时空相点移动对象数据索引 PM-Tree

汤 娜 朱展豪 李晶晶 汤 庸 叶小平
(华南师范大学计算机学院 广州 510613)

摘 要 随着移动定位技术和无线通讯技术发展,移动对象的应用领域越来越广阔.位置随时间而变化的移动对象产生的时空数据具有规模大、多维性、结构复杂和关系复杂等特点.由于移动对象的运动轨迹大多被限定在特定的交通网络中,因此基于路网的移动对象索引成为时空数据索引研究的一个重要应用分支.目前,针对移动对象历史数据的区域查询优化的研究重点是如何提高窗口查询的效率.这类索引通常以同一线路为单位来组织轨迹数据的存储.索引通常采用两层的 R-tree 索引结构,上层的 2D R-tree 用于索引在某个区域内的线路,下层的 2D R-tree 用于索引某个时间段内在这些区域的移动对象.这类索引在处理轨迹信息的时间维度的时候,仅仅是把时间维度等同于空间的维度来进行 R 树维度的扩展.由于 R 树算法不能有效地降低最小限定矩形的空间堆叠问题,尤其是在数据量较大、数据维数增加时表现得更为明显.所以,为了提高路网中移动对象时空信息的存储以及查询的效率,本文则将轨迹信息中的时间数据和空间数据整合起来,提出了一种移动对象数据索引 PM-tree(Phase-point Moving Object Tree).首先运用映射函数把路网中移动对象运动轨迹的二维时空矩形投影成带参数的一维"时空相点",并讨论了时空相点之间的偏序关系,建立了基于相点偏序划分的相点序分枝结构,为索引的建立提供了理论支撑.接着论文以 MON-tree 索引为基础,以相点序分枝结构来改进其下层索引结构,提出了时空相点移动对象数据索引,该索引能完成运动轨迹时空的一体化查询,能避免类 R-tree 索引中最小限定矩形堆叠导致的效率低下的问题,有效地缩小搜索空间.最后论文实现了索引的增量式动态更新管理.通过实验的对比分析,表明 PM-tree 索引不但能有效提高储存空间的利用率,"一次一集合"的查询模式还提高了查询性能.

关键词 时空矩形;路网;移动对象索引;时空映射;相点偏序
中图法分类号 TP311 **DOI 号** 10.11897/SP.J.1016.2021.00579

Temporal-Spatial Phase Point Moving Object Data Indexing: PM-Tree

TANG Na ZHU Zhan-Hao LI Jing-Jing TANG Yong YE Xiao-Ping
(*School of Computer Science, South China Normal University, Guangzhou* 510631)

Abstract With the development of mobile location technology and wireless communication technology, the application of moving objects has exhibited a broad application prospect. As moving objects' position varies as time goes on, the spatial data and temporal data generated continuously by moving objects has the characteristics of multi-dimension, complex data structure, massive data scale and complex data relationship. Usually the trajectory of moving objects was confined to a specific road network, so the index of moving objects based on the road network has become an important branch of the research of temporal spatial data index. At present, for the query optimization of the historical data of moving objects, the research focus on how to improve the efficiency of the window query. Usually such kind of indexes partitioned the

收稿日期:2019-11-16;在线发布日期:2020-11-06.本课题得到国家自然科学基金(61772211,U1811263)、国家重点研发计划(2018AAA0101300)、广东省教育厅创新团队(粤教科函 2018-64/8S0177)、广州市科技计划项目(国际合作)(201807010043)资助.**汤 娜**,博士,副教授,中国计算机学会(CCF)会员,主要研究方向为时态数据库、大数据管理、移动对象数据库和知识发现.E-mail:sinceretn@qq.com.**朱展豪**,硕士研究生,主要研究方向为数据处理与挖掘.**李晶晶**,博士,副教授,主要研究方向为计算智能与优化.**汤 庸**(通信作者),博士,教授,博士生导师,主要研究领域为时态数据管理、社交网络与协同计算.E-mail:ytang@scnu.edu.cn.**叶小平**,博士,教授,主要研究领域为时态数据库、移动对象数据库和知识发现.

trajectory data of moving objects by route, so the trajectory data of moving objects on a specific route was stored together. So this kind of indexes was a two-layer R-tree index structure, the upper layer was a 2D R-tree for indexing the routes in a region, and the lower one was also a 2D R-tree for indexing the moving objects in the ranges of routes in a certain period of time. In the view of these papers, the dimension of time in trajectory information was the same as the dimension of space. So dealing with the dimension of time, this kind of temporal spatial moving object index just extended the temporal dimension to R tree. However, because the algorithm of R tree cannot effectively reduce space overlapping of Minimal Bounding Rectangle (MBR), and it is more serious when the data volume is large and the dimension increases. In order to improve the efficiency of spatial-temporal trajectory information storage and query of moving objects in road network at some interval, this paper integrated temporal data and spatial data, and proposed a temporal-spatial phase point moving object data index (PM-Tree index). Firstly, this paper modeled the spatial trajectory of the moving object at some interval as a set of two-dimensional rectangles, and mapped it into a set of single-dimensional temporal and spatial phase points with parameters. Secondly, the paper discussed the partial order relationship among the temporal and spatial phase points on the phase plane. By partitioning the phase points with the partial order, a Phase-Point Order Branching was constructed. Then, based on Mon-tree index, the paper improved its lower layer of index structure by using the Phase-Point Order Branching structure, and proposed the spatial-temporal phase point moving object data index. This Index can realize the query optimization by the integration of spatial information and temporal information as spatial phase points, also it can avoid the low efficiency caused by MBR overlap in R tree and effectively reduce the search space. Finally, the paper realized the incremental dynamic update management of index. By comparing the performance of PM-tree index with that of Mon-tree index, the experimental results show that the PM-tree index can not only effectively improve the utilization of storage space, but also improve the query performance.

Keywords temporal and spatial rectangle; road network; moving-object indexing; temporal and spatial mapping; phase points partial-order

1 引 言

交通管理、目标跟踪等大量应用中都存在着基于位置的应用,需要处理大量随着时间而演变的空间数据,即移动对象或移动数据[1].移动对象数据库(Moving Object Databases,MOD)技术成为一个热门的研究领域之一.MOD 技术对移动对象的位置及其他相关信息进行表示、存储和管理,提供了对移动对象进行过去、现在查询和对未来预测等操作[2-3].MOD 实现的基本功能包括对移动对象数据的存储、查询和更新[4].移动对象数据具有时间与空间双重特性[5],并具有多维性、结构复杂性、规模海量性和关系复杂性等特点.因而研究移动对象数据索引对提高查询的效率尤为重要.在移动对象存取这个研究领域涌现了一大批工作[6].

根据索引时态信息的不同,移动对象索引可分为移动历史索引和当前及未来位置索引.

当前及未来位置索引研究是针对移动对象位置的不确定范围所做的研究,大多采用函数估值计算,采用的方法有原时空存取方法 PMR-quadtree[7]、空间转换方法[8]、参数化时空存取方法等.例如,TPR-tree[9]通过在 R-tree 上定义时参范围框形以覆盖移动对象集合,但随时间的推移,边界矩形不断扩大导致了矩形间重叠增加,致使查询性能下降,文献[10]改进了 TPR-tree 这个问题.由于基于当前和未来位置的应用往往具有实时性,而且移动对象的位置不断发生变化,所以这一类数据管理研究的其中一个重点在于如何有效地实现数据的更新与存储[11-12].范围查询也逐渐演变为概率范围查询[13-14]、连续范围查询[15]和预测范围查询[16]这三种[17].

而对于移动对象历史数据的查询,经典的查询

包括轨迹查询(某段时间,移动对象的移动轨迹变化)和区域查询两类,区域查询又包括时刻查询(找到时刻 t 时在线路 r 上的移动对象)和窗口查询(找到时刻 t_1 至 t_2 时在线路 r 上的移动对象).针对区域查询优化所建立的索引的研究重点是如何提高时刻查询和窗口查询的效率,优先考虑以同一空间为单位来组织数据和建立索引.有两类建立索引的方法,一类基于一般 R-tree[18]上进行时空扩展,把二维空间和一维时间的时空数据转化成"纯"三维空间数据处理,此时时间维度转化为空间的维度.如 3DR-tree[19]、RT-tree[20]、STR-tree[21].另一类的索引在每个更新时刻上建立一棵版本树.如 MR-tree[22]是在 R-tree 上利用重叠 B-tree 的思想.MV3R-tree[23]是基于多版本 B-tree 的思想,用一棵 MVR-tree 来处理时间戳查询和 3D R-tree 来处理长时间间隔.

由于移动对象位置不断变化,引起了数据的大量更新,在面向轨迹的查询应用中,这类索引在创建时往往优先考虑以同一个移动对象为单位来聚集数据,即同一移动对象的运动轨迹尽量存储在一起[24].如经典轨迹索引 TB-tree[25],它采用了类 R-tree 结构,并在 STR-tree 上进行扩展,把同一轨迹的线段储存在每个叶节点中以保存移动对象的运动轨迹.SETI 索引[26]将静态的空间区域进行非重叠分区,利用分区函数把数据同一轨迹的线段储存在同一分区中[27].

以上的基于移动历史的索引研究主要是针对移动轨迹没有任何限定的情况下所做的研究.在许多现实场景中,移动对象的运动轨迹并不是杂乱无章的,而是被限制在特定的或者具有一定规律的网络上,例如高速路上的汽车.因而它们的位置信息可以借助网络上固定线路的相对位置来表示.因而相比轨迹无限定的移动对象查询,基于路网的历史查询的复杂度相对降低[23].

这一类索引通常是一个两层的索引结构.均采用 R-tree索引结构或是其变种结构进行存储.如Frentzos 提出的路网移动对象经典索引 FNR-tree[28],它是一个两层混合索引结构:上层是一棵 2D R-tree,用于索引道路网络的路段;下层是 1D R-tree 森林,用于索引路段中运动的移动对象.FNR-tree 具有良好的窗口查询性能,但对于时间片查询和历史轨迹查询,则需要遍历整个 1D R-tree 森林.FNR-tree 还假定移动对象在路网中速度不变.但在现实的应用场景中往往对象的移动不是以同一个速度进行.郭景峰等人提出了 FNR$^+$-tree 索引结构[29],它在 FNR-tree 的基础上增添了一个哈希结构来储存对象的历史轨迹,从而改善了 FNR-tree 在轨迹查询上的效率.Pfoser 等人提出用 Hilbert 曲线把复杂的三维空间转化成用 R-tree 表示的低维子空间[30],虽然查询处理较 FNR-tree 要复杂,但可以把移动对象的运动方式表示得更具体.De Almeida 等人提出了具有两层结构的 MON-tree[31],上层是一棵用于索引线路/线段的 2D R-tree,下层是一个用于索引指定线路中移动对象位置和时间信息的 2DR-tree 森林.在道路表达上 MON-tree 提供了两种表达方式,一种是以道路作为基本元素,另一种是将道路表示为折线段,以折线段作为基本元素.当道路长度较大时,MON-tree 会产生大量的死空间,查询效率相对降低.实验表明,相比较于 FNR-tree,MON-tree 具有更好的性能,且 MON-tree 的两种不同表达方式索引中,基于道路的 MON-tree 的查询效率更高.

一些研究人员还针对一些混合模式进行研究:(1)同时处理移动对象过去、当前以及未来位置信息的索引模型 AMH[32]、R^{PPF}-tree[33]、PCI[34].这类索引往往是移动历史索引与当前未来位置两类索引技术的整合.但由于两种索引结构的更新效率是不同的,所以针对两种不同的查询,需要有两种数据结构来分别存储数据,并建立两种数据结构的联系,将现在的数据不断转化成过去的数据;(2)为了利用多核处理器的并行性以满足大数据处理的需求,提出了基于内存和磁盘的轨迹索引[35-36],这类工作的挑战是如何处理查询和更新上锁之间的争用.

本文针对在限定路网上的历史区域查询,提出一种基于时空相点的路网移动对象数据索引技术 PM-tree,目的在于提高路网中移动对象时空信息的存储以及历史区域查询的效率.本文首先将路网中的移动对象轨迹信息建模为时空数据矩形集合,通过映射函数将其投影成带参数的一维"时空相点"数据.其次,讨论了时空相点集合上基于相点序分支结构的数据结构.最后,构建了基于时空相点序分枝结构的路网移动对象索引,该索引改进了 MON-tree 的下层用于索引指定线路中移动对象位置和时间信息的 2DR-tree 森林,采用相点序分枝结构实现了指定线路中移动对象位置和时间信息的一体化存储和查询,同时可以避免最小限定矩形大量重叠导致的查询效率低下问题;最后讨论了该索引的查询和增量式更新算法.

本文主要贡献是:把二维的时空数据矩形通过映射函数投影成带参数的一维"时空相点"数据,实现了时空数据的降维以及时空数据的一体化查询;通过研究时空相点之间的关系提出相点序分支结

构,该结构可以有效缩减区域查询的搜索范围;构建了基于相点序分支结构的路网移动对象数据索引PM-tree,提出了"一次一集合"的查询模式和动态更新算法.

本文第2节基于路网的移动对象数据模型和时空相点集合的数据关系,提出相点序分支数据结构;第3节讨论以相点序分支数据结构为基础建立的路网移动对象索引模式PM-tree,并研究PM-tree的数据查询和更新算法;第4节是相应的实验仿真;第5节是对本文的总结与展望.

2 时空数据结构

线路 R 由一组固定的有序线段 $\{\langle a_0,a_l\rangle,\langle a_1,a_2\rangle,\cdots,\langle a_{n-2},a_{n-1}\rangle,\langle a_{n-1},a_n\rangle\}$ 组成,其中 $a_i(0\leq i\leq n)$ 为二维平面线段的点,a_0 和 a_n 分别为线路始点和终点.R 上点 a_i 的位置用 a_i 关于 a_0 的距离参数 $D_i=D(R,a_i)$ 表示,当 $a_i=a_0$ 时,$D(R,a_i)=0$;当 $a_i\neq a_0$ 时,$D(R,a_i)=D(R,a_{i-1})+d(a_{i-1},a_i)$,其中 $d(a_{i-1},a_i)$ 是 a_{i-1} 到 a_i 之间的欧式距离($1\leq i\leq n$).一条线路对应了地图中的一条道路.路网是由一组有序线路集合 $\{r_1,r_2,\cdots,r_i,\cdots,r_m\}$ 连接组成的图.移动对象 m 在线路 r_i 上运动所产生的运动轨迹可以一系列点 $\langle M_0,M_1,\cdots,M_n\rangle$ 来表示,$M_{i-1}=(d_{i-1},t_{i-1})$ 表示在时刻 t_{i-1} 位于点 M_{i-1},距离线路 r_i 的始点的距离为 d_{i-1},其中 $d_{i-1}=D(R,a_j)+d(a_j,M_{i-1})$,其中 a_j 是从道路的初始点 a_0 到 M_{i-1} 之间最靠近 M_{i-1} 的点,$d(a_{i-1},a_i)$ 是 a_j 到 M_{i-1} 之间的欧式距离.下一个时间点 t_i 的轨迹则为 $M_i=(d_i,t_i)$,即点 M_i 距离线路 r_i 的始点的距离为 d_i,相邻两个结点 M_{i-1} 和 M_i 组成一个折线段 $seg(M_{i-1},M_i)$.

定义 1. 时空数据矩形 TSDR(Temporal-Spatial Data Rectangle).移动对象 m 运动轨迹上的折线段 $seg(M_{i-1},M_i)$ 可用一个平行于坐标轴的时空数据矩形 $S_i=(d_{i-1},d_i;t_{i-1},t_i)$ 来表示,其中 $d_{i-1}\leq d_i\wedge t_{i-1}\leq t_i$,即 (d_{i-1},t_{i-1}) 和 (d_i,t_i) 分别表示 S_i 左下和右上顶点坐标,如图1所示.

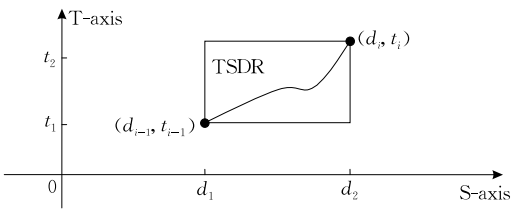

图1 时空矩形 TSDR

由上述定义可得,移动对象 m 在线路 r 上的运动轨迹 $\langle M_0,M_1,\cdots,M_n\rangle$ 可以建模为时空数据矩形 TSDR 序列 $\langle S_1,S_2,\cdots,S_n\rangle$,其中 $S_i=(d_{i-1},d_i;t_{i-1},t_i)$.

2.1 时空相点映射

TSDR 作为一个二维时空矩形,若直接对其进行数据操作,处理效率较低.本小节基于 TSDR 数据的固有特性运用数学映射方法把 TSDR 矩形映射成时空相点,从而实现提高移动对象运动信息的处理效率.

首先将时空矩形 TSDR 的左下和右上端点垂直投影到相点轴(Phase-axis)上,得到投影线段 $[a,b]$.参见图2所示.a 和 b 的值分别为从原点出发沿相点轴到点 a 及点 b 的距离.距离值的计算参见图3.$[a,b]$ 可以记作相平面中的时空相点坐标 (a,b) 在相点轴上的线段或是相点坐标 (a,b) 对应的区间.

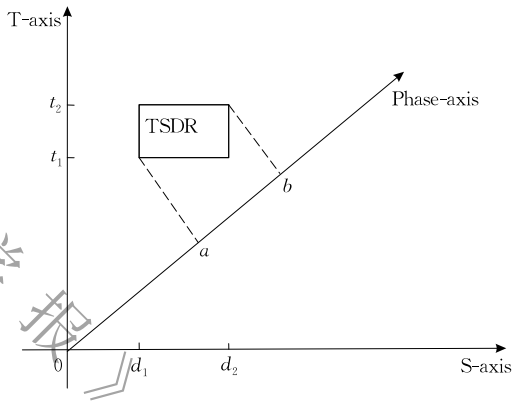

图2 时空矩形 TSDR 映射为线段

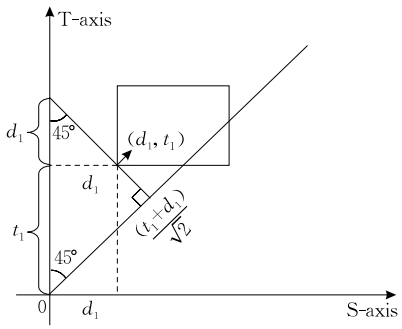

图3 点 a 值的计算

定义 2. 时空相点映射 Phase Points Mapping.相点映射定义如下:

$$S=(d_1,d_2;t_1,t_2)\rightarrow P=((a,b),d_1,d_2,t_1,t_2,O_k)$$

$$a=d_1\times\sqrt{2}+\frac{t_1-d_1}{\sqrt{2}}=\frac{t_1+d_1}{\sqrt{2}},$$

$$b=d_2\times\sqrt{2}+\frac{t_2-d_2}{\sqrt{2}}=\frac{t_2+d_2}{\sqrt{2}},$$

其中，P 称为时空数据矩形 TSDR 对应的时空相点(Temporal-Spatial Phase Point，TSPP)，(a,b)称为 P 的时空相点坐标，d_1，t_1，d_2，t_2称为 P 的时空判定参数，O_k为相点所属的移动对象. TSDR 与相点 P 的映射关系如图 4 所示.

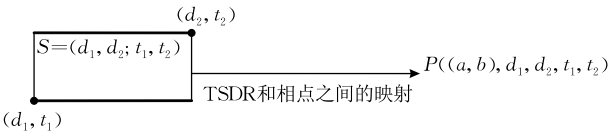

图 4　TSDR 与相点 P 的映射关系

为了简化计算和方便显示，把 a、b 均放大$\sqrt{2}$倍，则有 $a=t_1+d_1$，$b=t_2+d_2$.

定理 1.　TSDR 相交关系与相点相交等价性. 设 $TSDR_i$ 和 $TSDR_j$ 所对应的时空相点分别为 $P_i((a_i,b_i),d_{i1},d_{i2},t_{i1},t_{i2},O_k)$和 $P_j((a_j,b_j),d_{j1},d_{j2},t_{j1},t_{j2},O_m)$，由时空相点和其对应的区间的概念可以得到：

$$TSDR_i \cap TSDR_j \neq \varnothing \Rightarrow [a_i,b_i] \cap [a_j,b_j] \neq \varnothing.$$

证明.　$TSDR_i \cap TSDR_j \neq \varnothing$意味着两个矩形一定存在着相交的面积 S，此面积可以是点、线、面，则一定此面积投影在相点轴上至少有一个点. 即 $[a_i,b_i] \cap [a_j,b_j] \neq \varnothing$.　证毕.

定理 2.　TSDR 不相交关系的相点坐标判定. 设 $TSDR_i$ 和 $TSDR_j$ 所对应的时空相点分别为 $P_i((a_i,b_i),d_{i1},d_{i2},t_{i1},t_{i2},O_k)$和 $P_j((a_j,b_j),d_{j1},d_{j2},t_{j1},t_{j2},O_m)$，则有：

$$[a_i,b_i] \cap [a_j,b_j] = \varnothing \Rightarrow TSDR_i \cap TSDR_j = \varnothing.$$

证明.　假设$[a_i,b_i] \cap [a_j,b_j] = \varnothing$时 $TSDR_i \cap TSDR_j \neq \varnothing$. 由于 $TSDR_i \cap TSDR_j \neq \varnothing$意味着两个矩形一定存在着相交的面积 S，此面积可以是点、线、面，则此面积投影在相点轴至少有一个点. 即 $[a_i,b_i] \cap [a_j,b_j] \neq \varnothing$，则与假设矛盾. 所以可以推出 $[a_i,b_i] \cap [a_j,b_j] = \varnothing \Rightarrow TSDR_i \cap TSDR_j = \varnothing$. 证毕.

为叙述方便，下文把相点 $P_i((a_i,b_i),d_{i1},d_{i2},t_{i1},t_{i2},O_k)$与 $P_j((a_j,b_j),d_{j1},d_{j2},t_{j1},t_{j2},O_m)$对应的区间$[a_i,b_i] \cap [a_j,b_j]$记为 $P_i \cap P_j$.

定义 3.　移动对象时空相点模型(Moving Object Phase Point Model). 移动对象在线路 r_i上的运动轨迹数据 TSDR 序列可建模为相平面上的一个时空相点 TSPP 序列$\langle P_1,P_2,\cdots,P_n\rangle$.

例 1.　对于移动对象 $O_1,O_2,\cdots,O_{13}$在线路 r_i 上运动产生的数据可以建模如表 1 所示.

表 1　移动对象相点数据模型

	Temporal-Spatial Data Rectangle	Temporal-Spatial Phase Point
O_1	(0,2;0,2)(2,7;2,6)	((0,4),0,2,0,2,O_1), ((4,13),2,7,2,6,O_1)
O_2	(0,2;3,5)(2,7;5,8)	((3,7),0,2,3,5,O_2), ((7,15),2,7,5,8,O_2)
O_3	(0,5;5,9)	((5,14),0,5,5,9,O_3)
O_4	(2,5;5,9)	((7,14),2,5,5,9,O_4)
O_5	(2,7;3,8)	((5,15),2,7,3,8,O_5)
O_6	(2,5;3,6)(5,7;6,7)	((5,11),2,5,3,6,O_6), ((11,14),5,7,6,7,O_6)
O_7	(0,5;2,5)(5,7;5,7)	((2,10),0,5,2,5,O_7), ((10,14),5,7,5,7,O_7)
O_8	(5,7;0,4)	((5,11),5,7,0,4,O_8)
O_9	(0,2;7,9)	((7,11),0,2,7,9,O_9)
O_{10}	(0,2;2,3)(2,5;3,7)	((2,5),0,2,2,3,O_{10}), ((5,12),2,5,3,7,O_{10})
O_{11}	(2,5;2,4)(0,2;4,7)	((4,9),2,5,2,4,O_{11}), ((4,9),0,2,4,7,O_{11})
O_{12}	(0,2;2,4)(2,7;4,8)	((2,6),0,2,2,4,O_{12}), ((6,15),2,7,4,8,O_{12})
O_{13}	(2,5;6,8)	((8,13),2,5,6,8,O_{13})

2.2　时空相点序分枝结构

定义 4(相点偏序关系).　设 Σ 为相点集合，对于 $P_i,P_j \in \Sigma$，若 $P_i \subseteq P_j$，即$(a_j \Leftarrow a_i) \wedge (b_i \Leftarrow b_j)$，则称 P_i与 P_j具有关系“$\lesssim$”，记为 $P_i \lesssim P_j$，“$\lesssim$”是 Σ 集合上满足自反性、反对称和传递性的偏序关系.

定义 5.　相点序分枝(Phase-Point Order Branch，PPOB). 对于移动对象的相点数据集合 Σ，其对应的偏序划分记为 $P(\Sigma)=\langle L_1,L_2,\cdots,L_m\rangle$，称 L_i为 $P(\Sigma)$中相点序分支(Phase Point Order Branch，PPOB). 每一个 L_i是满足“$\lesssim$”的相点的偏序集合，是 Σ 偏序划分中的一个全序分枝，即 L_i中的每一个相点之间都满足“$\lesssim$”的全序关系，且每一个相点属于且仅属于一个 L_i.

例 2.　例 1 中移动对象的相点集合 Σ，由算法 1 中的相点偏序划分算法 1 中的函数 *GENERATE_PMTreeStr* 可得：

$L_1=\langle(0,4)\rangle$，

$L_2=\langle(2,10)(2,6)(2,5)\rangle$，

$L_3=\langle(3,7)\rangle$，

$L_4=\langle(4,13)(4,9)\rangle$，

$L_5=\langle(5,15)(5,14)(5,12)(5,11)(7,11)\rangle$，

$L_6=\langle(6,15)(7,15)(7,14)(8,13)\rangle$，

$L_7=\langle(10,14)(11,14)\rangle$.

最终可得相点偏序划分 $P(\Sigma)=\langle L_1,L_2,L_3,$

L_4,L_5,L_6,L_7〉.从本例起,为了更清晰地描述相点,采用相点坐标代表相点,省略了相点的时空判定参数和相点对应的移动对象 id.

定理 3. 相点序分枝相交定理.设有相点序分枝 $L_i=\langle p_1,p_2,\cdots,p_j,\cdots,p_{n-1},p_n\rangle$,对于任意相点 P,若有 $p_j\cap P\neq\varnothing$,则 L_i 中所有位于 p_j 前的相点均与 P 相交,即($p_1\cap P\neq\varnothing\wedge p_2\cap P\neq\varnothing\wedge\cdots\wedge p_{j-1}\cap P\neq\varnothing$).若有 $p_j\cap P=\varnothing$,则 L_i 中所有位于 P_j 后的相点与 P 均不相交,即($p_{j+1}\cap P=\varnothing\wedge\cdots\wedge p_{n-1}\cap P=\varnothing\wedge p_n\cap P=\varnothing$).

证明. 由定义 4 和定义 5 可得,对于 L_i 中的元素 p_k 和 p_j,若 $k<j$,则必有 $a_k\Leftarrow a_j\wedge b_j\Leftarrow b_k$.现假设 p_j 与相点 P 相交,即 $[a,b]\cap[a_j,b_j]\neq\varnothing$.由 $[a,b]\cap[a_j,b_j]\neq\varnothing\Leftrightarrow a_j\Leftarrow b\wedge a\Leftarrow b_j$,又 $a_k\Leftarrow a_j\wedge b_j\Leftarrow b_k$,则有 $a_k\Leftarrow b\wedge a\Leftarrow b_k$,因此 $p_k\cap P\neq\varnothing$.同理可得当 $k>j$ 时,若 $p_j\cap P=\varnothing$,则 $p_k\cap P=\varnothing$. 证毕.

例 3. 对于例 2 中的 $L_6=\langle(5,15)(5,14)(5,12)(5,11)(7,11)\rangle$,设相点 P 的坐标为(12,15).由 $[5,12]\cap[12,15]\neq\varnothing$,可得 $[5,15]\cap[12,15]\neq\varnothing$ 且 $[5,14]\cap[12,15]\neq\varnothing$;由 $[5,11]\cap[12,15]=\varnothing$,可得 $[7,11]\cap[12,15]=\varnothing$.

3 移动对象数据索引

3.1 时空相点移动对象数据 PM-tree 索引

PM-tree 索引(Phase-point Moving Object Tree Index)包括两层结构的建立,上层结构是一棵 R-tree.使用地图数据的线路单元的最小限定框 MBR(Minimal Bounding Rectangle)作为上层索引结构的构建单元,将地图的所有线路的最小限定的框建立的 R-tree 作为上层结构用于索引线路.如图 5 所示.下层是移动对象轨迹的索引结构,是由一个包含线路 Id 和该线路对应的 PM-tree 结构的哈希映射结构组成,这个哈希映射包含了每一条线路对应的 PM-tree 结构,哈希表中的每一个 PM-tree 结构负责索引其对应线路下的所有移动对象的轨迹数据.下面着重讨论 PM-tree 结构.

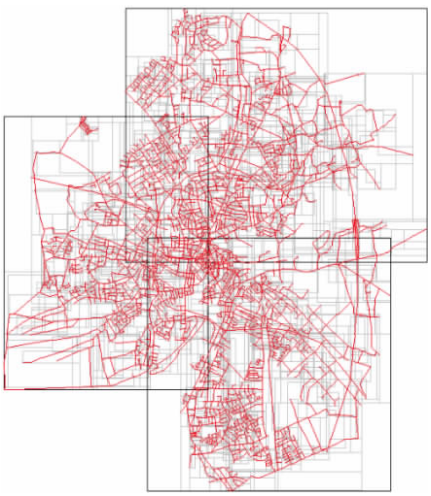

图 5 由线路的最小限定框构建的 R-Tree

定义 6. PM-Tree 结构 *PMTreeStr*.PM-tree 结构 *PMTreeStr* 是由 Root-level、Max-level、PPOB-level 和 O-level 构成的四层树形结构.如图 6 所示.

(1) Root-level.逻辑层,表示数据操作的入口.

(2) Max-level.由 PPOB-level 中各个 PPOB 中的最大、最小元 max(L_i) 和 min(L_i) 组成,且 max(L_i)在该层的排列顺序与 L_i 在算法 1 中的获取顺序相对应.

(3) PPOB-level.由各个 max(L_i) 相对应的 PPOB 构成,且 PPOB 中的每个相点均带有一个指向 O-level 对象的指针.

(4) O-level.由每个相点对应的移动对象构成,用于存储移动对象的具体信息.

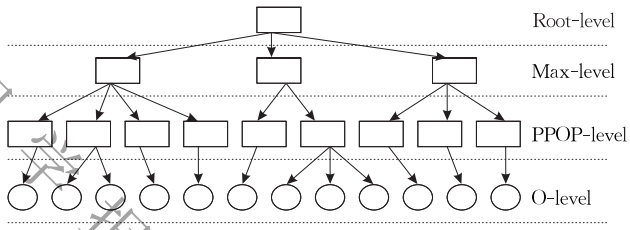

图 6 PM-tree 结构

例 4. 例 1 的移动对象运动轨迹数据所构成的 PM-tree 结构如图 7 所示.

算法 1. 索引建立算法 *build_tree*.

输入:*ED*(*EdgeData*)地图数据,*TDs*(*TSDRDatas*)各线路上移动对象的轨迹数据

输出:PM-tree 索引,其中 *HM*(*HashMap*)为将上层索引结构中的线路映射到到下层 PM-tree 结构的哈希结构

```
1.  FUNCTION build_tree(ED,TDs)
2.    /* 建上层索引结构 */
3.    FOR 每一条线路对应的 MBR∈ED DO
4.      RTree.Insert(MBR)
5.    END FOR
6.    /* 建下层结构 */
7.    FOR 每一条线路下的 TSDR 集合 TDL∈TDs DO
8.      GENERATE_PMTreeStr(TDL)
9.      HM add PMTreeStr
10.   END FOR
11. END FUNCTION
12. /* 相点偏序划分及 PMTreeStr 结构的构建算法 */
```

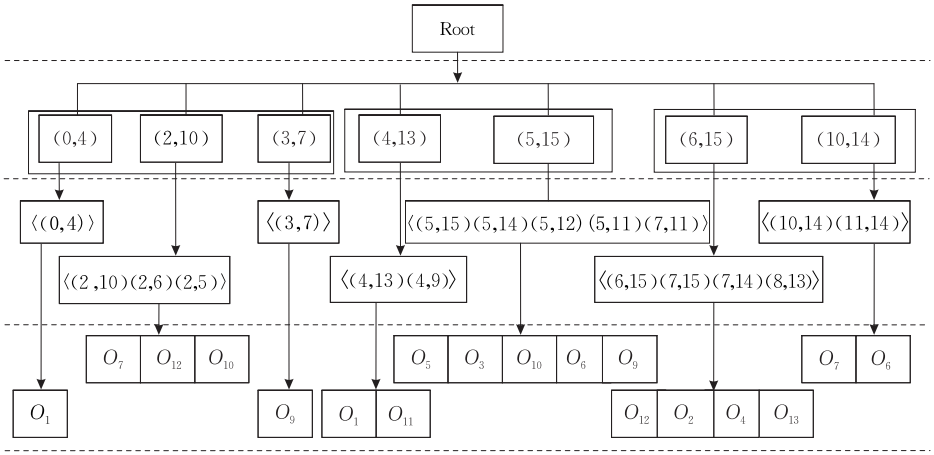

图 7　PM-tree 结构 *PMTreeStr* 实例

13. FUNCTION *GENERATE_PMTreeStr*(*TDL*)
14. /* 生成 *TDL* 对应的相点集合 Σ 的偏序划分，记为 $P(\Sigma)=\langle L_1,L_2,\cdots,L_m\rangle(1\leq i\leq m)$，并构建对应的 *PMTreeStr* */
15. 生成 *PMTreeStr* 的 Root-level
16. FOR ALL *TD* in *TDL* Do 计算其对应的相点，并按照 a 值的升序分成若干列，a 值相同的为同一列，同一列的点按照 b 按降序排列，形成相点平面上的相点集合 Σ；
17. 从 Σ“最左上方”点（即 a 最小，b 最大）开始，$P=(a_i,b_j)$，点 $P(a_i,b_j)$ 所在的列记为 $col(a_i)$. 列 $col(a_i)$ 右边最近的邻列记为 $RightC(a_i)$. 在列 $col(a_i)$ 中，点 $P=(a_i,b_j)$ 的列直接后继节点记为 $RowSucP(P)$. 若列 $col(a_P)$，其列值大于 $col(a_i)$，则该列中第一个满足 $b_q\Leftarrow b_j$ 的节点 $K(a_p,b_q)$ 称为 P 的在列 $col(a_P)$ 中的后继节点，记为 $ColSucP(col(a_P),P)$；$n=0$.
18. $n=n+1$，建立列表 L_n，且 $\max(L_n)=[P.a,P.b]$，
19. //$\max(L_n)$ 为 L_n 中的“最大元”，即“首”元素，$\min(L_n)$ 为 L_n“最小元”，即“尾”元素.
20. 将 P 加入 L_n，$\Sigma=\Sigma-\{P\}$；$col=col(P.a)$；$\min(L_n)=[P.a,P.b]$
21. 若 P 存在着列直接后继节点 $RowSucP(P)$，则令 $P=RowSucP(P)$，返回步骤 20；
22. 若 $RightC(col)$ 存在，则 $col=RightC(col)$，查找 $ColSucP(col,P)$，如果存在该节点，则令 $P=ColSucP(col,P)$，返回步骤 20；否则返回步骤 22；
23. 在 *PMTreeStr* 的 Max-level 增加一个新节点 $(\min(L_n),\max(L_n))$，并对该节点构建子树，PPOB-level 层为 L_n，O-level 层为 L_n 中每个相点所属的对象 id.
24. 若 $\Sigma=\varnothing$，退出算法；否则，查找 Σ“最左上方”点 K，并令 $P=K$，返回步骤 18.

相点偏序划分算法的平均时间复杂度为 $O(n\log(n))$.

3.2　数据操作

基于 PM-tree 索引的数据操作主要分为数据查询和更新两种操作.

路网移动对象的查询类型一般分为窗口查询、时间片查询和点查询. 窗口查询是指给定一个时间间隔和一个空间矩形区域，查找在该时间间隔中位于给定空间矩形区域上的移动对象. 而时间片查询和点查询均为窗口查询的特殊情况，因此小节仅讨论 PM-tree 索引的窗口查询操作.

PM-tree 索引的查询算法，需要下述定理支持.

定理 4.　TSDR 相交关系的相点参数判定. 设 $S_i=(d_{i1},t_{i1};d_{i2},t_{i2})\rightarrow P_i=((a_i,b_i),d_{i1},d_{i2},t_{i1},t_{i2},O_k)$，$S_j=(d_{j1},t_{j1};d_{j2},t_{j2})\rightarrow P_j=((a_j,b_j),d_{j1},d_{j2},t_{j1},t_{j2})$，假设 $d_{i1}\leq d_{j1}$，则 $S_i\cap S_j\neq\varnothing\Leftrightarrow(d_{j1}\leq d_{i2}\wedge t_{j1}\leq t_{i2})\wedge(d_{i1}\leq d_{j2}\wedge t_{i1}\leq t_{j2})$.

证明.　必要性. 首先将两矩形投影在 X 轴，由于 $S_i\cap S_j\neq\varnothing$，意味着两个矩形有交集，即线段 $[d_{i1},d_{i2}]$ 与线段 $[d_{j1},d_{j2}]$ 必须有交集. 两线段有交集要满足的条件为 $d_{j1}\leq d_{i2}\wedge d_{i1}\leq d_{j2}$.

再将两矩形投影在 Y 轴，由于 $S_i\cap S_j\neq\varnothing$，意味着两个矩形有交集，即线段 $[t_{i1},t_{i2}]$ 与线段 $[t_{j1},t_{j2}]$ 必须有交集. 两线段有交集要满足的条件为 $t_{j1}\leq t_{i2}\wedge t_{i1}\leq t_{j2}$. 即由 $S_i\cap S_j\neq\varnothing\Rightarrow(d_{j1}\leq d_{i2}\wedge t_{j1}\leq t_{i2})\wedge(d_{i1}\leq d_{j2}\wedge t_{i1}\leq t_{j2})$.

充分性. 已知 $(d_{j1}\leq d_{i2}\wedge t_{j1}\leq t_{i2})\wedge(d_{i1}\leq d_{j2}\wedge t_{i1}\leq t_{j2})$，假设 $S_i\cap S_j=\varnothing$，及两个矩形没有交集，如果 $d_{i1}<d_{j1}$，则必有 $d_{i1}<d_{i2}<d_{j1}<d_{j2}$；与前提中的 $d_{j1}\leq d_{i2}$ 矛盾. 如果 $d_{i1}>d_{j1}$，则必有 $d_{j1}<d_{j2}<d_{i1}<d_{i2}$ 与前提中的 $d_{i1}\leq d_{j2}$ 矛盾矛盾. 即命题 $(d_{j1}\leq d_{i2}\wedge t_{j1}\leq t_{i2})\wedge(d_{i1}\leq d_{j2}\wedge t_{i1}\leq t_{j2})\Rightarrow S_i\cap S_j\neq\varnothing$ 得证. 证毕.

当 $(d_{i1}=d_{j2})\vee(d_{j1}=d_{i2})\vee(t_{j1}=t_{i2})\vee(t_{i1}=t_{j2})$ 时，S_i 与 S_j 只有边相交，不满足窗口查询的定义，所以窗口查询的相交判断定理可以简化为

$S_i \cap S_j \neq \varnothing \Leftrightarrow (d_{j1} < d_{i2} \wedge t_{j1} < t_{i2}) \wedge (d_{i1} < d_{j2} \wedge t_{i1} < t_{j2})$.

结合定理 2(TSDR 不相交关系的相点坐标判定)和定理 4(TSDR 相交关系的相点参数判定)可以得到基于时空相点移动对象数据索引 PM-tree 的窗口查询算法.

3.2.1 PM-tree 索引的窗口查询

窗口查询算法,即设给定查询窗口 $Q=\langle x_1, x_2, y_1, y_2; t_1, t_2\rangle$,查找在 t_1 到 t_2 时间中位于区域(x_1, x_2, y_1, y_2)内的移动对象. 主要包含四个步骤:(1)找在索引的上层 R-树中找到包含区域(x_1, x_2, y_1, y_2)的所有最小限定矩形 MBR,取出包含在 MBR 中的所有线路;(2)计算每条线路对应的查询窗口;(3)查询与相点序分支相交的候选集合;(4)精确化查询.

算法 2. 窗口查询算法 *QueryAlgorithms*.

输入:*QW*(*QueryWindow*)查询窗口,*t*1,*t*2 时间跨度

输出:*MO*(*MovingObject*)查询到的移动对象

```
1.  /* 查询算法 */
2.  FUNCTION QueryAlgorithms(QW,t1,t2)
3.     /* ①通过上层 R 索引查询到 QW 包含的线路 */
4.     Edges←RTree.Search(QW)
5.     FOR Edge∈Edges DO
6.        /* ②调用算法 3 计算某线路对应查询窗口 */
7.        currQW←ComputeQW(Edge,QW,t1,t2)
8.        /* 获得该线路下的 PMTreeStr/
9.        PMTreeStr←HashMap.get(Edge.eId)
10.       /* ③查询相点序分支相交的候选集合 */
11.       Result←PhaseIntersect(PMTreeStr,currQW)
12.       /* ④精确化查询 */
          MO add GetResult(Result,currQW)
13.    END FOR
14. END FUNCTION
```

第一,对于给定的查询窗口,先在上层结构 R-tree 进行线路的索引,如图 8 所示.

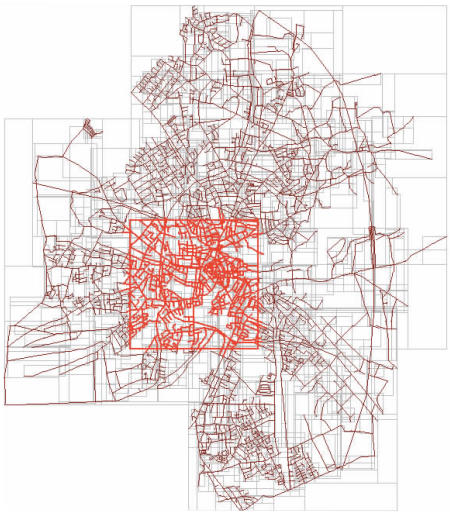

图 8 上层线路索引实例

第二,由于上层的查询窗口是绝对地理坐标,且不包含时间查询参数. 而下层的查询窗口则是取相对于线路起点的距离参数 d,以及时间参数. 所以需要对上下层查询窗口进行转换.

窗口转换存在两种情况:(1)查询窗口与线路最小限定框相交,此时相交区域作为下层查询窗口的空间参数;(2)查询窗口包含线路的最小限定框,此线路的最小限定框作为下层查询窗口.

算法 3. 窗口变换算法 *ComputeQW*.

输入:*Edge* 线路,*QW*(*QueryWindow*)查询窗口,*t*1,*t*2 时间跨度

输出:*QueryMap* 变换后的查询窗口

```
1.  /* 计算查询窗口 */
2.  FUNCTION ComputeQW(Edge,QW,t1,t2)
3.     IF QW 包含了线路对应的 MBR THEN
4.        S1←计算空间参数 S1
5.        S2←计算空间参数 S2
6.     END IF
7.     IF QW 与线路的 MBR 相交 THEN
8.        IA←计算相交区域
9.        S1←计算空间参数 S1
10.       S2←计算空间参数 S2
11.    END IF
12.    QueryMap add 查询窗口(S1,S2,t1,t2)
13.    RETURN QueryMap
14. END FUNCTION
```

第二,查询候选结果集,在 PM-tree 中,对其有序的每一个相点序分支进行遍历,将与查询窗口有交集的相点序分支对应的相序加入候选结果集. 如图 9 所示,C,D,E,F,G,H 都应加入候选结果集,因为它们所在的相点序分支与查询窗口相序有交集,而 A,B 所在的相点序分支与查询窗口相序并没有交集,所以 A,B 不加入候选结果集.

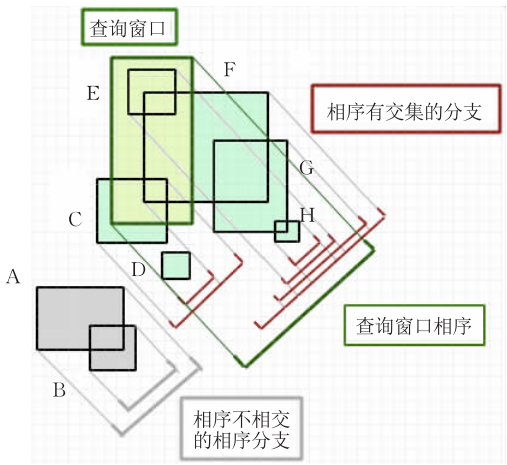

图 9 查询候选结果集

算法 4.　查询候选结果集算法 *PhaseIntersect*.

输入：*PMTreeStr*：当前线路的 PM-Tree 结构，*cQW* 当前线路查询窗口

输出：*PIR*(*Phase_Intersect_Result*)候选结果集

```
1.  FUNCTION PhaseIntersect(PMTreeStr,cQW)
2.    将 cQW 映射为时空相点 q,
3.    FOR PPOB∈PMTreeStr DO
4.      IF q 与 PPOB 最大元都不相交 THEN
5.        放弃该分支,结束本次循环;
6.      IF q 与 PPOB 最小元都相交 THEN
7.        PIR add PPOB 的所有相点,结束本次循环;
8.        index←二分查找第一个不相交的相点的位
          置,即该相点的相点坐标中的 a 值>相点 P
          的相点坐标中的 b 值
9.      END IF
10.     PIR add PPOB 的第一个相点到第 index−1 个
11.   END FOR
12. END FUNCTION
```

假设有交集的 *PPOB* 共有 m 个，每个 *PPOB* 中最多有 n 个元素，则查询算法的时间复杂度为 $O(m\log n)$.

第四，精确化候选结果集，由于候选结果集可能会存在不与查询窗口相交的 TSDR(时空相序矩形)，所以在候选结果集中还需要进一步精确化查询结果，对于候选结果集中的每一条记录进行判断，使得最终结果集的每一条记录都与查询窗口有交集.

算法 5.　精确化查询算法 *GetResult*.

输入：*PIR* 候选结集，*cQW* 当前线路查询窗口

输出：*MOs*(*MovingObject*)查询到的移动对象

```
1.  FUNCTION GetResult(PIR,cQW)
2.    FOR 每一个相点 P∈PIR DO
3.      IF P 的时空判定参数与 cQW 有交集则
4.        MOs add P 对应的移动对象 id
5.      END IF
6.    END FOR
7.    RETURN MOs
8.  END FUNCTION
```

3.2.2　数据更新

数据更新通常包括数据删除和数据插入. 由于 PM-tree 着眼于索引移动对象在路网中的运动信息，而移动对象运动信息的更新主要考虑新数据插入，因此本文仅研究基于数据插入的索引更新算法.

在索引结构中插入节点主要考虑以下两种情形：第一种情形，插入已有线路下移动对象 O 的运动信息，即哈希映射中已经包含该线路的 PM-tree. 第二种情形，插入一个新的线路下移动对象 O 的运动信息 (x,y,t)，此时，哈希映射中并不包含该线路下对应的 PM-tree. 则要现在上层 R 树中插入该线路的节点，并建立该节点新的 PM-tree 结构，新建一个 *PPOB*.

算法 6.　索引插入算法 *InsertRecord*.

输入：*iTD*(*insertTSDR*)用于插入的时空矩形

```
1.  FUNCTION InsertRecord (iTD)
2.    映射 iTD 对应的相点 P
3.    IF 哈希表中已存在该线路 THEN
4.      PMTreeStr←获取该线路对应的 PMTreeStr
5.      PMTreeStrInsert(PMTreeStr,P)
6.      更新哈希表
7.    ELSE
8.      /* 新线路下插入 */
9.      新建一个 PPOB
10.     PMTreeStr.insert(PPOB)
11.     HashMap add PMTreeStr
12.   END IF
13. END FUNCTION
```

定义 7.　相序错位 Phase-Displace. 如果一个相点 P_i 所对应的区间 $[a_i,b_i]$ 与某一个 $PPOB_k$ 不满足以下三个条件，(1) P_i 包含 $PPOB_k$ 的最大元 $\max(PPOB_k)$；(2) P_i 被 $PPOB_k$ 的最小元 $\min(PPOB_k)$ 所包含；(3) P_i 与 $PPOB_k$ 中某一相邻的两个相点 P_j 与 P_{j+1} 对应的区间 $[a_j,b_j][a_{j+1},b_{j+1}]$ 存在被包含与包含的关系(即：不存在 P_j 与 P_{j+1} 使得 $(a_j\le a_i$ 且 $b_j\ge b_i)$ 且 $(a_{j+1}\ge a_i$ 且 $b_{j+1}\le b_i)$)则称该相点 P_i 与相点序分支 $PPOB_k$ 错位.

算法 7.　*PPOB* 插入算法 *PMTreeStrInsert*.

输入：*P* 插入的相点，*PMTreeStr* 用于插入相点的 PM-Tree 结构

```
1.  FUNCTION PMTreeStrInsert(PMTreeStr,P)
2.    FOR PPOB∈PMTreeStr DO
3.      IF P 的区间与 PPOB 错位
        THEN
4.        IF 该 PPOB 是最后一个分支
5.        THEN 建立一个 PPOB 插入 P 退出算法
6.        ELSE 退出本次循环
7.      ELSE
8.        插入到 PPOB 的相应位置,程序结束
9.    END FOR
10. END FUNCTION
```

设 *PMTreeStr* 中共有 k 个分枝，每个分枝中共有 m 个元素，则算法 7 的平均时间复杂度均为 $O(k\log m)$.

4　数据仿真与评估

本节设计了对比仿真实验来评估 PM-tree 的基

本性能,实验采用的比较对象为 MON-tree. 实验使用文献[2]提出的基于路网的移动对象生成器. 我们使用的路网地图数据是下载自 Thomas Brinkhoff 移动生成器网站①上的奥尔登堡. 基于路网数据的移动对象生成器如图 10 所示.

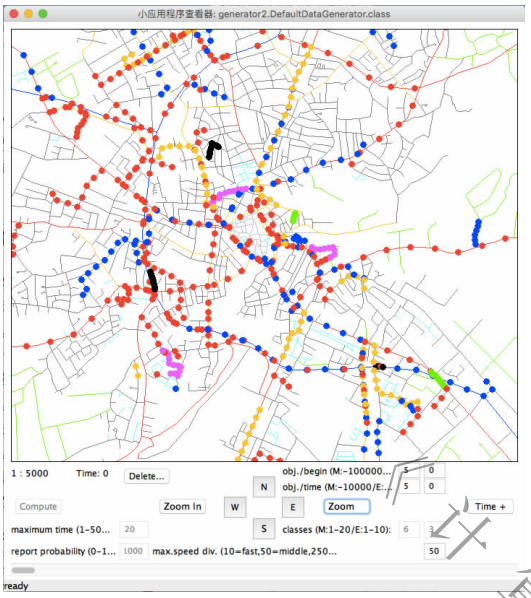

图 10 移动对象生成器

由于下载的移动生成器默认生成的数据集中的移动对象记录并没有记录移动对象对应所在的线路信息,所以在生成移动对象数据的同时,记录下移动对象对应的线路信息. 由移动对象生成器生成的数据样本字段如下:移动对象的状态、移动对象所在线路、移动对象 id、时间参数、移动对象 X 坐标,移动对象 Y 坐标.

由于文章提出的结构建模的数据单元是 TSDR(时空数据矩形),因此需要将上述数据集合的 X,Y 坐标数据建模成时空参数 d,建模及筛选后的数据样本如表 2 所示.

表 2 移动对象数据样本

所在线路	移动对象 id	时间参数	空间参数
193 601 937	5	26	4
501 105 012	4	366	4
194 001 941	3	144	4
493 504 965	2	57	5
646 706 534	1	233	7
36 800 369	23	74	6

实验使用移动对象生成器生成 200 万、400 万、600 万、800 万、1000 万条时空对象数据. 实验的硬件环境是:处理器 Intel Core™i5-2430M 2.40 GHz,内存 10 GB;软件环境为操作系统 Win10,编程语言 Java.

4.1 索引构建

本小节从构建索引的空间开销和时间开销两个方面来对 PM-tree 和 MON-tree 进行比较. 如图 11 所示,随着移动对象数据量的增长,PM-tree 和 MON-tree 的空间开销均呈近似线性增长,但 PM-tree 较 MON-tree 更为优越. 这主要是由于 MON-tree 的下层结构 R-tree 采用结点填充的方式建立索引,把有效数据均存储于叶结点上. 而 PM-tree 则采用紧凑的储存方式,在其索引树上的每个结点均为有效数据,因此 PM-tree 可取得更优越的空间储存效率.

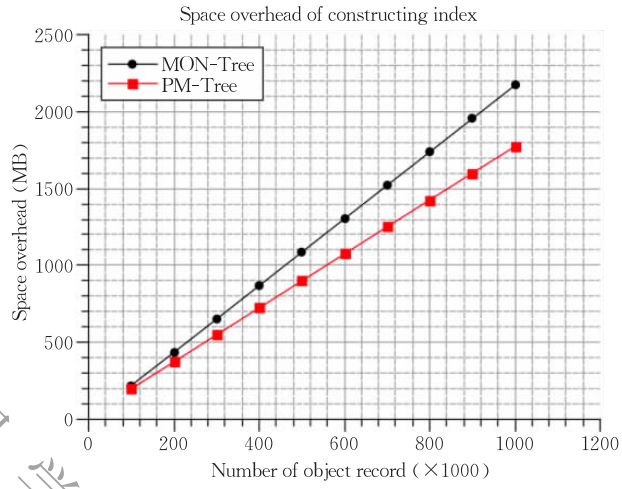

图 11 索引构建空间开销

如图 12 所示,在构建索引的时间开销方面,随着移动对象数据量的增长,PM-tree 的构建时间开销远比 MON-tree 构建时间开销要小. PM-Tree 的相序划分算法是类似于一种对 TSDR 进行分类的过程,每一次并不需要遍历完所有用于建序的 TSDR

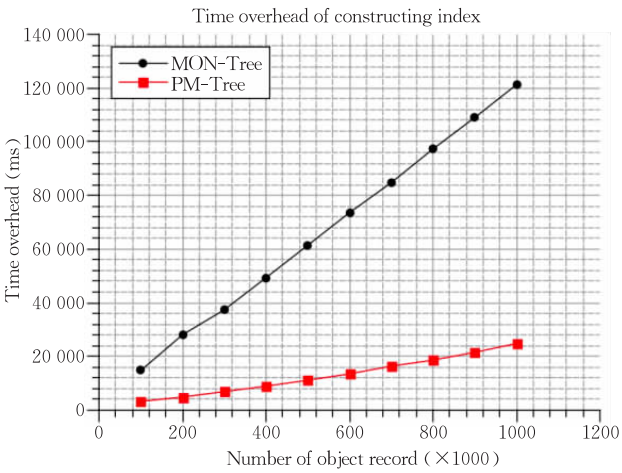

图 12 索引构建时间开销

① http://iapg.jade-hs.de/personen/brinkhoff/generator/

序列，所以十分节省时间，而 MON-tree 的下层结构 R-tree 采取了逐点插入的模式，而在建树的过程中，R-tree 存在大量的非叶子结点的动态计算，包括下溢、分裂等，相比于 PM-Tree 的这种归类的算法来说构建时间成本大大增加. 因此，MON-tree 较 PM-tree 需要更多的处理时间. 由图 12 可看出，随着移动对象数量的递增，PM-tree 的构建索引曲线增长很缓慢，在 1000 万移动对象记录的情形下，构建时间约为 MON-tree 的六分之一.

4.2　查询仿真

路网移动对象查询模式通常分为窗口查询、时间片查询和点查询，因此以下从上述三个方面设计仿真实验来对比 PM-tree 与 MON-tree 的查询性能.

4.2.1　窗口查询

窗口查询是指给定一个时间间隔和一个空间矩形区域，查找在该时间间隔中位于给定空间矩形区域上的移动对象.

实验随机生成查询窗口，对查询窗口进行 1000 次查询实验，考察评估这 1000 次查询分别在 200 万、400 万、600 万、800 万、1000 万数据量上的平均查询时间消耗. 如图 13 所示，随移动对象数据量的增长，PM-tree 的窗口查询时间消耗比 MON-tree 少. 这主要是由于 PM-tree 索引能迅速定位目标数据集所在的相点序分支，快速获得查询结果的候选集. 而 MON-tree 则对于每个查询窗口均需要从上到下对其节点（包括其非有效数据中间节点）进行查找匹配直到找到所有满足查询要求的叶节点，这种迭代遍历结点的方式效率不高.

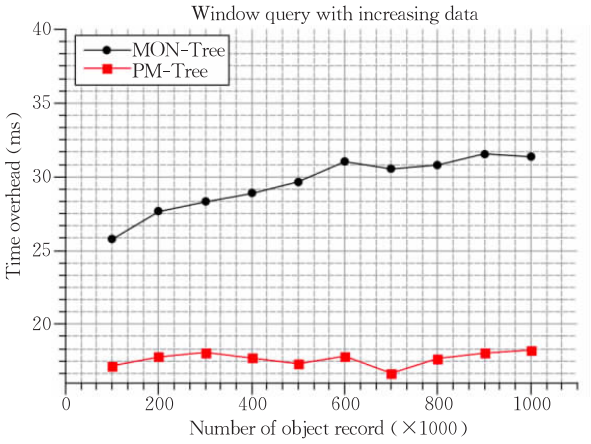

图 13　移动对象数据量增加对窗口查询性能影响

衡量索引性能其中一个重要指标是查询窗口参数变化对索引性能影响. 因此实验考察当移动对象数据量设为 1000 万时，查询窗口的时间和空间间隔参数都分别设为其最大查询跨度 T 的 1%、5%、10%、15%、20%、25%时，索引的窗口查询效率是否受到影响.

如图 14 所示，随着查询窗口的时间区间和空间区域跨度的增加，PM-tree 的查询时间小于 MON-tree 的查询时间. 这是主要由于 PM-tree 利用映射函数把二维时空数据矩形投影成带参数的一维时空相点并建立相应的相点偏序结构，因此在查询比较时，PM-tree 仅需要对一维区间进行操作且利用偏序结构所实现“一次一集合”的查询模式，从而 PM-tree 可取得较 MON-tree 更高的过滤效果以及更稳定的查询性能.

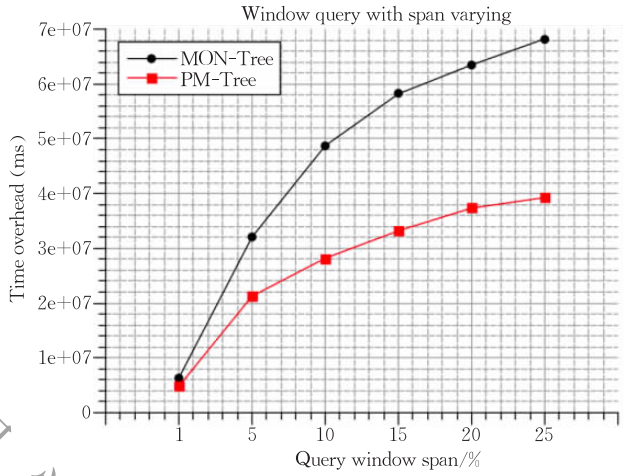

图 14　查询窗口变化对窗口查询性能影响

PM-tree 作为一种时空数据索引，它能同时支持时间区间查询和空间区域查询. 当查询窗口的空间区域取最大查询跨度 T 时，窗口查询会蜕化成纯时间区间查询；当查询窗口的时间区间取最大查询跨度 T 时，窗口查询会蜕化成纯空间区域查询. 如图 15 和图 16 所示，无论是纯时间区间查询还是纯空间区域查询，随着数据量的增长，PM-tree 的查询效率均高于 MON-tree.

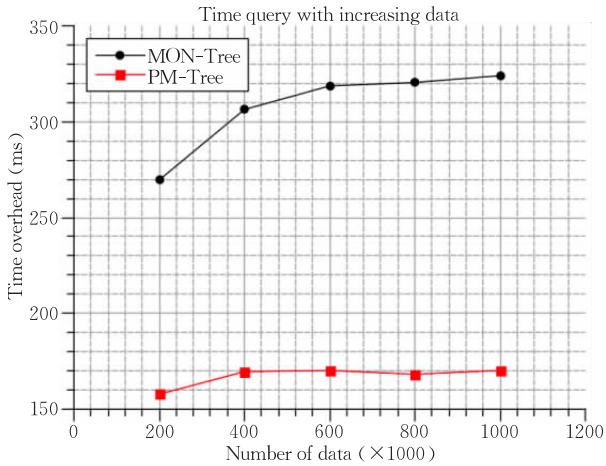

图 15　纯时间区间查询性能

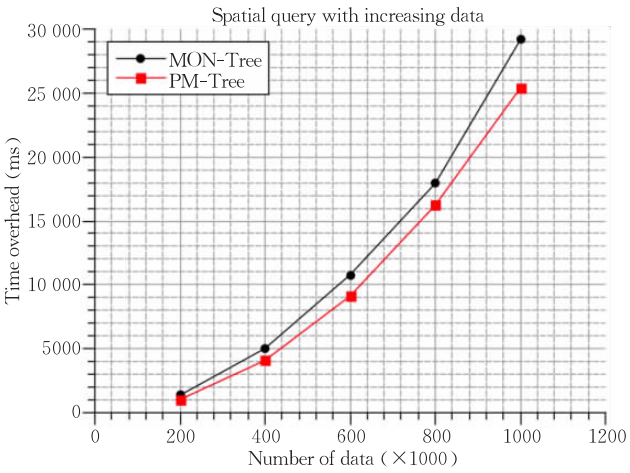

图 16 纯空间区域查询性能

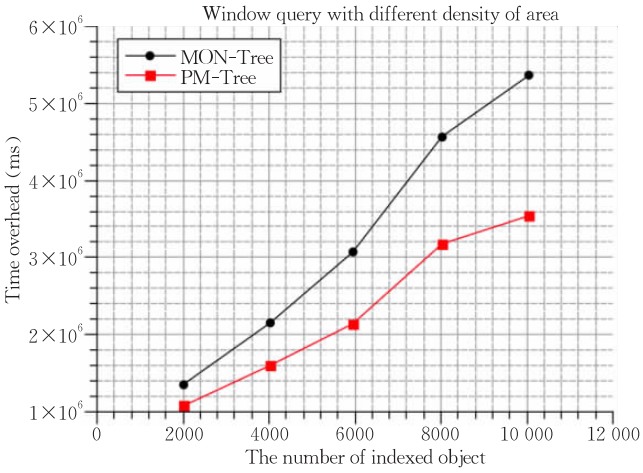

图 18 不同密集度索引性能

4.2.2 区域密集度查询

由于现实生活中，在真实的路网中，不同区域在不同的时间段的移动对象的数量会有所不同. 这取决于现实生活中的交通情况、天气情况或者特殊事件. 如图 17 所示，因此考量一个索引在不同密集度的情形下的查询效率对于一个索引来说显得尤为重要. 我们在地图上的不同区域随机生成 1000 个尺寸不变的查询窗口. 对这个 1000 个查询窗口在相同数据量的查询情况进行分析，挑选索引到的移动对象数据量约为 1000，2000，3000，4000，5000，6000，7000，8000，9000，10 000 的查询窗口，这些索引到不同数据量的窗口代表密集度不一样的区域. 对这些区域进行仿真查询，结果如图 18 所示，对于不同密集度的区域的查询结果，PM-tree 都要优于 MON-tree.

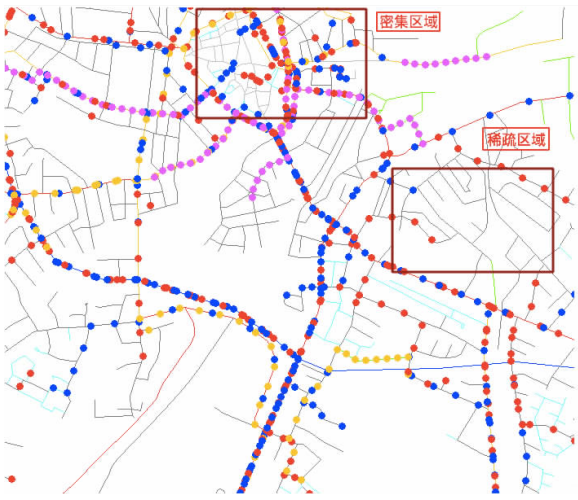

图 17 密集度不同区域

4.2.3 查询性能分析

MON-tree 的基本构件是 R-Tree，在当移动对象数量越多的时候，R-tree 的非叶子结点数量和划分的分支数量急剧增加，导致查询的时候需要递归地遍历所有与查询窗口相重叠的查询分支以及该分支下的所有非叶子结点，即非有效数据节点. 而 PM-tree 的相点序分支不会导致非有效数据的递增，而且被遍历到的相点序分支数量并不会随对象数量的增加而显著增多. 为了验证我们的分析，我们实验了在不同移动对象密集度的在相同查询窗口条件的情况下，查询所遍历过的分支数目.

由图 19 可知，随着查询到的移动对象的数量越多，MON-tree 所需遍历的非叶子结点数量急剧增加，而 PM-tree 所需遍历的相序数量增加不明显. 如图 20，通过计算机的视觉模拟，我们也可看出随着数量增加，R-tree 的非有效数据节点数量急剧增加.

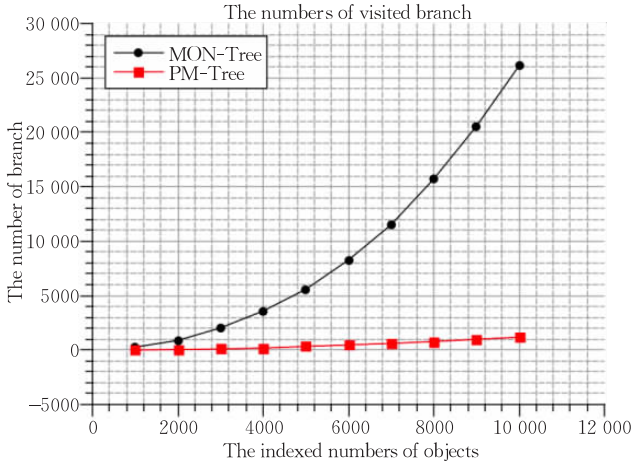

图 19 索引到不同数量移动对象所遍历的分支数目

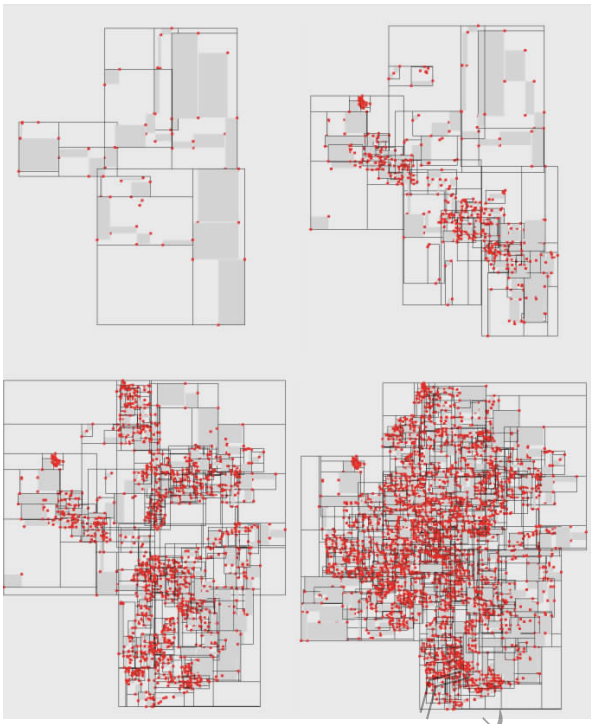

图 20　不同移动对象数量对于 R-tree 非叶子结点数影响

4.3　更新仿真

由于 PM-tree 和 MON-tree 主要应用于历史信息管理范畴，因此索引的更新只考虑插入更新. 插入更新包含了在新的线路中插入移动对象和在现有线路中插入移动对象产生的新数据两种不同的情形，因此实验分别考察在不同情况下的索引更新效率.

如图 21 所示，当在现有线路下插入移动对象所产生的运动信息时，PM-tree 比 MON-tree 取得高效的更新效率. 由于 MON-tree 下层结构 R-tree 在插入数据时，结点由于上溢或者下溢使得分裂会传递到根结点，而 PM-tree 不存在这种调整模式，使得 PM-tree 拥有极其良好的插入性能. 如图 22 所示，

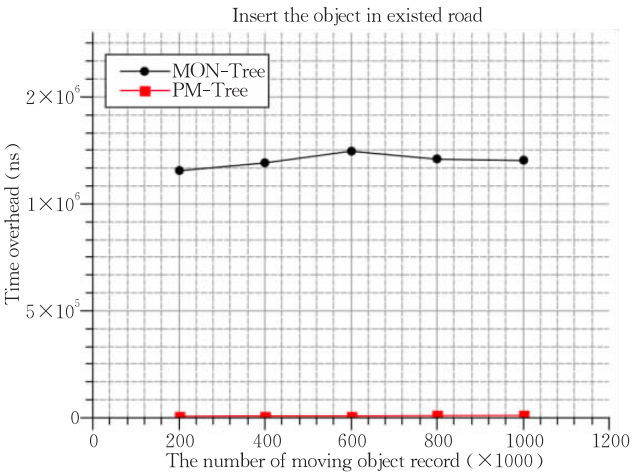

图 21　在线路中插入移动对象

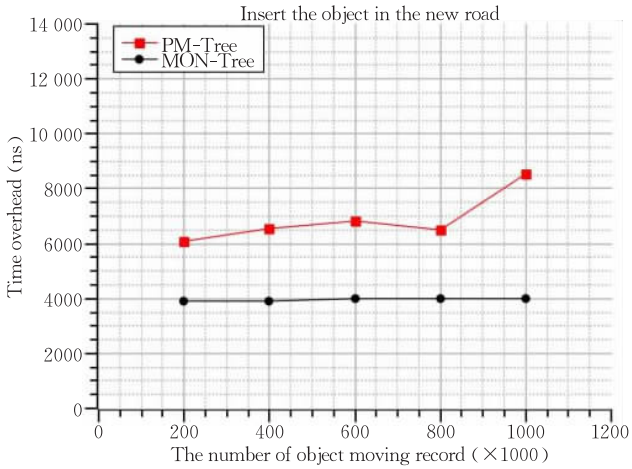

图 22　插入新线路中的对象

当在新的线路中插入新的移动对象所产生的运动信息时，PM-tree 的更新效率略逊于 MON-tree. 这是由于在一棵 PM-tree 中，插入一条记录需要两步，一步是将 TSDR 构建成 PPOB，再将 PPOB 插入新 PM-tree 中. 而在 MON-tree 的下层结构 R-tree 中则只需要一步，直接插入即可，但是插入的时间都控制在 50000 ns 以内. 并且现实情况中第一种插入情况要更频繁些.

5　总结与展望

本论文研究基于路网场景下移动对象数据索引技术，属于移动对象的历史数据管理范畴. 为了避免对时间维度和空间维度信息的分别过滤，论文把移动对象时间和空间的二维信息映射成一维相点区间，同时把相点区间组织成偏序结构，并以此为基础建立了路网移动对象索引. 通过这种时空信息统一处理的索引，从而实现了“一次一集合”的查询模式以及增量式更新的管理机制. 实验表明了 PM-tree 的可行性和有效性. 论文研究的基于时空映射方法以及相点序结构具有良好的扩展性，可适用于其他的应用场景. 本文下一步的工作是将算法从单机环境移植到并行计算环境中去，以适应大数据环境下的路网数据集及大规模并行计算的要求.

参　考　文　献

[1] Guting Ralf Harmut, Schneider Markus. Moving Objects Databases. Jin Pei-Quan, Yue Li-Hua Trans. Beijing: Higher Education Press, 2009(in Chinese)
(古廷，施耐德. 移动对象数据库. 金培权，岳丽华译. 北京：高等教育出版社，2009)

[2] Sistla A P, Wolfson O, Chamberlain S, Dao S. Modeling and querying moving objects//Proceedings of the 13th International Conference on Data Engineering(ICDE). Birmingham, UK, 1997: 422-432

[3] Zhou Ao-Ying, Yang Bin, Jin Che-Qing. 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. 基于位置的服务: 架构与进展. 计算机学报, 2011, 34(7): 1155-1171)

[4] Hao Zhong-Xiao. Theoretical Basis of Mobile Object Database. Beijing: Science Press, 2012(in Chinese)
(郝忠孝. 移动对象数据库理论基础. 北京: 科学出版社, 2012)

[5] Chen Jidong, Meng Xiaofeng, Guo Yanyan, Xiao Zhen. Update-efficient indexing of moving objects in road networks. GeoInformatica, 2009, 13(4): 397-424

[6] Nguyen-Dinh L V, Aref W G, Mokbel M F. Spatio-temporal access methods: Part 2 (2003－2010). Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 2010, 33(2): 46-55

[7] Tayeb J, Ulusoy Ö, Wolfson O. A quadtree-based dynamic attribute indexing method. The Computer Journal, 1998, 41(3): 185-200

[8] Kollios G, Gunopulos D, Tsotras V J. On indexing mobile objects//Proceedings of the 18th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems. Philadelphia, USA, 1999: 261-272

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

[10] Procopiuc C M, Agarwal P K, Har-Peled S. STAR-TREE: An efficient self-adjusting index for moving objects//Proceedings of the International Workshop on Algorithm Engineering and Experiments. San Francisco, USA, 2002: 178-193

[11] Zhu Tong-Yu, Wang Chen, Lv Wei-Feng, Huang Jian. Continuous range monitoring of moving objects in road networks//Proceedings of the 10th International Conference on Intelligent Systems Design and Applications. Cairo, Egypt, 2010: 1412-1417

[12] 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 Databases. Berlin, Germany, 2003: 790-801

[13] Zheng K, Trajcevski G, Zhou X, Scheuermann P. Probabilistic range queries for uncertain trajectories on road networks//Proceedings of the 14th International Conference on Extending Database Technology. Uppsala, Sweden, 2011: 283-294

[14] Wang J, et al. SMe: Explicit & implicit constrained-space probabilistic threshold range queries for moving objects. GeoInformatica, 2016, 20(1): 19-58

[15] Wang H, Zimmermann R. Processing of continuous location-based range queries on moving objects in road networks. IEEE Transactions on Knowledge and Data Engineering, 2011, 23(7): 1065-1078

[16] Xu X, Xiong L, Sunderam V, Xiao Y. A Markov chain based pruning method for predictive range queries//Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. Burlingame, USA, 2016: 1-10

[17] Shi Y, Huang S, Feng J, et al. A probabilistic range query of moving objects in road network. IEEE Access, 2019, 7: 40165-40174

[18] Guttman A. A dynamic index structure for spatial searching//Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data. New York, USA, 1984: 47-57

[19] Theodoridis Y, Vazirgiannis M, Sellis T. Spatio-temporal indexing for large multimedia applications//Proceedings of the 3rd IEEE International Conference on Multimedia Computing and Systems. Hiroshima, Japan, 1996: 441-448

[20] Xu X, Han J, Lu W. RT-tree: An improved R-tree index structure for spatioltemporal databases//Proceedings of the 4th International Symposium on Spatial Data Handling. Zurich, Switzerland, 1990: 1040-1049

[21] Leutenegger S T, Lopez M A, Edgington J. STR: A simple and efficient algorithm for R-tree packing//Proceedings of the 13th IEEE International Conference on Data Engineering. Birmingham, UK, 1997: 497-506

[22] Kim K C, Yun S W. MR-Tree: A cache-conscious main memory spatial index structure for mobile GIS//Web and Wireless Geographical Information Systems. Berlin, Germany: Springer, 2005: 167-180

[23] Tao Y, Papadias D. MV3R-Tree: A spatio-temporal access method for timestamp and interval queries//Proceedings of the 27th International Conference on Very Large Data Bases. Roma, Italy, 2001: 431-440

[24] Abbasifard M R, Ghahremani B, Naderi H. A survey on nearest neighbor search methods. International Journal of Computer Applications, 2014, 25(95): 39-52

[25] Pfoser D, Jensen C S, Theodoridis Y. Novel approaches to the indexing of moving object trajectories//Proceedings of the 26th International Conference on Very Large Databases. Cairo, Egypt, 2000: 395-406

[26] Chakka V P, Everspaugh A C, Patel J M. Indexing large trajectory data sets with SETI//Proceedings of the 1st Biennial Conference on Innovative Data Systems Research (CIDR). Asilomar, USA, 2003, 1001: 164-175

[27] Han B, Liu L, Omiecinski E. NEAT: Road network aware trajectory clustering//Proceedings of the IEEE 32nd International Conference on Distributed Computing Systems(ICDCS). Macau, China, 2012: 142-151

[28] Frentzos E. Indexing objects moving on fixed networks//Advances in Spatial and Temporal Databases. Berlin, Germany: Springer, 2003: 289-305

[29] Guo Jing-Feng, Wang Jian-Chao, Dong Hong-Yu, et al. Indexing scheme of moving objects on fixed networks. Computer Science, 2006, 33(7): 68-70(in Chinese)
(郭景峰，王建朝，董宏宇等. 基于路网的移动对象索引机制研究. 计算机科学，2006，33(7)：68-70)

[30] Pfoser D, Jensen C S. Indexing of network constrained moving objects//Proceedings of the 11th ACM International Symposium on Advances in Geographic Information Systems. New Orleans, USA, 2003: 25-32

[31] De Almeida V T, Güting R H. Indexing the trajectories of moving objects in networks. GeoInformatica, 2005, 9(1): 33-60

[32] Sun Jimeng. Querying about the past, the present, and the future in spatio-temporal databases//Proceedings of the 20th IEEE International Conference on Data Engineering. Los Alamitos, USA, 2004: 202-213

[33] Pelanis M, Šaltenis S, Jensen C S. Indexing the past, present, and anticipated future positions of moving objects. ACM Transactions on Database Systems, 2006, 31(1): 255-298

[34] Abbasifard M R, Naderi H, Alamdari O I. Efficient indexing for past and current position of moving objects on road networks. IEEE Transactions on Intelligent Transportation Systems, 2017, 19(9): 2789-2800

[35] Cha C I, Kim S W, Won J I, et al. Efficient indexing in trajectory databases. International Journal of Database Theory and Application, 2008, 1(1): 21-28

[36] Šidlauskas D, Šaltenis S, Jensen C S. Parallel main-memory indexing for moving-object query and update workloads// Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. Scottsdale, USA, 2012: 37-48

TANG Na, Ph. D., associate professor. Her research interests include temporal databases, big data management, moving objects database and knowledge discovery.

ZHU Zhan-Hao, M. S. candidate. His research interests include data processing and mining.

LI Jing-Jing, Ph. D., associate professor. Her research interests include computational intelligence and optimization.

TANG Yong, Ph. D., professor, Ph. D. supervisor. His research interests include Temporal data management, social networking and collaborative computing.

YE Xiao-Ping, Ph. D., professor. His research interests include temporal databases, moving objects database and knowledge discovery.

Background

The paper studied the technology of moving object data indexing based on the network. To the best of our knowledge, the current works on the performance optimization of historical window query are based on the two-layer R tree structure by which temporal dimension is treated as the additional spatial dimension. The processing of temporal dimension is same as that of spatial dimension. And low efficiency caused by the overlap of minimum bounding rectangle should be avoided as far as possible. By adapting the R tree structure on the second layer of Mon-tree index with a special data structure-Phase-Point Order Branch, which we proposed basing on the theoretical basis of mathematics, we designed a temporal-spatial phase point moving object data indexing PM-Tree. The contributions of the paper include: the two-dimensional spatial-temporal data rectangle was mapped into parameters of one-dimensional spatial points, so the dimensions of data were reduced and the spatial data and temporal data are integrated. Based on the partial order of the spatial points on the phase plane, a Phase-Point Order Branch was constructed, which can reduce the search space. By studying of correlation of spatial rectangles and spatial phase points, we also discussed new methods of data query and update, and implemented the so-called "one set at a time" query mode as well as dynamic update management.

This work is supported by the National Natural Science Foundation of China (Nos. 61772211 and U1811263), the National Key Research and Developemt Program (No. 2018AAA0101300), Innovation Team of Guangdong Education Department (No. 2018-64/8S0177), and Science and Technology Program of Guangzhou (International Cooperation, No. 201807010043).