基于路网的移动对象动态双层索引结构

乔少杰[1)] 韩 楠[2)] 王 超[1)] 祝 峰[3)] 唐常杰[4)]

[1)](西南交通大学信息科学与技术学院 成都 610031)

[2)](西南交通大学生命科学与工程学院 成都 610031)

[3)](闽南师范大学福建省粒计算及其应用重点实验室 福建 漳州 363000)

[4)](四川大学计算机学院 成都 610065)

摘 要 为了支持对大规模不确定性移动对象当前及将来位置的查询,亟需设计更加有效和高效的索引结构.当前索引算法主要考虑索引建立和维护的效率问题或关注基于索引进行查询时的准确性,对索引建立维护以及查询时性能综合考虑的研究较少.针对已有方法的不足,提出基于路网的移动对象动态双层索引结构 DISC-tree,对静态路网信息采用 R*-tree 索引,对实时更新的移动对象运动轨迹采用结点更新代价较小的 R-tree 进行索引,设计哈希表和双向链表辅助结构对索引协同管理.成都市真实地图数据集上的实验结果表明:相比于经典的 NDTR-tree,DISC-tree 在索引建立和维护方面时间代价平均减少 39.1%,移动对象轨迹查询时间代价平均减少 24.1%;相比于 FNR-tree,DISC-tree 的范围查询准确率平均提高约 31.6%.

关键词 移动对象数据库;路网;索引;范围查询

中图法分类号 TP311 **DOI 号** 10.3724/SP.J.1016.2014.01947

A Two-Tiered Dynamic Index Structure of Moving Objects Based on Constrained Networks

QIAO Shao-Jie[1)] HAN Nan[2)] WANG Chao[1)] ZHU Feng[3)] TANG Chang-Jie[4)]

[1)](*School of Information Science and Technology, Southwest Jiaotong University, Chengdu* 610031)

[2)](*School of Life Science and Engineering, Southwest Jiaotong University, Chengdu* 610031)

[3)](*Laboratory of Granular Computing, Minnan Normal University, Zhangzhou, Fujian* 363000)

[4)](*College of Computer Science, Sichuan University, Chengdu* 610065)

Abstract In order to support efficient query for the current and future position of massive and uncertain moving objects, it urgently needs to propose effective and efficient index structures. The existing algorithms focus on the efficiency of creating and maintaining the index structure or the accuracy of index-based query processing, and the algorithms which take into consideration the performance of both index maintenance and query performance are rarely reported. Aiming to handle the drawbacks of existing algorithms, a new index called DISC-tree is proposed, which uses R*-tree to manage the static road networks and employs low-cost R-tree to index trajectories of moving objects that update in real time. Moreover, an associated structure integrates hash map and doubly linked list is applied to index dynamic trajectories. The experiments are conducted on synthetic datasets generated on the real map of Chengdu, and the results show that the DISC-tree

收稿日期:2013-06-07;最终修改稿收到日期:2014-02-25.本课题得到国家自然科学基金(61100045,61165013)、高等学校博士学科点专项科研基金(20110184120008)、中国博士后科学基金特别资助项目(201104697)、教育部人文社会科学研究青年基金(14YJCZH046)及中央高校基本科研业务费专项资金(2682013BR023)资助.**乔少杰**,男,1981 年生,博士,博士后,副教授,中国计算机学会(CCF)高级会员,主要研究领域为移动对象数据库、轨迹数据挖掘.E-mail:sjqiao@swjtu.edu.cn.**韩 楠**(通信作者),女,1984 年生,博士,工程师,主要研究方向为移动对象数据库、生物信息学.E-mail:hannan@swjtu.edu.cn.**王 超**,男,1985 年生,硕士,主要研究方向为移动对象数据库、索引.**祝 峰**,男,1962 年生,博士,教授,主要研究领域为粒计算.**唐常杰**,男,1946 年生,硕士,教授,博士生导师,主要研究领域为数据库.

can reduce the time cost of index creation and maintenance by an average gap of 39.1% compared with the classical NDTR-tree, as well as reduce the time cost of trajectory query by an average gap of 24.1%. In addition, compared to the FNR-tree, the accuracy of range query is averagely improved by about 31.6%.

Keywords moving objects databases; constrained networks; index; range query

1 引 言

随着无线通信、传感器网络和定位技术的高速发展,人们对持续移动物体所处空间位置的跟踪能力不断加强,推动相关应用领域的研究,如智能交通控制、智能导航和旅游路线推荐等不断深入.此类应用系统往往基于移动对象数据库(Moving Objects Databases,MODs).作为移动计算技术的重要分支以及位置服务类应用的底层支撑技术,移动对象数据库技术得到了广泛的关注和研究[1].相比于时空数据库,移动对象数据库不仅支持查询当前时间有效的空间对象,而且可以对随时间而发生位置变化的移动对象的历史轨迹进行管理.对移动对象的位置信息提供高效的索引和查询,在地理信息系统、精确定位技术方面具有很高的科学意义.

现实生活中人们真实的运动场景往往被限制在某一固定区域内,即道路交通网络(简称路网),因此研究路网中移动对象索引结构具有现实意义.路网是一个动态复杂的系统,系统中存在规模巨大、种类繁多的移动对象,而且构成网络的街道错综复杂,并不都是汇集成十字型路口,某些道路往往是三分支或者五分支的.利用 R-tree 或者 R*-tree 对移动对象进行索引,往往忽略了静态路网自身的结构特点,得不到令人满意的查询结果.

基于路网的移动对象时空索引和范围查询技术目前是一个充满挑战性的课题.衡量索引性能的主要指标包括:索引建立和维护的效率、轨迹查询时间、时空范围查询的效率和准确性.面临的主要困难是:简单索引结构能够提高索引维护的效率,减少时空范围查询访问结点的 I/O 代价,但是会降低查询的准确性;复杂索引有助于提高查询准确率,但增加了维护的时间开销.针对当前路网索引结构的不足,本文提出一种新型索引结构 DISC-tree(a two-tiered Dynamic Index Structure of moving objects based on Constrained networks).相比于具有代表意义的 FNR-tree[2],克服了 FNR-tree 在时空范围查询精度上的不足;相比于 NDTR-tree[1],索引更新时间开销和 I/O 代价更小,克服 NDTR-tree 对移动对象轨迹查询时间性能上的不足.

2 相关工作

索引技术是移动对象数据库领域的关键研究内容,移动对象的位置查询、基于位置的服务、移动对象位置预测都依赖于移动对象索引[3].移动对象的静态索引已经有比较成熟的技术[4],如 R-tree、R+-tree、R*-tree、KD-tree 等.移动对象索引按照空间范围分为两类:不受空间约束和受限路网中位置索引.不受空间约束是指移动对象的运动不依赖于具体的道路,其可以在空间任意范围内运动.这类索引技术在国内外已经取得一系列研究成果[5].

根据移动对象当前位置建立索引的技术瓶颈是提高索引更新的效率,为了解决这一问题,吴岑等人[6]提出了一种基于 GRID 文件的移动对象索引方法,其优势在于:在结点合并和分裂过程中,分裂位置选择具有随机性,因此索引更新的代价较小.但相比基于 R-tree 的索引,其查询时间提高不大.为了进一步减少移动对象索引更新的代价,Silva 等人[7]提出了一种基于内存的 R-tree 索引结构 RUM-tree,在进行结点更新操作时仅需要一次结点插入操作.Saltenis 等人[8]提出 TPR-tree 对移动对象未来位置进行索引,结点分裂时不仅考虑对象的位置而且考虑包含速度和方向的运动矢量.Tao 等人[9]对 TPR-tree 中结点插入和更新操作进行改进,提出了 TPR*-tree,借助改进的压缩技术对结点进行删除.Chen 等人[10]提出一种动态的称为自调整单元(Adaptive Unit)的结构来扩展 R-tree,根据移动对象的运动特征(位置、速度和方向)对其进行分组,达到预测整组对象运动轨迹的目的.

相比于不受空间约束移动对象的索引技术,基于受限路网的索引技术尚处于起步阶段,国内外成果较少,具有代表意义的工作如下:

针对 Pfoser 等人[11]提出的受限公路网络中移

动对象索引方法的不足，Frentzos 等人[2]提出双层索引结构 FNR-tree，由一个 2D R-tree 和一个 1D R-tree 森林构成，2D R-tree 用于管理整个路网中的边. FNR-tree 存在以下两方面不足[5]：(1) 上下两层索引的粒度相同，均针对于路网中的边，上层叶子结点记录与下层 R-tree 的对应关系为 1∶1，导致下层 R-tree 的规模庞大；(2) 由于仅存储对象进入和离开边的数据，会遗漏掉在边上运动并停止的移动对象位置信息，导致窗口范围查询结果的准确性较低.

MON-tree[12]是对 FNR-tree 进行改进而提出的索引结构，对道路网络建立索引不再基于直线边，而是基于折线道路，减小了对象跨越不同边的更新代价. 不足在于：对路网建立索引是基于道路信息，会造成结点 MBR(Minimum Boundary Rectangle)的重合. 查询效率随结点 MBR 重合程度增加而降低.

NDTR-tree[1]的上层结构对路网中原子路段建立索引，下层 R-tree 用于索引在原子路段上运动的移动对象轨迹片段. 这一结构的不足在于[5]：(1) 索引维护代价较高，每当发生位置更新请求时，需要进行两次结点插入和一次删除操作. 此外，每次删除操作均需遍历整个 R-tree 森林. (2) 不支持对移动对象轨迹的高效查询，当查询指定的运动轨迹时，需遍历整个下层 R-tree 森林. (3) 实际路网中道路的相交部分较多，采用 R-tree 组织索引，结点插入时会产生大量索引空间的重叠，不利于查询.

近期移动对象索引研究仍然得到学术界的广泛关注. 针对移动对象速度分布的偏离特性，Nguyen 等人[13]提出了新型速度划分技术，借助 PCA 和 k-means 聚类技术识别 DVA(Dominant Velocity Axe)，然后基于 DVA 建立索引，提高了移动对象查询的效率. 为了查询移动对象过去、当前及未来位置，Zhu 等人[14]利用哈希表自下而上对移动对象进行位置更新，对于路网中道路采用邻接矩阵存储每条道路相邻道路信息. 但是这一结构没有考虑移动对象频繁更新的索引维护的代价. Appathurai 等人[15]利用线性函数刻画移动对象的位置信息，设计了新型结点更新和合并算法，可以在某一时刻处理更多对象的位置更新信息，达到提高结点更新效率的目的. 但是，对象速度并不总是遵循线性分布，不能应用于真实的路网环境中.

3　受限路网中移动对象运动模型

本节将介绍 DISC-tree 中采用的数学模型和主要概念，并给出形式化定义，如下所示.

定义 1.　路网. $G=(E,N)$，E 是路网中所有封闭路段的集合，N 是路段交点的集合.

定义 2.　道路交点. $n=(nid,x,y)$，nid 是道路交点的标识，x 和 y 为交点在二维空间平面上的坐标.

定义 3.　封闭路段. $e=(eid,rid,n_s,n_e,len,w,pos_s)$，$e$ 表示路网中的一条封闭路段，“封闭”的含义指从路段起点到终点的范围内不含有任何道路的交点，eid 是该封闭路段的标识，rid 为该封闭路段所在道路的标识，n_s 和 n_e 表示封闭路段的起点和终点，len 为此路段的长度，$w=e.len/r.len$(r 为该封闭路段所属的道路)，表示该封闭路段在其所属道路中的比重，$pos_s\in[0,1]$为起点在街道中的位置.

定义 4.　道路. $r=(rid,S,l)$，r 表示路网中的一条道路，道路通常包含一条或者多条封闭路段，rid 为道路的标识，S 为该道路所包含封闭路段的集合，l 表示道路的长度.

本文通过封闭路段、道路、道路交点来描述复杂交通网络和交通网络中的道路联接关系.

定义 5.　移动对象位置更新信息. $m=(mid,type,t,v,x,y)$，mid 表示移动对象标识，$type$ 为移动对象类型，t 为采集运行矢量的时间点，v 表示移动对象的速度，x 和 y 为对象所处二维位置信息.

定义 6.　移动对象运行矢量. $mv=(mid,type,t,v,rid,pos)$，$mid$ 为移动对象的标识，$type$ 表示移动对象类型，t 为采集该运行矢量的时间，v 表示该移动对象在 t 时刻的速度，rid 为该移动对象所处道路的标识，$pos\in[0,1]$为对象所处道路的相对位置.

已知查询窗口与封闭路段的交点 c 到封闭路段起点的距离为 k，封闭路段长度为 e，那么运行矢量中 pos 取值为移动对象当前位置距离道路起点的长度除以整条道路的长度，计算方法如式(1)所示：

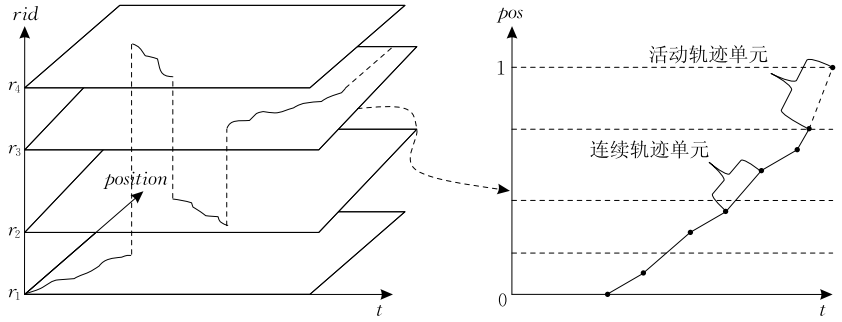
$$pos = pos_s + w \times (k/e) \tag{1}$$

其中，pos_s 表示对象初始位置，w 为封闭路段在其所属道路中的比重，计算方法见定义 3.

定义 7.　移动对象轨迹单元. 两个运行矢量之间的线段称为轨迹单元 U. 连续轨迹单元表示为 $U(mv_s,mv_e)$，mv_s 和 mv_e 表示运动矢量的起点和终点. 活动运行矢量所形成的轨迹单元是对应点作为端点发出的一条射线，记为活动轨迹单元 $U(mv_n)$.

移动对象的轨迹利用 $rid\times pos\times t$ 三维坐标系统描述，如图 1(a)所示. 随时间的变化移动对象在

道路中的位置不断变化，对于 rid 坐标的每一个切面形成一个 $pos \times t$（位置和时间）的坐标平面，表示移动对象在具体道路上的时空轨迹. 移动对象可以跨越不同的道路，其轨迹可以表示为一条 $rid \times pos \times t$ 三维空间中不断变化的曲线. 参考定义 7，图 1(b)表示移动对象在具体道路上的轨迹，其由若干个轨迹单元构成，虚线部分为活动轨迹单元.



(a) 移动对象时空轨迹　　(b) 道路r_3上的时空轨迹

图 1　路网下移动对象三维时空轨迹及二维投影[5]

定义 8.　链表轨迹单元. 用 $LU=(U, uid)$ 表示，U 表示为 $U(mv_s, mv_e)$ 或者 $U(mv_n)$，uid 表示轨迹单元在下层 R-tree 中的位置标识.

链表轨迹单元保存在移动对象所对应的双向链表中，目的是加速下层 R-tree 索引的建立和维护.

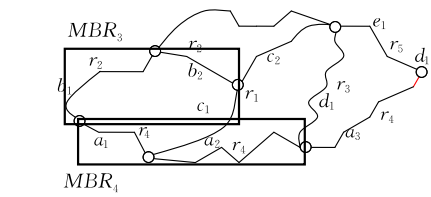
4　DISC-tree 的数据结构

4.1　DISC-tree 双层索引结构

DISC-tree 分为上下两层，上层是基于 xy 平面的 R*-tree 索引，用于索引路网中的封闭路段，其叶子结点的数据结构为〈M, id, e〉. M 表示包含该封闭路段的最小边界矩形 MBR，id 为该叶子结点在索引中的标识，e 为表示封闭路段. 中间结点用〈M, id', p〉表示，M 为包含下层结点的最小边界矩形，id' 为结点在索引中的标识，p 为指向下层结点的指针. 其中，一个 MBR 包含一条封闭路段. 采用 R*-tree 的主要原因在于，相比于 R-tree，在索引建立时，R* 树不仅考虑了索引空间的“面积”，而且考虑了空间的重叠部分. 此外，本文采用“强制重新插入”的方法使树的结构得到优化.

图 2 为交通网络在 DISC-tree 上层索引中的对应关系，道路 r_4 包含 3 个封闭路段($r_4.a_1$，$r_4.a_2$，$r_4.a_3$)，于是形成 3 条记录存储于包含各封闭路段 MBR 信息的叶子结点中. 图 2 上半部分矩形框示例给出 MBR_3 和 MBR_4 的范围.

DISC-tree 下层结构是一组与每条道路对应的 R-tree 索引，基于 $pos \times t$ 平面，用于索引移动对象轨迹单元，根据位置更新信息动态地进行维护. 下层



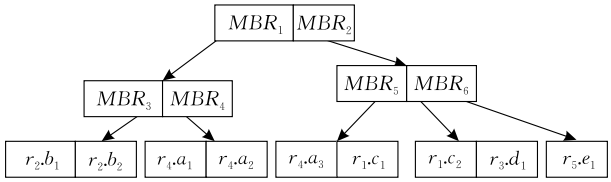
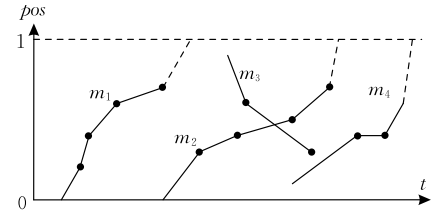


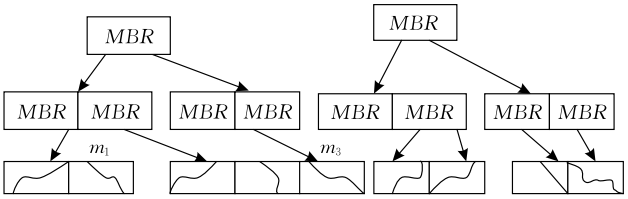
图 2　DISC-tree 上层索引结构

叶子结点的数据结构为〈M, id, mv_s, mv_e, o〉，M 表示包含该轨迹的最小边界矩形，id 为叶子结点的标识，mv_s 和 mv_e 分别表示轨迹单元起始和结束运行矢量，如果为活动轨迹单元，则 mv_e 为空. o 表示移动对象信息，由对象标识 mid 和对象类型 $type$ 组成. 中间结点数据结构为〈$M_{pos \times t}, id', p$〉，$M_{pos \times t}$ 是包含下层结点的最小边界矩形，id' 为中间结点标识，p 为指向下层结点的指针. DISC-tree 下层结构如图 3 所示，下层 R-tree 森林管理移动对象基于 $pos \times t$ 平面的轨迹片段. 例如移动对象 m_1 仅在道路 r 上运动，于是其所有轨迹单元被道路 r 对应的 R-tree 管理(参见图 3(b)). 图 3(a)中虚线部分表示活动轨迹单元，如果对象从一条道路运动到另一条道路，那么它的轨迹单元将被两条道路的 R-tree 管理.

DISC-tree 上下两层 R-tree 的对应关系为 1∶n，1 代表上层索引仅由一个 R*-tree 构成，n 为路网中道路 r 的数目，下层索引含有 n 个 R-tree. 利用上层索引叶子结点中的信息可以搜索到下层索引中的某



(a) 道路r上包含移动对象的轨迹片段



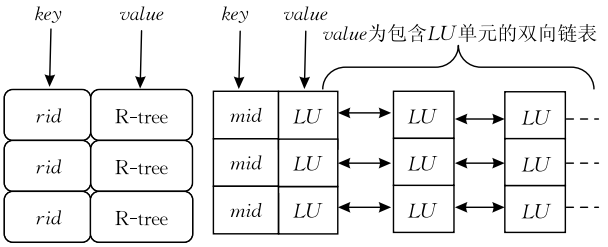
(b) 道路r的R-tree索引

图 3　DISC-tree 下层索引结构

个 R-tree. 例如，已知交大路包含两个封闭路段，则上层 R*-tree 叶子结点中有两条记录指向下层索引中属于交大路的 R-tree. 查询某个时间段 Δt 内交大路上所有移动对象的轨迹时，首先从上层 R*-tree 自顶向下查询，找到叶子结点中交大路的所有封闭路段，然后定位到下层索引中交大路的 R-tree，根据查询条件搜索 Δt 时间内的下层索引.

4.2　DISC-tree 的辅助结构

为了提高索引维护效率，本文采用如下 2 个哈希表和 1 个双向链表作为 DISC-tree 的辅助结构[5].



(a) 索引的哈希表　　(b) 轨迹单元哈希表及双向链表

图 4　DISC-tree 应用的辅助结构

图 4(a)中的哈希表 RoadHashTable 对 DISC-tree 下层基于道路的 R-tree 森林进行管理，其中 *key* 的值为道路标识 *rid*，*value* 为道路对应的 R-tree 索引. 图 4(b)中的哈希表 TrajectoryHashTable 用于管理移动对象的轨迹单元，*key* 取值为移动对象标识 *mid*，*value* 为一个双向链表，用 *linkedlist* 表示，包含具体移动对象的所有轨迹单元 *LU*，其按时间先后顺序加入到链表中，根据移动对象提供的位置信息不断进行更新. 当收到一条位置更新信息时，首先形成轨迹单元 *U*，将其插入到 R-tree 中，并返回在 R-tree 中的位置标识 *uid*，形成链表轨迹单元 $LU(U, uid)$ 插入到 *linkedlist* 中，在维护索引的同时对 *linkedlist* 和 TrajectoryHashTable 进行维护.

应用本文所提出的辅助结构具有如下优势：

(1) 因为移动对象轨迹具有不确定性，遍布在整个路网中，导致其轨迹可能被相当多条道路的 R-tree 索引，如果采用不恰当的数据结构，对移动对象的轨迹查询将是极其低效的. 例如，移动对象的运动轨迹跨越了多条道路，为了查询其历史运动轨迹，需要自上向下遍历整个 R-tree 森林. 而采用哈希表和双向链表构成的辅助结构后，仅需将其加载到缓存中进行维护，查找移动对象轨迹不需要遍历 R-tree 森林，极大地提高了查询的效率.

(2) 在无哈希冲突的情况下，由 *LU* 构成的哈希双向链表插入、查找、删除 n 个结点的时间复杂度均为 $O(n)$，构建哈希双向链表的时间复杂度为 $O(n)$，n 表示轨迹单元的数量. 辅助结构的建立和维护时间代价较低.

4.3　DISC-tree 与 NDTR-tree 对比分析

NDTR-tree 的轨迹查询效率不高，查找移动对象的历史轨迹需遍历一个甚至多个 R-tree 的所有结点，找到移动对象的轨迹单元. DISC-tree 采用哈希表加双向链表的数据结构，提高轨迹查找和删除的效率. 双向链表中包含移动对象的轨迹单元，当需要访问对象在某些时刻的轨迹信息时，仅需访问哈希表，取出移动对象对应的链表进行查询. 此外，双向链表的好处在于可以快速访问某一时刻的前一时刻和后一时刻的轨迹单元，为查询提供便利.

DISC-tree 结合 R-tree 和 R*-tree 索引，对静态路网建立 R*-tree 索引，对动态轨迹单元建立 R-tree 索引. 对静态路网，采用更新操作中结点分裂次数较少的 R*-tree，查询对象将更加高效和准确. R-tree 索引建立和维护的计算代价较小，适用于管理频繁更新的移动对象轨迹单元. 而 NDTR-tree 中对变化较小的路网和更新频率较高的移动对象轨迹单元均采用 R-tree 索引.

在 NDTR-tree 中为了寻找移动对象的前一个运动轨迹单元，需要遍历多条道路所对应的多个 R-tree 索引. DISC-tree 对此做出改进[5]：将每个对象上一次更新后的最后一个轨迹单元在 R-tree 中的索引标识 *uid* 存储到与其对应的链表结点 *LU* 中，需要对其操作时可以直接定位到 R-tree 的某个叶子结点，进而找到轨迹单元，极大地提高了索引维护的效率，第 6.3.2 节的实验证明了这一点.

5 DISC-tree 中应用的算法

5.1 索引建立和维护算法

算法 1. DISC-tree 的建立和维护算法.

输入:移动对象位置更新信息 m.

输出:更新后的 DISC-tree.

1. FOR (每条更新信息 $m=(mid,type,t,v,x,y)$)
2. 遍历上层以封闭路段为索引单位的 R*-tree,找到移动对象所在的封闭路段 e;
3. $mv\leftarrow(mid, type, t, v, rid, pos)$;
4. $R\leftarrow searchRoadTable(rid)$;
5. $linkedlist\leftarrow searchTrajectoryTable(mid)$;
6. $t\leftarrow moveToTail(linkedlist)$;
7. IF ($t==null$) THEN
8. $LU\leftarrow insert(R, U(mv))$;
9. $insertLinkedList(LU)$;
10. ELSE
11. $U(mv_n)\leftarrow t.uid$;
12. $delete(U(mv_n))$;
13. $LU1\leftarrow insert(R, U(mv_s, mv_e))$;
14. $LU2\leftarrow insert(R, U(mv_n))$;
15. $delete(linkedlist)$;
16. $insertTail(LU1)$;
17. $insertTail(LU2)$;
18. END IF
19. END FOR

DISC-tree 首先利用真实路网数据建立上层静态 R*-tree 索引,“静态”是指路网信息的更新几率远小于移动对象轨迹单元的更新概率,大多数情况下仅需一次构建,不需要频繁更新.然后,根据移动对象提交的位置更新信息建立并维护下层动态索引.“动态”是指整个索引结构不断更新.本文重点关注下层动态索引的建立和维护,DISC-tree 建立和维护过程如算法 1 所示,主要步骤为:

(1)根据移动对象位置更新信息 m 查找基于 xy 平面的上层索引,进而找到移动对象所处的封闭路段 e,取出其相关信息(第 1～2 行);

(2)求取对象在 e 所属道路 r 上的位置 pos,形成运行矢量 $mv=(mid,type,t,v,rid,pos)$(第 3 行);

(3)通过 rid 访问辅助结构 RoadHashTable,取得道路所对应的 R-tree(第 4 行);

(4)根据 mid 访问哈希表 TrajectoryHashTable,获得轨迹单元链表,访问链表尾结点 t(第 5～6 行);

(5)若 t 为空,表示初次收到该对象位置信息,则插入活动轨迹单元 $U(mv_n)$ 到 R-tree 中,并记录它在 R-tree 中的位置 uid,通过 (U,uid) 生成链表轨迹单元 LU,并插入到的链尾(第 7～9 行);

(6)若 t 不为空,根据 t 中的 uid 信息找到并删除 R-tree 中的活动轨迹单元 $U(mv_n)$.插入连续轨迹单元 $U(mv_s,mv_e)$,插入活动轨迹单元 $U(mv_n)$,$LU1$,$LU2$ 为生成的链表轨迹单元(第 10～14 行),最后在链表中删除 t(第 15 行);

(7)将 $LU1$ 和 $LU2$ 相继添加到链表尾部,进而建立索引的同时并维护链表信息(第 16～17 行).

5.2 移动对象时空范围查询

时空范围查询是衡量移动对象索引结构好坏的指标.本文采用如下所示的时空范围查询算法[5].

算法 2. 移动对象时空范围查询算法.

输入:查询区域 $(\Delta x,\Delta y,\Delta t)$,索引哈希表 RoadHashTabel.

输出:满足查询条件的移动对象标识集合 S.

1. 根据 $(\Delta x,\Delta y)$ 查询上层 R*-tree 求得查询窗口与封闭路段的多个交点 $(e_i,x_i,y_i)_{i=1}^n$;
2. $(rid,pos)\leftarrow calculatePos(e,x,y)$;//通过 e 中的属性及其与道路交点的 x, y 坐标,求取交点的位置和道路标识;
3. 对每一个交点重复步骤 2 的计算,得到一系列偶对的集合 $Q=(rid_i,period_i)_{i=1}^n$;
4. FOR EACH (q in Q)
5. $R\leftarrow searchRoadTable(rid)$;
6. 根据 $period_i\times\Delta t$ 查询 R 中叶子结点记录 U;
7. IF ($U\bigcap(period_i\times\Delta t)\neq null$)
8. $I\leftarrow U$ 对应的 mid;
9. END IF
10. END FOR

以成都路网中指定范围查询为例,具体过程:

(1)根据查询窗口 $[(x_1,y_1),(x_2,y_2)]$,搜索上层 R*-tree,遍历到叶子结点,借助叶子结点中封闭路段信息 e,求得查询窗口与封闭路段的交点.对于一个封闭路段,查询窗口与之相交的个数可能不唯一.例如,查询窗口与万和路的交点有 2 个,形成两条查询信息.从索引文件中读出存储该封闭路段信息的字符数组,反序列化为封闭路段 e 的信息,交点信息用 (e,x,y) 表示(第 1 行).

(2)计算出交点在街道中的位置 pos(第 2 行),需要用到定义 3 中的属性:w,len,pos_s.

(3)查询区域与一个道路可能有多个交点,对应于一个偶对 $(rid,period)$(第 3 行),其中 $period$ 包含一个或者多个 pos 信息.查询区域与多条道路相交,则对应于一组偶对,在 rid 对应的下层 R-tree 中查询与 $(period,\Delta t)$ 相交的轨迹单元,最终输出轨

迹单元对应移动对象的标识(第 4～10 行).

6　实验及性能分析

6.1　实验环境及数据集描述

实验采用的数据集来源于成都市真实矢量地图,包含成都二环内的 232 条主干道,将地图数据进行转换,得到如表 1 所示的数据格式.

表 1　轨迹数据格式

对象标识	速度	x 坐标	y 坐标	移动对象类型	道路标识	所属封闭路段标识
7	20	468	264	5	123	362

在实验中,通过加载 Edge(路网边集合)和 Node(路网结点集合)两个文件后生成成都市二环内的真实路网. Edge 文件包含每个封闭路段边的信息,表示为(i,n_s,n_e,eid,rid),i 为自增字段,n_s和 n_e 为封闭路段的起点和终点标识,eid 为封闭路段标识,rid 为道路标识. *Node* 文件表示为(nid,x,y),nid 表示道路交点标识,x 和 y 为二维平面下的横纵坐标. 本文通过对文献[16]提出的移动对象轨迹生成器改进后设计实现了一个新的轨迹生成系统,改进之处在于:生成轨迹数据时考虑道路位置信息,不再基于边生成轨迹片段,而是基于道路生成轨迹,使生成轨迹的数据结构与本文所提算法相符.

本文中所有算法利用 Java 程序设计语言实现. 实验硬件平台为 AMD Athlon 5000＋,2.6 GHz CPU,2 GB 内存,操作系统平台为 Windows XP. 表 2 给出轨迹生成器的参数设置.

表 2　轨迹生成器参数设置

参数	值
地图宽度	23514(像素)
地图长度	25672(像素)
移动对象数量	500～10000
时间间隔	20
移动对象种类	5
道路数量	232

实验中主要对比 DISC-tree、FNR-tree 和 NDTR-tree 的性能差异,其中 FNR-tree 是基于路网的移动对象索引方法中性能较好的一种,能支持高效的 kNN 查询和时空范围查询;NDTR-tree 是移动对象时空范围查询性能较好的索引,因此选择这两种索引进行对比实验. NDTR-tree 与 DISC-tree 的索引基于道路构建,FNR-tree 的轨迹索引基于封闭路段. 为了保证算法可比性,FNR-tree 进行轨迹查询时将不会用到表 1 中移动对象所属道路标识.

6.2　PathFinder 轨迹查询系统简介

为了便于比较上述 3 种索引结构,本文开发了基于路网的移动对象轨迹查询系统 PathFinder[5],包含 3 个主要功能模块:(1)轨迹生成模块,根据静态路网生成均匀分布在整个路网中移动对象的动态运动轨迹;(2)索引构建,构建基于路网的索引结构;(3)对象查询,提供时空范围等查询功能.

图 5 界面左侧为利用轨迹生成器生成的成都二环内路网数据,圆点表示移动对象,利用不同颜色来区分移动对象的类型,例如:红色代表速度较快的移动对象,如小型车. 以移动对象时空范围查询模块为例,首先,用户需要确定查询的窗口范围;然后,选择查询的时间片段;最后,点击“开始查询”进行时空范围查询. 查询返回结果信息包括:移动对象标识、速度、移动对象所处的街道等,结果在系统界面中输出. 本文应用该系统比较 DISC-tree、NDTR-tree、FNR-tree 进行索引构建和时空范围查询的性能.

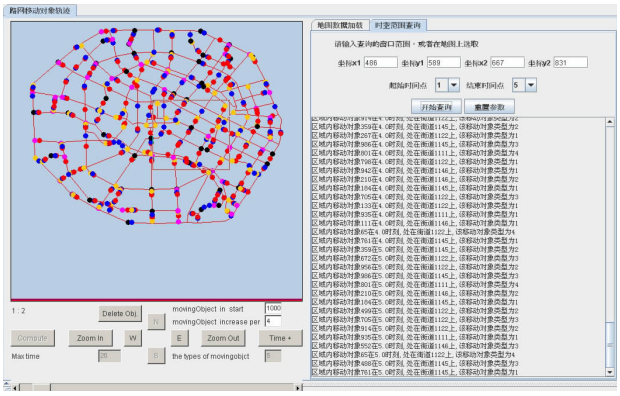


图 5　移动对象范围查询结果示例

6.3　索引建立和维护代价比较

6.3.1　索引建立及维护代价理论分析

式(2)～式(4)给出了 FNR-tree、DISC-tree、NDTR-tree 在索引建立和维护过程中结点的访问次数. 其中,T_i,T_i'和 T_i^* 表示 3 种索引中第 i 个移动对象的轨迹数目;N_{insert},N'_{insert}和 N^*_{insert}分别表示 3 种索引进行一次插入操作结点访问的次数;N_{delete},N'_{delete},N^*_{delete}表示 3 种索引进行一次删除操作结点访问的次数;n 表示移动对象的数量,m_j表示树中第 j 层包含结点的数目,h 为树的高度.

$$NodeAccess(FNR-tree)=\sum_{i=1}^{n}T_i\times N_{insert} \quad (2)$$

$$NodeAccess(DISC-tree)=\sum_{i=1}^{n}T'_i\times (2\times N'_{insert}+N'_{delete}) \quad (3)$$

$$NodeAccess(NDTR-tree)=\sum_{i=1}^{n}T_i^*\times\left(\sum_{j=1}^{h}m_j+2\times N_{insert}^*+N_{delete}^*\right)\quad(4)$$

随着移动对象数量的增加，DISC-tree 与 NDTR-tree 索引建立和维护时结点访问的次数远高于 FNR-tree，原因在于：

(1) 只有移动对象离开封闭路段时 FNR-tree 才触发一次轨迹更新事件，当移动对象运动在封闭路段上时，其活动轨迹单元没有保存在 FNR-tree 中. 相反，对于 DISC-tree 和 NDTR-tree，不管移动对象是否发生路段转移事件，均需记录对象的全部轨迹信息. 对象在 FNR-tree 的轨迹数目要少于其在 DISC-tree 和 NDTR-tree 下的轨迹数目，即式(2)中的 T_i 值小于式(3)和式(4)中的 T_i' 和 T_i^* 值.

(2) 在 DISC-tree 与 NDTR-tree 中，下层索引中的每个树均基于包含多个封闭路段的道路 r 构建，而 FNR-tree 的下层索引基于封闭路段 e 构建，其空间单位较小. 由于 r 与 e 的关系是 1∶n(n 值由 r 中包含 e 的个数确定)，DISC-tree 与 NDTR-tree 的下层索引中每个树的规模远大于 FNR-tree，因此在 FNR-tree 中进行一次插入操作需要访问的结点数目要小于同样的操作在 DISC-tree 和 NDTR-tree 中所访问的结点次数，式(2)中的 N_{insert} 小于式(3)和式(4)中的 N_{insert}' 和 N_{insert}^*.

(3) DISC-tree 和 NDTR-tree 每次更新索引时都需要删除之前的活动轨迹单元，并插入新的活动轨迹单元，FNR-tree 无需进行上述操作.

6.3.2 索引建立及维护代价实验

本节统计不同移动对象数量下，FNR-tree，NDTR-tree，DISC-tree 索引建立和维护的 I/O 及时间开销. 3 种索引基于 6.1 节介绍的成都二环路路网生成相同的轨迹数据集，避免由于轨迹随机性造成实验环境的不同，结果如图 6 和图 7 所示.

如图 6 所示，随着移动对象数目增多，DISC-tree 索引建立及维护的性能明显优于 NDTR-tree，原因在于：NDTR-tree 每次接收到位置更新请求对下层 R-tree 进行更新时，需要遍历下层 R-tree 找到移动对象的活动轨迹单元并进行删除，式(4)中 $\sum_{j=1}^{h}m_j$ 表示每次遍历过程中 NDTR-tree 访问结点的数目. 移动对象数目越多，下层 R-tree 规模越庞大，将产生巨大的 I/O 开销. 而 DISC-tree 采用了哈希表和双向链表辅助结构，寻找移动对象活动轨迹单元仅需访问链表尾结点，查找其在下层 R-tree 中的位置，删除活动轨迹单元. 此操作不需要遍历 R-tree，极大地减少了 I/O 开销. FNR-tree 的 I/O 要远低于其他两种索引结构，图 6 中几乎显示不出来. 当对象数量达到 5000 时，NDTR-tree 的 I/O 是 FNR-tree 的 262.7 倍，是 DISC-tree 的 5.3 倍.

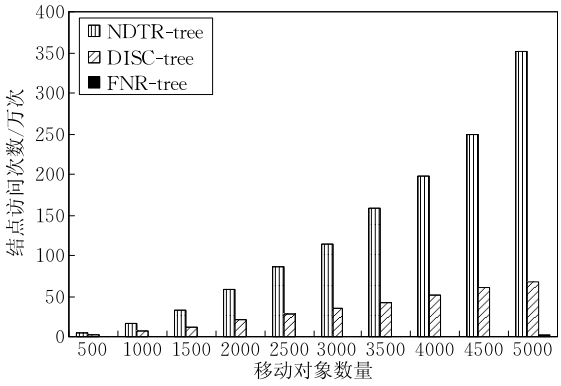


图 6 索引建立和维护时 I/O 开销比较

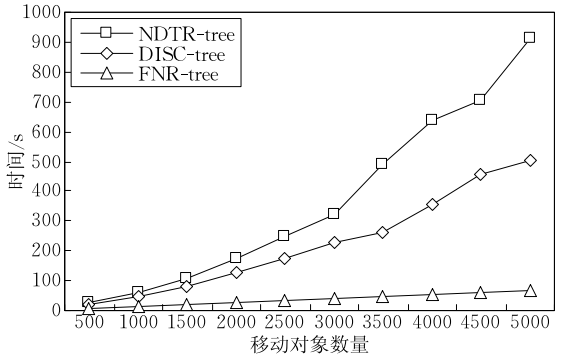


图 7 索引建立和维护时间比较

如图 7 所示，当移动对象数量规模较小，为 500 左右时，DISC-tree 索引建立时间略低于 NDTR-tree. 因为 DISC-tree 采用 R*-tree 索引静态路网，相比于 NDTR-tree 采用 R-tree 索引结构，虽然前者的一次建立开销略大于后者，但是采用辅助结构节省删除轨迹单元的时间开销. 当移动对象数量增加到 1500 后，DISC-tree 的辅助结构作用明显，极大地节省了删除活动轨迹单元的时间代价. 相比于 NDTR-tree，DISC-tree 在索引建立和维护方面时间代价平均减少 39.1%. FNR-tree 仍然具有时间性能上的优势，移动对象数量从 500 增加到 5000 的过程中，DISC-tree 的平均时间约为 FNR-tree 的 5.14 倍，NDTR-tree 的平均时间约为 FNR-tree 的 9.78 倍. FNR-tree 索引建立和维护的时间性能优势不如 I/O 性能优势大的原因在于：FNR-tree 需要维护的动态索引树基于封闭路段构建，而 NDTR-tree 和

DISC-tree 的动态索引树基于道路构建，一条道路包含多个封闭路段的实际情况导致 FNR-tree 下层索引中虽然每个树所占空间较小，但是整个森林规模更为庞大，增加了时间开销.

6.4　移动对象轨迹查询性能比较

轨迹查询是指查询在给定道路上所有出现的移动对象的运动轨迹单元. 相比于 NDTR-tree 和 DISC-tree，如果移动对象一直没有离开某条封闭路段，FNR-tree 不记录其轨迹信息，其所包含的轨迹数量少而且不准确，因此实验中比较 NDTR-tree 与 DISC-tree 的轨迹查询性能. 为了获取移动对象的运动轨迹，NDTR-tree 需要访问下层所有 R-tree(n 条道路对应 n 个 R-tree)，将包含移动对象轨迹单元的所有记录取出，进而得到完整轨迹. 而 DISC-tree 首先根据移动对象标识访问哈希表，取出移动轨迹单元链表，然后访问双向链表，逐一取出链表单元，按时间顺序将轨迹片段连接起来形成完整轨迹.

如图 8 所示，由于 DISC-tree 无需访问下层 R-tree 森林，其轨迹查询时间一直维持在一定范围内，当对象数量大于 2500(横坐标为 5)时，近似呈线性增长；而 NDTR-tree 的查询时间随对象的增加及 R-tree 结构增大，变得越来越长. 相比 NDTR-tree，DISC-tree 的平均查询效率提高约 24.1%.

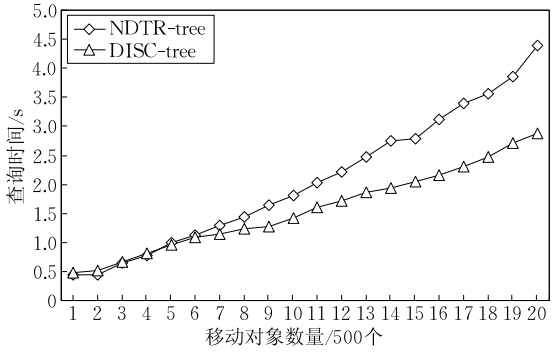


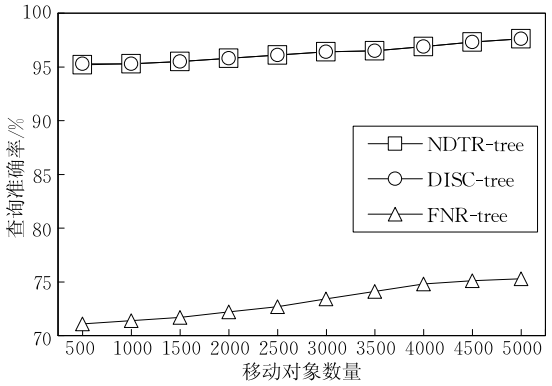
图 8　轨迹查询时间比较

6.5　时空范围查询性能分析

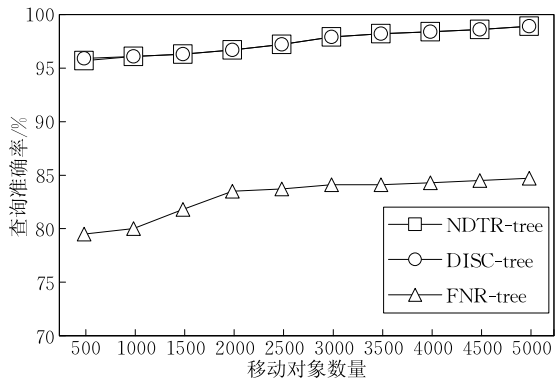
为了比较 FNR-tree、NDTR-tree、DISC-tree 时空范围查询的准确性和时间效率，实验分别以查询窗口大小占地图的 10%、20%、30%、40%范围进行查询，比较查询准确率和时间两个性能指标.

6.5.1　时空范围查询准确性分析

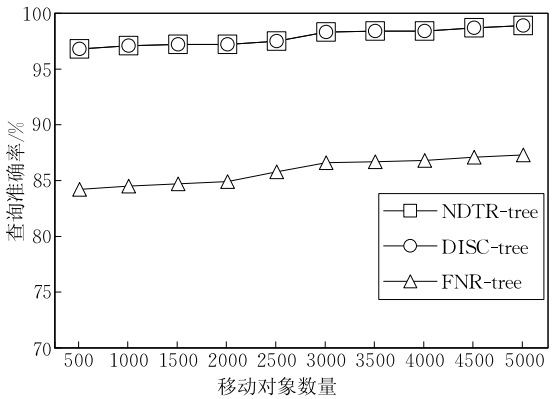
3 种索引结构在不同查询窗口下查询结果的准确率如图 9 所示. 虽然 NDTR-tree 和 DISC-tree 采用不同的索引结构，但是均基于道路构建双层索引，因此这两种索引得到的查询结果一致，即两种索引的查询准确率曲线重合. FNR-tree 基于封闭路段构



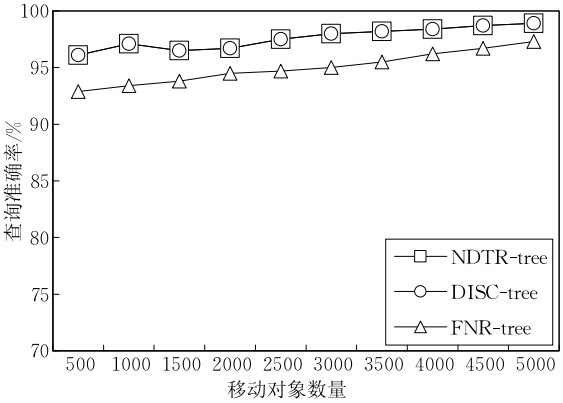
(a) 查询窗口大小为10%



(b) 查询窗口大小为20%



(c) 查询窗口大小为30%



(d) 查询窗口大小为40%

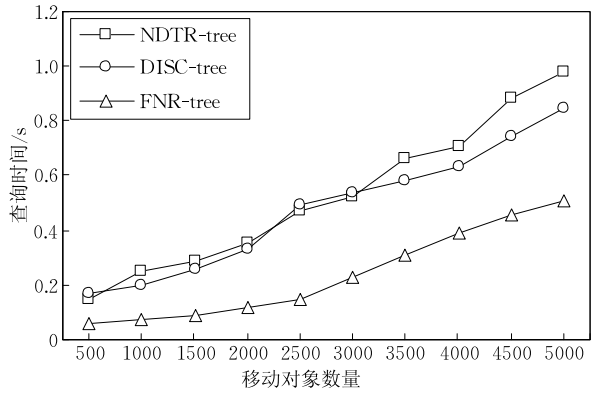
图 9　时空范围查询准确性比较

建索引,查询返回的结果要少于上述两种索引,导致预测不够准确.当查询窗口为地图的10%时,FNR-tree的查询准确率平均为73.2%,而NDTR-tree和FNR-tree的查询准确率较高,维持在95%以上.当查询窗口增大到20%时,NDTR-tree和DISC-tree保持平均98%以上的准确率,FNR-tree的准确率为83%.当查询窗口增大到40%时,3种索引查询准确率相差不大.

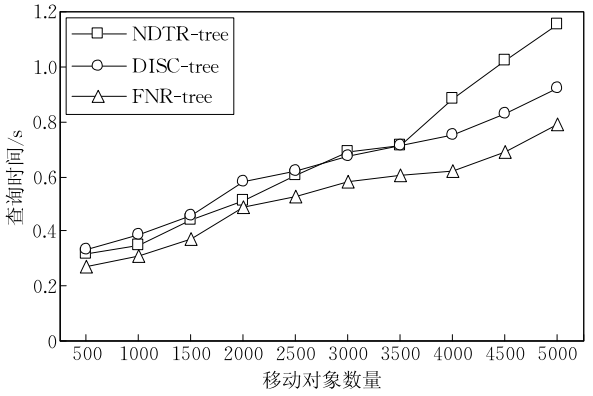
在小范围窗口下,FNR-tree查询准确率不高的原因在于[5]:其仅保存移动对象进入和离开一个封闭路段时的轨迹信息.假设移动对象 o 运动到道路 r 的子路段 $r.d_1$ 则到达目的地,并没有离开该路段,于是 o 在 $r.d_1$ 上的运动轨迹不会保存在FNR-tree中,当查询窗口较小时包含的路段较少,出现漏查的情况.随着查询窗口的增大,其包含的封闭路段数目将会增多,仍以道路 r 为例子,查询窗口增大后,可能包含 r 的多个封闭路段,在 $r.d_1$ 中查不到移动对象 o,但在其他路段上可能会查找到 o.实验发现:当查询窗口小于10%时,DISC-tree相比于FNR-tree在查询准确率上平均提高约31.6%.

6.5.2 时空范围查询时间代价分析

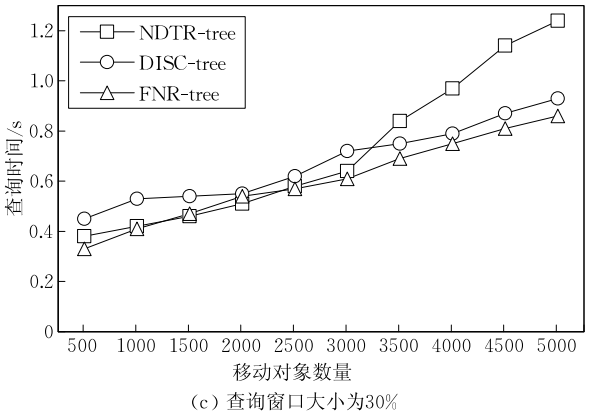
图10显示了在不同的查询窗口下FNR-tree、NDTR-tree、DISC-tree三种索引时空范围查询的时间开销.可以发现:当移动对象从500增加到5000时,DISC-tree时空范围查询时间性能优于NDTR-tree.原因在于时空范围查询过程中,首先通过指定窗口范围对路网进行查询,由于DISC-tree采用R*-tree索引管理路段,查询效率要高于NDTR-tree.然后,通过($period$,Δt)遍历指定道路的R-tree索引,这一步中两者查询效率相当.而FNR-tree的下层索引是基于封闭路段构建的R-tree森林,相比于NDTR-tree和DISC-tree基于道路的R-tree森林,其下层索引中包含更多的R-tree,规模较大不利于查询,查询时间与DISC-tree相当.但是FNR-tree下层索引中每个树包含的记录数量小于NDTR-tree和DISC-tree,在查询过程中,NDTR-tree和DISC-tree对树的遍历操作更消耗时间.由于后一因素起主要作用,因此FNR-tree的查询时间优于NDTR-tree和DISC-tree.如图10所示,随着查询窗口的增大,FNR-tree查询时间增加的趋势要大于NDTR-tree和DISC-tree,原因在于NDTR-tree和DISC-tree下层每个R-tree对应的是一条道路,上层叶子结点与下层R-tree的对应关系为 $n:1$.而FNR-tree的上层索引基本单位是封闭路段,每个下



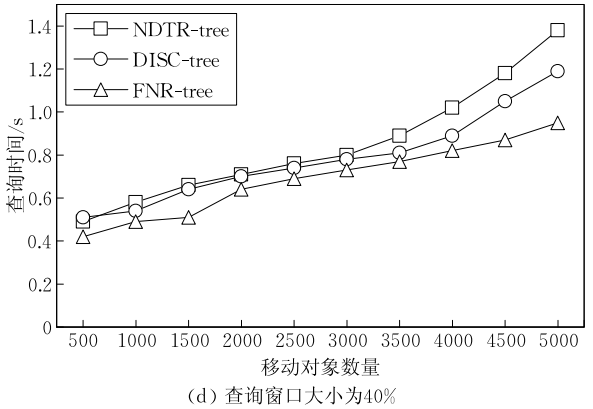
(a) 查询窗口大小为10%



(b) 查询窗口大小为20%



(c) 查询窗口大小为30%



(d) 查询窗口大小为40%

图10 时空范围查询时间比较

层 R-tree 对应一个封闭路段，上层叶子结点与下层 R-tree 的对应关系为 1∶1. 随着查询窗口增大，包含的封闭路段增多. 例如，查询到道路 r 的 3 个封闭路段($r.b_1$，$r.b_2$，$r.b_3$)，在 FNR-tree 中需要对 $r.b_1$，$r.b_2$，$r.b_3$ 在下层索引森林中的 3 个树分别进行查询，而 NDTR-tree 和 DISC-tree 只需要对下层索引森林中 r 所在的树进行查询即可.

为了进一步比较当移动对象数量规模较大时 3 种索引查询时间的性能差异，图 11 给出了查询窗口大小占地图的 30%，移动对象数量从 5500 变化到 10 000 时，NDTR-tree 和 DISC-tree 索引时空范围查询的时间代价.

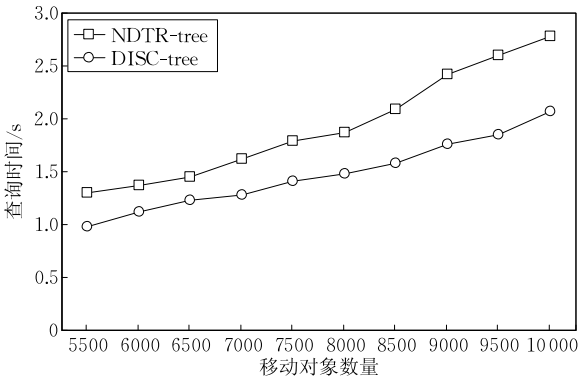


图 11　大规模移动对象下时空范围查询时间比较

实验结果表明：(1) DISC-tree 随着移动对象数量增加其查询时间波动没有 NDTR-tree 剧烈；(2) 当移动对象数量规模较大时(从 5500 变化到 10 000 时)，DISC-tree 性能优势明显，查询效率可以提高 23.5%. 原因与本节之前介绍的小规模移动对象数量下的时空范围查询相同.

7　结　论

目前路网中移动对象索引技术的研究成果较少，仍处于起步阶段. 考虑到现有路网中移动对象索引技术的不足，本文在提高索引建立维护和移动对象轨迹查询性能方面进行了深入研究. 针对 NDTR-tree 在索引维护和轨迹查询方面的不足，FNR-tree 针对时空范围查询准确性不高的缺点，提出了新型基于路网的移动对象动态双层索引结构 DISC-tree，对更新较少的静态路网信息采用 R*-tree 索引管理，对实时更新代价较高的移动对象运动轨迹采用结点更新代价较小的 R-tree 进行索引. 大量的实验结果证明了 DISC-tree 在索引维护和查询性能上领先于 NDTR-tree，在查询精度上优于 FNR-tree.

由于 DISC-tree 建模中没有考虑真实客观环境因素，如红绿灯，高峰路段，车辆限行等，这些因素对路网中移动对象运动趋势会产生影响，未来工作将对影响移动对象运动的主客观因素建模，集成到 DISC-tree 中，进一步提高查询结果的准确性.

参　考　文　献

[1] Ding Z, Li X, Yu B. Indexing the historical, current, and future locations of network-constrained moving objects. Journal of Software, 2009, 20(12): 3193-3204(in Chinese)
(丁治明，李肖南，余波. 网络受限移动对象过去、现在及将来位置的索引. 软件学报，2009，20(12)：3193-3204)

[2] Frentzos E. Indexing objects moving on fixed networks// Proceedings of the 8th International Symposium on Advances in Spatial and Temporal Databases. Santorini Island, Greece, 2003: 289-305

[3] Meng X, Ding Z. Mobile Data Management: Concepts and Techniques. Beijing: Tsinghua University Press, 2009 (in Chinese)
(孟小峰，丁治明. 移动数据管理：概念与技术. 北京：清华大学出版社，2009)

[4] Zhou A, Yang B, Jin C, Ma Q. Location-based services: Architecture and progress. Chinese Journal of Computers, 2011, 34(7): 1155-1171(in Chinese)
(周傲英，杨斌，金澈清，马强. 基于位置的服务：架构与进展. 计算机学报，2011，34(7)：1155-1171)

[5] Wang C. Research on two layers of indexing structures of moving objects based on constrained networks[M. S. dissertation]. Chengdu: Southwest Jiaotong University, 2013(in Chinese)
(王超. 受限路网中移动对象双层索引结构研究[硕士学位论文]. 成都：西南交通大学，2013)

[6] Wu C, Bai Y, Ding Z, Meng X. Grid file based indexing method for moving objects. Computer Science, 2002, 29(8): 203-206(in Chinese)
(吴岑，白芸，丁治明，孟小峰. 基于 GRID 文件的移动对象索引方法. 计算机科学，2002，29(8)：203-206)

[7] Siliva Y N, Xiong X, Walid G. The RUM-tree: Supporting frequent updates in R-trees using memos. The International Journal on Very Large Data Bases, 2009, 18(3): 719-738

[8] Saltenis S, Jensen C S, Leutenegger S T, Lopez M A. Indexing the position of continuously moving objects//Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data. Dallas, USA, 2000: 331-342

[9] Tao Y, Papadias D, Sun J. The TPR*-tree: An optimized spatio-temporal access method for predictive queries// Proceedings of the 29th International Conference on Very Large Data Bases. Berlin, Germany, 2003: 790-801

[10] Chen J, Meng X. Update-efficient indexing of moving objects in road networks. Geoinformatica, 2009, 13(4): 397-424

[11] Pfoser D, Jensen C S, Theodoridis Y. Novel approaches in query processing for moving object trajectories//Proceedings of the 26th International Conference on Very Large Data Bases. Cairo, Egypt, 2000: 395-406

[12] Almeida V T, Guting R H. Indexing the trajectories of moving objects in networks//Proceedings of the 16th International Conference on Scientific and Statistical Database Management. Santorini Island, Greece, 2004: 115-118

[13] Nguyen T, He Z, Zhang R, Ward P. Boosting moving object indexing through velocity partitioning. Proceedings of the VLDB Endowment (PVLDB), 2012, 5(9): 860-871

[14] Zhu Z, Yang Q, Pi D. Past, current and future positions index of moving objects in networks//Proceedings of the 2nd International Conference on Future Control and Automation. Changsha, China, 2012: 335-342

[15] Appathurai K, Karthikeyan S. Moving object indexing using crossbreed update. International Journal of Computer Applications, 2013, 69(16): 25-30

[16] Brinkoff T. Generating network-based moving objects//Proceedings of the 12th International Conference on Scientific and Statistical Database Management. Berlin, Germany, 2000: 253-255

QIAO Shao-Jie, born in 1981, Ph. D., post-doctor, associate professor. His current research interests include moving objects databases and trajectory data mining.

HAN Nan, born in 1984, Ph. D., engineer. Her current research interests include moving objects databases, bioinformatics.

WANG Chao, born in 1985, master. His current research interests include moving objects databases, indexing.

ZHU Feng, born in 1962, Ph. D., professor. His current research interests include granular computing.

TANG Chang-Jie, born in 1946, master, professor, Ph. D. supervisor. His current research interests include databases.

Background

This work is a part of the "Research on Key Techniques for Predicting Uncertain Trajectories of Moving Objects with Dynamic Environment Awareness", which is mainly supported by the National Natural Science Foundation of China under Grant No. 61100045. The project mainly aims to solve the problem of predicting uncertain trajectories of moving objects. In order to design a general schema of trajectory prediction on moving objects data, we explore the techniques including spatio-temporal index, frequent trajectory patterns mining and continuous time Bayesian networks, model the dynamical environmental factors, disclose the characteristic and rules of dynamical behavior of moving objects, study the effect of location, speed, direction, traffic situation, and propose new concepts, specific theories and models to efficiently and effectively predict the future trajectories under dynamic environment in order to set up the methodology of real-time tracking and accurate positioning techniques. The research group including the authors of this paper has proposed several approaches related to trajectory indexing, query optimization, and trajectory prediction in MODs.

This paper presents a two-tiered dynamic index structure of moving objects based on constrained networks, called DISC-tree, which considers the performance of index maintenance and query performance. Empirical studies demonstrate that the DISC-tree outperforms the NDTR-tree in time performance of index creation and maintenance by an average gap of 39.1%, and can reduce query time by an average gap of 24.1%. In addition, compared to the FNR-tree, the accuracy of range query is averagely improved by about 31.6%.