query processing framework based on Z-order. VLDB Journal, 2010,19(3):333−362. [doi: 10.1007/s00778-009-0166-x]

[7] Stanoi I, Agrawal D, Abbadi AE. Reverse nearest neighbor queries for dynamic databases. In: Proc. of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery. 2000. 44−53.

[8] Tao YF, Papadias D, Lian X. Reverse *k*NN search in arbitrary dimensionality. In: Proc. of the 30th Int'l Conf. on Very Large Data Bases. Toronto: Morgan Kaufmann Publishers, 2004. 744−755.

[9] Wu W, Yang F, Chan CY, Tan KL. Finch: Evaluating reverse *k*-nearest-neighbor queries on location data. Proc. of the VLDB Endowment, 2008,1(1):1056−1067.

[10] Cheema MA, Zhang WJ, Lin XM, Zhang Y, Li XF. Continuously reverse *k* nearest neighbors queries in euclidean space and in spatial networks. VLDB Journal, 2012,21(1):69−95. [doi: 10.1007/s00778-011-0235-9]

[11] Zhang J, Papadias D, Mouratidis K, Zhu ML. Spatial queries in the presence of obstacles. In: Proc. of the 9th Int'l Conf. on Extending Database Technology. LNCS 2992, Heidelberg: Springer-Verlag, 2004. 366−384. [doi:10.1007/978-3-540-24741-8_22]

[12] Xia CY, Hsu D, Tung AKH. A fast filter for obstructed nearest neighbor queries. In: Proc. of the 21st British National Conf. on Databases. LNCS 3112, Edinburgh: Springer-Verlag, 2004. 203−215. [doi: 10.1007/978-3-540-27811-5_19]

[13] Gao YJ, Zheng BH. Continuous obstructed nearest neighbor queries in spatial databases. In: Proc. of the 28th ACM SIGMOD Int'l Conf. of Management of Data. Providence: ACM Press, 2009. 577−590.

[14] Gao YJ, Zheng BH, Chen G, Chen C, Li Q. Continuous nearest-neighbor search in the presence of obstacles. ACM Trans. on Database System, 2011,36(2):Article 9. [doi: 10.1145/1966385.1966387]

[15] Wang YF, Gao YJ, Chen L, Chen G, Li Q. All-Visible-*k*-Nearest-Neighbor queries. In: Proc. of the 23rd Int'l Conf. on Database and Expert Systems Applications. LNCS 7447, Vienna: Springer-Verlag, 2012. 392−407. [doi: 10.1007/978-3-642-32597-7_34]

[16] Cao KY, Wang GR, Han DH, Yuan Y, Hu YC, Qi BL. Clustering algorithm of uncertain data in obstacle space. Journal of Frontiers of Compter Science and Technology, 2012,6(12):35−45 (in Chinese with English abstract).

[17] Yu XN, Gu Y, Zhang TC, Yu G. A method for reverse *k*-nearest-neighbor queries in obstructed space. Chinese Journal of Computers, 2011,34(10):1917−1925 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2011.01917]

[18] Gao YJ, Yang JC, Chen G, Zheng BH, Chen C. On efficient obstructed reverse nearest neighbor query processing. In: Proc. of the 19th Int'l Conf. on Advances in Geographic Information Systems. Chicago: ACM Press, 2011. 191−120.

附中文参考文献:

[16] 曹科研,王国仁,韩东红,袁野,胡雅超,齐宝雷.障碍空间中的不确定数据聚类算法.计算机科学与探索,2012,6(12):35−45.

[17] 于晓楠,谷峪,张天成,于戈.一种障碍空间中的反 *k* 最近邻查询方法.计算机学报,2011,34(10):1917−1925. [doi: 10.3724/SP.J.1016.2011.01917]

谷峪(1981－),男,辽宁鞍山人,博士,副教授,CCF 高级会员,主要研究领域为空间数据管理,图数据管理.
E-mail: guyu@ise.neu.edu.cn

于戈(1962－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为数据库理论与技术.
E-mail: yuge@ise.neu.edu.cn

于晓楠(1987－),女,硕士,主要研究领域为空间数据管理.
E-mail: yuxiaonananshan@163.com