ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.19, No.10, October 2008, pp.2696−2705
DOI: 10.3724/SP.J.1001.2008.02696

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于 U-tree 的不确定移动对象索引策略*

丁晓锋+, 卢炎生, 潘 鹏, 洪 亮, 魏 琼

(华中科技大学 计算机科学与技术学院,湖北 武汉 430074)

U-Tree Based Indexing Method for Uncertain Moving Objects

DING Xiao-Feng+, LU Yan-Sheng, PAN Peng, HONG Liang, WEI Qiong

(College of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China)

+ Corresponding author: E-mail: dxf@smail.hust.edu.cn

Ding XF, Lu YS, Pan P, Hong L, Wei Q. U-tree based indexing method for uncertain moving objects. *Journal of Software*, 2008,19(10):2696−2705. http://www.jos.org.cn/1000-9825/19/2696.htm

Abstract: This paper proposes a novel, U-tree based indexing technique that addresses the problem of managing uncertain data in a constantly evolving environment. The technique called TPU-tree is capable of indexing the moving objects with uncertainty in multi-dimensional spaces. Along with the data models capturing the temporal and spatial uncertainty, a modified *p*-bound based range query (MP_BBRQ) algorithms for probabilistic queries is also developed. Experimental evaluations demonstrate that the TPU-tree supports queries on uncertain moving objects quite efficiently. It yields rather good update performance even under frequent update environments, and has a practical value.

Key words: location management for moving objects; index structure; data uncertainty; TPU-tree; MP_BBRQ (modified *p*-bound based range query) algorithm

摘 要: 通过在U-tree中添加时间戳和速度矢量等时空因素,提出一种基于U-tree的高效率当前及未来不确定位置信息检索的索引结构 TPU-tree,可以支持多维空间中不确定移动对象的索引,并提出了一种改进的基于*p*-bound 的 MP_BBRQ(modified *p*-bound based range query)域查询处理算法,能够引入搜索区域进行预裁剪以减少查询精炼阶段所需代价偏高的积分计算.实验仿真表明,采用 MP_BBRQ 算法的 TPU-tree 概率查询性能极大地优于传统的 TPR-tree 索引,且更新性能与传统索引大致相当,具有良好的实用价值.

关键词: 移动对象位置管理;索引结构;不确定性数据;TPU 树;MP_BBRQ(modified *p*-bound based range query)算法

中图法分类号: TP392 **文献标识码**: A

随着移动终端功能的逐渐完备,无线通信与GPS等空间定位技术的迅速发展,诸多应用,如实时交通信息管理与导航、军事监控和民航管制等要求对移动终端的实时位置信息进行监控和管理.传统数据库索引技术是为存储更新周期相对较长的精确数据而设计的,其索引结构中存储移动对象的精确位置,不断变化的移动对象位置信息会使得更新过于频繁而导致系统资源枯竭.由于带宽受限和移动终端的节能机制,在数据库中存储移动

* Supported by the Pre-Research Project of the 'Eleventh Five-Year-Plan' of China under Grant No.513150402 (国家'十一五'预研基金)

Received 2007-07-30; Accepted 2008-02-25

对象的精确位置信息几乎是不可实现的,而基于此类信息的检索查询,在没有任何质量系数的保证下显得毫无意义. 在索引结构中存储移动对象的不确定运动信息已成为目前时空数据库领域的研究热点.

近年来,针对如何高效管理移动对象实时变化的位置信息的问题,研究人员提出了一些新的索引模型,大致可以分为两类:一类是针对移动对象历史位置信息的索引;另一类是针对移动对象当前及未来位置信息的索引.其中包括许多基于参数化的索引方法来对移动对象当前及未来位置信息进行管理,如 TPR-tree[1]及其变种 TPR*-tree[2],R^{EXP}-tree[3]等.TPR-tree 及其变种 TPR*-tree 充分地将传统 R-tree 的查询、插入及删除等优势算法与自身的需求相结合,到达了良好的索引和查询性能,得到了业界与学术界的一致认可,从而成为目前广泛使用的移动对象当前及未来位置信息索引方法,但其固有的自顶向下(top-down)更新模式,由于采用传统的删除加插入两阶段更新策略,引起较大的磁盘 I/O 次数而难以满足大量并发更新的要求.R^{EXP}-tree 借用 TPR-tree 的时间参数策略,通过在 R*-tree 上添加数据的有效期属性,提高了无效数据的删除效率,从而表现出较好的查询和更新性能.针对如何有效地结合相关索引的优点以解决全索引问题,最近研究人员提出能够同时对移动对象历史、当前及未来位置信息进行索引的模型(BB^x-tree[4],R^{PPF}-tree[5]),并通过实验证明了其索引的有效性和实用性.但上述索引方法均缺乏对不确定地理位置信息的支持和查询,从而在此类应用中由于不恰当的信息存储而引起很大的查询和更新代价.Tao 等人[6]提出了基于 R*-tree 的不确定对象索引策略 U-tree,其固有的良好的动态结构可以使数据对象以任何次序更新或插入,而且对不确定数据本身的概率密度分布(probability density function,简称 PDF)没有任何限制,实验结果表明,U-tree 具有良好的动态更新性能和域查询性能,域查询的磁盘 I/O 和 CPU 计算时间可以得到最大程度的优化.但 U-tree 本身只是针对不确定静止对象的索引,包括 ORION[7]和 TRIO[8]项目在内的索引策略也尚未实现对不确定移动对象信息的支持,因此,针对不确定移动对象位置信息的索引是当前学术界亟待解决的问题之一.

本文针对支持频繁位置更新的不确定移动对象当前及未来位置索引方法,提出了一种基于 U-tree 的高效率当前及未来不确定位置信息检索的索引结构 TPU-tree,并提出了一种改进的基于 p-bound 的域查询(modified p-bound based range query,简称 MP_BBRQ)处理算法.TPU-tree 在基本 U-tree 结构上增加了记录移动对象不确定状态特征的数据结构,通过利用概率密度函数描述移动对象在不确定区域(uncertain region,简称 UR)的位置分布,在保留原有位置记录的情况下加入时间特性,这样就可以预测移动对象在未来时间段内的大概位置信息,从而为当前及未来不确定位置信息检索提供可靠保证.实验仿真结果表明,采用 MP_BBRQ 算法的 TPU-tree 概率域查询效率可以得到很大程度的提高,且动态更新性能与传统索引大致相当.

1 数据模型及相关查询

本节首先给出了文献[6,9,10]中引出的相关不确定数据模型概念,这些不确定数据模型可以满足大部分实际应用的需求.然后基于不确定移动对象数据模型,本节介绍了概率相关的查询定义,并在后续部分直接使用这些基本定义.

1.1 不确定对象数据模型

针对不确定数据模型的研究已经有了相当广泛的基础.Cheng 等人[9]提出较为流行的不确定数据模型:在一定时间段 T 内,对象分布在距离原始记录点不超过特定距离 d 的有效区域内,如果对象的移动超过了距离 d,那么就对数据对象进行更新,并重新确定数据原始记录点和有效区域.在此基础上,人们主要提出了两种不确定移动数据模型:不确定直线数据模型和不确定自由运动数据模型.其中,不确定直线数据模型假设对象总是沿特定直线运动(比如公路网中的某路段),对象位置在任何时间段内总是分布在指定的距离段内,距离段 d 会随着对象记录中心的变化自行调整.而不确定自由运动数据模型并没有对移动对象的运动轨迹有特殊限制,研究者同样提出了相关的不确定移动数据模型,而移动对象的实际运动轨迹对此并没有很大意义.结合不确定自由运动数据模型的特点,本文给出如下不确定移动数据模型的相关定义:

定义 1(不确定区域)[9,10]. 移动对象 M_i 在 t 时刻的不确定区域是一个封闭区域:$Ur_i(t)$,M_i 只可能处于 $Ur_i(t)$ 之内的任何位置.

定义 2(概率密度分布)[10]. 移动对象 M_i 在不确定区域 $Ur_i(t)$内的概率密度分布函数记为 $PDF_i(x,t)$.它描述了移动对象 M_i 在 t 时刻处于位置 x 的概率.

由于移动对象 M_i 在 t 时刻只能处于 $Ur_i(t)$之内的任何位置,所以在 $Ur_i(t)$之外分布的 $PDF_i(x,t)$值为 0,也即有概率密度分布的重要特性: $\int_{UR_i(t)} PDF_i(x,t)\mathrm{d}t=1$.

定义 3(移动速度). 移动对象 M_i 在 t 时刻的移动速度记为 $V_i(t)$.

定义 4(概率值集合). 概率 p_i 组成的势为 m 的集合,记为 $U\text{-}catalog=\{p_1,\ldots,p_m\}=\{0,0.5/(m-1),1/(m-1),\ldots,0.5\}$.

不确定移动对象的实例如图 1 所示.其中白色圆圈表示移动对象 M_i 在数据库中的实际记录位置,而 M_i 有可能出现在其不确定区域 UR(灰色区域)中的任何位置,其具体分布与 $PDF_i(x,t)$直接相关.最常见的概率密度分布函数为均匀分布,它描述了一般或最坏情况下的概率分布,即移动对象 M_i 均匀地分布在 $Ur_i(t)$之内,并不是在其中某一点上具有较高的出现机率.均匀分布有助于减少域查询的 CPU 计算时间和磁盘 I/O 次数.针对不确定区域内的高斯分布,学者们也相应提出了基于平均值和方差的查询计算方法.不同的概率密度分布函数具有相应的查询处理方法,特定处理方法不适应于其他概率密度分布.因此,如何针对不同概率密度分布函数提出统一的查询处理方法是索引方法的关键点.Tao 等人[6]提出的 U-tree 在不确定概率密度分布的情况下,利用 p-bound 技术提出统一的域查询处理方法,并使得域查询的磁盘 I/O 和 CPU 计算时间得到最大程度的优化.但此方法仅适应于静止的不确定对象,本文将在后续部分给出不确定移动对象的查询处理方法.

定义 5(概率限定性区域)[9,10]. 移动对象 M_i 的概率限定性区域(probability constrained region,简称 PCR)以概率值 p_i 为参数,记为 $M_i.PCR(p_i)$,它在 d 维空间中由 $2d$ 个线段:$\{L_1(p_j),U_1(p_j),\ldots L_d(p_j),U_d(p_j)\}$组合而成.其中 $L_i(p_j)$和 $U_i(p_j)$分别将 $UR_i(t)$分成两部分,而 M_i 出现在 $L_i(p_j)$和 $U_i(p_j)$线以左(下)和以右(上)区域的概率均等于 p_i.

图 2 所示为二维空间中移动对象 M_i 的概率限定性区域 $PCR(0.3)$.其中,M_i 出现在阴影区域中的概率,以及出现在 $U_1(0.3)$以右、$L_2(0.3)$以下和 $U_2(0.3)$以上的概率均等于 0.3.

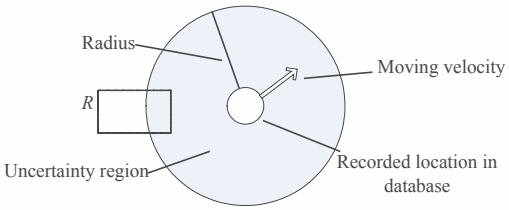


Fig.1 Example of uncertain moving object

图 1 不确定移动对象的实例结构

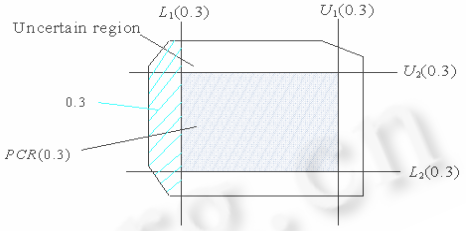


Fig.2 Example of probability constrained region

图 2 概率限定性区域的实例

1.2 概率查询

基于上述不确定移动对象数据模型,几种重要的时空数据库查询类型有了更新的诠释.移动对象位置的不确定性导致查询结果发生了实质性变化,此结果集中不能再以简单的布尔类型(是/否)来判定一个移动对象是否满足特定条件的查询,而是以一定的概率来满足查询条件.

为了便于陈述,本文假定系统中的移动对象都具有唯一标识符, 所有对象的集合表示为 $M=\{M_1,M_2,\ldots,M_n\}$,其中,n 是系统中所有移动对象的个数,$M_i(1\le i\le n)$为第 i 个移动对象的标识符.因此,不确定移动对象查询结果集中应包含这样的元组(M_i,p_i),它表示移动对象 M_i 满足某种查询的概率为 p_i.

假定如图 1 所示的查询区域为 R,本文相关的域查询可定义如下:

定义 6(不确定性域查询).不确定性域查询(imprecise range query,简称 IRQ)就是对于一个给定的查询区域 R,在一定时间段 T 内找出所有与 R 相交的移动对象 M_i,其查询结果集为

$$IRQ(M,R,T)=M'=\{(M_{i1},p_{i1}),(M_{i2},p_{i2}),\ldots,(M_{im},p_{im})\},$$

且满足条件,对 $\forall(M_j,p_j)\in M'$,有 $M_i\in M, p_i\neq 0, UR_i(T)\cap R\neq\varnothing$.

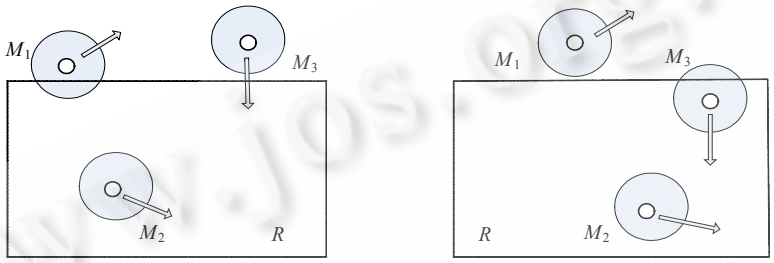
事实上,作为不确定性域查询的一般情况,限定性不确定域查询在实际应用需求中更加广泛.用户更青睐概率较高的查询结果,因此,对于指定的概率阈值 P_c,更一般的限定性不确定域查询定义如下:

定义 7(限定性不确定域查询).限定性不确定域查询(constrained imprecise range query,简称 CIRQ)就是对于一个给定的查询区域 R,在一定时间段 T 内找出所有与 R 相交的移动对象 M_i,其查询结果集为

$$CIRQ(M,R,T,P_c)=M'=\{(M_{i1},p_{i1},)(M_{i2},p_{i2}),...,(M_{im},p_{im})\},$$

且满足条件,对 $\forall(M_j,p_j)\in M'$,有 $M_i\in M, p_i\geq p_c, UR_i(T)\cap R\neq\varnothing$.

本文将不确定性域查询和限定性不确定域查询统称为概率查询.由于数据对象的移动性,即使是在数据库没有更新的情况下,相同的概率查询在不同时刻仍会产生异样的结果.如图 3(a)和图 3(b)所示,不确定移动对象 M_1,M_2,M_3 是 IRQ 分别在时刻 T_1,T_2 的查询对象,其查询区域均为 R.假设所有移动对象的 $PDF_i(x,t)$服从均匀分布,则在时刻 T_1 的查询结果大约(因此时 p_i 为估计值)为 $\{(M_1,0.2),(M_2,1)\}$.随着时间的变化,在时刻 T_2 移动对象如图 3(b)所示,同样的查询区域 IRQ 就有了不同的查询结果:$\{(M_3,0.8),(M_2,1)\}$.



(a) Situations at time T_1 (b) Situations at time T_2

(a) T_1 时刻的情况 (b) T_2 时刻的情况

Fig.3 Example of imprecise range query

图 3 不确定性域查询实例

显然,概率查询的关键点是如何计算移动对象集中每个对象 M_i 的概率因子 p_i,然后依据概率因子 p_i 构建查询结果集.计算 p_i 的传统方法是在移动对象的不确定区域 $UR_i(t)$与查询区域 R 的交集上对概率密度分布函数 $PDF_i(x,t)$进行积分,其公式为

$$p_i=\int_{UR_i(t)\cap R}PDF_i(x,t)\mathrm{d}x \tag{1}$$

如第 2.1 节所述,由于 $PDF_i(x,t)$的不确定性,积分公式(1)具有较高的计算代价,即使在 $PDF_i(x,t)$均匀分布的前提下,概率查询效率仍很难得到进一步的提高.为了避免这种计算代价很高的积分运算,本文将在第 3.3 节介绍一种新的基于 p-bound 的查询处理算法 MP_BBRQ,在保证更新性能的情况下,极大地提高了概率查询效率.

2 TPU-tree 移动对象索引

U-tree 是 Tao 等人[6]提出的一种针对不确定对象当前位置信息管理的索引模型.U-tree 依据存储在 U-catalog 中的 m 个概率 p_i 预先计算出不确定对象的 $PCR(p_i)$,然后利用 $PCR(p_i)$和相关判定准则可以迅速地对不确定对象进行定位并返回给查询结果集.如图 2 所示,对于给定限定性不确定域查询 Q:查询区域为 R 以及概率阈值 p_c= 0.3,如果 R 能包括所有的阴影区域,或者 $U_1(0.3)$以右、$L_2(0.3)$以下和 $U_2(0.3)$以上区域中的任何一个,那么移动对象 M_i一定能够满足查询 Q.相反地,如果 R 只与阴影区域的一部分相交,并且 R 只出现在 $L_1(0.3)$以左的区域,那么移动对象 M_i 一定不满足查询 Q.

U-tree 采用与 R*-tree 相同的原理进行构建,它们具有同样的优化准则,只不过由于 U-tree 本身的复杂性和实际需要,它在相关算法中使用优化参数的总和替代了 R*-tree 相对应的优化参数:① $\sum_m MARGIN(MBR)$,

② $\sum_m AREA(MBR)$,③ $\sum_m OVERLAP(MBR,MBR')$,④ $\sum_m CDIST(MBR,MBR')$.

鉴于此,本文提出一种基于 U-tree 的高效率当前及未来不确定位置信息检索的索引结构 TPU-tree.根据基于 p-bound 的查询处理思想,索引结构必须能够区分具有不同概率因子 p_i 的移动对象位置记录 $PCR(p_i)$.在保证 TPU-tree 稳定、高效的前提下,为了将不同移动对象位置记录 $PCR(p_i)$快速插入到索引树中,本文提出了改进的插入/删除优化算法.针对如何利用 $PCR(p_i)$以提高 TPU-tree 的查询效率,本文介绍了一种改进的基于 p-bound 的域查询处理算法,并通过实验证明了算法的有效性.

2.1 TPU-tree索引结构

由于每一个不确定移动对象所包含的 $PCR(p_i)$本身的复杂性,在 TPU-tree 中能否有效地管理和维护所有不确定移动对象的 $PCR(p_i)$是影响索引树性能的关键.在 U-tree 基本索引结构基础上,每个 TPU-tree 叶子节点的记录项都添加了一个标识该记录插入时间的 *time-stamp* 属性和移动速度矢量,记录形式为〈*oid*,$PCR(p_i)$,*VBR*,*ptr*,*time-stamp*〉,其中 *oid*,$PCR(p_i)$,*VBR*,*ptr*,*time-stamp* 分别表示移动对象标识、不确定区域、p_i 限定性区域、概率密度分布函数、速度包围框、节点磁盘页面地址、时间戳.特别地,*ptr* 指向的磁盘页面地址存储 $UR_i(t)$和 $PDF_i(x,t)$.

所有 TPU-tree 中间节点的记录项都添加了一个标识时间的 *time-stamp* 属性和一个保守速度包围框(conservative velocity bounding rectangle,简称 CVBR),CVBR 只可能扩张或平移而不会缩小,从而保证了父亲节点的最小包围框在任意时刻总是包围着所有的子节点.中间节点记录项的记录形式为〈*MBR*∨,*MBR*∧,*CVBR*,*ptr*,*time-stamp*〉,其中 *MBR*∨,*MBR*∧,*CVBR*,*ptr*,*time-stamp* 分别表示包含所有子节点 $PCR(p_1)$的最小边界包围框、包含所有子节点 $PCR(p_m)$的最小边界包围框、保守速度包围框、子节点磁盘页面地址以及时间戳.

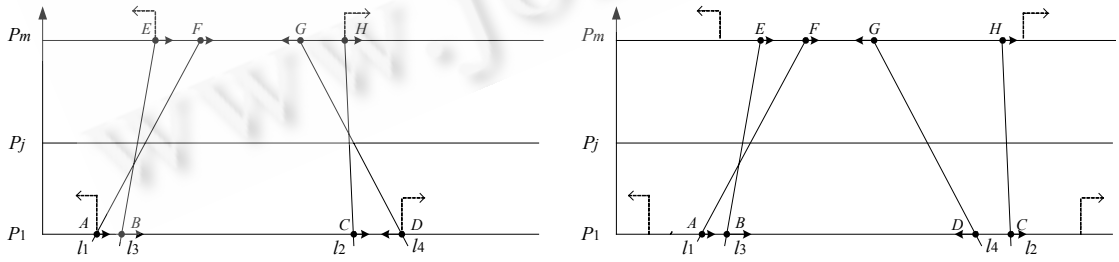
图 4 所示为单维空间中两个不确定移动对象 M_1 和 M_2 在 TPU-tree 中的索引结构.其中,图 4(a)为初始时刻 T=0 的情形,线段 l_1 和 l_2 表示移动对象 M_1 对应的 $PCR(p_i)$,线段 l_3 和 l_4 表示移动对象 M_2 对应的 $PCR(p_i)$.此时,直线段 *AC* 对应于 $M_1.PCR(p_1)$,*BD* 对应于 $M_2.PCR(p_1)$;直线段 *FH* 对应于 $M_1.PCR(p_m)$,*EG* 对应于 $M_2.PCR(p_m)$.假设移动对象 M_1 和 M_2 是中间节点记录项 *e* 唯一的两个子节点,那么 *e.MBR*∨就是线段 *AC* 和 *BD* 的最小边界包围框 *AD*, *e.MBR*∧就是线段 *FH* 和 *EG* 的最小边界包围框 *EH*,并且 *AD* 和 *EH* 沿着指定的 CVBR 移动,这样可以确保在未来的某个时刻 *AD* 和 *EH* 仍然处于其子节点的 $PCR(p_i)$内,如图 4(b)所示.

在给出各种算法之前,我们给出如下定义和定理:

定义 8(p_i最小边界包围框). 对$\forall p_i \in U\text{-}catalog$,索引树中间节点记录项 *e* 的 p_i 最小边界包围框就是 *e* 包含的所有子节点 $PCR(p_i)$的最小边界包围框,记为 $e.MBR(p_i)$.

定理 1. 在索引树中间节点记录项 *e* 的 *MBR*∨和 *MBR*∧已知的情况下,对$\forall p_i \in U\text{-}catalog$,$e.MBR(p_i)$是可计算的.

证明:在任意时刻 $t\in[T_s,T_s+H]$,根据定义 7,$e.MBR(p_1)=e.MBR\vee$,$e.MBR(p_m)=e.MBR\wedge$,则 $e.MBR(p_i)$是满足形参为 p_i 的线性函数:$e.MBR(p_i)=\alpha-\beta\cdot p_i$.当 p_1=0 时,可以推出$\alpha=e.MBR\vee$,$\beta=(e.MBR\vee-e.MBR\wedge)/p_m$,因此,$e.MBR(p_i)=e.MBR\vee-(e.MBR\vee-e.MBR\wedge)p_i/p_m$ 是可计算的. □



(a) Situation at time T=0 (b) Situation at time T=5

(a) T=0 时刻的情况 (b) T=5 时刻的情况

Fig.4 Node representation of TPU-tree

图 4 TPU-tree 中的节点表示

2.2 插入/删除算法

当移动对象发出位置更新请求时,新的位置记录信息要求被 TPU-tree 索引,本文中 TPU-tree 采用传统的删除加插入两阶段更新策略,这等价于在索引树中删除旧记录并插入新记录.因此,插入/删除记录在 TPU-tree 中具有与 U-tree 相似的处理过程,其关键在于如何处理更为复杂的 m 个位置信息记录 $MBR(p_i)$.本文提出的插入/删除算法思想如下:首先,由于 TPU-tree 支持对未来时间段 H 内的信息索引,就需要对此时间段内的优化参数进行积分,以达到全程监控和系统最优.然后在原始算法中将 m 个积分结果求和取代 U-tree 对应的优化参数进行各种插入/删除操作.具体优化参数为:

① $\sum_m\left(\int_t^{t+H} MARGIN(MBR(P_j),t)\mathrm{d}t\right)$,其中 $MARGIN$ 计算 $MBR(p_i)$在 t 时刻的空白面积;

② $\sum_m\left(\int_t^{t+H} AREA(MBR(P_j),t)\mathrm{d}t\right)$,其中 $AREA$ 计算 $MBR(p_i)$在 t 时刻的占用面积;

③ $\sum_m\left(\int_t^{t+H} OVERLAP(MBR(P_j),MBR'(P_j),t)\mathrm{d}t\right)$,其中 $OVERLAP$ 计算两个中间节点 $MBR(p_i)$在 t 时刻的重叠面积;

④ $\sum_m\left(\int_t^{t+H} CDIST(MBR(P_j),MBR'(P_j),t)\mathrm{d}t\right)$,其中 $CDIST$ 计算两个中间节点 $MBR(p_i)$在 t 时刻的垂直距离.

节点分裂算法在 U-tree 之上有所改进,在确定分裂节点 e 的 $MBR(p_{\lceil m/2\rceil})$之后,需要计算移动对象在不同时刻的位置以供排序.在依据空间位置排序的同时,分裂算法也可以根据速度矢量进行排序.如果节点分裂后出现下溢情况,算法会删除节点中所有的记录项并重新插入.

2.3 概率查询处理

TPU-tree 的查询处理过程类似于 U-tree,主要由 3 个阶段组成:(1) 裁剪阶段,排除不满足查询要求移动对象,并形成部分结果集和候选集;(2) 精炼阶段,利用传统的方法检查候选集中的移动对象是否满足查询要求;(3) 合成阶段,形成最终结果集并返回.其中裁剪阶段的关键是本文提出的改进基于 p-bound[11]的域查询处理算法.MP_BBRQ 查询算法首先从 TPU-tree 根节点进行开始搜寻,根据查询区域 R 和指定概率 P_c 排除不符合要求的记录项;对每个剩余的记录项,利用上述步骤重复检索其子节点直到叶子节点;然后,对叶子节点中每个移动对象进行判断,将不能判定是否满足查询要求的对象存入候选集中以备后用.MP_BBRQ 查询处理算法和具体过程如下所示:

算法 1. MP_BBRQ 查询处理算法.

输入:限定性不确定域查询 Q,查询区域 R,查询时间 T,概率阈值 P_c ;TPU-tree 索引 TI.

输出:部分查询结果集 PAS,查询候选集 RAS.

步骤:

1. 对 TPU-tree 索引 TI 进行深度优先搜索,从根节点开始,对于每个访问节点:

若为中间节点,则对节点中所有的 $MBR(p_i)$:

1.1.1. 若 $p_c\in U\text{-}catalog$,则令 $p_i=p_c$,否则 p_i 取 U-catalog 中小于 p_c 的最大值;

1.1.2. 若 $R\cap e.MBR(p_i)=\varnothing$,则裁剪掉 e 所指向的子节点;

1.1.3. 若 $R\cap e.MBR(p_i)\neq\varnothing$,则继续对 e 所指向的子节点进行深度优先搜索.

若为叶子节点,则对其中包含的所有不确定移动对象 M_i:

若 $p_c>1-p_m$,且 p_i 取 U-catalog 中不小于 $1-P_c$ 的最小值:

若 $M_i.PCR(p_{ii})$不能完全包含于 R 中,则裁剪掉移动对象 M_i;

若 $M_i.PCR(p_i)$完全包含于 R 中,则继续跟进 M_i.

若 $p_c>0.5$,且 p_i 取 U-catalog 中不大于 $1-p_c$ 的最大值:

若 $UR_i(T)$出现在 $L_i(p_i)$线以右或 $U_i(p_i)$线($\exists i\in[1,d]$)以左的区域完全包含于 R 中,则将 M_i 写入到部分查询结果集 PAS 中;

否则继续跟进 M_i.

若 $P_c \le 1-p_m$,且 p_i 取 U-catalog 中不大于 P_c 的最大值:

1.2.3.1 若 $M_i.PCR(p_i) \cap R=\varnothing$,则裁剪掉移动对象 M_i;

1.2.3.2 若 $M_i.PCR(p_i) \cap R\neq\varnothing$,则继续跟进 M_i.

若 $P_c \le 0.5$,且 p_i 取 U-catalog 中不小于 P_c 的最小值:

1.2.4.1 若 $UR_i(T)$出现在 $L_i(p_i)$线以左或 $U_i(p_j)$线($\exists i \in [1,d]$)以右的区域完全包含于 R 中,则将 M_i 写入到部分查询结果集 *PAS* 中;

1.2.4.2 否则继续跟进 M_i.

将 p_i 取 U-catalog 中不大于$(1-P_c)/2$ 的最大值,若 $UR_i(T)$出现在 $L_i(p_i)$线和 $U_i(p_i)$线($\exists i \in [1,d]$)之间的区域完全包含于 R 中,则将 M_i 写入到部分查询结果集 *PAS* 中,否则将 M_i 写入到查询候选集 *RAS* 中.

2. 返回部分查询结果集 *PAS* 和查询候选集 *RAS*.

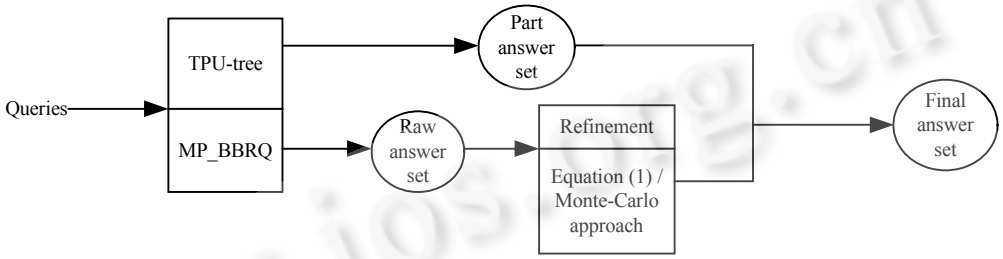


Fig.5 Query process of TPU-tree

图 5 TPU-tree 查询处理过程

3 实验仿真与性能分析

为了评价 TPU-tree 索引和基于 TPU-tree 索引的查询与动态更新性能,我们设计了一组实验,对文中提出的算法给出了验证分析,同时与 TPR-tree 进行了性能比较.

实验数据集使用文献[12]中基于道路网络的移动对象产生器随机生成.在 10000Km×10000Km 空间区域内模拟 100K 移动对象的运动情况.每个移动对象的不确定区域是半径为 250Km 的圆圈,概率密度分布为平均分布或高斯分布.在初始时刻 t 每个移动对象选择一个距离其最近的道路节点为目的地开始运动,到达目标之后发出位置更新请求并重新随机选择一个新目的地运动.其中,移动对象运动目的地设为 5 000 个 MBR,移动速度在[20,50]之间均匀分布.假设初始阶段所有的移动对象都在运动当中,不会消失,也不会在实验进行当中增加新的移动对象.

对移动对象数据集,我们基于 Gist 分别构建 TPR-tree 及 TPU-tree 索引,节点大小均设置为 1KB,中间节点扇出两者均为 27,TPU-tree 中叶子节点保存的移动对象个数随 U-catalog 的大小而变化,页面缓存大小均为 100KB,索引树高度均为 4.索引树根节点常驻缓存中,并使用最近最少使用(least recently used,简称 LRU)缓存替代策略.实验硬件环境是:Intel(R) Pentium(R)4 1.70GHz 的 CPU,内存为 512MB RAM;软件环境是:Windows XP 操作系统和 Visual C++ 6.0 集成开发环境.

U-catalog 势的大小 m 直接影响到 TPU-tree 的查询和更新性能.在预备实验中,我们选择不同的 m 值以保证其设置使 TPU-tree 得到最大程度的优化.图 6 比较了不同 m 设置情况下 TPU-tree 的查询处理时间.从预备实验可以看出,在 U-catalog 势的初始增加阶段,索引树的查询时间呈急剧下降趋势.这是由于较大的 m 值会产生节点记录项中较多的 $PCR(p_i)$,在很大程度上增加了 TPU-tree 的裁剪效率,从而缩短了移动对象的查询时间.不难发现,在 m 值增至 10 时,TPU-tree 的查询时间逐步稳定在 1s 上下,这是由于过大的 m 值减少了节点扇出,引起较多的磁盘 I/O.这就为 TPU-tree 产生较为理想的更新与查询效率提供了最佳时机,因此在下面的实验中,我们均设定 m=10.

3.1 更新代价

一般情况下,索引树的更新性能会随着时间的推进而逐渐下降,因此我们比较了这两种索引方法在每隔2.5K 次更新后的索引性能.图 7 比较了在不同移动对象更新次数情况下的 TPU-tree 与 TPR-tree 动态更新所需平均磁盘 I/O 次数.其中,移动对象位置更新保持较高的频率,平均每秒钟增加的更新次数在 10K 左右.可以看出,TPU-tree 的更新所需平均磁盘 I/O 只是略高于 TPR-tree,其更新性能不是随着更新次数的增加呈迅速下降趋势,而是具有很好的动态更新性能.这是由于 TPU-tree 和 TPR-tree 都是采用自顶向下(top-down)的更新模式进行更新,而 TPU-tree 索引由于在节点记录项中包含较多的 $PCR(p_i)$记录使得节点扇出减少,致使其节点包含记录项的个数也相对减少.因此,在节点包括范围减小的情况下,TPU-tree 删除移动对象旧记录所需搜索旧记录的过程会增加额外的磁盘 I/O,从而导致更新性能的相对下降.特别地,在 TPU-tree 的实现中我们采取了 TPR*-tree 的主动紧缩技术,以尽量避免各个节点记录项 e 包含的 $MBR^{\wedge}$之间的重叠,从而使得更新性能的下降是有限的,表现出较为稳定的更新性能.

3.2 概率查询代价

我们同时比较了 TPU-tree 和 TPR-tree 回答 100 个 CIRQ 查询所需要的平均索引节点磁盘 I/O 次数和 CPU 计算时间.图 8(a)和图 8(b)所示分别为固定 CIRQ 查询时间 T=50 和概率阈值 P_c=0.6 时,TPU-tree 与 TPR-tree 回答查询所需要的磁盘 I/O 代价和 CPU 计算时间.从中可以看出,随着查询区域 R 的增加,两种索引树的查询性能均会下降,但基于 TPU-tree 索引的 MP_BBRQ 查询算法具有较好的查询性能,基于 TPR-tree 索引的查询算法显然次之.这是由于,基于 TPU-tree 索引的 MP_BBRQ 查询算法可以快速裁剪掉不符合查询要求的移动对象,只有一小部分对象需要进入精炼阶段进行代价很高的积分计算,因而具有较好的查询性能.而 TPR-tree 的查询算法只是对查询域进行了裁剪,并不能裁剪掉概率值不符合要求的移动对象,它只有通过代价很高的积分计算来判断所有裁剪域内的移动对象是否满足 CIRQ 查询条件,从而影响了查询性能.

图 9(a)和图 9(b)所示分别为固定 CIRQ 查询时间 T=50 和查询区域大小 R=1500 时,TPU-tree 与 TPR-tree 随着 CIRQ 查询概率阈值 P_c 的变化回答查询所需要的磁盘 I/O 代价和 CPU 计算时间.从中可以看出,随着概率阈值 P_c 的增加,TPU-tree 索引的查询性能均呈略微上升态势,且基于 TPU-tree 索引的 MP_BBRQ 算法在磁盘 I/O 次数和 CPU 计算时间上远优于基于 TPR-tree 索引的查询算法.这是由于,基于 TPU-tree 索引的 MP_BBRQ 查询算法可以根据查询区域 R 和查询概率阈值 P_c 快速裁剪掉不符合查询要求的移动对象,且概率阈值 P_c 越高,MP_BBRQ 算法的裁剪效率越好,因而其查询性能逐渐提高.而 TPR-tree 的查询算法首先根据查询域 R 进行了裁剪,并不能根据查询概率阈值 P_c 裁剪掉不符合查询条件的移动对象,然后逐一地对符合条件的查询域 R 内所有移动对象通过代价很高的积分计算以得到最终的查询结果,因此其查询性能不受查询概率阈值 P_c 的影响.

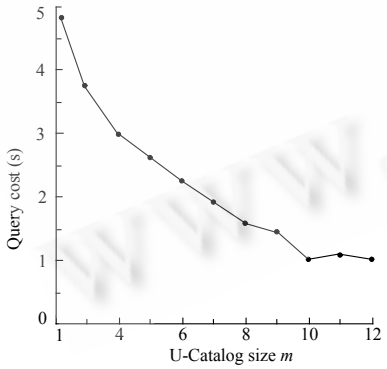


Fig.6 Effect of m to query cost
图 6 势 m 对查询时间的影响

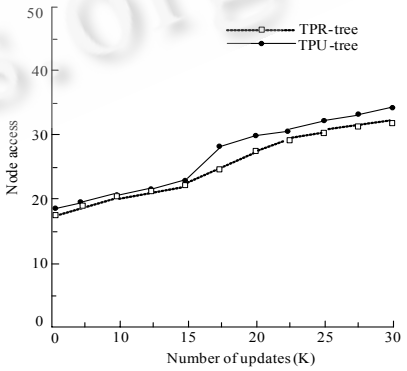
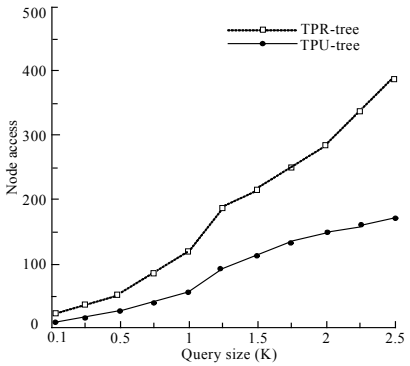
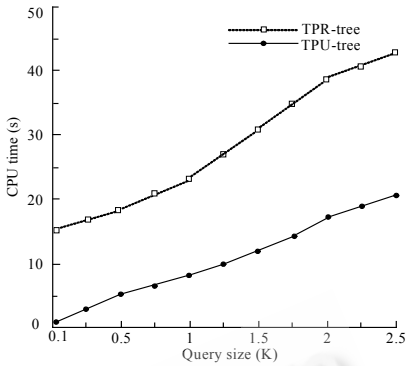


Fig.7 Effect of update number to update performance
图 7 更新次数对更新性能的影响

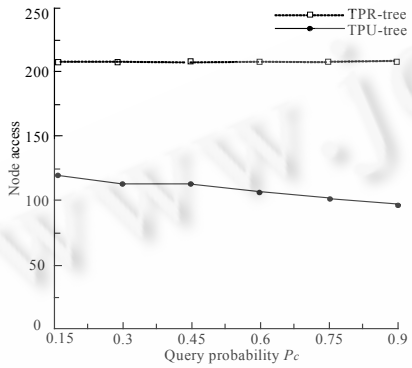


(a) Node access vs. query size
(a) 磁盘 I/O 对比查询区域大小

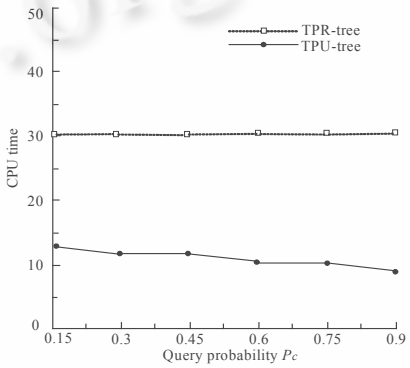


(b) CPU time vs. query size
(b) CPU 时间对比查询区域大小

Fig.8 Effect of query region size to query performance
图 8 查询区域大小对查询性能的影响



(a) Node access vs. P_c
(a) 磁盘 I/O 对比概率阈值



(b) CPU time vs. P_c
(b) CPU 时间对比概率阈值

Fig.9 Effect of probability thresholds P_c to query performance
图 9 概率阈值 P_c 对查询性能的影响

4 结 论

本文在当前流行的不确定静止对象位置信息管理的索引技术 U-tree 的基础上,针对不确定移动对象当前及未来位置索引问题,提出了一种高效率当前及未来不确定位置信息检索的索引结构——TPU-tree.与传统的索引方法相比,TPU-tree 可以支持多维空间中不确定移动对象的索引,并基于 TPU-tree 索引提出了一种改进的基于 p-bound 的 MP_BBRQ 域查询处理算法,能够引入搜索区域进行预裁剪以减少查询精炼阶段所需代价偏高的积分计算,且具有良好的可伸缩性,可以支持移动对象的插入、删除等动态更新操作.实验仿真结果表明,基于 TPU-tree 索引的 MP_BBRQ 域查询处理算法极大地减少了查询所需要访问的磁盘页面和 CPU 计算时间,优于目前最好的基于 TPR-tree 的移动对象索引方法.在基于位置的服务、移动计算等具有不确定性数据要求的应用领域,TPU-tree 具有较高的实用价值.在未来的工作中,我们将进一步优化 TPU-tree 索引结构,通过引入辅助索引结构,在不影响 TPU-tree 查询性能的前提下,完善 TPU-tree 的更新性能,以解决频繁更新应用领域中的新问题.

References:

[1] Saltenis S, Jensen CS, Leutenegger S, Lopez MA. Indexing the positions of continuously moving objects. In: Proc. of the ACM SIGMOD Int'l Conf. Management of Data. New York, 2000. 331−342. http://www.cs.auc.dk/~tbp/Teaching/DAT5E01/simonasSIGMOD2000.pdf

[2] Tao Y, Papadias D, Sun J. The TPR*-tree: An optimized spatio-temporal access method for predictive queries. In: Proc. of the Int'l Conf. on Very Large Data Bases (VLDB) 2003. San Francisco, 2003. 790−801. http://www.vldb.org/conf/2003/papers/S24P01.pdf

[3] Saltenis S, Jensen CS. Indexing of moving objects for location-based services. In: Proc. of the IEEE Int'l Conf. on Data Engineering (ICDE). 2002. 463−472. http://www.cs.auc.dk/research/DP/tdb/TimeCenter/TimeCenterPublications/TR-63.ps.gz

[4] Lin D, Jensen CS, Ooi BC, Saltenis S. Efficient indexing of the historical, present, and future positions of moving objects. In: Proc. of the Int'l Conf. on Mobile Data Management (MDM). 2005. 59−66. http://www.comp.nus.edu.sg/~spade/pub/mdm05.pdf

[5] Pelanis M, Saltenis S, Jensen CS. Indexing the past, present, and anticipated future positions of moving objects. ACM Trans. on Database Systems, 2006,31(1):255−298.

[6] Tao Y, Cheng R, Xiao X, Ngai WK, Kao B, Prabhakar S. Indexing multi-dimensional uncertain data with arbitrary probability density functions. In: Proc. of the Int'l Conf. on Very Large Data Bases (VLDB). Trondheim, 2005. 922−933. http://www.vldb2005.org/program/paper/fri/p922-tao.pdf

[7] Singh S, Mayfield C, Prabhakar S, Shah R, Hambrush S. Indexing uncertain categorical data. In: Proc. of the IEEE Int'l Conf. on Data Engineering (ICDE). 2007. 616−625. http://www.cs.purdue.edu/homes/sunil/pub/catInd.pdf

[8] Widom J. Trio: A system for integrated management of data, accuracy and lineage. In: Proc. of the Int'l Conf. on Innovative Data Systems Research (CIDR). 2005. 262−276. http://www.cidrdb.org/cidr2005/papers/P22.pdf

[9] Cheng R, Kalashnikov DV, Prabhakar S. Querying imprecise data in moving object environments. IEEE Trans. on Knowledge and Data Eng., 2004,16(9):1112−1127.

[10] Ding X, Lu Y. Indexing the imprecise positions of moving objects. In: Proc. of the ACM SIGMOD 2007 Ph.D. Workshop on Innovative Database Research. Beijing, 2007. 45−50. http://idke.ruc.edu.cn/phd-idar2007/idar_pdf/regular-idar28.pdf

[11] Cheng R, Xia Y, Prabhakar S, Shah R, Vitter JS. Efficient indexing methods for probabilistic threshold queries over uncertain data. In: Proc. of the Int'l Conf. on Very Large Data Bases (VLDB). Toronto, 2004. 876−887. http://www.vldb.org/conf/2004/RS22P2.PDF

[12] Brinkhoff T. A framework for generating network based moving objects. Geoinformatica, 2002,2(6):153−180.

丁晓锋(1982－),男,河南禹州人,博士生,主要研究领域为时空数据库,移动计算.

洪亮(1982－),男,博士生,主要研究领域为移动计算,传感器网络.

卢炎生(1949－),男,教授,博士生导师,主要研究领域为数据库技术,软件测试.

魏琼(1980－),女,博士生,主要研究领域为隐私保护,移动计算.

潘鹏(1976－),男,博士,讲师,主要研究领域为空间数据库,移动实时数据库.