第 45 卷第 2 期
2013 年 4 月

南 京 航 空 航 天 大 学 学 报
Journal of Nanjing University of Aeronautics & Astronautics

Vol. 45 No. 2
Apr. 2013

基于预计算的连续 k 近邻查询处理的性能优化

李艳红

(中南民族大学计算机科学学院,武汉,430074)

摘要:现有的路网中连续 k 近邻(Continuous k-nearest neighbor,CkNN)查询方法一般分为两类:一类是利用某种数据结构监控查询 q 的 kNN 可能存在的区域;另一类是基于预计算的。基于预计算的 CkNN 查询处理方法不容易扩展到大路网结构,这是因为大路网的大量预计算信息不得不存储在外存,由此引发的内外存之间的大量的信息交换会大大地降低查询算法的性能。为了克服这个问题,本文提出了一种有效的优化技术,以减少查询处理时的内外存交换次数、提高查询处理的效率。实验证明所提出的优化方法的有效性,而且优化技术的采用使得查询处理算法具有更好的可扩展性。

关键词:连续 k 近邻查询;路网;预计算;性能优化

中图分类号:TP391　　**文献标志码**:A　　**文章编号**:1005-2615(2013)02-0290-07

Performance Optimization on Continuous k-Nearest Neighbor Query Processing Based on Pre-calculation

Li Yanhong

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

Abstract: The existing continuous k-nearest neighbor (CkNN) query processing methods in road networks are generally divided into two classes: (1) The methods using a data structure to monitor the area where the kNNs of the query could exist; (2) The methods based on pre-calculation. People always consider that CkNN query methods based on pre-computation cannot easily be extended to handle CkNN queries in large road networks. This is because that pre-computed information of a large road network always has a too large size to be stored in memory, thus lots of data swapping between the main memory and the auxiliary storage will greatly degrade the performance of query methods. In order to overcome this shortcoming, an efficient optimization technique is proposed to greatly reduce data swapping between the main memory and the auxiliary storage in query processing and improve the efficiency of query processing. Experimental result shows the efficiency of the technique. Moreover, the use of optimization technique could improve the scalability of the query processing methods.

Key words: continuous k-nearest neighbor query; road network; pre-calculation; performance optimization

最近邻查询(Nearest neighbour,NN)是最重要的位置相关查询之一。而路网中的 NN 查询又是 NN 查询很重要的组成部分。在现实生活当中,移动对象和移动查询点常常活动在某种网路(如公路网或铁路网)中,研究路网 NN 查询具有更广泛的现实意义。近年来,研究者们深入研究路网中的连续 k 近邻(Continuous k-nearest neighbor,CkNN)查询问题。一个路网中 CkNN 查询的示例

基金项目:国家自然科学基金(61173049)资助项目;湖北省自然科学基金(2012FFB07401)资助项目。

收稿日期:2012-04-13;**修订日期**:2012-12-10

通信作者:李艳红,女,博士,讲师,1973 年 4 月生,E-mail:anddylee@163.com。

是:一个在路网中行走的步行者,他要求得到未来 10 到 20 min 内离他最近的、在路上行驶的两辆出租车。

现有的路网中的 CkNN 查询方法一般分为两类:一类是利用某种数据结构监控查询 q 的 kNN 可能存在的区域,这类方法假定对象的移动是离散的,在每个位置更新发生时根据对象位置的变化情况对查询结果进行修正,从而达到对查询结果进行连续监控的目的。而实际应用中,对象的运动应该是连续的,这样在连续时间点之间的查询结果有可能不准确[1];另一类基于预计算,预先计算路网中结点间的路网距离(即连接着两个结点间的最短路径的距离值),在查询处理中,首先用静态的 kNN 方法求得在起始时刻查询 q 的 kNN,然后根据可能影响查询 q 的 kNN 结果的那些对象的移动,来对查询结果进行修正,从而保证查询结果的持续有效性。这类方法既可以处理对象定速移动的情况[2],也可以处理对象变速移动的情况[3]。

文献[3]中,作者提出一种路网中可变速对象连续 k 近邻查询处理算法,该算法能处理满足以下各个条件的连续 k 近邻查询:(1)所有的对象和查询点均在路网中运动,对象和查询点之间的距离是路网距离;(2)对象和查询点的移动速度允许在一定范围内变化;(3)能够给出查询监控的整个时间段内任意时刻的具体查询结果。实验表明,文献[3]所提出的方法较文献[1]的增量式连续监控算法(Incremental monitoring algorithm, IMA)更加高效、精确。

研究者普遍认为,基于预计算的 kNN 查询处理方法不容易扩展到大路网结构。其原因在于大路网的预计算信息通常占用大量的存储空间,以至于不能存放在内存中,由此引发的内外存之间的大量信息交换会极大地降低查询算法的性能。为了克服这个问题,本文提出了一种有效的优化技术,以减少查询处理时的内外存交换次数,提高查询处理的效率。

首先,根据路网中结点的分布情况对路网进行四分划分,使得划分后的每个单元内的结点数不超过一个给定的阈值。然后,采用一种特别的双层编号方式给每个结点进行编号。每个结点的编号包括两个部分:第一部分是结点所在的单元的编号,第二部分是结点在其所在单元内的编号。对于单元的编号,采用了一种类似于 Hilbert 曲线的空间填充方式对二维的路网空间中的单元进行编号。由于 Hilbert 曲线具有良好的局部性保留性能,所获得的单元编号具有这样的特点,即编号相邻的单元在物理位置上也相邻。此外,即便是对于那些结点非均匀分布的路网,所提出的方法也能得到连续的单元编号,因此比普通的 Hilbert 填充曲线更有效。类似地,可以得到结点的第二部分编号,即它们在所在单元内的编号。通过合理的组织和存储预计算的路网结点间的距离信息,可以大大地减少 CkNN 查询处理时的外存访问,提高算法处理的性能。最后,通过模拟实验来验证所提优化技术的有效性。实验结果表明,所提出的优化技术能使查询处理时间减少 5.06%,并且使查询处理的平均磁盘访问块数减少 92.44%。

1　相关工作

最近邻查询作为位置相关查询处理技术中最重要的类型之一,已经成为空间与时空数据库领域的一个研究重点与热点。研究者们对最近邻查询进行了广泛、深入地研究,并提出了许多最近邻查询处理算法[4-5]。

1.1　路网空间的静态最近邻查询方法

人们最早研究的是静态路网最近邻查询问题。Papadias 等人[6]提出了两种基于欧氏空间的 k 近邻查询方法,即增量式欧氏距离限制(Incremental Euclidean restriction, IER)算法和增量式路网扩展(Incremental network expansion, INE)算法,以处理路网中 k 近邻查询问题。INE 方法的主要思想是以查询点 q 为中心逐步进行路网扩展,在扩展过程中比较所有遇到的对象到查询中心的距离。INE 算法的效率取决于待查询对象的密度,如果整个路网范围比较大,而待查询的对象又比较少且很分散,那么该方法的效率就很低。Kolahdouzan 等人[7]提出了基于 Voronoi 图的路网最近邻(Voronoi network nearest neighbour, VN^3)算法,它是基于路网 Voronoi 图(NVD)的路网 k 近邻查询方法。这种方法的主要思想是将整个路网划分为若干个 Voronoi 单元。每个 Voronoi 单元以数据对象 p 为中心,并事先计算出每个 p 所在的单元中所有边界点到 p 的距离以及相邻 Voronoi 单元的边界点之间的距离,并将预计算的距离值加以保存。但是,VN^3 只适合稀疏的数据集。

1.2　路网空间的连续最近邻查询方法

Mouratidis 等人[1]研究了路网移动对象最近邻连续监控问题,提出了一种增量式连续监控算法。对于一个查询点 q,该方法通过用一棵 q 树来界定查询 q 的监控范围,在更新发生的时间点上对

q树和查询结果进行修正,以获得连续时间段内的查询结果。该算法能处理对象和查询点在路网中任意移动情况下的k近邻连续监控。但由于实际应用中对象和查询点位置更新是连续而不是离散的,在查询的两个连续时间点内的kNN结果变化情况是未知的,因此可能在kNN结果发生变化的两个连续时间点内返回不正确的查询结果。Huang等人[2]提出了在路网上连续监测移动对象而获得kNN查询结果的方法。该方法先计算在一个时间段内不可能成为kNN结果的剪枝距离,然后根据计算的剪枝距离进行剪枝过滤。然而,该方法假定对象的移动速度固定不变。

2 路网中kNN查询处理的优化技术

如前所述,基于预计算的CkNN查询处理方法不容易扩展到大路网结构。基于预计算的CkNN查询处理方法中,一般预先计算并保存路网中各结点间路网距离值,假设用矩阵 ***ND*** 表示。其中,***ND*** 的元素 $\boldsymbol{ND}_{ij}$ 存储的是结点 n_i 和 n_j 之间的最短路网距离。对于一个大型路网,这个矩阵将占用大量的空间。例如,如果路网有 10^5 条边,则矩阵 ***ND*** 的元素个数为 $10^5\times10^5$,也即 10^{10}。很明显,不可能为矩阵 ***ND*** 分配这么多内存空间。

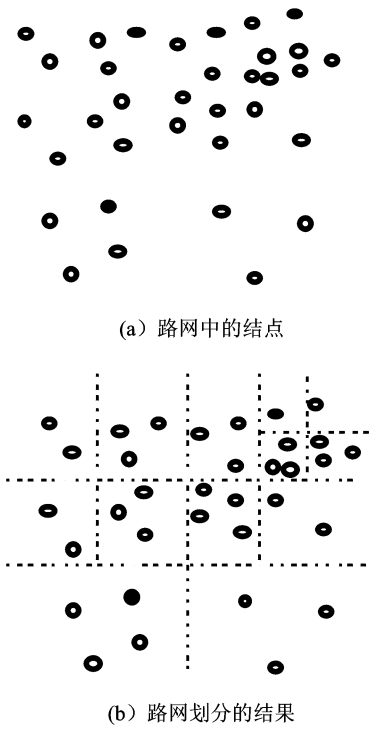
这里采用以下四分方式来划分路网。首先,将整个路网作为一个整体,用一个矩形区域来表示(也称为单元)。若路网中结点的数目超过了一个预先确定的阈值δ,进一步将该单元对称地分成均等的4个小矩形区域(或单元)。这个过程不断地进行,直到每个单元中包含的结点数均不超过 δ 个。特别地,δ 的取值由磁盘块的大小以及矩阵 ***ND*** 的元素大小所决定,使得 $\delta\times\delta$ 个距离值正好占据一整个磁盘块。因此,有

$$\delta=\sqrt{\frac{\text{磁盘块大小}}{\text{矩阵 }\boldsymbol{ND}\text{ 的元素的大小}}} \tag{1}$$

式中:磁盘块大小是指磁盘块的字节数;矩阵 ***ND*** 的元素的大小是指矩阵 ***ND*** 的一个元素所占据的字节数;δ 取整数值。

图1给出了路网中的结点以及路网划分结果的示例。这里假定 δ 为4,空间划分后每个单元所包含的结点数均不超过4。

第二步是给路网中结点进行编号。这里采用一种特别的双层编号方式,每个结点的编号包括两个部分:第一部分是结点所在的单元的编号,第二部分是结点在其所在单元内的编号。首先考虑给单元编号。单元是位于一个二维或三维的空间内,



(a) 路网中的结点

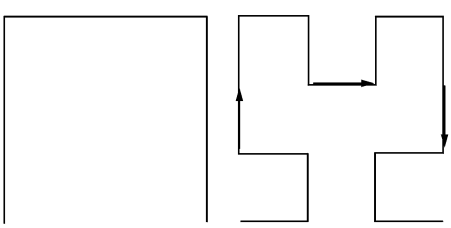
(b) 路网划分的结果

图1 路网中的结点以及路网划分结果的示例

而单元的编号是一个线性的序列。为了使高维空间内邻近的单元(特别是单元内的结点)的编号也是邻近的,可以采用空间填充曲线方法,根据各单元在填充曲线中的次序来对各单元进行编号。

这里所选择的空间填充曲线与Hilbert曲线(HC)[8]很类似。不同的地方是,所选择的填充曲线是若干不同级别Hilbert曲线的组合。Hilbert曲线具有良好的局部性保留性能,第1级和第2级Hilbert曲线的示例见图2。

Hilbert曲线的基本元素是"帽子"(即一边开口的方形)和"连接"(即连接两个帽子的箭头)。"帽子"的开口可以是向上、向下、向左或者向右。类似地,"连接"也有方向性:向上、向下、向左或者向右。一级的HC就是一个"帽子"(见图2(a))。二级的HC用4个小"帽子"来代替一级HC的帽子,这4个小"帽子"由3个"连接"组合在一起(见



(a) H_1 (b) H_2

图2 第1级和第2级Hilbert曲线示例

图 2(b))。每一个下一级的 HC 都是通过重复将一个“帽子”替换为 4 个小“帽子”和 3 个“连接”而获得。HC 的划分规则见图 3。

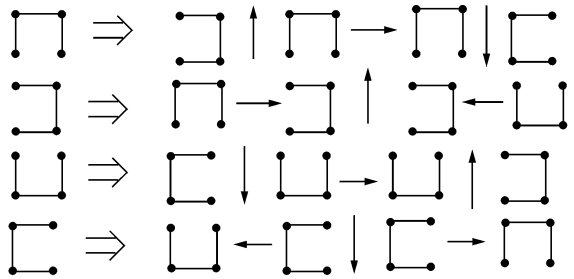


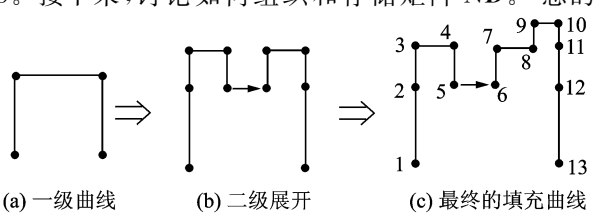
图 3　HC 的划分规则

所采用的空间填充曲线的构建方法如下:

首先,用一个一级 HC 来表示第一层的 4 个单元。如果某个单元包含有子单元,则根据图 4 所列的规则将该单元所对应的点替换为一个小“帽子”。而且还应根据需要加入“连接”。这个过程不断进行,直到每个单元均不包含子单元。按照这种方法,可以得到一个由不同级别的 HC 构成的空间填充曲线。然后,顺序地对曲线上的点进行编号。这样,可以得到每个单元的编号。继续图 2 的例子,可以得到相应空间填充曲线以及每个单元的编号,具体见图 4。图 4(a) 给出了一级曲线。由于第一层的左上角和右上角的两个单元(即对应于一级曲线上的左上角和右上角的两个点)均包含子单元,则这两个点被替换为两个开口向下的“帽子”以及一个向右的“连接”(见图 4(b))。最终的填充曲线以及每个单元的编号见图 4(c)。

接下来,讨论结点的第二部分编号(即它们在所在单元内的编号)的获取方法。在这里,采用跟第一部分编号的处理完全相同的方法。但是,由于每个单元拥有的结点数很少,第二部分编号比较容易获得。

到目前为止,已经讨论了如何对路网中的单元和结点进行编号。例如,图 2 的实心结点是位于单元 1 中,则它的第一部分编号是 1。假定该结点在单元 1 中的编号是 3,则这个灰结点的编号就是 1-3。接下来,讨论如何组织和存储矩阵 ***ND***。 总的



(a) 一级曲线　(b) 二级展开　(c) 最终的填充曲线

图 4　得到空间填充曲线和各单元的编号

来说,将位于同一个单元内的结点一起考虑,得到如下所示 ***ND*** 的结构。假定总共得到 m 个单元,则有

$$\boldsymbol{ND}=\begin{bmatrix}\boldsymbol{ND}_{\mathrm{Cell}_1,\mathrm{Cell}_1} & \boldsymbol{ND}_{\mathrm{Cell}_1,\mathrm{Cell}_2} & \cdots & \boldsymbol{ND}_{\mathrm{Cell}_1,\mathrm{Cell}_m}\\ \boldsymbol{ND}_{\mathrm{Cell}_2,\mathrm{Cell}_1} & \boldsymbol{ND}_{\mathrm{Cell}_2,\mathrm{Cell}_2} & \cdots & \boldsymbol{ND}_{\mathrm{Cell}_2,\mathrm{Cell}_m}\\ \cdots & \cdots & \ddots & \cdots\\ \boldsymbol{ND}_{\mathrm{Cell}_m,\mathrm{Cell}_1} & \boldsymbol{ND}_{\mathrm{Cell}_m,\mathrm{Cell}_2} & \cdots & \boldsymbol{ND}_{\mathrm{Cell}_m,\mathrm{Cell}_m}\end{bmatrix} \tag{2}$$

$$\boldsymbol{ND}_{\mathrm{Cell}_i,\mathrm{Cell}_j}=\begin{bmatrix}d_{\mathrm{Cell}_i^1,\mathrm{Cell}_j^1} & d_{\mathrm{Cell}_i^1,\mathrm{Cell}_j^2} & \cdots & d_{\mathrm{Cell}_i^1,\mathrm{Cell}_j^\delta}\\ d_{\mathrm{Cell}_i^2,\mathrm{Cell}_j^1} & d_{\mathrm{Cell}_i^2,\mathrm{Cell}_j^2} & \cdots & d_{\mathrm{Cell}_i^2,\mathrm{Cell}_j^\delta}\\ \cdots & \cdots & \ddots & \cdots\\ d_{\mathrm{Cell}_i^\delta,\mathrm{Cell}_j^1} & d_{\mathrm{Cell}_i^\delta,\mathrm{Cell}_j^2} & \cdots & d_{\mathrm{Cell}_i^\delta,\mathrm{Cell}_j^\delta}\end{bmatrix} \tag{3}$$

特别地,$d_{\mathrm{Cell}_i^k,\mathrm{Cell}_j^l}$ 是单元 Cell_i 中结点 k 与单元 Cell_j 中结点 l 之间的距离,每个元素 $\boldsymbol{ND}_{\mathrm{Cell}_i,\mathrm{Cell}_j}$ 包含 $\delta\times\delta$ 个这样的距离值。

如前所述,特别地选择 δ 的取值,使得 $\delta\times\delta$ 个距离值正好占据一整个磁盘块。又考虑到矩阵 ***ND*** 和每个子矩阵 $\boldsymbol{ND}_{\mathrm{Cell}_i,\mathrm{Cell}_j}$ 都是相对于主对角线对称分布的,所以只需要保存上(或下)三角矩阵。这里选择下三角矩阵。因此,可以修改 δ 的取值以进一步节省空间,δ 的取值具体由下述方程确定

$$\delta^2+\delta=\frac{\text{磁盘块的大小}}{\text{矩阵 }ND\text{ 的元素的大小}} \tag{4}$$

式中:磁盘块大小是指磁盘块的字节数,矩阵 ***ND*** 的元素的大小是指矩阵 ***ND*** 的一个元素所占据的字节数。δ 取正整数值。

按照这种方式,矩阵 ***ND*** 的一个元素 $\boldsymbol{ND}_{\mathrm{Cell}_i,\mathrm{Cell}_j}$ 将占据一个磁盘块。接下来,在连续$((m\times m)+m)/2$ 个磁盘块中,按行优先的顺序依次存储矩阵 ***DD*** 的下三角各元素矩阵,具体顺序是 $\boldsymbol{ND}_{\mathrm{Cell}_1,\mathrm{Cell}_1}$, $\boldsymbol{ND}_{\mathrm{Cell}_2,\mathrm{Cell}_1}$, $\boldsymbol{ND}_{\mathrm{Cell}_2,\mathrm{Cell}_2}$, $\boldsymbol{ND}_{\mathrm{Cell}_3,\mathrm{Cell}_1}$, $\boldsymbol{ND}_{\mathrm{Cell}_3,\mathrm{Cell}_2}$, $\boldsymbol{ND}_{\mathrm{Cell}_3,\mathrm{Cell}_3}$, …, $\boldsymbol{ND}_{\mathrm{Cell}_m,\mathrm{Cell}_1}$, $\boldsymbol{ND}_{\mathrm{Cell}_m,\mathrm{Cell}_2}$,…, $\boldsymbol{ND}_{\mathrm{Cell}_m,\mathrm{Cell}_m}$

考虑到同一个单元内的结点通常彼此非常靠近,而且在 CkNN 查询处理的过程中同一单元内结点间的距离会经常被使用。因此,将 m 个特殊元素 $\boldsymbol{ND}_{\mathrm{Cell}_i,\mathrm{Cell}_i}$(保存单元 i 内各结点间的距离值,$1\leqslant i\leqslant m$)存放在内存中。这样,能够大大地减少矩阵 ***ND*** 的磁盘访问。

最后,总结本节中为尽量减少磁盘访问而采用的各种技术,具体如下:

(1)根据空间内的结点的位置来划分路网空间。将同一个单元内的各相邻结点作为一个整体,并根据结点所在的单元来组织、存放结点间的距离值。

(2)用一种类似于 Hilbert 曲线的方式,对单元进行编号,这样空间位置上相邻的单元在其编号上也是相邻的。此外,所提出的空间填充曲线比 Hilbert 曲线更有效,其原因在于即便是对于那些结点非均匀分布的路网,所提出的方法也能得到连续的单元编号。

(3)结点的编号包括两个部分:结点所在单元的编号和结点在其所在单元内的编号。给定一组结点,很容易知道这些结点所属的单元,因此也很容易确定要访问哪些磁盘块。而且,这些待访问磁盘块往往是一些连续的磁盘块,因此能进一步减少磁盘访问时间。

(4)存储 m 个特殊的元素 $\boldsymbol{ND}_{\mathrm{Cell}_i,\mathrm{Cell}_i}(1 \leqslant i \leqslant m)$在内存中,以进一步减少磁盘访问。

3 实验评估

通过模拟实验评估所提出的优化方法的效果。在文献[3]中,作者提出了一种处理路网中不定速移动对象连续 k 近邻查询的算法,称为 $\mathrm{CUkNN}_{\mathrm{PLS}}$。本节将以 $\mathrm{CUkNN}_{\mathrm{PLS}}$为参照对象,来考察所提出的优化方法。为了区别起见,称采用了优化技术的 $\mathrm{CUkNN}_{\mathrm{PLS}}$算法为 $\mathrm{CUkNN}_{\mathrm{PLS\_Opt}}$。通过测量处理路网中 CkNN 查询所需的平均 CPU 时间和平均磁盘访问块数,来评价这两个算法的性能。

3.1 实验环境和实验参数

用一个实际路网来测试所提出的方法。为了建模现实生活中的真正路网,采用了美国三藩市地区的路网的真实数据,构建了一个包含有 175 343 个结点和 223 308 条边的路网。这个路网含有一组服从随机分布的查询点和数据对象。每一个对象或查询点在路网中以不确定速度移动,速度的最大和最小值的差别为30 m/h。表 1 给出了控制实验的参数,而表中强调字体给出的是缺省的参数值。

表 1 实验参数表

参数	取值
对象数/10^5	1,3,5,7,9
查询点数/10^3	1,3,5,7,9
NN 的数目(k 值)	1,5,10,15,20
查询时间段的长度	10,30,50,70,100

3.2 实验结果

第 1 组实验用来评价查询时间段的长度变化对 $\mathrm{CUkNN}_{\mathrm{PLS}}$ 和 $\mathrm{CUkNN}_{\mathrm{PLS\_Opt}}$ 算法的 CPU 时间和访问的磁盘块的数目的影响。

如图 5(a)所示,这两种算法的 CPU 时间随着查询时间段的增长而增加。这是由于一个较长的时间段意味着更多的对象将到达路网的结点,因此会花费更多的查询处理时间。从图中可以看出,$\mathrm{CUkNN}_{\mathrm{PLS\_Opt}}$ 的 CPU 时间较 $\mathrm{CUkNN}_{\mathrm{PLS}}$ 少 5.06%,这是因为 $\mathrm{CUkNN}_{\mathrm{PLS_Opt}}$中计算待访问的磁盘块的编号操作较 $\mathrm{CUkNN}_{\mathrm{PLS}}$简单些。

图 5(b)表明,$\mathrm{CUkNN}_{\mathrm{PLS}}$访问磁盘块的数目远多于 $\mathrm{CUkNN}_{\mathrm{PLS\_Opt}}$,$\mathrm{CUkNN}_{\mathrm{PLS_Opt}}$的平均访问磁盘块数为 $\mathrm{CUkNN}_{\mathrm{PLS}}$的 7.56%。而且,变化的查询时间段对 $\mathrm{CUkNN}_{\mathrm{PLS}}$的影响要大于对 $\mathrm{CUkNN}_{\mathrm{PLS_Opt}}$的影响。对 $\mathrm{CUkNN}_{\mathrm{PLS}}$而言,若增加查询时间段的长度,则查询处理将会涉及更多的边和结点,因此更多的磁盘块将被访问。对 $\mathrm{CUkNN}_{\mathrm{PLS_Opt}}$而言,由于采用了特殊的优化技术来组织大型的预计算结构,即矩阵 $\boldsymbol{ND}$,因此被访问的磁盘块的数目减少至 7.56%。

在第 2 组实验中,将考察 k 值的变化对各算法性能的影响。

如图 6(a)所示,这两种算法的 CPU 时间随着 k 值的增加而增加。这是因为随着 k 值的增加,查询处理所需要的对象距离值的计算以及距离值的比较也相应增多。总的来说,$\mathrm{CUkNN}_{\mathrm{PLS\_Opt}}$ 相对它的参照者来说,其性能在任何情况下都优于其参

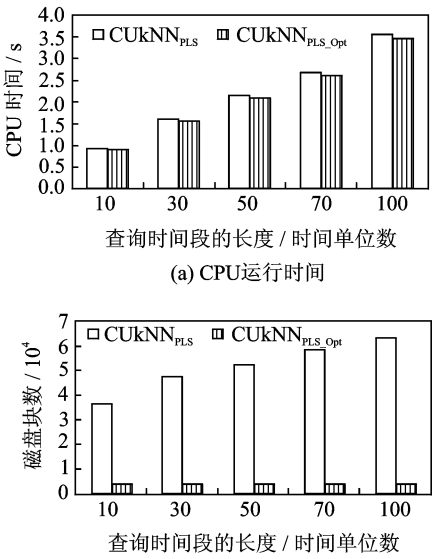
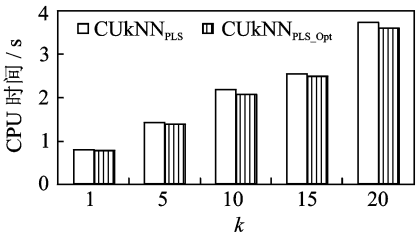
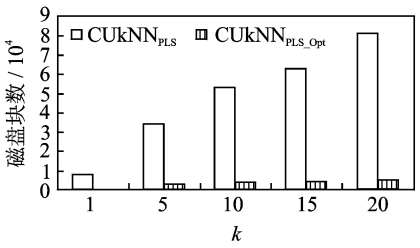


图 5 查询时间段的长度对算法性能的影响

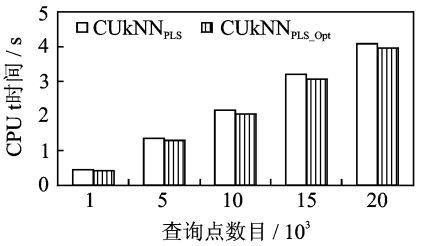


(a) CPU运行时间

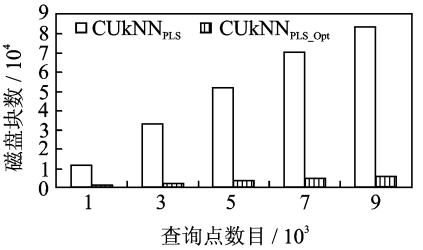


(b) 访问的磁盘块数

图 6　k 值对算法性能的影响



(a) CPU运行时间



(b) 访问的磁盘块数

图 7　查询点数目的变化对算法性能的影响

照者。

图 6(b)表明,k 值对算法 CUkNN$_{PLS}$的访问磁盘块数有很大的影响。然而,k 值对算法 CUkNN$_{PLS_Opt}$的访问的磁盘块数的影响较小。如图所示,当 k 值由 10 变为 20,CUkNN$_{PLS}$访问的磁盘块数增加了 55.1%,而 CUkNN$_{PLS_Opt}$访问的磁盘块数仅仅增加了 25.1%。这是因为 CUkNN$_{PLS_Opt}$采用了空间填充曲线来组织单元的编号,使得相邻的结点尽可能地位于相邻的单元中,当 k 值变大时,所需要访问的磁盘块的数目的增长要缓慢得多。

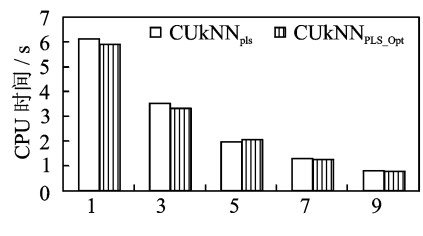
如图 7(a)所示,随着查询点数的增加,这两种算法的 CPU 运行时间几乎是线性增长的。这是因为总的运行时间就是系统中各查询运行时间的总和。

图 7(b) 表明查询点数目的多少对这两种算法的磁盘访问块数有着相当不同的影响。对 CUkNN$_{PLS}$而言,当增加系统中查询的数目时,访问的磁盘块数几乎是线性增长的;而对于 CUkNN$_{PLS_Opt}$而言,这种增长趋势要平缓些。如图所示,当查询数由 5 000 增至为 9 000 时,CUkNN$_{PLS_Opt}$访问的磁盘块数只增加了 35.0%。这是因为 CUkNN$_{PLS_Opt}$算法按位置邻近的原则来组织路网结点,位置相邻的查询点可全部或部分共享有关磁盘块的信息,因此当查询数变大时,所需要访问的磁盘块的数目增长要缓慢些。

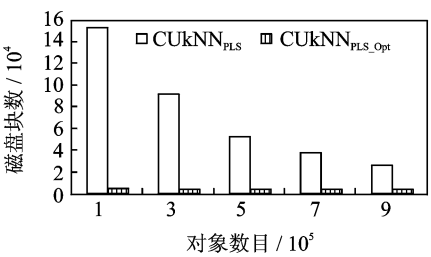
最后考察变化的移动对象数目对这些算法性能的影响。

图 8(a)表明,随着系统中移动对象的数目的增加,这两种算法的 CPU 运行时间有减少的趋势。这是因为随着对象数目的增加,系统中对象的密度也增加,此时一个查询的监控范围随之缩小,那么查询所需要的距离计算和距离比较也相应减少。因此,运行时间减少。

图 8(b)表明,对象数目的变化对这两种算法的访问磁盘块数有着不同的影响。对 CUkNN$_{PLS}$而言,随着对象数目的增加,访问的磁盘块数明显地减少。而算法 CUkNN$_{PLS_Opt}$的减少趋势却是比



(a) CPU运行时间



(b) 访问的磁盘块数

图 8　对象数目的变化对算法性能的影响

较平缓的。其原因与图6(b)类似。

从实验结果看,采用了优化技术的CUkNN$_{PLS_Opt}$算法的平均CPU时间比没采用优化技术的CUkNN$_{PLS}$算法少5.06%,CUkNN$_{PLS_Opt}$算法的平均磁盘访问块数只占没有采用优化技术的CUkNN$_{PLS}$算法的7.56%;此外,CUkNN$_{PLS_Opt}$算法对查询时间段、k值、对象数和查询点数发生变化时,其变化趋势比CUkNN$_{PLS}$算法要平缓些,这说明优化技术的采用使得查询处理算法具有更好的可扩展性。

4 结束语

本文讨论了基于预计算的连续k近邻查询方法扩展到大路网时所存在的问题,提出了CUkNN$_{PLS_Opt}$算法以减少查询处理时的内外存交换次数、提高查询处理的效率。实验表明所提出的优化技术能使查询处理时间减少5.06%,并且使查询处理的平均磁盘访问块数减少92.44%,因此是有效的。此外,优化技术的采用使得查询处理算法具有更好的可扩展性。

参考文献:

[1] Mouratidis K, Yiu M L, Papadias D, et al. Continuous nearest neighbor monitoring in road networks [C]//Proc of VLDB. New York: ACM Press, 2006: 43-54.

[2] Huang Yuan-Ko, Chen Zhi-Wei, Lee Chiang. Continuous K-nearest neighbor query over moving objects in road network [C]//Proc of APWeb-WAIM. Suzhou: Springer, 2009: 27-38.

[3] Li Guohui, Li Yanhong, Shu LihChyun. CkNN query processing over moving objects with uncertain speeds in road networks[C]//Proc of APweb. Beijing: Springer, 2011: 65-76.

[4] Zheng B, Xu J, Lee W C, et al. Grid-partition index: a hybrid method for nearest-neighbor queries in wireless location-based services [J]. VLDB Journal, 2006, 15(1): 21-39.

[5] Xiong Xiaopeng, Mokbel M F, Aref W G. SEA-CNN: Scalable processing of continuous k-nearest neighbor queries in spatio-temporal databases[C]// Proc of ICDE. Tokyo: IEEE, 2005: 643-654.

[6] Papadias D, Zhang J, Mamoulis N, et al. Query processing in spatial network databases [C]//Proc of VLDB. San Francisco: Morgan Kaufmann, 2003: 802-813.

[7] Kolahdouzan M R, Shahabi C. Voronoi-based K nearest neighbor search for spatial network databases [C]//Proc of VLDB. San Francisco: Morgan Kaufman, 2004: 840-851.

[8] Gotsman C, Lindenbaum M. On the metric properties of discrete space-filling curves[J]. IEEE Trans Image Process, 1996, 5(5): 794-797.