道路网中的移动对象连续 K 近邻查询

赵 亮[1)] 陈 荦[1)] 景 宁[1)] 廖 巍[2)]

[1)](国防科学技术大学电子科学与工程学院 长沙 410073)

[2)](海军工程大学电子工程学院 武汉 430033)

摘 要 已有道路网中的连续 k 近邻查询处理算法采用增量式的查询处理机制,当数据频繁更新时性能急剧下降.结合多核多线程技术,提出了一种基于多线程的连续查询处理框架.该框架周期性重计算所有查询结果,将查询处理分为顺序执行的数据更新阶段和查询执行阶段,分别使用任务并行和数据并行的方法执行各阶段的操作.设计了数据更新阶段使用的数据结构,提出了查询处理阶段的 k 近邻查询处理策略,包含离线预计算和在线 k 近邻查询处理算法两个部分.对 k 近邻算法复杂性及多线程处理框架的加速比进行了理论分析.实验结果表明,提出的算法在数据频繁更新下,串行执行时性能优于已有算法,而基于多线程处理框架的并行执行在任何参数配置下性能均优于已有算法;且基于多线程处理框架的并行执行具有较好的性能扩展性,加速比可以达到1.51~1.7.

关键词 移动对象;道路网;连续 k 近邻查询;多线程;频繁更新

中图法分类号 TP392 **DOI号**: 10.3724/SP.J.1016.2010.01396

Continuous K Nearest Neighbor Queries of Moving Objects in Road Networks

ZHAO Liang[1)] CHEN Luo[1)] JING Ning[1)] LIAO Wei[2)]

[1)](*College of Electronic Science and Engineering, National University of Defense Technology, Changsha* 410073)

[2)](*College of Electronic Engineering, Navy University of Engineering, Wuhan* 430033)

Abstract Existing algorithms for continuous k-nearest neighbor (kNN) queries of moving objects in road networks adopt the incremental query mechanism, which is verified to be inefficient when data update frequently. Considering the ubiquitous multi-core and multi-threading technologies, a multi-threading based framework for continuous kNN queries of moving objects is proposed. In the framework, all the queries are re-evaluated periodically, and the query process is divided into two sequential phases: the data updating phase and the query execution phase, task parallel and data parallel mechanism are used respectively in each phase to carry out the corresponding operations. In the updating phase, the data structures used in the framework are designed. Moreover, in the execution phase a kNN query strategy is proposed which includes an off-line pre-computation and an on-line query algorithm. The computational complexity of the algorithm and the speedup of the framework are analyzed theoretically. Experimental results show that, under the frequent update environment, the proposed query algorithm when serially executed has better performance than the traditional algorithms, however, the multi-threading based parallel execution is better in all kinds of parameter setups; what's more, the multi-threading based parallel execution maintains good performance scalability, and the speedup can reach to 1.51~1.7.

Keywords moving objects; road network; continuous k-nearest neighbor queries; multi-threading; frequent update

---

收稿日期:2010-06-11.本课题得到国家自然科学基金(40801160,60902036)、国家"八六三"高技术研究发展计划项目基金(2008AA12A211)和中国博士后科学基金项目(20080431384)资助.**赵 亮**,男,1982年生,博士研究生,主要研究方向为时空数据库、移动对象数据库.E-mail:liangzhao2010@gmail.com.**陈 荦**,男,1973年生,博士,副教授,主要研究方向为地理信息系统与数据库技术.**景 宁**,男,1963年生,博士,教授,博士生导师,主要研究领域为地理信息系统、空间数据库.**廖 巍**,男,1980年生,博士,讲师,主要研究方向为时空数据库.

1 引　言

随着无线通信、计算技术、GPS 空间定位等技术的快速发展以及众多具有定位功能的无线手持和车载设备的大量普及，位置服务目前已经成为了一个独具特色、前景无限的新兴产业，得到了越来越多的关注. 位置服务在交通调度管理、救援服务、战场指挥等领域有着广泛的应用前景[1]. 尤其是在城市道路网络环境下，许多新型应用如路边协助、交通导航、位置感知广告服务等由于切合用户的实际需求，引起了人们广泛的兴趣. 其中一个典型的需求是对道路网中的移动对象进行连续查询，比如连续查询“距目前位置最近的 10 辆出租汽车”.

移动对象的连续查询是近年来移动对象数据库领域的研究热点，然而其中大多数方法[2-6]都假设移动对象在欧氏空间中自由运动，因而以欧氏距离来判断移动对象是否在查询结果集中. 这些方法虽然有其特定的应用领域(如战场监控、航空管理等)，但是由于现实中绝大多数用户和移动对象的运动都受限于道路网络，用户和移动对象之间的距离通常以网络距离衡量，因而如果仍然使用欧氏距离来计算查询结果，则显然不能得出正确的查询结果. 目前典型的道路网中移动对象并发连续 k 近邻查询处理算法包括文献[7]中提出的 IMA/GMA 算法和文献[8]中提出的 ER-CkNN 算法. 这两种算法是对传统的道路网中静态空间对象查询处理算法[9]的时空扩展. 然而，在移动对象数据频繁更新的情况下，上述算法性能急剧恶化，甚至不如采用每个周期重复计算所有查询结果的方式.

目前，多核处理器逐渐成为市场的主流. 在多核处理器平台上，上述传统的单线程算法均无法充分利用处理器的计算资源(在双核平台上运行已有单线程算法，处理器的使用率最高为 50%). 本文的研究旨在利用多核处理器的多线程并行计算能力提高道路网上移动对象连续 k 近邻查询处理的性能. 因此，重点提出了一种基于多线程的查询处理框架，设计了包含离线预计算及在线查询处理算法两部分的 k 近邻查询处理策略.

2 相关工作及问题描述

2.1 相关工作

目前典型的道路网中移动对象连续 k 近邻查询处理算法有 IMA/GMA 算法[7]和 ER-CkNN 算法[8].

IMA/GMA 算法将移动对象数据、多用户并发查询、道路网数据全部组织在内存中. 对于每一个查询来说使用网络扩张的方法获得其初始结果集，即从查询所在的位置开始，遍历周围的边及其上的移动对象，根据到移动对象的网络距离不断地更新查询结果集. 此外，IMA 算法将网络扩张中遍历过的结点组织成一个称为查询扩张树的数据结构，基于这种数据结构，IMA 算法通过判断边权重、移动对象位置、查询位置的变化，对扩张树进行修剪，从而重用扩张树中的查询结果. GMA 算法则将路径(起点和终点的度数不等于 2)上的查询组成一组，同一路径上查询的结果集是路径两个端点的 k 近邻查询结果和该路径上移动对象并集的子集. 基于这一性质 GMA 算法利用 IMA 算法监控每条路径两个端点的 k 近邻查询结果，并判断数据的更新会影响哪些查询的结果，而后重新计算这些受影响的查询. 通过分析和实验，发现 IMA/GMA 算法的不足可以概括为：(1) IMA/GMA 会进行大量计算以判断是否需要对查询重计算，当数据频繁更新时，绝大多数查询都需要重计算，因此性能急剧下降；(2) GMA 算法不能处理 k 值不同情况下的多用户并发查询；(3) 当道路网规模较大时，其基本的网络扩张算法性能下降.

ER-CkNN 算法的提出者注意到了 IMA/GMA 算法由于网络扩张造成的性能不佳，因而提出了预计算一种称为 Edge Bitmap Encoding 的数据结构，通过这种结构能够快速计算给定两点的最短路径. 在初始结果计算时，除了利用这一结构外，还采用了文献[9]中的欧氏距离限制的思想，即快速找到候选结果集，而后利用欧氏范围查询不断对结果集精炼得到最终结果. 在维护数据更新时，与 IMA/GMA 算法类似需要进行大量判断，实现增量式的结果更新机制. ER-CkNN 算法的不足可以概括为：(1) 当移动对象数据频繁更新时，性能急剧下降；(2) Edge Bitmap Encoding 数据结构在道路网中边权重更新时需要重新计算.

此外，文献[10]中针对自由运动移动对象的连续 k 近邻查询处理问题，提出了基于流水线式多线程的查询处理框架和 k 近邻查询处理算法，证明了多核平台上多线程技术能够提高连续查询处理的性能. 但是，道路网连续查询算法中的数据依赖使得流水线式多线程处理框架难以发挥流水线的优势，而正如引言中所述，基于欧氏距离的算法不适用于道路网上的距离计算.

正是注意到了已有算法在面对数据频繁更新时的性能低下问题，且多线程技术能够带来算法性能的提高，开展了本文的研究.

2.2 问题描述

在给出本文解决方案之间，首先对道路网中的连续 k 近邻查询问题做出形式化的定义.

定义 1(道路网). 一个道路网络可以表示为一个无向带权图 $G=(N,E,W)$，其中，N 是顶点(结点)的集合；E 是边的集合；W 为正的实数集合($W:E\rightarrow R^+$)，表示边所对应的权值.

定义 2(移动对象). 道路网中的一个移动对象位于网络对应的无向带权图中的边 $e(e\in E)$ 上，移动对象在图中的位置可以表示为一个三元组(n_i，n_j，pos)，其中，n_i 和 n_j 是移动对象所在边的两个结点；$pos\in[0,W(e)]$ 是移动对象与结点 n_i 之间的距离.

定义 3(网络距离). 道路网中任意两个对象 $p_1=(n_{1i},n_{1j},pos_1)$，$p_2=(n_{2i},n_{2j},pos_2)$ 之间的网络距离定义为

$$d_N(p_1,p_2)=\min\begin{Bmatrix} pos_1+d_N(n_{1i},n_{2i})+pos_2 \\ pos_1+d_N(n_{1i},n_{2j})+W(e_2)-pos_2 \\ W(e_1)-pos_1+d_N(n_{1j},n_{2i})+pos_2 \\ W(e_1)-pos_1+d_N(n_{1j},n_{2j})+W(e_2)-pos_2 \end{Bmatrix}.$$

定义 4(连续 k 近邻查询). 道路网中的连续 k 近邻查询 q 位于网络对应的无向带权图中的边 $e(e\in E)$ 上，查询在图中表示为一个四元组(n_i，n_j，pos，k)，其中，n_i 和 n_j 是查询所在边的两个结点；$pos\in[0,W(e)]$ 是查询与结点 n_i 之间的距离；k 表示该 k 近邻查询需要返回的移动对象个数. 连续 k 近邻查询的含义是连续地给出距离 q 位置最近的 k 个移动对象. 即，考虑移动对象集合 $S=\{p_1,p_2,\cdots,p_n\}$，连续 k 近邻查询 q 的结果集 $S'\subseteq S$，对于任意 $p'\in S'$ 以及 $p\in S-S'$，$d_N(p',q)\leq d_N(p,q)$. 其中 d_N 表示查询点与移动对象之间的网络距离，其计算方法与定义 3 相同.

3 基于多线程的连续查询处理框架

解决并发连续查询处理的一般方法是以批处理的形式，周期性地给出所有查询的结果. 与已有的移动对象增量式连续查询处理技术相比，本文采用多线程技术在每个周期重复计算所有查询结果，力求利用多核处理器的并行处理能力提高连续查询处理的性能. 为此，提出了一种基于多线程的并发连续查询处理框架，如图 1 所示. 该框架的主要思想是：在多核处理器平台中，将连续查询转换为周期性地查询，在每个周期将包括数据更新和查询处理在内的操作分为两个阶段(数据更新、查询处理)，在每个阶段针对该阶段内的操作采用多线程进行处理. 需要注意的是该框架并非基于流水线策略，而是第一个阶段内的所有多线程操作执行完才能继续执行下一个阶段的操作. 下面将分别说明各个阶段在框架中的作用和其中使用的数据结构.

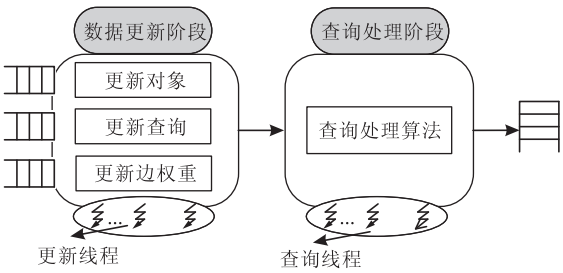


图 1 基于多线程的连续查询处理框架

3.1 数据更新阶段

在连续查询处理问题中，移动对象位置信息和查询的位置信息都是不断变化的. 因此，需要对这些信息在查询处理之前进行更新，保证结果的即时性和正确性. 为此，在数据更新阶段采用任务并行的方式，分别使用对象更新线程、查询更新线程和边更新线程更新系统中的 3 种数据. 由于采用周期性的查询处理策略，因此需要对一个周期内发生更新的数据进行缓存，如图 1 中数据更新阶段左侧作为输入的 3 个缓存队列. 在这一阶段设计了如下内存数据结构来存储各种数据：

首先，使用边表(Edge Table，ET)存储边的信息. 该结构是一个基于内存的 Hash 表，存储如下信息：(1) 边的 id；(2) 每条边的起始和终止结点；(3) 每条边的权重. 其次，使用格网索引存储移动对象信息. 将移动对象的运动空间划分为均匀的单元格，按照移动对象的空间位置将其映射到对应的单元格中，每个单元格维护一个列表保存指向移动对象数据的指针. 移动对象真实数据存储在线性数组中，其中每一条记录包含如下信息：(1) 移动对象 id；(2) 移动对象空间位置；(3) 移动对象所在边的 id；(4) 移动对象在边上的位置. 最后，使用查询表(Query Table，QT)存储连续 k 近邻查询的信息. 该结构是一个基于内存的线性结构，存储如下信息：(1) 查询的 id；(2) 查询的空间坐标；(3) 查询的 k 值；(4) 查询结果集；(5) 查询所在的边的 id；(6) 查询在边上的位置.

在本阶段的多线程执行中，移动对象格网索引更新线程、边更新线程和查询表更新线程之间由于

操作的数据不同，因此不存在写操作的同步问题.

3.2　查询处理阶段

查询处理阶段在获得前一个阶段准备的数据后，调用查询处理算法计算每个查询的结果，如图 1 中查询处理阶段及右侧查询结果列表. 为此，采用数据并行的方式将并发的多用户查询均匀划分给多线程，每个线程调用 k 近邻查询处理算法依次处理每个查询得出最终的查询结果. 查询处理阶段使用的数据结构在第 4 节具体讲述.

由于多线程之间针对不同的查询进行计算，因此不会存在对相同数据同时进行写操作的同步问题.

4　k 近邻查询处理策略

本节设计了包含离线预计算与在线查询处理两部分的 k 近邻查询处理策略.

4.1　离线预计算

离线预计算首先对道路网中的边利用 PMR-Quad 树构建索引；其次对移动对象的运动空间用格网进行划分，计算每条边经过的单元格，形成边和单元格之间的映射表. 经过这些预计算得到的辅助数据结构有助于在查询处理算法中快速地定位可能的查询结果. 离线预计算在整个框架及算法应用过程中只需执行一次，无论是 PMR-Quad 树索引还是边和单元格之间的映射表都不会发生改变. 利用 PMR-Quad 树构建边索引的方法可见文献[11]，这里重点给出计算每条边经过的单元格的算法.

给定一条网络边 $e(n_1,n_2)$，n_1 和 n_2 的空间位置分别为 (n_{1i},n_{1j})，(n_{2i},n_{2j})，则该网络经过的单元格可用算法 1 计算.

算法 1.　计算交叉的单元格.

Input：n_1，n_2

Output：*cellList*

1. $s=(n_{2j}-n_{1j})/(n_{2i}-n_{1i})$，$b=n_{1j}-s\times n_{1i}$
2. $beginX=\lfloor n_{1i}/l \rfloor$，$beginY=\lfloor n_{1j}/l \rfloor$
3. $endX=\lfloor n_{2i}/l \rfloor$，$endY=\lfloor n_{2j}/l \rfloor$
4. while $beginX!=endX$
5.　if $endX>beginX$
6.　$nextX=beginX+1$
7.　else
8.　$nextX=beginX-1$
9.　$nextY=\lfloor (s\times nextX\times l+b)/l \rfloor$
10.　for $i=beginY$ to $nextY$
11.　$cellList=cellList\cup c(beginX,i)$
12.　$beginX=nextX$，$beginY=nextY$
13. for $i=beginY$ to $endY$
14.　$cellList=cellList\cup c(endX,i)$
15. return *cellList*

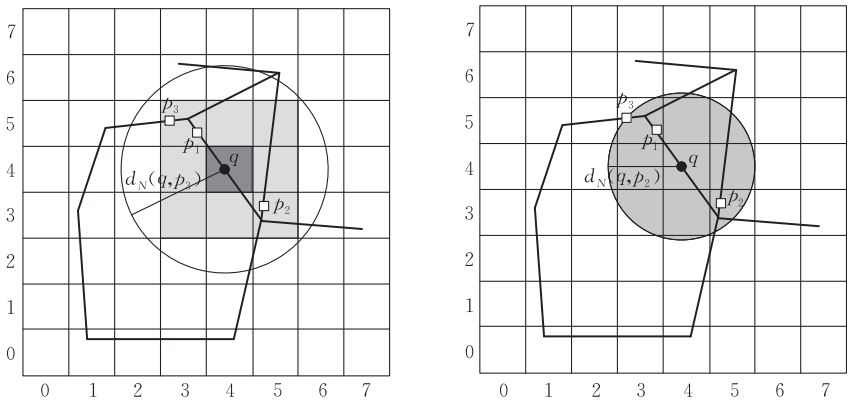
算法 1 首先计算由边两个端点组成的线段的斜率和截距(第 1 行)，而后计算起点和终点所在的单元格(第 2～3 行). 接下来算法在 x 轴方向每次推进一个单元格，在每次循环中将 y 轴方向所有与线段相交的单元格加入结果集中(第 5～12 行). 最后算法将终点所在列与线段相交的单元格加入结果集(第 13～14 行).

通过算法 1 可构造边和单元格之间的映射表.

4.2　在线查询处理算法

在线处理部分提出了过滤精炼的两步查询处理算法. 首先在过滤阶段，使用边和单元格的映射表快速过滤不可能的查询结果，并确保所有可能的结果都在候选结果集中；在精炼阶段，不断利用基于欧氏距离的范围查询排除不正确的结果.

具体来说，过滤阶段的实现可用图 2(a)进行解释. 以 q 为 2 近邻的查询为例，首先确定 q 所在的单元格，并判断该单元格内是否存在 k 个对象，若不存在则在该单元格周围的 8 个单元格中进行搜索，直到找到至少 k 个为止. 而后以这些对象中距查询 q



(a) 过滤阶段示意图　　(b) 精炼阶段示意图

图 2　k 近邻查询处理算法示意图

最远的网络距离为半径作圆，并以该圆为一个空间范围查询，通过 PMR-Quad 树查找与该圆相交的边，通过边和单元格的映射关系找到格网索引中的移动对象作为候选结果集. 此时的候选结果集包含了全部可能的查询结果.

精炼阶段可用图 2(b)进行解释. 在得到了候选结果集后，依次利用候选结果集中移动对象与查询的网络距离进行欧氏范围查询，并排除不正确的结果. 即在图 2(b)中用 q 到 p_2 的网络距离进行范围查询，得到结果集中的移动对象个数为 2 时停止，这两个移动对象即为最终的基于道路网距离的 k 近邻查询结果.

对于每一个查询 q 来说，其 k 近邻查询处理算法如算法 2 所示.

算法 2. kNN 算法.

Input: q, *EdgeCellTable*, *GridIndex*, *EdgeTable*, *Adjacency List*(*AL*)
Output: *ResultSet*
//Filter Step
1. *TempCellList*=*CellExpansion*(q)
2. *kthNDT*=*maxNDT*(*TempCellList*)
3. *edgeList*=*euclideanRangeQuery*(q, *kthNDT*)
4. *candidateResult*=*retrieveObjects*(*edgeList*)
//Refinement Step
5. if(*candidateResult*.*length*=k)
6. *ResultSet*=*candidateResult*
7. else
8. while(*candidateResult*.*length*>k)
9. p=*nextEuclideanNN*(*candidateResult*)
10. if($d_N(q,p)$<*kthNDT*)
11. *updateResult*(p)
12. *kthNDT*=$d_N(q,p)$
13. *edgeList*=*eculideanRangeQuery*(q,*kthNDT*)
14. *candidateResult*=*retrieveObjects*(*edgeList*)
15. *ResultSet*=*candidateResult*
16. return *ResultSet*

本文提出的 k 近邻查询处理策略中，离线预计算阶段只需进行一次计算，在以后的查询处理过程中不需要对离线预计算阶段的结果进行更改；而在线 k 近邻查询处理算法则通过欧氏距离限制过快的速度得到候选结果集，使得在计算网络距离时不必在整个道路网中遍历.

5 理论分析

5.1 k 近邻算法复杂性

为了简化分析，假设道路网和移动对象在单位正方形空间内均匀分布，则移动对象的空间位置坐标在$[0,1]\times[0,1]$区间内取值. 格网单元格的边长为 l，则运动空间总计包含 $1/l\times 1/l$ 个单元格. 道路网的边数为 E，移动对象总数为 M.

对每个查询 q，k 近邻查询处理算法按算法 2 所示步骤进行处理，依次来分析每条语句的计算复杂性. 第 1 行进行单元格扩张以初选 k 个移动对象，因此包含 k 个对象的区域面积为$\frac{k}{M}$，边长为$\sqrt{\frac{k}{M}}$需要遍历$\frac{k}{Ml^2}$个单元格，复杂度为 $O\left(\frac{k}{Ml^2}\right)$. 第 2 行计算单元格中移动对象与 q 的最大网络距离，相当于在$\frac{kE}{M}$条边组成的子图中计算两点之间的最短路径，根据 Dijkstra 算法得到其复杂度为 $O\left(\frac{kE}{M}\lg\frac{kE}{M}\right)$. 第 3 行进行基于 PMR-Quad 树的欧氏范围查询，其复杂度为 $O(2E^{1/2})$[12]. 第 4 行基于边和单元格映射表获取候选数据集，由于该表已预计算得出，因此只与边数有关，其复杂度为 $O\left(\frac{kE}{M}\right)$. 第 5～15 行排除候选结果集中的不正确结果. 在最好情况下，候选结果集中的对象个数刚好为 k，算法终止. 如果个数大于 k，由于候选结果集中最多包含$\left(\sqrt{\frac{k}{M}}+l\right)^2 M$个对象，所以需要进行$\left(\sqrt{\frac{k}{M}}+l\right)^2 M-k$次循环. 在最差情况下第 11～14 行的复杂度等于第 2～4 行.

综上所述，在最好情况下，k 近邻算法复杂度为

$$O\left(\frac{k}{Ml^2}+\frac{kE}{M}\lg\frac{kE}{M}+\frac{kE}{M}+2E^{1/2}\right) \tag{1}$$

在最差情况下，k 近邻算法复杂度为

$$O\left\{\frac{k}{Ml^2}+\frac{kE}{M}\lg\frac{kE}{M}+2E^{1/2}+\frac{kE}{M}+\left[\left(\sqrt{\frac{k}{M}}+l\right)^2 M-k\right]\left(\frac{kE}{M}\lg\frac{kE}{M}+2E^{1/2}+\frac{kE}{M}\right)\right\} \tag{2}$$

事实上，算法计算时间主要由两部分决定，一是单元格扩张时间(第 1 行)，二是计算网络距离并获取移动对象时间(第 2～4 行). 相对来说单元格扩张时间对算法性能影响较小. 因此，综上由式(2)可以得出以下结论：

(1) 算法计算时间随 k、E 的增大而增她长；

(2) l 较大时，计算时间较长；l 越小计算时间越短；但是 l 小到一定程度后，由式(2)知计算网络距离并获取移动对象时间变化不大，因此算法计算时间趋于平稳.

(3) M 较小时，计算时间较长；M 越大计算时间

越短；但是 M 大到一定程度后，由式(2)知计算网络距离并获取移动对象时间变化不大，因此算法计算时间趋于平稳.

5.2　多线程处理框架的加速比

接下来分析多线程处理框架的加速比. 通常并行系统的加速比可表示为

$$S=\frac{T_s}{T_p} \tag{3}$$

其中 T_s 为串行算法的执行时间，T_p 为并行算法的执行时间.

在本文的多线程并行处理框架中，连续 k 近邻查询处理算法的串行执行时间可以表示为

$$T_s=(T_{ou}+T_{eu}+T_{qu})+T_{qe} \tag{4}$$

其中，T_{ou}、T_{eu}、T_{qu} 分别表示更新移动对象、更新边的权重、更新查询的执行时间，T_{qe} 表示针对所有并发的查询执行查询处理算法的时间.

基于多线程处理框架的并行执行时间可以表示为

$$T_p=\max(T_{ou},T_{eu},T_{qu})+T_{qe}/n+T_c \tag{5}$$

其中 n 表示处理器核数，T_c 表示多线程启动、通信等的开销. 由于数据更新阶段使用任务并行的机制，所以该阶段的执行时间等于最大的数据更新时间；而由于查询处理阶段使用数据并行机制，所以该阶段的执行时间等于原执行时间除以并行执行的线程数(一般设置为处理器核数)；此外，多线程在启动、终止、上下文切换、访问共享缓存等方面存在开销.

综上，框架的并行执行时间等于上述三者之和. 将式(4)、(5)代入式(3)可得，本文提出的基于多线程的并行执行框架的加速比为

$$S=\frac{(T_{ou}+T_{eu}+T_{qu})+T_{qe}}{\max(T_{ou},T_{eu},T_{qu})+T_{qe}/n+T_c} \tag{6}$$

需要特别说明的是在传统的单核处理器平台上，只能将多个指令流交错执行，因此多线程一般都被当做是一种能够实现延迟隐藏的编程手段. 在单核平台上执行本框架并不能提高算法性能，具体来说在单核平台上式(5)中 T_c 所占比例会大大增加. 在多核平台上，其真正的多线程执行硬件环境能够极大降低 T_c 的值，因而能够提高查询处理性能. 但是多核平台上例如双核平台上通常也只具备 2 个线程并行执行的能力，因此在执行数据并行任务时，一般设置线程数等于处理器核数.

6　实验结果与分析

实验的硬件环境为：Intel Pentium Dual E2200 2.2GHz CPU，1MB L2 高速缓存，1GB 内存，操作系统采用 Windows XP SP2.

实验中分别采用上海市和德国 Oldenburg[13] 两个城市的道路网来生成移动对象和查询. 上海市道路网数据为本实验室购买的 shapefile 文件，该文件经处理后共有 175710 个道路结点和 180660 条道路边. 自行设计了道路网上移动对象生成算法，随机生成移动对象和查询. 德国城市 Oldenburg 的道路网数据由文献[13]获得，该道路网为移动对象研究中广泛采用的测试数据集，共有 6105 个道路结点和 7035 条道路边. 采用标准的移动对象生成器生成移动对象和查询. 表 1 列出了实验中采用的各种参数的范围及其默认值. 其中数据的灵敏性是指每个周期发生变化的数据占总数据量的比例，体现了数据频繁更新的程度.

表 1　实验参数

实验参数	默认值	参数范围
移动对象数目(N)	100k	10，50，100，150，200(k)
查询数目(Q)	10k	1，5，10，25，50(k)
对象分布	Uniform	Gaussian，Uniform
查询分布	Gaussian	Gaussian，Uniform
移动对象灵敏性	50%	10%，30%，50%，70%，100%
查询灵敏性	50%	10%，30%，50%，70%，100%
边的灵敏性	4%	1%，2%，4%，8%，16%

实验中将本文提出的不采用多线程执行(即仍采用两阶段的重计算所有查询的机制，但在每个阶段内串行执行每个操作)的 k 近邻算法称为 S-kNN，将基于多线程处理框架的 k 近邻查询处理称为 P-kNN，此外为了与已有算法进行比较，实现了第 2 节分析的 IMA 和 ER-CkNN 算法. 这些算法均采用 C++语言在 Visual Studio.net 2005 开发环境中编程实现. 由于框架和算法中使用的全部数据结构都是基于内存，因此在算法比较中以 CPU 执行时间作为评价指标，该值的获取采用运行 10 个周期取平均的方法.

6.1　数据分布及网格划分对框架性能的影响

首先利用上海市道路网和默认数据量及灵敏性，测试数据分布和格网划分对本文提出的基于多线程的连续查询处理框架性能的影响. 本实验中 $k=2$.

图 3 显示了数据分布对框架性能的影响，可见当查询为高斯分布时框架性能较好，而移动对象的分布对框架性能影响不大. 这是因为，查询比较集中时计算网络距离的数据重用性较高. 在以后的实验中使用均匀分布的移动对象和高斯分布的查询. 需要说明的是 IMA 和 ER-CkNN 算法当查询为高斯

分布时同样性能达到最优. 图 4 显示了随单元格数量变化框架的性能变化情况. 可见随着单元格数量的增加框架性能越来越好,但是当单元格数目大于一定数目时,性能提高不明显,与第 5 节理论分析结果相吻合. 在以后的实验中设置将运动空间划分为 400×400 个单元格.

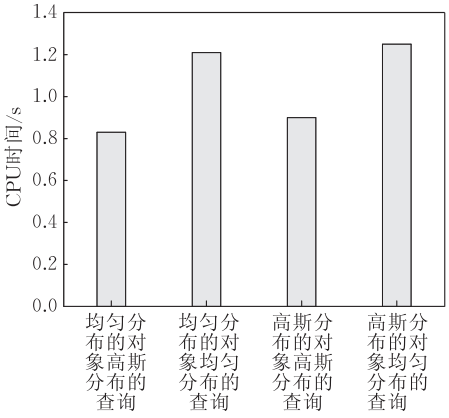


图 3 数据分布对框架性能的影响

6.2 数据灵敏性对性能的影响

接下来利用上海市道路网和默认数据量,测试数据灵敏性对已有算法和本文提出框架性能的影响. 本实验中 $k=2$.

图 5 显示了实验结果. 可以看到在 3 种数据的灵敏性逐渐增大时,S-kNN算法性能逐渐优于传统的 IMA 算法和 ER-CkNN 算法,其中图 5(c)所示在对象和查询灵敏性设置为默认的 50%时,S-kNN 始终优于已有的算法. 这是因为 IMA 和 ER-CkNN 算法采用了增量式的查询处理机制,但是当数据频繁更新时,判断是否需要重新计算查询结果的代价越来越高,抵消了结果重用带来的性能提升. 此外,还可以看到,无论在何种数据设置条件下,基于多线程查询处理框架的 P-kNN 性能始终优于已有算法. 在 3 种情况下,随着灵敏性的提高 S-kNN 和 P-kNN 的 CPU 执行时间略有上升,这是因为灵敏性越高需要更新的数据量越多,在数据更新阶段的时间代价越大,但由于整个框架中查询处理阶段的时间代价占很大比例,因此更新代价的提高并不会严重影响整个框架的性能.

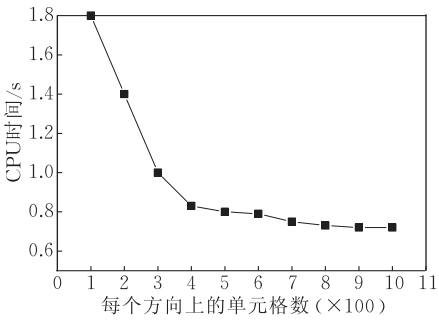
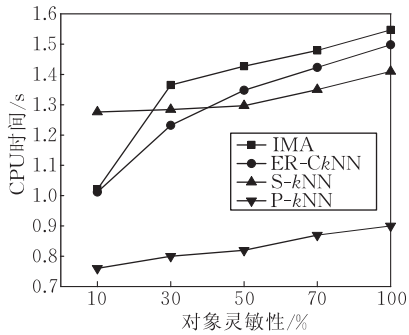
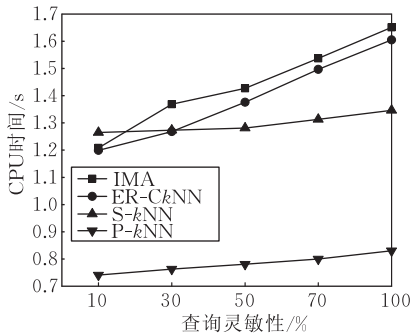


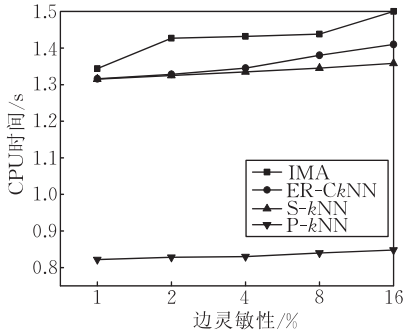
图 4 格网划分对框架性能的影响



(a) 对象灵敏性对性能的影响



(b) 查询灵敏性对性能的影响



(c) 边灵敏性对性能的影响

图 5 数据灵敏性对算法性能的影响

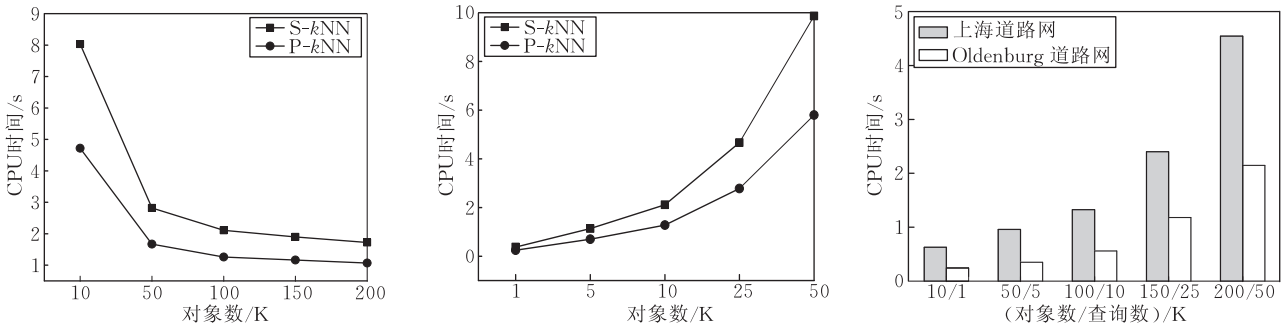
6.3 数据量对性能的影响

最后,在默认数据灵敏性和不同数据量下测试本文提出的 k 近邻查询处理算法在串行执行和多线程并行执行框架下的性能比较. 本实验中 $k=10$.

图 6(a)、(b)显示了采用上海道路网的实验结果. 图 6(a)显示了移动对象数目对框架执行性能的影响. 可以看到,随着移动对象数据量的增加,本文提出的框架执行时间越少,且在数目大到一定程度后性能趋于平稳. 这是因为对象较少时,在 k 近邻算法中采用 Dijkstra 算法计算网络距离时的代价越大,而这种代价是整个框架执行中代价最大的部分. 相反的,当对象数目较多时,计算网络距离的代价就越小. 在图 6(a)显示的实验结果中,在不同对象数目情况下加速比变化不大,平均为 1.61. 图 6(b)显示了查询数目对框架执行性能的影响. 可以看到随着查询数目的增多查询处理的时间越大,且并行执行的性能加速比逐渐变大. 这是因为在基于多线程的查询处理框架中的查询处理阶段将查询集合均匀划分给多线程执行,查询数目越多查询处理阶段在整个框架执行代价中的比重越高,由式(4)知,加速

比越接近 n. 在图 6(b)显示的实验结果中，平均加速比为 1.67. 在图 6(a)、(b)中，加速比在 1.51～1.7 之间变化.

图 6(c)显示了在不同移动对象和查询数据量设置下，多线程处理框架在不同道路网规模下的性能比较. 可以看到，上海道路网的规模虽然为 Oldenburg 道路网的 20 多倍，但是查询处理的时间仅为其 2～4 倍. 这表明，框架对道路网的规模具有较好的性能扩展性.



(a) 对象数目对性能的影响　(b) 查询数目对性能的影响　(c) 网络规模对性能的影响

图 6　数据量对框架性能的影响

7　结　论

道路网中连续 k 近邻查询处理的难点在于：(1) 如何对并发的查询进行周期性维护；(2) 如何快速有效地计算两点之间的网络距离或者减少距离计算的次数. 本文采用基于多线程的查询处理框架周期性地重计算所有查询结果，解决了第 1 个难点；通过设计了包含离线预计算和在线查询处理算法两部分的 k 近邻查询处理策略解决了第 2 个难点.

在实际道路应用中，道路网往往有很多约束，例如单行路、有些路口不允许左右转弯等，因此如何基于更加复杂的道路网模型进行算法设计，使研究成果更加贴近实际应用，是下一步重点研究的内容. 此外，在设计时空连续查询处理算法时应充分考虑多核体系结构带来的并行计算能力.

参　考　文　献

[1] Wolfson Ouri. Moving objects information management: The database challenge//Proceedings of the International Workshop on Next Generation Information Technologies and Systems. Caesarea, Israel, 2002: 75-89

[2] Xiong Xiaopeng, Mohamed F M, Walid G A. SEA-CNN: Scalable processing of continuous k-nearest neighbor queries in spatio-temporal databases//Proceedings of the International Conference on Data Engineering. Tokyo, Japan, 2005: 643-654

[3] Yu Xiaohui, Ken Q P, Nick Koudas. Mointoring k-nearest neighbor queries over moving objects//Proceedings of the International Conference on Data Engineering. Tokyo, Japan, 2005: 631-642

[4] Mouratidis K, Hadjieleftheriou M, Papadias D. Conceptual partitioning: An efficient method for continuous nearest neighbor monitoring//Proceedings of the International Conference on Management of Data. Baltimore, USA, 2005: 634-645

[5] Hu Haibo, Xu Jianliang, Dik L L. A generic framework for monitoring continuous spatial queries over moving objects//Proceedings of the International Conference on Management of Data. Baltimore, USA, 2005: 479-490

[6] Hsueh Y L, Zimmermann R, Wang Haojun et al. Partition-based lazy updates for continuous queries over moving objects//Proceedings of the International Symposium on Advances in Geographic Information Systems. Seattle, USA, 2007

[7] Mouratidis K, Yiu M L, Papadias D et al. Continuous nearest neighbor monitoring in road networks//Proceedings of the International Conference on Very Large Data Bases. Seoul, Korea, 2006: 43-54

[8] Demiryurek U, Banaei K F, Shahabi C. Efficient continuous nearest neighbor query in spatial networks using euclidean restriction//Proceedings of the International Symposium on Spatial and Temporal Database. Aalborg, Denmark, 2009: 25-43

[9] Papadias D, Zhang Jun, Mamoulis N et al. Query processing in spatial network databases//Proceedings of the International Conference on Very Large Data Bases. Berlin, Germany, 2003: 802-813

[10] Liao Wei, Wu Xiao-Ping, Yan Cheng-Hua et al. Research on multi-threading processing of concurrent multiple continuous k-nearest neighbor queries. Journal of Computer Applications, 2009, 29(7): 1861-1864(in Chinese)
(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. 计算机应用，2009，29(7)：1861-1864)

[11] Hoel E G, Samet H. Efficient processing of spatial queries in

line segment databases//Proceedings of the International Symposium on Large Spatial Databases. Zurich, Switzerland, 1991

[12] Samet H. Foundations of Ultidimensional and Metric Data Structures. San Francisco, USA: Morgan Kaufmann, 2006

[13] Brinkhoff T. A framework for generating network based moving objects. GeoInformatica, 2002, 6(2): 153-180

ZHAO Liang, born in 1982, Ph. D. candidate. His current research interests include spatio-temporal databases, moving objects databases.

CHEN Luo, born in 1973, Ph. D., associate professor. His research interests include GIS and database.

JING Ning, born in 1963, Ph. D., professor, Ph. D. supervisor. His research interests include GIS and spatial database.

LIAO Wei, born in 1980, Ph. D., lecturer. His research interests focus on spatio-temporal databases.

Background

With the rapid advances of wireless communications and electronic technologies, Location Based Services (LBS), which track the location of mobile users and provide useful information associated with locations, is becoming a new and important application area. In the mobile environment, moving objects database is used to manage the large amount of moving objects. Indexing, querying, privacy, uncertainty are the four research hotspots in the moving object management. Our work focus on efficient query processing of moving objects, for it can substantially improve the quality of the services and there arises the great need of new query processing technologies in mobile environments.

In this paper we consider the problem of continuously monitoring multiple k nearest neighbor (kNN) queries over moving objects in road networks. Existing algorithms for continuous kNN queries of moving objects in road networks adopt the incremental query mechanism, which is verified by us to be inefficient when data update frequently. However, with the advent of modern Chip Multi-Processors (CMPs), continuous kNN algorithms need to be adapted to exploit such multi-core designs. Thus, by considering the ubiquitous multi-core and multi-threading technologies, a multi-threading based framework for continuous kNN queries of moving objects is proposed. Unlike traditional algorithms, the framework periodically re-evaluate all the queries, with the help of the parallel computing capabilities of CMPs and the efficient kNN query strategy proposed in this paper, we can get far better performance than traditional algorithms in all parameter setups.

This work is supported by the National Natural Science Foundation of China under grant Nos. 40801160, 60902036 and the National High Technology Research and Development Program (863 Program) of China under grant No. 2008AA12A211 and the Post-Doctoral Science Foundation of China under grant No. 20080431384. These four projects are all about efficient managing and querying spatial and spatio-temporal databases. Our research group has a lot of experiences and achievements in studying the moving object database. Former research of us is focused on indexing of the current and future positions of moving objects, and the predicted querying of moving objects. Some of our researches are published in Chinese Journal of Computers, Journal of Software, etc. The techniques proposed in this paper can greatly help with the tracking and monitoring of moving objects in road networks, thus could improve the quality of mobile LBS. Future research of us will be focused on: continuous queries in complicated road network model, and multi-threading based query algorithms.