高度动态环境下移动对象连续K近邻查询算法

牛剑光 陈 荦 赵 亮 谭 洁
(国防科技大学电子科学与工程学院 长沙 410073)

摘 要 针对面向高度动态移动对象集的多用户连续K近邻查询,提出了基于查询索引的多用户连续K近邻查询处理(Query Index based Multiple Continuous K-Nearest Neighbor Queries,QI-MCKNN)算法,阐述了查询索引的概念和构建方法,分析了格网大小对查询性能的影响,给出了相应的查询处理算法。实验表明,算法在面对高度动态的移动对象集时,查询处理性能优于基于移动对象格网索引的SEA-CNN算法。

关键词 高度动态,连续K近邻查询,格网索引,QI-MCKNN算法
中图法分类号 TP392 **文献标识码** A

Processing Continuous K Nearest Neighbor Queries on Highly Dynamic Moving Objects

NIU Jian-guang CHEN Luo ZHAO Liang TAN Jie
(College of Electronic Science and Engineering, National University of Defense Technology, Changsha 410073, China)

Abstract To evaluate multiple continuous k nearest neighbor queries on highly dynamic moving objects, we proposed an algorithm for processing Query Index based Multiple Continuous K-Nearest Neighbor Queries (QI-MCKNN), discussed the conception of Query Index and the solution to construct the grid index, analyzed the effect of grid size on query processing, and provided the corresponding procedure. Finally experimental results verify that when dealing with highly dynamic moving objects, the proposed algorithm is much more effective than the traditional SEA-CNN algorithm which is based on the objects index.

Keywords Highly dynamic, Continuous k nearest neighbors queries, Grid index, QI-MCKNN algrithm

1 引言

随着无线通信、计算技术、空间定位等技术的快速发展,以及众多具有定位功能的手持和车载设备的大量普及,对大量的移动空间对象进行监控和管理成为现实。个人位置服务、交通运输管理、商业位置服务、公共救援服务等应用需要依据移动对象位置进行查询。而个人用户由于移动电话等终端功能的日趋强大,对位置服务的需要越来越多,对实时性的要求也越来越高。因此应用系统中服务器端面对的将是频繁更新的海量移动对象和大量并发的多用户查询。

连续K近邻查询是查询结果随着查询及移动对象位置的变化不断更新的一种K近邻查询,近年来在移动对象管理领域得到广泛关注[1]。该方面的研究成果有:查询驱动更新的通用连续查询处理框架[2];通过引入查询搜索区域并利用共享查询执行思想进行多用户连续K近邻查询批更新处理的SEA-CNN算法[3];采用移动对象层次网格索引和查询索引以提高连续K近邻查询更新性能的YPK-CNN算法[4];利用概念空间划分技术以支持多用户连续K近邻查询处理的CPM算法[5];结合流水线处理策略利用多线程技术提高多用户连续查询处理并行性的MCkNN算法[6]。

上述多用户连续K近邻查询处理技术面向多用户环境,采用基于内存的查询处理框架,利用格网索引对移动对象进行存储和管理,采用不同的查询搜索策略来减少对移动对象集的重复访问,并采用共享查询执行机制,提高多用户连续K近邻查询处理的整体性能。

然而,在面对高度动态的移动对象数据集时,上述方法的效率将变低,这是由于所建立的移动对象索引的维护和更新代价变高,对查询结果重复利用的可靠性将受到影响。其中高度动态的含义是:(1)集合中移动对象状态更新频率更高,因为应用的实时性要求移动对象频繁更新;(2)由于移动对象具有目的性,每一个更新周期都会有一定数量的移动对象随机出现和消失。

面对高度动态环境下移动对象连续K近邻查询问题,本文提出一种新的索引构建方案以及对应的查询处理算法,通过对移动对象的空间分布进行预统计,建立K近邻查询的格网索引,并通过优化查询策略,进一步提高查询处理效率。

2 问题描述

设在一个平面空间S内,随机分布着N个位置不断更新的移动对象$O_i(1\leqslant i\leqslant N)$。现要对$S$中随机分布的$M$个不断更新的连续K近邻查询进行求解,其中每个查询表示为$Q_j$$(1\leqslant j\leqslant M)$,其中$(x,y)$是查询$Q_j$的中心位置坐标;$k$表示要

到稿日期:2010-04-19 返修日期:2010-09-29 本文受国家863项目(2007AA12Z208)资助。
牛剑光(1985-),男,硕士生,主要研究方向为移动对象数据库,E-mail:niujian-007@163.com;陈 荦(1973-),男,博士,副教授,主要研究方向为地理信息系统与数据库技术;赵 亮(1982-),男,博士生,主要研究方向为移动对象数据库;谭 洁(1973-),女,讲师。

求查询 Q_j 返回距其最近的移动对象个数。

在此，我们先明确若干概念：

查询中心(Searching Center)：查询 Q 的基准位置，称为 Q 的查询中心，记为 $SC(Q)$。

搜索区域(Searching Region[3])与搜索距离(Searching Distance)：S 中以 Q 的查询中心为圆心，以 R 为半径构成的区域，称为 Q 的搜索区域，记为 $SR(Q)$，R 称为搜索距离。

结果半径(Answer Radius)：查询 Q 的 k 个结果中距 $SC(Q)$ 最远的移动对象到 $SC(Q)$ 的距离，记为 $AR(Q)$。

本文研究假设 S 为二维平面，因此 $SC(Q)$ 表示为 (x,y)，搜索区域是个圆形。

以图 1 来说明基于查询的格网索引(后文简称查询索引)概念：平面空间已存在如图格网划分，存在两个 K 近邻查询 Q_1 和 Q_2，圆形区域表示 Q_1 和 Q_2 的搜索区域。单元格的数据结构中维护一个 QueryList 列表以存储其索引到的查询。图中单元格(2,4)被 Q_1 的搜索区域覆盖，因而将 Q_1 添加至单元格(2,4)的 QueryList 列表中；单元格(4,3)被 Q_1 和 Q_2 的搜索区域覆盖，因而将 Q_1 和 Q_2 添加至单元格(4,3)的 QueryList 列表中。对一个移动对象执行判断，首先计算其所在单元格，若其处于单元格(4,3)，则查询 Q_1，Q_2 对该移动对象进行查询判断。

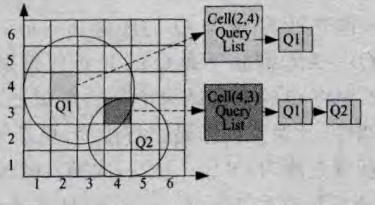


图 1　基于查询的格网索引

由于 K 近邻查询 Q 的搜索区域与移动对象的空间分布有关，至少须保证搜索区域内移动对象数大于 k，因此首先对移动对象空间分布进行统计，得到各单元格中移动对象数量，再根据各查询 Q 的查询中心确定搜索区域。

面对连续查询处理问题时，由于查询具有连续性，并且一般情况下 $M<<N$，基于多用户查询构建索引具有索引构建快、维护代价低的优点，为此本文提出基于查询索引的多用户 K 近邻查询处理算法(QI-MCKNN)，该算法更适合在高度动态移动环境中解决多用户连续 K 近邻查询问题。

3　基于查询索引的多用户连续 K 近邻查询处理算法(QI-MCKNN)

3.1　算法描述

QI-MCKNN 算法主要数据结构如下。

查询索引表(QueryTable)：内存中的一个 $n*n$ 的格网表，数据空间被划分为 $n*n$ 的格网单元即单元格(Cell)。K 近邻查询集依据一定的规则存储在表中。将查询集添加至查询索引表称为索引构建。

单元格(Cell)：查询索引表的存储单元，数据内容包含移动对象数量(Count)和查询列表 QueryList。

QI-MCKNN 算法如图 2 所示。步骤 3 为查询索引的构建，步骤 4 为查询处理策略，以下两节分别予以说明。

QI-MCKNN 算法

输入：移动对象集，查询集

输出：查询集

Repeat

1. 更新移动对象和查询，初始化空间格网索引；
2. 遍历移动对象集，对每一个移动对象 $O_i(1\leqslant i\leqslant N)$，
   (a) 通过 O_i 空间坐标计算得到所属单元格 Cell(m,n)；
   (b) Cell(m,n)中移动对象数 $Count+=1$；
3. 将查询 $Q_j(1\leqslant j\leqslant M)$依次添加入格网，构建查询索引；
4. 遍历移动对象集，对每一个移动对象 $O_i(1\leqslant i\leqslant N)$，
   (a) 通过 O_i 空间坐标计算得到所属单元格(m,n)；
   (b) 对于单元格(m,n)的 QueryList 查询集中的每个查询，对 O_i 进行判断；

图 2　QI-MCKNN 算法

3.2　查询索引的构建

构建查询索引对应于图 2 的步骤 3，关键在于 K 近邻查询 Q 搜索区域的确定。各单元格中移动对象分布数已得到，可以根据查询中心 $SC(Q)$ 和查询近邻数 k 确定对应的查询索引构建方法。

为了讨论方便，首先给出单元格颜色和象限的说明。单元格以着色程度区分为深色、浅色及无色。如图 3(a)中(2,2)单元格为深色，(1,2)单元格为浅色，(1,1)单元格为无色。深色及浅色统称为灰色。针对单元格象限需要说明的是，查询处于单元格内不同象限时对应的索引构建方法不同，为便于叙述，故提出单元格象限的定义，如图 3 所示：以单元格中心为坐标原点，平行于单元格边建立坐标轴，则单元格被划分为 4 个部分，分别按坐标系中的位置称为单元格的一、二、三、四象限。

下面分两种情况讨论查询索引构建方法。

(1) 查询所处单元格包含不小于 k 个移动对象

由于查询 Q 所在单元格已包含超过 k 个移动对象，因此搜索区域要且仅要覆盖此单元格即可。为说明 Q 在单元格中位置不同会导致不同的索引结果，以图 3 中 4 种情况为例，其中查询 Q 的查询中心均以×标示：

图 3(a)Q 位于单元格中心；

图 3(b)Q 位于单元格第一象限，靠近单元格右上角；

图 3(c)Q 位于单元格第一象限，靠近单元格中心；

图 3(d)未考虑 Q 在单元格内位置。

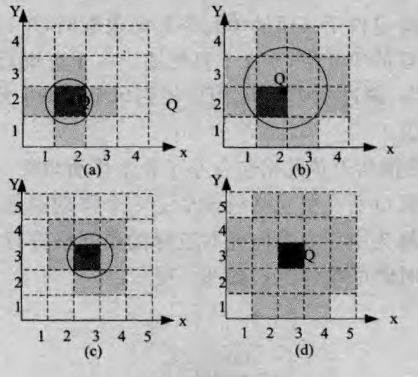


图 3　查询索引构建方法

图 4 示出单元格象限的定义。

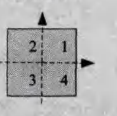


图 4　单元格象限的定义

图中灰色单元格表示被搜索区域覆盖，深色单元格表示

其中包含超过 k 个移动对象。可以看到，研究查询在单元格中的位置，可以合理缩小搜索区域，有效减少被搜索的单元格。

如图 5 所示，查询 Q 位于深色单元格第一象限，距单元格内最远点 O 的距离为 r，到单元格(1,3)的下边界距离为 d，当 $r>d$ 时，查询 Q 的搜索区域会覆盖单元格(1,3)和(2,3)。为找到 Q 在单元格中的位置使其满足 $r>d$，建立图 5 中的坐标系，设单元格边长为 1，Q 坐标为 (x',y')，依据 $r=d$ 列方程，$(x',y')(x',y')$ 满足：$\sqrt{x'^2+y'^2}=2-y'$。

变换方程得到曲线公式

$$y'=1-\frac{x'^2}{4}$$

若查询 $Q(x',y')$ 满足

$$y'>1-\frac{x'^2}{4} \tag{1}$$

则 Q 的搜索区域覆盖(1,3)和(2,3)两单元格。

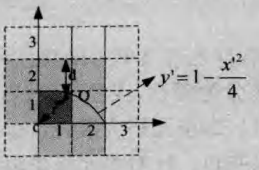


图 5　查询 Q 在单元格中位置分析

同理，得到

$$x'^2=1-\frac{y'^2}{4}$$

若 $Q(x',y')$ 满足下公式：

$$x'>1-\frac{y'^2}{4} \tag{2}$$

则 Q 的搜索区域覆盖(3,1)，(3,2)两单元格。

综上所述，当查询位于单元格第一象限时，可以确定索引建立方法：

· 首先在图 5 所示 8 个灰色单元格的 QueryList 中添加 Q；

· 若 Q 在单元格中相对坐标 $(x,,y,)$ 满足式(1)，则在单元格(1,3)和(2,3)的 QueryList 中添加 Q；

· 若 Q 在单元格中相对坐标 $(x,,y,)$ 满足式(2)，则在单元格(3,1)和(3,2)的 QueryList 中添加 Q。

当查询 Q 位于其所处单元格其他象限时，可按上述方法确定添加 Q 的单元格，本文不再赘述。由于查询 Q 在单元格中随机分布，通过概率分析，可以得到单个查询平均被 8.3 个单元格索引。

(2) 查询所处单元格包含小于 k 个移动对象

若查询 Q 所在单元格包含小于 k 个移动对象，就要以 Q 所在单元格为中心由内向外对近邻的单元格进行扩展搜索，直至移动对象总数大于 k，如图 6 所示。

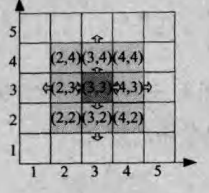


图 6　单元格扩展搜索

设查询 Q 位于图中深色单元格(3,3)中，搜索至近邻的 8 个浅色单元格称为向外扩展一层，以此类推。若向外扩展一层的所有单元格内移动对象总数大于 k，则停止搜索。建立索引时，若要使 Q 的搜索区域完全覆盖这些单元格，则 Q 的搜索区域必须包含向外扩展三层的(扩展三层之后是一个 7 * 7 的单元格方阵，共计 49 个)单元格。这是因为：一般地，如图 7 所示，查询 Q 处于深色单元格 (m,n) 中，设单元格边长为 1。已知扩展至第 i 层时，所有单元格中移动对象总数已达到 k，则需要将向外扩展 $\lceil\sqrt{2}(1+i)\rceil$ 层的所有单元格添加对查询 Q 的索引。一般情况下，考虑到索引建立的速度，格网划分不会过于密集，单元格扩展搜索一层之后移动对象数即大于 k，则建立索引时需要向向外扩展 3 层的单元格添加该查询，这种情况下，单个查询将被 49 个单元格索引。

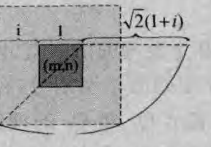


图 7　单元格搜索扩展公式

3.3　查询处理策略及其优化

查询处理策略对应图 2 的步骤 4，在基本思路不变的情况下，通过在索引当中增加一个数据结构和适当调整查询处理过程，可以提高查询处理性能。

对于 K 近邻查询处理算法，K 近邻查询 Q 会维护变量 AR(Answer Radius，结果半径)。当一个新的移动对象 m 要参与判断时，首先计算 m 与 Q 的距离 distance(m,Q)。若 distance$(m,Q)>AR$，则 m 不是 Q 的 K 近邻结果；若 distance$(m,Q)<AR$，则将 Q 的查询结果中最远点删去，m 插入至 Q 的查询结果当中，并对查询结果进行排序，更新 AR。进行一次比较的时间复杂度为 O(1)；完成一次插入的时间复杂度为 O(k)。设搜索区域存在 $M'(M'>k)$ 个待处理的移动对象，最好情况下，前 k 次恰巧是最近的 k 个移动对象参与判断，则整个查询处理过程只需 k 次插入操作和 $(M'-k)$ 次判断操作；最坏情况下，M' 个移动对象由远及近参与判断，则整个查询处理过程需要 M' 次插入操作。

由前所述，可知良好的查询处理策略可以大大降低整体时间复杂度。对于第 3.2 节所述索引方式，查询 Q 所在的单元格中的移动对象最有可能成为 Q 的 K 近邻结果。由此思路，如图 8 所示，将每个单元格中维护的查询列表 QueryList 分裂为两个查询列表：MainList 和 SecondList。对于查询所在单元格，将查询添加至 MainList 当中；对于其他单元格，则将查询添加至 SecondList 当中。因此，图 2 步骤 4 增加一次移动对象遍历过程，其流程更改为：

(1)遍历移动对象集，对每一个移动对象 $O_i(1\leqslant i\leqslant N)$，

(a)通过 O_i 空间坐标计算得到所属单元格 Cell(m,n)；

(b)对于单元格 Cell(m,n) 的 MainList 查询集中的每个查询，对 O_i 进行判断；

(2)遍历移动对象集，对每一个移动对象 $O_i(1\leqslant i\leqslant N)$，

(a)通过 O_i 空间坐标计算得到所属单元格 Cell(m,n)；

(b)对于单元格 Cell(m,n) 的 SecondList 查询集中的每个查询，对 O_i 进行判断。

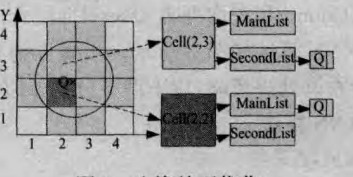


图 8　查询处理优化

上述查询处理过程不影响查询处理结果，但优化的查询处理策略使查询处理性能较前有很大提高。

4 格网划分对算法性能影响的分析

影响格网索引性能的一个关键因素是单元格的大小。一般地，稀疏的格网划分会导致单个单元格内移动对象过多；密集的格网划分按照 3.2 节中的索引方式不能有效减少查询处理的计算量。简单来讲，本文中一个执行周期的查询处理过程为：(1)更新移动对象和查询；(2)对多用户查询进行索引；(3)查询处理。当移动对象和查询更新完毕情况下，整体计算任务与移动对象所在单元格索引到查询数正相关。所以选择单元格大小的目标，就是使每个移动对象参与判断的查询数均值 e 最小。

由前文假设，按照 $n*n$ 的格网划分方式建立格网索引，平均每个单元格中移动对象数量为：

$$\bar{m}=\frac{N}{n^2}$$

设单个单元格中移动对象实际数量 m 大于等于 k 的概率为 p，小于 k 的概率为 $(1-p)$。据前文所述，当 $m>k$ 时，平均每个查询添加到 8.3 个格网当中；当 $m<k$ 时，平均每个查询添加到 49 个单元格当中。照此推论，格网索引中包含查询总数为：$8.3*M*p+49*M*(1-p)$。

平均每个单元格当中存在查询数亦即每个移动对象平均执行查询数为：

$$e=\frac{8.3*M*p+49*M*(1-p)}{n^2}=\frac{49*M-40.7*M*p}{n^2} \tag{3}$$

在移动对象总数 M 确定的情况下，式(3)中概率值 p 是格网划分规模 n 的函数：$p=f(n)$。目前的问题是：为使 e 取最小值，求 n 的取值。

对于单个单元格，其中移动对象数量为一随机变量，设为 η。由于移动对象在空间的随机分布，可以认为 η 服从参数为 $(N,1/n^2)$ 的二项分布，由 De Moivre-Laplace 定理[7]，可知对于任意 x，有

$$\lim_{N\to\infty}P\{\frac{\eta-N/n^2}{\sqrt{N(1-1/n^2)/n^2}}\leqslant x\}=\Phi(x)$$

由于 $n^2\gg1$，$N\gg1$，$E(\eta)=N/n^2$，上式可以化简为：

$$P\{\frac{\eta-E(\eta)}{\sqrt{E(\eta)}}\leqslant x\}=P\{\eta\leqslant x\sqrt{E(\eta)}+E(\eta)\}=\Phi(x)$$

令

$$k=x\sqrt{E(\eta)}+E(\eta)$$

得到

$$p=P\{\eta>k\}=\Phi([E(\eta)-k]/\sqrt{E(\eta)}) \tag{4}$$

将式(4)代入式(3)，并且 $E(\eta)=N/n^2$，可以得到：

$$e=\frac{49*M-40.7*M*\Phi((N/n^2-k)/\sqrt{N/n^2})}{n^2}$$

要取适当的整数 n，使得 e 取值最小。据前分析，n 的取值有一个前提，那就是要使得 N/n^2 不至于太小，能够保证以图 6 方式扩展一层之后 9 个单元格中移动对象总数大于 k。一般地，可以从 $n=\sqrt{\frac{2N}{k}}$ 起自大至小遍历 n 值，结合正态分布表，取到一合适的 n 值使得 e 取值最小。如：若移动对象总数为 $N=100,000$，查询总数 $M=1000$，查询近邻数 $k=10$，按照上述方法可以得到，$n=75$ 时，$e=1.71$ 为最小。为简单起见，可取

$$n=\lceil 0.8\sqrt{\frac{N}{k}}\rceil \tag{5}$$

实验证明其查询性能接近最佳。

5 实验结果及分析

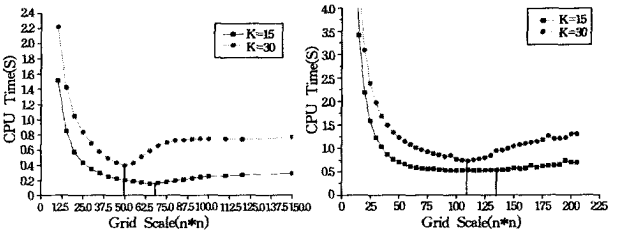
下面首先对本文提出的 QI-MCKNN 算法格网划分公式进行验证；然后在不同查询数和移动对象数情况下对比 QI-MCKNN 和 SEA-CNN 算法的整体性能。

实验系统运行环境为 Intel(R) Pentium(R) Dual E2180 2.00GHz 的双核 CPU、1GB 内存、Windows XP 操作系统。移动对象均匀分布在 10km * 10km 的空间范围内，速度矢量随机，其速率在初始阈值 20m/s 内随机分布。连续 K 近邻查询生成方式与移动对象数据集相同，k 值按照要求设置。每个查询周期将移动对象和查询全部更新。

5.1 格网划分对查询性能影响

本文第 4 节给出不同移动对象规模和查询近邻数情况下格网划分计算公式。为验证结论，采用以下两种数据设置方案设计实验：(1)移动对象规模 100000，查询数 1000，查询近邻数 k 分别为 15 和 30。(2)移动对象规模 500000，查询数 1000，查询近邻数 k 分别为 15 和 30。

在第一种数据设置方案中，依据式(5)得到格网划分应分别为 65 * 65 和 47 * 47，图 9(a)表明当格网划分分别为 63 * 63 和 50 * 50 时，查询处理耗时最少；在第二种数据设置方案当中，依据式(5)得到格网划分应为 145 * 145 和 103 * 103，图 9(b)表明当格网划分分别为 135 * 135 和 108 * 108 时，查询处理耗时最少。实验结果证明了公式的正确性。



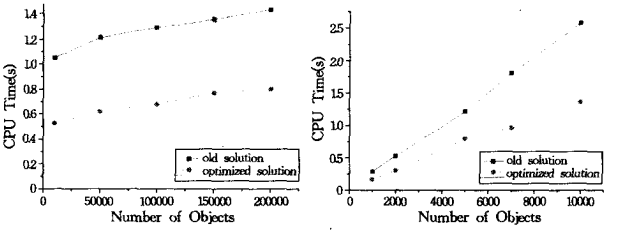
(a)100k 移动对象　(b)500k 移动对象

图 9 格网划分对查询性能的影响

5.2 查询策略优化对查询性能的影响

为验证第 3.3 节提出的查询优化策略对算法性能的影响，本节实验通过查询和移动对象数量不同情况下查询策略优化前后的查询处理性能对比，证明查询策略优化的有效性。

(1) 生成 100k 移动对象，查询近邻数 k 取 15，查询数按照表 1 所示范围设置，格网划分方式为 65 * 65。查询优化前后性能对比如图 10(a)所示。



(a)不同查询数　(b)不同移动对象数

图 10 查询策略优化对查询性能的影响

(2)生成5k查询,近邻数取15,移动对象数和对应的最优格网划分为:10k(20 * 20),50k(46 * 46),100k(65 * 65),150k(80 * 80),200k(95 * 95),查询优化前后性能对比如图10(b)所示。

由图10可以看到,查询策略的优化极大地提高了查询性能,优化后的算法查询处理性能较之前提高了约80%。

5.3 算法性能比较

为了与传统的SEA-CNN算法进行比较,本节模拟高度动态的移动对象环境,更新移动对象采取以下策略:(1)每个周期全部更新;(2)每个周期将比率为 r 的移动对象重置,即随机删除移动对象集中 $N*r$ 个移动对象并重新生成 $N*r$ 个移动对象。这里称 r 为重置率。查询处理整体性能对比实验采用参数如表1所列。

表1 实验参数

	范围	默认
查询	1,2,5,7,10(k)	5k
移动对象	10,50,100,150,2 00(k)	100k
r	0.001,0.005,0.01,0.05	0.005
K	15	15

(1)首先,生成5k查询和100k移动对象,分别对QI-MCKNN和SEA-CNN采用65 * 65和120 * 120的格网进行索引,在不同重置率 r 的情况下对两算法进行比较。

图11表明,随着重置率的增大,SEA-CNN算法性能急剧下降。SEA-CNN算法的主要思想是对查询结果高效的重复利用,面对高度动态的移动对象集,旧的查询结果可靠性很差,使得搜索区域增大,并且移动对象的高度动态性使得每个周期索引结构变化很大,导致了查询处理效率急剧降低。QI-MCKNN算法基于查询构建索引,移动对象动态性对查询处理过程并无影响,而且查询本身数量相对较少,基于查询的索引结构更新速度很高,对查询处理整体性能影响很小,面对高度动态的移动对象集时表现出较好的性能。

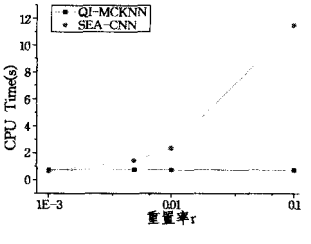


图11 不同重置率下算法性能比较

(2)生成100k移动对象,保持重置率为0.005,对不同查询数比较两算法性能。图12表明,查询处理时间基本随着查询数增长而线性增长。QI-MCKNN算法的查询处理性能高于SEA-CNN算法。

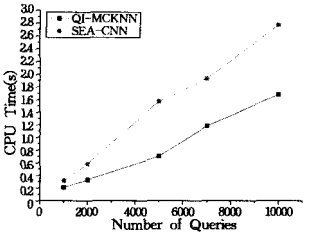


图12 不同查询数情况下查询性能比较

(3)生成5k查询,保持重置率为0.005,对不同移动对象规模比较两算法性能。

由图13可以看到,查询处理所需时间随着移动对象数量的增加基本呈线性增长趋势,但SEA-CNN算法对移动对象数量的敏感性高于QI-MCKNN。当只有10k移动对象时,SEA-CNN完成5k查询的时间甚至少于QI-MCKNN;随着移动对象数增多,SEA-CNN算法的查询处理性能与QI-MCKNN相比开始逐渐变差。据前所述,QI-MCKNN适用于查询数远少于移动对象数的情况。随着移动对象数目增多,QI-MCKNN的优势也随之明显。

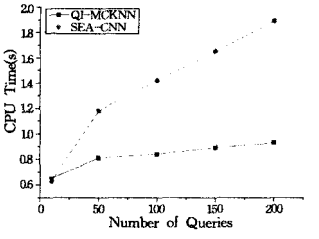


图13 不同移动对象数情况下查询性能比较

结束语 本文针对面向高度动态移动对象集的多用户连续K近邻查询,首先提出了一种基于查询的索引方案,给出了格网索引的划分公式;在查询处理策略上,以降低时间复杂度为目标,提出了一种性能良好的查询处理策略,并在索引结构上进行了优化;最后利用实验证明了算法相对于某经典算法的优越性,说明该算法更适合于高度动态、频繁更新的移动对象K近邻查询应用。

本文提出的算法在查询处理上表现出较好的性能,下一步的工作是继续研究在高度动态移动环境下,如何对旧的查询结果信息进行有效利用,进一步提高查询处理效率。

参 考 文 献

[1] Tao Yu-fei, Papadias D. Spatial Queries in Dynamic Enviroment [J]. ACM Transactions on Database Systems TODS, 2003, 28 (2): 101-139

[2] Hu Hai-bo, Xu Jian-liang, Lee D L. A Generic FrameworK for Monitoring Continuous Spatial Queries over Moving Objects[C]// Proc. of the 2005 SIGMOD Intl. Conf. on Management of Data. Baltimore, Maryland, 2005: 479-490

[3] Xiong Xiao-peng, MoKbel M F, Aref W G. SEA-CNN: Scalable Processing of Continuous K-Nearest Neighbor Queries in Spatio-temporal Databases[C]// Proc. of the 21st ICDE Intl. Conf. on Data Engineering. ToKyo, Japan, 2005: 643-654

[4] Yu Xiao-hui, Pu K Q, Koudas N. Mointoring K-Nearest Neighbour Queries over Moving Objects[C]// Proc. of the 21st ICDE Intl. Conf. on Data Engineering. ToKyo, Japan, 2005: 631-642

[5] Mouratidis K, Hadjieleftheriou M, Papadias D. Conceptual Partitioning: An Efficient Method for Continuous Nearest Neighbor Monitoring[C]// Proc. of the 2005 SIGMOD Intl. Conf. on Management of Data. Baltimore, Maryland, 2005: 634-645

[6] 廖巍,吴晓平,严承华,等. 多用户连续k近邻查询多线程处理技术研究[J]. 计算机应用,2009,29(7):1861-1864

[7] 盛骤,谢式千,潘承毅,等. 概率论与数理统计(第三版)[M]. 北京:高等教育出版社,2001,12:150-151