路网中双色数据集上连续反向 k 近邻查询处理的研究

李艳红[1] 李国徽[2] 杜小坤[1]

(中南民族大学计算机科学学院 武汉 430074)[1] (华中科技大学计算机科学与技术学院 武汉 430074)[2]

摘 要 近年来,反向最近邻查询(RNN)算法研究得到了普遍的关注,成为了数据库领域的一个研究热点。欧氏空间中提出了较多的高效算法,而路网中的反向最近邻处理方面所做的工作不够,有关这方面的成果较少。路网中查询点和数据对象之间以及不同数据对象之间的距离受到路网连通性的影响,欧氏空间中的反向最近邻方法在路网中不适用。反向最近邻查询有两种类型:单色反向最近邻查询(Monochromatic RNN,MRNN)和双色反向最近邻查询(Bichromatic RNN,BRNN)。到目前为止,仍然没有有效的算法来处理路网中双色数据集上的连续反向 k 近邻查询。因此,研究路网中双色数据集上连续反向 k 近邻查询是很有意义的。

关键词 反向最近邻查询,路网,双色数据集,连续监控

中图法分类号 TP391 **文献标识码** A

Study on Continuous Bichromatic Reverse k Nearest Neighbor Query Processing in Road Networks

LI Yan-hong[1] LI Guo-hui[2] DU Xiao-kun[1]

(College of Computer Science, South-Central University for Nationalities, Wuhan 430074, China)[1]

(School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China)[2]

Abstract In recent years, reverse nearest neighbor (RNN) query processing has received wide concern and has become a hot research topic in the database field. Many efficient RNN query processing methods in Euclidean spaces have been proposed. However, there is few related research findings in road networks. RNN processing methods in Euclidean spaces are unsuitable for RNN processing in road networks where the query-object distance and object-object distance are determined by the connectivity of the road network. RNN query can be roughly classified into two types: Monochromatic RNN(MRNN) query and Bichromatic RNN(BRNN) query. Up to now, there is still a lack of efficient algorithms for processing continuous bichromatic reverse k nearest neighbor (CBRkNN) queries in road networks. Thus, it is essential to examine CBRkNN queries in road networks and present efficient methods for CBRkNN query processing in road networks.

Keywords Reverse nearest neighbor query, Road network, Bichromatic data set, Continuous monitoring

1 引言

位置相关查询(LDQ)是位置相关服务的一个重要技术支持。近年来,有关位置相关查询的研究在国际上引起了广泛的关注,成为数据库领域的一个研究热点。反向最近邻查询作为LDQ中最重要的类型之一,受到了人们的普遍关注。欧氏距离空间中,提出了较多高效的算法。路网中查询点和数据对象之间以及不同数据对象之间的距离受到路网连通性的影响,欧氏空间中的反向最近邻方法在路网中并不适用。一般度量空间中的反向最近邻算法是依据距离的三角不等式来消减查询空间,也没有考虑到路网中节点间的连通性对查询的影响,对路网仍然不适合。

反向最近邻查询有两种:单色反向最近邻查询(Monochromatic RNN,MRNN)和双色反向最近邻查询(Bichromatic RNN,BRNN)。单色反向最近邻查询中,所有移动数据对象和查询对象均属于相同的类型。其定义是:对于查询对象 q、数据对象 o 而言,若不存在其他数据对象 o',使得 $dist(o,o')<dist(o,q)$,则 o 是 q 的反向最近邻。双色反向最近邻查询中,有两种不同的对象类型 A 和 B,查询对象类型为 A,数据对象类型为 B。其定义是:对于 A 类型的查询对象 q_A 和 B 类型的移动对象 o_B,若不存在其他 A 类型的对象 $o_{A'}$,使得 $dist(o_B,o_{A'})<dist(o_B,q_A)$,则 o_B 是 q_A 的反向最近邻。

LIANG 等人[1]提出了路网中双色数据集反向最近邻连续查询的处理方法,该方法以一个查询点与其他相邻查询点之间的距离的中点为边界,边界之内的区域为查询点的 RNN 结果集的监控范围。作为连续算法,它考虑了道路权值的变化以及对象的移动。但是,该方法假设查询点是静止的,不能算是真正意义上的动态 RNN 算法。而且,该方法只适合于 $k=1$ 的情况,不能扩展到 $k>1$ 的情形。

文献[2]讨论了路网中单色数据集上连续反向 k 近邻查

到稿日期:2012-06-30 返修日期:2012-07-20 本文受国家自然科学基金项目(61173049)资助。

李艳红(1973—),女,博士,讲师,主要研究方向为现代数据库,E-mail:anddylee@163.com;李国徽(1973—),男,博士,教授,主要研究方向为现代数据库、实时系统;杜小坤(1980—),男,博士,主要研究方向为现代数据库。

询问题，提出了一种采用称为两层多路树(DLM-tree)的数据结构来界定查询 q 的监控区域，通过监视监控区内数据对象和查询点的移动来修正查询结果，使查询结果连续有效的处理方法。本文将扩展文献[1]的方法，以处理路网中双色数据集上的连续反向 k 近邻查询问题(CBRkNN)。

所提出的 CBRkNN 查询处理方法改造了文献[2]提出的两层多路树(DLM-tree)，以表示 CBRkNN 查询 q_A 的反向 k 近邻的有效监控范围。在查询处理中，从 q_A 出发，搜索可能成为 q_A 的反向 k 近邻的 B 类数据对象(称为候选对象)，并将候选对象可能存在的范围用一棵以 q_A 为根的多路数表示，称为范围树；并通过构建各候选对象的验证范围，即下层验证树，来验证该候选对象是否真的是 q_A 的反向 k 近邻。为了方便描述，称验证后的确是查询点 q_A 的 RkNN 的候选对象为真解；而称那些验证后不是 q_A 的 RkNN 的候选对象为假解。而以 q_A 为根的上层范围树连同各候选对象的下层验证树所构成的两层多路树(DLM-tree)就成为了查询 q_A 的 BRkNN 的有效监控范围。双层多路树的生成意味着初始结果集的产生。为了实现对路网上 BRkNN 查询的连续监控，这里考虑移动查询点 q_A、A 类型对象以及 B 类型对象的位置变更。为了避免重复，先考虑范围树中的变动，再考虑验证树中的变动。

本文第 2 节讨论了相关工作；第 3 节给 CBRkNN 查询处理方法；第 4 节对所提出的算法的性能进行模拟实验；最后给出本文的结论。

2 相关工作

2.1 欧氏空间的反向最近邻查询方法

最初的 RNN/RKNN 算法是基于预计算的，构建 RNN 树[3]、RDNN 树[4]等索引结构，通过对索引结构的查询来实现静态的 RNN/RkNN 查询。基于预计算的方法存在的主要问题有：只适合于 k 值固定的 RkNN 查询，且索引构建和维护的代价很高，插入和删除一个对象可能会影响到多个对象。

为了消除预计算，出现了快照 RNN。快照类 RNN 的主要目的是利用一些措施来削减查询空间(称为过滤步)，然后对得到的候选对象进行精炼，以确定它们是否为查询的 RNN。有两种过滤的方法，即六分空间削减[5]和 TPL 削减[6]。

六分空间削减[5]由 Stanoi 等人提出，其利用获取 RNN 时的一些性质来消除预计算。他们将以查询点 q 为中心的查询空间分成 6 个大小相等的扇区 s_1-s_6，令 p 为区域 s_i ($i=1..6$)内 q 的 NN，可以证明：1)要么 $p\in$RNN(q)；2)要么在 s_i 区 q 无 RNN。该方法分为两个阶段：过滤阶段，在每个扇区进行一个受限的 NN 查询以找扇区内的 NN；精炼阶段，所有候选者用 kNN 查询来评估，去掉超过 kNN 的候选者，剩下的构成 q 的 RNN/RkNN 集。该算法的效率源于有较少的候选对象，二维空间数据集的 RNN 至多有 6 个候选对象。但随着维数的增加，候选对象的数目会呈指数增加。

Tao 等人[6]提出 TPL 算法，即利用空间的半平面特性来寻找候选者。该方法从查询点 q 出发，寻找一个尚未访问过的 q 的 NN(假设为 p)，以 p 与 q 的垂直平分线构建半平面。对于任何落在含 p 的半平面内的数据对象 p'，必定有 $dist(p,p')<dist(q,p')$。所以，若某个数据点被 k 或多于 k 个半平面所覆盖，则它必定不是 RkNN 的结果数据对象，因此，它能被安全地去掉，而不需做详细的检查。有效区间内不再有 q 的 NN 存在时，过滤阶段结束，此时，所有的候选者已被找到，而无关者已被去除。文献[6]的试验显示：正常情况下，TPL 剪切后候选者的数目低于 $4*k$。在精炼阶段，对每个候选者进行 NN 查询以去掉错误的候选者。这种方法能保证结果的完备性。然而，它只适合于静态 RNN 查询。

Tian Xia 等人[7]提出一种连续反向最近邻算法(CRNN)，它是一种增量的、可扩展的方法。该方法基于六分空间削减策略。对于一个查询点 q，CRNN 监控区由至多 6 个 π 区域和 6 个圆形区域构成。它将空间划分的理念引入到监控区域的划分和候选对象的获得中。由于它通过识别并处理落入监控区域的更新来连续地获得 RNN 结果集，因此它比直接方法更高效。但这种方法只能处理单色数据集，且为每个查询维持的监控区域较复杂，处理较麻烦。

James M. Kang 等[8]提出了基于 TPL 削减策略的 IGERN(增量式、通用、连续反近邻查询方法)。它提供一种统一模式，以同时处理单色数据集上和双色数据集上的 CRNN，为每个查询只维持一个有界监控区域和少量移动对象，所以比基于六分空间削减策略的方法更高效。

Wei wu 等人[9]指出：CRNN/CRkNN 查询可划分为连续过滤和连续精炼两个阶段，而通过分析可知，连续精炼在 $k>1$ 时占用查询时间的主要部分。提出 cRange-k 连续精炼算法，该算法验证查询 q 的 CRkNN 的候选者 p 所采用的方法是：连续监控离 p 的距离小于 $dist(o,q)$ 的对象数是否小于 k。该算法不要求任何精确的信息，而且，它既以范围(Range)来界限查询范围，又以 k 值来界限查询范围。只要其中之一条件满足，就可以中止算法，故处理高效。

2.2 路网空间的反向最近邻查询方法

Man Lung Yiu 等[10]首先提出了大型图中的反向最近邻问题(路网是图的一种应用)。他们处理了有限制图和无限制图上任意 k 值的单色 RNN 和双色 RNN 查询，基于图的一些特性，提出了 RNN 查询的优化技术和相应的算法。但是，这种方法只适合于静态 RNN。

Maytham Safar 等[11]基于 Network Voronoi Diagram (NVD)和路网扩展解决了路网上 RNN 查询，该方法虽然处理的对象和查询均可移动，但实质是一种快照 RNN，没能解决道路网上连续 RNN 监控。

SUN Huan-liang 等[1]提出了一种方法来解决道路网中双色数据集上移动对象连续反向最近邻查询问题。该方法利用 PMR quad 树和 Multi-way 树来分别索引路网和查询 q 的监控区域。其结果集的获得分成两个阶段：初始结果集的获得和结果集的连续监控。连续监控部分，考虑了路网中的两种基本更新：移动对象位置更新和交通情况更新。但这种方法没有考虑查询点的位置更新，且只适用于双色数据集。此外，该方法只适合于 $k=1$ 的情况，不能扩展到 $k>1$ 的情形。

3 CBRkNN 查询处理

本文讨论路网中双色数据集上连续反向 k 近邻查询(CBRkNN)的处理。图 1 给出了一个 CBRkNN($k=3$)查询的

示例，图1中有一组A类型对象o_{A1}到o_{A9}、一组B类型对象o_{B1}到o_{B9}和一个A类型的查询点q_A在路网中自由移动。其中，三角符号表示A类型对象，实心圆点表示B类型对象，空心圆点表示路网结点。假定查询q_A需要获得它的反向k(k=3)近邻，则查询结果是$\{o_{B1}, o_{B2}, o_{B11}, o_{B4}\}$。

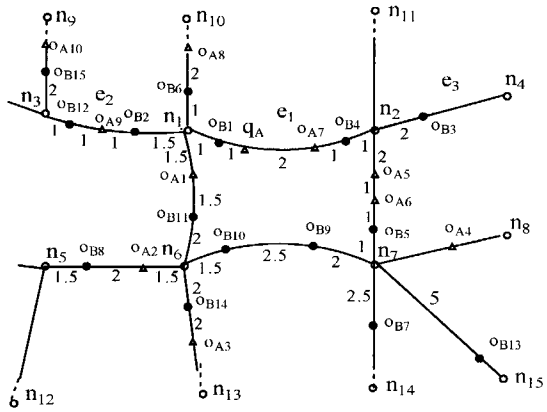


图1　一个CBRkNN查询示例

接下来，给出路网中CBRkNN查询的处理方法。该方法由两阶段构成：初始结果集的生成、查询结果的渐增维护。

3.1　生成初始结果集

初始阶段的任务包括：(1)构建q_A的范围树；(2)识别出一组A类监控对象，并将其加入监控对象集MO_Set$_A$；识别出一组B类候选对象，并将其加入候选对象集Cand_Set$_B$；(3)为每个候选对象构建验证树，判断该候选对象是真解还是假解，将真解加入RkNN结果集(RkNN_Set)；(4)输出q_A的RkNN结果集。

1. 构建范围树

构建范围树的具体步骤是：1)从查询点q_A出发，沿各路径进行路网扩展，对于每条可以探寻的路径而言，以找到k个A类型对象或者到达路网边界或者以没有不包含在范围树内的其他邻接点为探寻结束的依据。路网探寻过程遇到的A类对象作为查询q_A的监控对象，并将其加入监控对象集MO_Set$_A$；所遇到的B类对象作为查询q_A的候选对象，并将其加入候选对象集Cand_Set$_B$。2)为记录每条路径上的A类型的对象个数，对范围树中每个结点node均用计数器$count_A$进行计数，以统计从q_A到node的路径上已找到的A类型的对象个数。计数器$count_A$的初值为k，每找到一个A类型对象，则结点的$count_A$值减1，树中每个结点的$count_A$值等于其父亲结点的$count_A$值减去连接该结点到其父亲结点的边上的A类对象的个数。当$count_A$减为0时，该方向探寻中止，因为此时已找到了k个A类型对象。3)范围树中的结点可能是路网结点或对象。其中，中间结点只能是路网结点，叶子结点则可能是路网结点或对象。对范围树中的每条边，分别保存该边上的A类型对象和B类型对象，形式为：对象的id、对象在边中的位置。

2. 剪枝方案

为了减少不必要的路网探寻，可在满足下述3个条件之一时，采用有关引理来削减空间：1)对某结点node而言，若$count_A$减少至1时，已经找到一个以node为参考结点的特殊监控对象(将A类型监控对象区分为：一般监控对象和特殊监控对象，具体见定义1)；2)某结点node的$count_A$为x，且已经找到$count_A-x$个以node为参考结点的特殊监控对象(这x个特殊监控对象可以在node的一条邻接边上，也可以分布在node的多条邻接边上)；3)在探寻的过程中找到一个假解。若满足上述3个条件之一，则该方向探寻结束。以下分别介绍3个有关的引理和一个定义。

引理1　给定一个A类型查询点q_A、一个图结点n和一个A类型对象o_A，满足$d(q_A,n)>d(o_A,n)$。假定在从q_A到n的最短路径上已经有另外$k-1$个A类型对象o_{Ai}($1\leqslant i\leqslant k-1$)。则对于任意$B$类型对象$o_B$，若$o_B$到$q_A$的最短路径经过$n$，则有$o_B\notin$RkNN($q_A$)。

证明：$d(q_A,o_B)=d(q_A,n)+d(n,o_B)>d(o_A,n)+d(n,o_B)\geqslant d(o_A,o_B)$。由于假定在从$q_A$到$n$的最短路径上已经有另外$k-1$个$A$类型对象$o_{Ai}$($1\leqslant i\leqslant k-1$)，则对这$k-1$个$A$类型对象$o_{Ai}$而言，有$d(o_{Ai},n)<d(q_A,n)$，也有$d(o_{Ai},o_B)\leqslant d(n,o_B)+d(n,o_{Ai})<d(n,o_b)+d(q_A,n)=d(q_A,o_B)$。因此，至少存在$k$个$A$类型对象，这$k$个对象到$o_B$的距离均小于$d(q_A,o_B)$，所以有$q_A\notin$kNN($o_B$)，即$o_B\notin$RkNN($q_A$)。本文中，对于满足本引理的$A$类型对象$o_A$而言，本引理所指定的结点$n$称为$o_A$的参照结点，标记为$o_A.n_{ref}$。

引理2　给定一个A类型查询q_A、一个图结点n和一组A类型对象$o_{A1}, o_{A2}, \cdots, o_{Ar}$，满足$d(q_A,n)>d(o_{Ai},n)$($1\leqslant i\leqslant r$)。假定在从$q_A$到$n$的最短路径上已经有另外$k-r$个$A$类型对象$o_{Ai}$($1\leqslant i\leqslant k-r$)。则任意$B$类型对象$o_B$，若$o_B$到$q_A$的最短路径经过$n$，则有$o_B\notin$RkNN($q_A$)。

证明：$d(q_A,o_B)=d(q_A,n)+d(n,o_B)>d(o_{Ai},n)+d(n,o_B)\geqslant d(o_{Ai},o_B)$($i=1..r$)。由于假定在从$q_A$到$n$的最短路径上已经有另外$k-r$个$A$类对象$o_{Ai}$($1\leqslant i\leqslant k-r$)，则对这$k-r$个$A$类对象$o_{Ai}$而言，有$d(o_{Ai},n)<d(q_A,n)$，也有$d(o_{Ai},o_B)\leqslant d(n,o_B)+d(n,o_{Ai})<d(n,o_B)+d(q_A,n)=d(q_A,o_B)$。因此，至少存在$k$个$A$类对象，这$k$个对象到$o_B$的距离均小于$d(q,o_B)$。所以有$q_A\notin$kNN($o_B$)，即$o_B\notin$RkNN($q_A$)。本文中，对于满足本引理的$A$类监控对象$o_A$而言，本引理所指定的结点$n$称为$o_A$的参照结点，标记为$o_A.n_{ref}$。

定义1　称均不满足上述两个引理的A类型监控对象为一般监控对象；称满足上述两个引理之一的A类型监控对象为特殊监控对象。

引理3　给定一个A类查询q_A和两个B类对象o_B和$o_{B'}$，从q_A到$o_{B'}$的最短路径经过o_B。则若$o_B\notin$RkNN(q_A)，那么$o_{B'}\notin$RkNN(q_A)。

证明：若$o_B\notin$RkNN(q_A)，则$q_A\notin$kNN(o_B)。因此，至少存在k个A类对象o_{Ai}($1\leqslant i\leqslant k$)，满足$d(o_{Ai},o_B)<d(q_A,o_B)$。又假定从q_A到$o_{B'}$的最短路径经过o_B，则有$d(q_A,o_{B'})=d(q_A,o_B)+d(o_B,o_{B'})$，因此$d(o_{Ai},o_{B'})<d(o_{Ai},o_B)+d(o_B,o_{B'})<d(q_A,o_B)+d(o_B,o_{B'})=d(q_A,o_{B'})$($1\leqslant i\leqslant k$)。所以，$q_A\notin$kNN($o_{B'}$)，也即$o_{B'}\notin$RkNN($q_A$)。

3. 构建验证树

构建验证树的步骤是：以B类候选对象$cand$为中心，以距离值$d(q_A,cand)$为扩展范围进行路网扩展，得到以$cand$为根的多路树。如前所述，称验证后的确是查询点q_A的RkNN的候选对象为真解；而称那些验证后不是q_A的RkNN

的候选对象为假解。对于假解，只需扩展找到 *cand* 的 kNN 即可。对于真解，则需监控 $d(q_A, cand)$ 范围内的整个区域。

4. 为了连续监控的方便，对 *cand* 设置一个属性 NN_count_A，以记录在候选对象 *cand* 的 $d(q_A, cand)$ 的距离范围内已找到的 A 类最近邻(NN)的个数。则在构建 *cand* 的验证树时，只要满足以下两个条件之一就中止路网的探寻和验证树的构建：1)已找到 k 个 A 类 NN；2)扩展距离已到达 $d(q_A, cand)$。对于假解 *cand*：扩展时找到 *cand* 的 k 个 A 类 NN 后，还保留第 kNN 所在边上的其他未被包含到 kNN 中 A 类的对象(如果有，且在 $d(q_A, cand)$ 范围内)，并也将这些对象的个数加到 NN_count_A 属性中。这样在连续监控中，若有 A 类对象落入验证树，则增加 NN_count_A 的值；若有 A 类对象离开，则减少 NN_count_A 的值；若 A 类对象在验证树内移动，则不处理。而且只有当 NN_count_A 的值从 $k-1$ 增大为 k，或由 k 减少为 $k-1$ 时，才有可能改变 *cand* 的性质。此时，才需要调整验证树。

5. 为减少验证树的数目，对范围树中各 B 类候选对象采用从上到下的方式构建验证树。即：先构建位于范围树高层次的对象的验证树，再沿着从根结点到叶子结点的路径，从上到下构建各候选对象的验证树。若有某个候选对象为假解，则其所有子孙结点对应的 B 类候选对象均为假解，它们的验证树都不需再构建。

按照上述方法，可以构建图 1 中查询 q_A 的 DLM 树(见图 2)。其中，上层为界定 B 类候选对象存在范围的范围树，下层为各 B 类型候选对象的验证树，三角符号表示 A 类型对象，实心圆点表示 B 类型对象，空心圆点表示路网结点，黑色短线表示 DLM 树的边界。

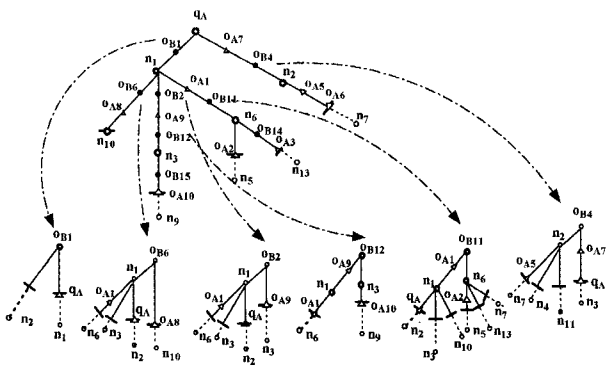


图 2　q_A 的 DLM 树

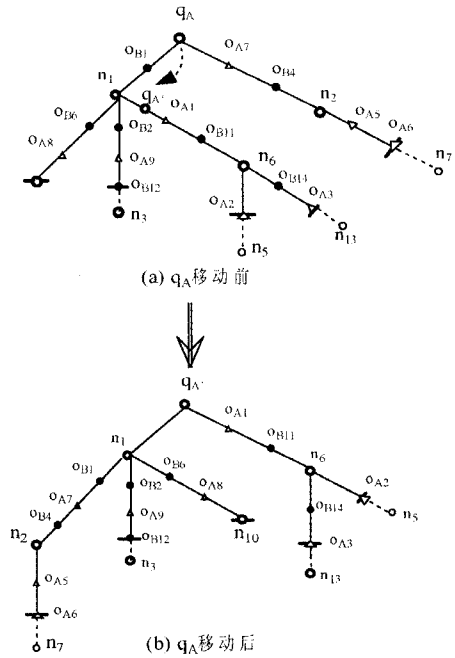
3.2　连续监控

以下讨论如何渐增地维护各 BRkNN 查询，使各监控区域和查询结果持续有效。为了实现对路网上 BRkNN 的连续监控，这里考虑两种基本的更新，即：查询点位置更新、对象位置更新。

1. 处理查询 q_A 的移动

q_A 在路网中移动到一个新位置，若新位置在原范围树外，则 q_A 原范围树失效，且必须重构；若新位置在原范围树内，则原范围树部分有效。为描述方便，用 $q_{A'}$ 表示 q_A 移动后的新位置。令 $e(n_i, n_j)$ 为 q_A 所在的边、$e(n_{i'}, n_{j'})$ 为 $q_{A'}$ 所在的边，则可由 q_A 原范围树通过“根变换”操作，得到以 $q_{A'}$ 为根的新范围树。根变换操作具体是：a)令 $q_{A'}$ 为新树的根；b)令 $n_{i'}, n_{j'}$ 为新树中 $q_{A'}$ 的两个孩子结点，并将在原树中连接到 $n_{i'}$ 和 $n_{j'}$ 的部分分别作为 $n_{i'}, n_{j'}$ 的子树。继续图 1 的例子，假定如图 3(a)所示，查询点 q_A 移动至边 $e(n_1, n_6)$，则新范围树见图 3(b)。

对位于 q_A 到 $q_{A'}$ 路径上的每个结点(或对象)，计算它们到 $q_{A'}$ 的新距离。将那些新距离小于原来到 q_A 的旧距离的结点或对象，连同它们的子树都视为有效部分；其他部分视为无效的，将重构。



(a) q_A 移动前　　(b) q_A 移动后

图 3　q_A 的移动对范围树的影响

2. 处理对象的移动

由于查询 q_A 的监控范围(其 DLM 树)是由一棵范围树和若干验证树组成，对象的移动分别需要考虑范围树内对象的移动以及验证树内对象的移动，具体有以下 3 种情况。其中先处理前两种情况，再处理最后一种情况。

1) A 类型对象相对于查询 q_A 的范围树的移动

a)对象 o_A 从 q_A 的范围树外移入范围树内(见图 4 中的对象 o_{A4})。首先，计算 o_A 的 $count_A$ 值，并将 o_A 加入到监控集 MO_Set_A 中。若 o_A 的 $count_A$ 值为 0，则将 o_A 连同它的子树视为无效并去掉。否则，将以 o_A 为根的子树上所有结点的 $count_A$ 值减 1，并将减 1 后 $count_A$ 值变为 0 的部分去掉。

b)对象 o_A 从 q_A 的范围树内移出(见图 4 中的对象 o_{A1})。则将以 o_A 为根的子树上的每个结点的 $count_A$ 值加 1。此外，还须要对该子树上的叶子结点以及特殊监控对象的参考结点进行扩展。

c)对象 o_A 在 q_A 的范围树内移动(见图 4 中的对象 o_{A8})。则处理为旧对象的离开和新对象的进入。

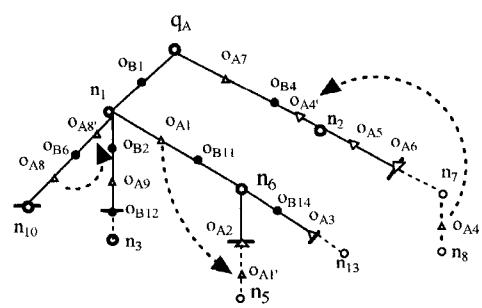


图 4　A 类对象在范围树内的移动

2) B 类型对象相对于查询 q_A 的范围树的移动

a)对象 o_B 从 q_A 的范围树外移入范围树内(见图 5 中的对象 o_{B7})。首先检查 o_B 是否是进入了 q_A 的范围树的特殊边(即,特殊监控对象所在的边),如果是,则不需要任何操作;否则,将 o_B 加入 Cand_Set$_B$ 集,并构建其验证树。

b)对象 o_B 从 q_A 的范围树内移出(见图 5 中的对象 o_{B11})。首先,检查 o_B 是否是作为范围树边界的假解候选对象,如果是,则将它从 Cand_Set$_B$ 内删除,丢弃其验证树,并从 o_B 的位置进行扩展;否则,检查 o_B 的原位置是否在查询 q_A 的特殊边上,如果是,则不需要任何操作;如果不是,则将 o_B 从 Cand_Set$_B$ 中删除,并丢弃其验证树。

c)对象 o_B 在 q_A 的范围树内移动(如图 5 中的对象 o_{B6})。则首先检查 o_B 是否为范围树边界的假解,如果是,则在 o_B 原位置扩展。

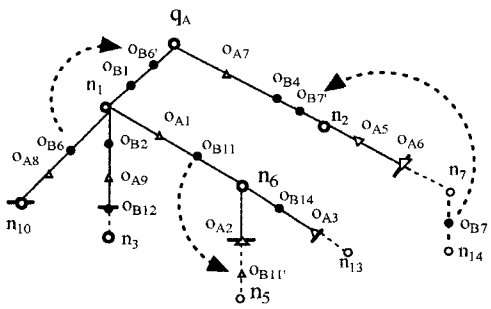


图 5 B 类对象在范围树内的移动

3) A 类型对象相对于候选对象 *cand* 的验证树的移动

a)对象 o_A 从外面移入到假解 *cand* 的验证树内。则将 *cand* 的 NN_count$_A$ 增 1,*cand* 仍是一个假解。例如,在图 6(a)中,o_{A2} 移入 o_{B12} 的验证树。

b)对象 o_A 从外面移入到真解 *cand* 的验证树内。则先将 *cand* 的 NN_count$_A$ 增 1。若 NN_count$_A$ 从 $k-1$ 增至 k,*cand* 将从一个真解变为一个假解;否则,*cand* 仍是一个真解。例如,图 6(b)中 o_{A7} 移入 o_{B1} 的验证树。

c)对象 o_A 从假解 *cand* 的验证树内移出。首先,检查从 o_A 到 *cand* 的新距离值是否仍小于 $d(q_A, cand)$,如果小于,则 *cand* 仍为假解;否则 NN_count$_A$ 减 1。若减 1 后 NN_count$_A$ 大于或等于 k,则 *cand* 仍为假解;否则 NN_count$_A$ 从 k 减为 $k-1$,扩展 *cand* 的验证树到距离 $d(cand, q_A)$ 或者直到扩展时遇到一个新的 A 类对象。若扩展中遇到新的 A 类对象,则 *cand* 仍为假解,其 NN_count$_A$ 从 $k-1$ 恢复到 k;否则,*cand* 从假解变为真解。例如,图 6(c)中,o_{A1} 移入 o_{B12} 的验证树。

d)对象 o_A 从真解 *cand* 的验证树内移出(见图 6(d)中的对象 o_{A1}),则 NN_count$_A$ 减 1,*cand* 仍为真解。

e)对象 o_A 在候选对象 *cand* 的验证树内移动,结果保持不变。

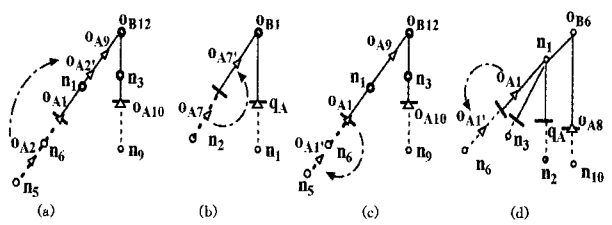


图 6 A 类对象在验证树内的移动

4) B 类型对象相对于候选对象 *cand* 验证树的移动。此时,对查询结果无影响,不需要任何操作。

4 实验评估

本节评估以上提出的双色数据集上路网连续反向 k 近邻查询方法的性能。由于所提方法是路网中双色数据集反向最近邻连续查询的第一个解决方案,选择一种静态的、在每个时间点运行所提算法的初始结果集获得步骤以获得 RkNN 结果集的方法作为对比算法。为公平起见,对比算法只扩展路网,以寻找候选对象和验证候选对象,而所有构建范围树和验证树的步骤均不执行,这主要考虑到对比算法不需要这些树进行连续监控。为方便描述,本节中用 NAV$_B$ 指代对比算法,用 CBRkNN 指代所提算法。此外,用 30 个时间单位的处理时间之和除以 30 而获得的平均处理时间作为度量的标准。对于 CBRkNN 算法而言,30 个时间单位包含一个获得初始结果集的周期和 29 个连续监控的周期。

4.1 实验参数

这里从文献[12]获取美国三藩市海湾交通网路图,并截取了包含 10k 条边的子网作为测试路网。子网含有的查询点数、A 类对象数和 B 类对象数分别作为系统参数,在试验时将用不同的查询点数、A 类和 B 类对象数目来进行测试。在每个时间点,部分查询点和对象将会改变其位置,使用 P_{object}、P_{query} 分别表示自上个时间点以来变动了位置的对象和查询的百分率。表 1 给出了实验的各项数据,其中粗体字表示缺省值。

表 1 实验参数表

参数	取值
A 类对象数	40,60,**80**,100,120(k)
B 类对象数	40,60,**80**,100,120(k)
查询点数	1,35,7,9(k)
RNN 的数目(k 值)	1,15,30,45,60
发生位置更新的查询的百分比	0,5,10,15,20
发生位置更新的对象的百分比	0,5,10,15,20

4.2 实验结果

图 7 表明随着对象数的增加,NAV$_B$ 和 CBRkNN 的处理时间明显减少。这是因为随着对象数的增加,对象的密度也增加,因此,搜寻候选对象和验证候选对象所需的路网扩展减少。

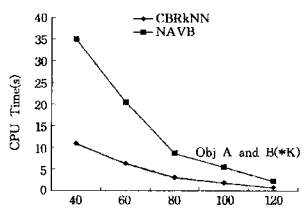


图 7 对象数的变化对算法性能的影响

图 8 表明,NAV$_B$ 和 CBRkNN 的处理时间随着查询数的增加而线性增加。这是因为总运行时间等于各查询处理时间之和。CBRkNN 在各种查询数的情况下,性能均好于 NAV$_B$。其原因在于 CBRkNN 只需要监控结果集,而不需要像 NAV$_B$ 那样,在每个时间点重新计算查询结果集。

图 9 表明,NAV$_B$ 和 CBRkNN 的性能受 k 值变化的影响

较大。这是很显然的，因为随着 k 值的增加，所需的路网扩展也增加。然而，不论 k 值是大还是小，CBRkNN 的性能均优于 NAV_B。

图 10 研究了对象的可移动性对算法运行时间的影响。当对象的可移动性增大时，CBRkNN 算法的处理时间将变长。这一点显而易见，因为对象频繁的位置更新会导致结果集的较大部分失效，从而增加维护代价。而 NAV_B 算法在对象的可移动性变化时，处理时间没有变化。这是因为 NAV_B 算法在每一个时间点都重新计算结果集，每一个时间点对它而言都是全新的。同样地，NAV_B 算法在不同的查询点可移动性下，其处理时间维持不变。

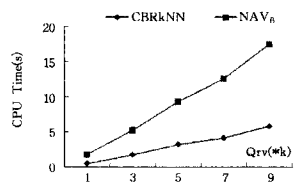


图 8 查询数的变化对算法性能的影响

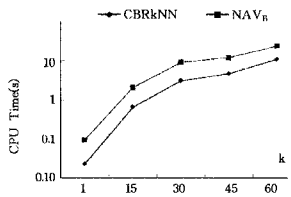


图 9 k 值变化对算法性能的影响

图 11 给出了各算法对于不同的查询点可移动性所表现出来的特征。如图 11 所示，当发生位置更新的查询点数目增加时，CBRkNN 算法的处理时间也增加。这是因为当查询点移动后，会使得它的 DLM 树部分或全部失效，从而触发失效部分的重建操作。

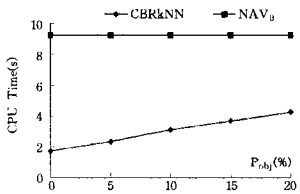


图 10 对象的可移动性对算法性能的影响

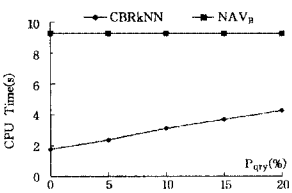


图 11 查询点的可移动性对算法性能的影响

平均来说，所提 CBRkNN 的性能优于参照算法 NAV_B 1.95 倍。而且，CBRkNN 对 k 值、对象数、查询数、对象和查询的可移动性等参数的变化都是健壮的。因此，它是高效和可扩展的。

结束语 本文首次讨论了路网中双色集上反向 k 近邻查询连续监控问题(CBRkNN)。所提 CBRkNN 查询处理方法改造了文献[2]所提出的两层多路树(DLM-tree)，来表示 CBRkNN 查询 q_A 的反向 k 近邻的有效监控范围。为进一步提高算法的效率，提出了相关引理来尽可能地削减查询空间，并提出一系列技术来高效地处理范围树和验证树的更新。实验结果表明，所提出的方法比直接方法高效约 1.95 倍。

参 考 文 献

[1] Sun Huan-liang, Jiang Chao, Liu Jun-ling, et al. Continuous Reverse Nearest Neighbor Queries on Moving Objects in Road Networks[C]// The Ninth International Conference on Web-Age Information Management. 2008:238-245

[2] Li Guo-hui, Li Yan-hong, et al. Continuous Reverse k Nearest Neighbor Monitoring on Moving Objects in Road Networks [J]. Information Systems, 2010, 35(8):860-883

[3] Korn F, Muthukrishnan S. Influence Sets Based on Reverse Nearest Neighbor Queries[C]// Proc. ACM SIGMOD '00. 2000: 201-212

[4] Yang C, Lin K-I. An Index Structure for Efficient Reverse Nearest Neighbor Queries [C]// Proc. 17th Int'l Conf. Data Eng. (ICDE' 01). 2001:485-492

[5] Stanoi I, Agrawal D, El Abbadi A. Reverse Nearest Neighbor Queries for Dynamic Databases[C]// ACM SIGMOD Wo-rkshop Research Issues in Data Mining and Knowledge Discovery (DMKD). 2000:44-53

[6] Tao Y, Papadias D, Lian X. Reverse kNN Search in Arbitrary Dimensionality[C]// Proc. 30th Int'l Conf. Very Large Data Bases (VLDB '04). 2004:744-755

[7] Xia T, Zhang D. Continuous Reverse Nearest Neighbor Monitoring [C]// Proc. 22nd Int'l Conf. Data Eng. (ICDE '06). 2006: 77

[8] Kang J M, Mokbel M F, Shekhar S, et al. Continuous Evaluation of Monochromatic and Bichromatic Reverse Nearest Neighbors [C]// ICDE. 2007:806-815

[9] Wu Wei, Yang Fei, Chan C-Y, et al. Continuous Reverse k-Nearest-Neighbor Monitoring[C]// The 9th International Conference on Mobile Data Management. 2008:132-139

[10] Yiu M L, Papadias D, Mamoulis N, et al. Reverse Nearest Neighbors in Large Graphs[J]. TKDE, 2006, 18(4):540-553

[11] Safar M, Al-Saleh A. PINE based RNN queries in Road Networks[C]// Campos do Jordão, São Paulo, Brazil. VII Brazilian Symposium on Geoinformatics, INPE. November 2005:356-367

[12] http://www.fh-oow.de/institute/iapg/personen/brinkhoff/generator/

(上接第 110 页)

[11] 王金朋，侯贵宾，邓成玉，等. 基于 Pi-演算的扩展有向图工作流模型及验证[J]. 计算机工程与设计，2010，31(10)：2399-2404

[12] 周建涛，史美林，叶新铭. 一个基于 Petri 网化简的工作流过程语义验证方法[J]. 软件学报，2005，16(7)：1242-1251

[13] 胡乃静，赵亮，胡金化. 基于 Petri 网的工作流结构正确性化简验证方法[J]. 小型微型计算机系统，2007，28(6)：1076-1079

[14] Henry H B, Zhao LJ. Applying propositional logic to workflow verification [J]. Information Technology and Management, 2004, 5(3/4):293-318

[15] Sadiq W, Orlowska M. Modeling and verification of workflow graphs[R]. Tech. Rep. 386. Department of Computer Science, University of Queensland, 1996

[16] Bae J, Caverlee J, Liu Ling, et al. Process Mining by Measuring Process Block Similarity [C]// Business Process Management Workshops. 2006:141-152

[17] Lawrence P. Workflow Handbook 1997[M]// John Wiley and Sons. Workflow Management Coalition, New York, 1997