doi:10.3969/j.issn.1001-4616.2017.03.006

时空轨迹数据存储方法研究

郑浩泉[1],何浩奇[2],刘伽椰[2],赵　斌[2*],吉根林[2],俞肇元[3]

(1.国网电力科学研究院,江苏 南京 211100)
(2.南京师范大学 计算机科学与技术学院,江苏南京 210023)(3.南京师范大学 地理科学学院,江苏 南京 210023)

[**摘要**]　时空轨迹数据的存储方法是轨迹数据管理中的重要课题,直接影响轨迹数据挖掘算法的性能. 本文根据轨迹数据访问方式的不同提出了 3 种轨迹数据的存储方法,分别是原序保持的轨迹存储方法、空间属性优先的轨迹存储方法和时间属性优先的轨迹存储方法. 存储的原则是每次数据访问所涉及的数据应该尽可能被连续存储. 将上述 3 种轨迹数据存储方法加以实现,基于真实数据集的实验表明,按照数据访问的特点为轨迹数据挖掘算法选择合适的轨迹存储方法,可以有效地提高挖掘算法的执行效率,更好地支撑轨迹数据分析挖掘任务.

[**关键词**]　轨迹,磁盘存储,列存储,轨迹数据挖掘
[**中图分类号**]TP392　[**文献标志码**]A　[**文章编号**]1001-4616(2017)03-0038-07

Research on Storage Methods of Spatio-Temporal Trajectories

Zheng Haoquan[1],He Haoqi[2],Liu Jiaye[2],Zhao Bin[2*],Ji Genlin[2],Yu Zhaoyuan[3]

(1.State Grid Electric Power Research Institute,Nanjing 211100,China)
(2.School of Computer Science and Technology,Nanjing Normal University,Nanjing 210023,China)
(3.School of Geography Science,Nanjing Normal University,Nanjing 210023,China)

Abstract:Spatiotemporal trajectory data storage is an important research issue in trajectory data management,which directly affects the performances of trajectory data mining algorithms. In this paper,we propose three trajectory storage methods according to data access methods,including the trajectory storage based on chronological orders,the trajectory storage based on spatial attributes and the trajectory storage based on temporal attributes. The basic principle of trajectory storage is that the data involved in one operation should be stored as close as possible. The three trajectory storage methods are implemented,and the experimental results based on a real data set show that,employing appropriate trajectory storage methods according to the characteristics of data access may significantly improve the efficiency of trajectory data mining algorithms and help to better support trajectory data analysis and mining tasks.

Key words:trajectory,disk storage,column store,trajectory data mining

近年来,随着全球定位技术和网络通信技术的发展和成熟,移动终端产生的时空轨迹数据的规模越来越大. 在综合考虑存储设备的技术性能与价格成本之后,通常轨迹数据采用辅助存储设备(如磁盘)进行存储. 但是,轨迹数据的挖掘算法常常在执行过程中引发大量的随机 I/O 操作,这成为了影响算法性能提升的重要因素. 本文通过研究时空轨迹数据在辅助存储设备中的存储方法,减少随机 I/O 操作次数,缩短执行时间,有效提升挖掘算法的执行性能.

关于轨迹数据的存储方案,工业界普遍采用成熟的数据库技术. 例如,甲骨文公司的 Oracle Spatial 产品基于对象-关系模式存储和管理空间数据,其中的 SDO_GEOMETRY 对象类型可以用来存储轨迹数据. 又如,微软公司的 SQL Server 同样提供了对空间数据的支持功能. 开源数据库 PostgreSQL 的扩展插件 PostGIS 提供的空间数据类型也可以构造出 Trajectory 数据类型. 上述软件产品大多采用基于关系数据库扩展出的空间数据库来存储轨迹数据. 虽然此类解决方案提供了成熟的空间数据类型和空间访问接口,使用方便. 但是,轨迹数据本质上不是结构化数据,并不完全适合关系数据库. 因而,这样的存储方法无法

收稿日期:2017-03-16.
基金项目:智能电网生产调度领域大数据应用研究(524606160204)资助、国家自然科学基金(41471371)资助.
通讯联系人:赵斌,博士,副教授,研究方向:数据挖掘、数据库及其应用. E-mail:zhaobin@outlook.com

针对轨迹的时空特性提供高效率的访问方法,从而影响数据访问的性能.

学术界针对此问题也展开了相关研究. Mediano M[1]等人提出采用轨迹形式表示二维空间长矢量,在经过分段处理后按照原始顺序依次存储. 实际操作时仅需读取所需片段,避免非必要数据的读取,提升了数据访问效率. Chakka V[2]等人也采用了基于轨迹段的存储方法,但其存储方法采用在线算法实现. 基本思想是将添加的轨迹分段处理后按照空间关系进行组织,将邻近的轨迹段归为一组,同组轨迹段在磁盘上连续存放,这样的存储方法可以有效提升邻近空间轨迹的磁盘访问效率. 通常时空数据在空间分布上具有偏斜的特性,Botea V[3]等人基于此观察采用空间四象限划分的方法将历史时空点数据分组到不同大小的空间网格中,每个网格存储的数据规模大致相同,并且相邻时空点数据在磁盘上邻近存储. 另一个相近的研究工作是 Mauroux P 提出的 TrajStore[4],它也研究轨迹数据的存储问题,不同之处是 TrajStore 的存储方法支持数据压缩和新轨迹的添加功能. Wang HZ[5]等人研究了内存中轨迹数据的组织方法. 按照时间顺序以"帧"为单位进行分组,并且采用压缩和 Cache 优化的技术提升数据访问性能. 由于该存储方法是针对内存设备而设计的,因此无法直接移植到磁盘设备上.

轨迹数据存储方法对轨迹数据挖掘算法的性能影响很大. 现有的轨迹数据挖掘包含轨迹模式挖掘、轨迹聚类、轨迹分类和时空图挖掘等研究内容[6-8]. 其中不同的挖掘算法访问轨迹数据的方式各不相同,因而在存储设备上轨迹数据按照轨迹访问方式进行存储与组织最有利于算法性能的发挥. 例如,轨迹伴随模式算法的核心操作是对每个时刻的点数据集进行聚类运算. 打破原始轨迹的点数据组织方式,按照时间属性对点数据进行重新分组,显然这种方法更优. 由此可见,轨迹数据存储方法的设计应该充分考虑数据访问的方式.

根据轨迹数据挖掘算法中数据访问方式的不同,轨迹数据的存储方法可以分成 3 种. 第一种,按照轨迹原有序列顺序存储点数据. 此方法适合依序对轨迹数据进行访问的算法. 第二种,按照空间关系组织轨迹数据. 对轨迹进行空间上的划分,在磁盘上将相同空间区域的轨迹段连续存储. 此方案适合基于空间维度的数据访问操作,如空间窗口查询. 第三种,按照时间属性组织轨迹数据. 将同一时刻的空间点数据在磁盘上邻近存储. 此方案适合基于时间维度的数据访问操作,如指定时刻的空间查询. 本文的大量实验表明,只有根据轨迹数据挖掘算法的数据访问方式选择适合的存储方法,才能显著提升轨迹数据挖掘算法的运行效率.

1 时空轨迹数据存储方法

在时空轨迹挖掘算法中,核心操作的数据访问方式决定了应该选择的数据存储方法. 目前,轨迹数据挖掘主要包括轨迹模式挖掘、轨迹分类、轨迹聚类和时空异常检测等[6-8]. 其中,轨迹聚类是轨迹数据挖掘[7,9-10]中最常见的算法之一. 例如,在群体运动模式挖掘时,普遍采用聚类算法获得移动对象簇. 为了保证聚类算法的高效执行,同一时刻的点数据在存储设备中应该集中存放. 又如,轨迹分类[11-12]通过对轨迹段进行特征提取的方法构建分类模型. 显然,按照依照时序访问轨迹的点数据并不适合,而采用轨迹分段存储的方法更适合. 由此可见,合适的存储方案有助于充分发挥轨迹挖掘算法的性能.

经过分析常见的轨迹数据访问方式之后,本文提出了 3 种典型的轨迹存储方法,分别是原序保持的轨迹存储方法、空间属性优先的轨迹存储方法和时间属性优先的轨迹存储方法.

第一类存储方法保持轨迹原有序列的完整性,按照轨迹序列顺序对点数据连续存储,是一种最简单的轨迹存储方法. 该方法适合顺序访问轨迹数据的操作. 第二类存储方法从空间维度重新组织轨迹数据. 按照空间区域的划分将轨迹分割成多条轨迹段,然后按照空间邻近性原则存储轨迹段. 该方法适合优先关注轨迹空间属性的数据访问方式. 第三类存储方法从时间维度重新组织轨迹数据. 同一时刻的点数据连续存放. 此方法适合优先考虑时间属性的数据访问方式.

为了便于深入讨论轨迹数据存储方法,我们将时空轨迹形式化定义如下:给定时空轨迹数据集 $\mathrm{Traj}_{\mathrm{DB}}=(\mathrm{Traj}_1,\mathrm{Traj}_2,\cdots,\mathrm{Traj}_n)$,其中,轨迹 $\mathrm{Traj}_i=<p_{i1},p_{i2},\cdots p_{im},>$,$p_{ij}$代表轨迹 Traj_i 中的第 j 个点,每个点包含 x,y 和 t 3 种属性,代表在 t 时刻移动对象的经纬度坐标 x 和 y. 除了可以将轨迹表示成点序列,还可以表示成由轨迹段组成的序列,即 $\mathrm{Traj}_i=(S_{i1},S_{i2},\cdots,S_{in_i})$. 即 S_{ij}代表轨迹 Traj_i 中的第 j 条轨迹段.

1.1　原序保持的轨迹存储方法

最简单直接的轨迹存储方法是按照轨迹时序在存储设备上依次存储点数据,即在时间维度上相互邻近的点数据在存储设备上邻近存放. 这种存储方法实现简单,可以保证数据的完整性,但是数据访问效率不高.

为了提高轨迹数据的访问效率,采用数据压缩技术进行处理. 不难发现,轨迹序列中相邻的点数据在空间中也比较接近. 因而采用差分编码技术对轨迹数据进行压缩,即采用较小的差值替代较大的原数值. 经过这样的优化处理后,轨迹数据占用的存储空间更小,访问数据的 I/O 效率可以提高.

1.2　空间属性优先的轨迹存储方法

原序保持的轨迹存储方法虽然实现简单,并且保证数据的完整性,但是访问轨迹空间属性的效率很低. 有鉴于此,本文提出了第二种轨迹存储方法,即空间属性优先的轨迹数据存储方法.

该方法的基本思想是,通过空间划分对轨迹进行分段处理,相邻的轨迹段在存储设备上邻近存放. 如此这样,依照空间属性访问轨迹数据集可以有效降低数据访问的 I/O 代价.

按照空间划分方法的不同,轨迹分段处理分为均匀网格划分和密度敏感的网格划分[4]两种. 由于基本原理相同,因而本文采用均匀网格划分的方法. 具体步骤为,首先采用均匀网格对轨迹进行分段处理,将轨迹与网格的交叉点作为轨迹的分割点. 其次,在完成所有轨迹分段操作之后,同一网格的轨迹段被集中组织. 最后,以“块”作为存储单元将网格中的轨迹段存入存储设备,并且记录下块的相关信息. 如图 1 所示,轨迹 Traj_1、Traj_2、Traj_3 和 Traj_4 被空间网格 Grid 分割成多个轨迹段,其中网格 Grid(1,2)包含轨迹段 S_{13}、S_{33}和 S_{44},它们被存储在同一块中.

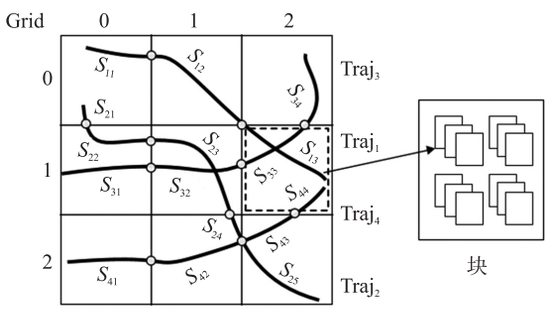


图 1　基于空间划分的轨迹分段

Fig. 1　Trajectory segmentation based on spatial partition

当访问指定空间区域中的轨迹时,查询算法通过比对各网格的空间信息来快速定位候选的轨迹段,避免大量不相关轨迹段的访问,有效提高了查询处理的执行效率. 此外,由于轨迹数据挖掘算法中存在大量的分析型查询,因而提高查询性能可以对轨迹挖掘算法在运行效率上有实质性地提升.

1.3　时间属性优先的轨迹存储方法

在时空数据挖掘中有一类查询需要基于时间维度进行计算. 例如,为了进行空间点聚类,需要查询给定时刻移动对象的位置. 前两种存储方法都无法高效地支持这类查询操作. 因为相同时刻的移动对象的点数据在存储设备上并非集中存储,这样必然引发大量随机 I/O 的产生,导致数据访问效率低下. 有鉴于此,本文提出第三种轨迹存储方法,即时间属性优先的轨迹存储方法.

该方法依据时间属性对轨迹的点数据进行重新组织,同一时刻的点数据集中存储. 为了便于讨论,本文将移动对象同一时刻所有的点数据构成的结构称之为“帧”. 按照时刻将轨迹组织成帧数据,一帧对应于一个时刻,每一帧包含该时刻所有轨迹的点数据,同一帧数据在存储设备中邻近存储.

表 1　帧结构的示例

Table 1　Example of frame based structure

时刻	9:01	9:02	9:03	9:04	9:05
帧	第 1 帧 $(1,P_{11})$ $(2,P_{21})$	第 2 帧 $(1,P_{12})$ $(2,P_{22})$	第 3 帧 $(1,P_{13})$ $(2,P_{23})$	第 4 帧 $(1,P_{14})$ $(2,P_{24})$	第 5 帧 $(1,P_{15})$ $(2,P_{25})$

如表 1 所示,两条轨迹 $\mathrm{Traj}_1(p_{11},p_{12},\cdots p_{15})$ 和 $\mathrm{Traj}_1(p_{21},p_{22},\cdots p_{25})$,第 i 帧由 Traj_1 的 p_{1i}点和 Traj_2 的 p_{2i}点构成,$i=1,\cdots,5$.

当执行基于时间的移动对象查询时,首先按照时间参数定位对应的帧数据. 然后,只需要有限次 I/O 就可以读出指定数据. 由于轨迹点数据的采样频率可能不同,因此需要采用线性插值的方法对缺失的点数据进行补齐.

1.4 轨迹存储方法比较

3 种轨迹存储方法设计的主要目的都是为了支持高效的轨迹数据访问操作. 由于应用场景的不同,因而轨迹存储方法差别很大. 表 2 详细列出了 3 种存储方法在不同特性上的比较情况.

表 2 轨迹存储方法比较

Table 2 Comparison of three trajectory storage methods

存储方法	特性					
	数据完整性	方法实现	重新组织轨迹	支持空间属性访问	支持时间属性访问	可否压缩
原序保持的轨迹存储方法	完整	简单	否	不支持	不支持	可以
空间优先的轨迹存储方法	完整	复杂	是	支持	不支持	可以
时间优先的轨迹存储方法	完整	复杂	是	不支持	支持	不可以

从中可以发现以下几点.

(1)一种存储方法不能满足所有情况,不同的存储方法有各自的优势,应该根据应用场景的需求选择合适的轨迹存储方法.

(2)为了满足时空场景的数据访问要求,需要对原轨迹数据进行重新组织,并且有针对性地设计存储的方法和结构. 当然,这也增加了存储方法实现的复杂性. 下节中将通过大量的实验验证 3 种存储方法在时空访问方面的性能.

(3)第二、三种存储方法虽然改变了原轨迹数据的组织结构,但是依然保证了轨迹数据的完整性.

2 实验与分析

2.1 实验设置

本文实验选取 2012 年 11 月 2 日至 8 日北京市 12 408 辆出租车 2 500 条 GPS 数据作为数据集. 在经过线性插值后,最终得到 1.13G 的测试数据集. 所有实验程序采用 Java 语言开发实现. 实验环境配置为 Intel Core i5 处理器、2.6GHz 主频、8G 内存和 500G 转速 7 200 r/s 的硬盘.

为了对比 3 种存储方法在面对不同数据访问要求时的性能,本文设计了 3 个实验,分别为窗口查询、K 最近邻查询和 DBSCAN[14] 聚类算法(轨迹数据挖掘中最具有代表性的聚类算法). 空间查询中的窗口尺寸是以北京市的面积为参考基准根据特定比例随机产生(见表 3). 空间属性优先的轨迹存储方法中的均匀网格规模为 60×60. 而 DBSCAN 的密度阈值设为 5,距离阈值设为 200 m. 具体参数信息参考表 3. 本文所有的查询实验都是取 50 次测试的平均值作为最终测试结果. 由于 DBSCAN 测试非常耗时,因而取 5 次测试的平均值作为最终结果.

为了测试轨迹存储方法的有效性和可行性,本文实现了 3 种存储方法. 按照是否采用压缩技术,又对前两种方法进一步分为压缩版和非压缩版. 为了表述方便,后续实验分析将采用表 4 中的英文简称代替存储方法的中文全称.

表 3 实验参数设置情况

Table 3 Parameters setting

参数	默认值	实验参数变化范围
轨迹数量(条)	2500	500~2500
时间间隔(h)	5	1~9
空间窗口比例	5%	1%~9%
最近邻的 K 值	10	1~50

表 4 轨迹存储方法列表

Table 4 Trajectory storage methods

存储方法简称		存储方法全称
CT	CTS	原序保持的轨迹无压缩存储方法
	CTCS	原序保持的轨迹压缩存储方法
ST	STS	空间属性优先的轨迹无压缩存储方法
	STCS	空间属性优先的轨迹压缩存储方法
TT		时间属性优先的轨迹存储方法

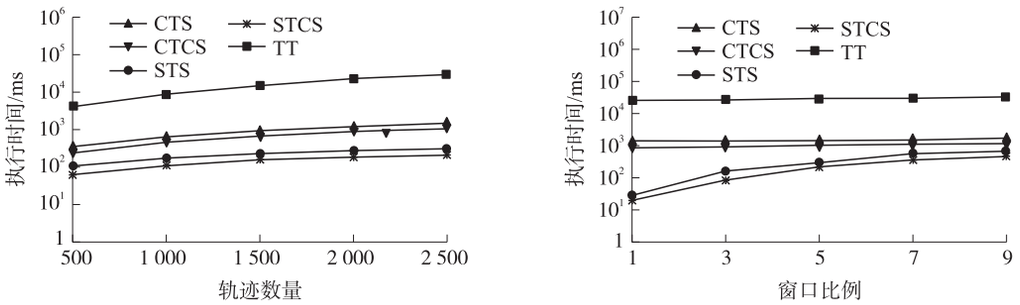
2.2 实验结果与分析

本文实验将依次测试 3 种存储方法在窗口查询、K 最近邻查询和聚类算法中的性能.

2.2.1 窗口查询实验

窗口查询是指查找指定空间范围内的轨迹段. 以下将从轨迹数据量变化和空间窗口比例两方面测试 3 种存储方法的性能. 不难发现,在两种情况下落入窗口的轨迹段数量都会随之增加,因此所有存储方法的查询时间也相应增加. 但是,由于 3 种存储方法的轨迹数据组织方式不同,因而查询时间的变化情况并

不一样. 如图 2(a)、(b)所示,TT 的时间代价最高,主要是由于指定窗口中的轨迹段可能分布于整个数据文件中,需要扫描所有数据块才能得到正确的查询结果. 其次是 CT 的时间代价较高. 这是因为在 CT 中窗口范围内的轨迹段非集中存储,数据访问的 I/O 代价自然很高. 此外,STCS 在所有方法中表现最好. 这是由于 ST 方法中空间窗口中的轨迹段在存储设备中连续存放,有效降低了数据访问时 I/O 的次数,压缩版本的 ST 方法表现自然更出色.



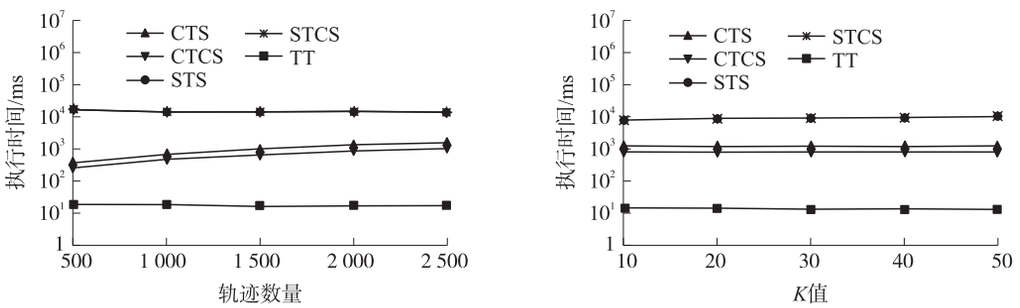
(a) 轨迹数据量的变化影响　　(b) 窗口比例的变化影响

图 2　窗口查询的时间性能

Fig. 2　Runtime of window queries

2.2.2　*K* 最近邻查询实验

K 最近邻查询是指给定时刻查找距离查询点最近的 *K* 个移动对象. 以下将从轨迹数据量变化和近邻数 *K* 值变化两方面测试 3 种存储方法的性能. 不难发现,轨迹数据量的增加会造成 *K* 近邻计算的候选集规模的变大. 如图 3(a)所示,ST 方法的时间代价最高,而 TT 方法的时间代价最低. 这是因为在 ST 中相同时刻的点数据分布于整个数据文件中,所以扫描该候选集几乎等同于扫描整个文件. 在 TT 中,同一时刻的候选集数据在存储设备中集中存放,因而数据访问效率最高. 另一方面,由于近邻数 *K* 值的变化不改变候选集的规模,因此各方法在时间代价上基本没有变化,并且时间代价上的排序和轨迹数据量变化的完全相同,如图 3(b)所示.



(a) 轨迹数据量的变化影响　　(b) 近邻数*K*值的变化影响

图 3　*K* 最近邻查询的时间性能

Fig. 3　Runtime of KNN queries

2.2.3　DBSCAN 聚类实验分析

DBSCAN 算法对给定时刻的移动对象进行聚类处理,主要的数据访问操作是在指定时刻查找指定空间范围内的点数据. 如图 4 所示,TT 方法的时间代价应该最低,因为它首先按照时刻对轨迹数据进行了重新组织,同一时刻的数据邻近存储,这都有利于 DBSCAN 中的数据访问操作. ST 和 CT 方法相比,表现要更好一些. 这主要是因为 ST 方法中点数据按照网格进行了重新组织,由网格构成的邻域十分有利于 DBSCAN 的范围查询执行,所以 ST 比 CT 的时间代价更小.

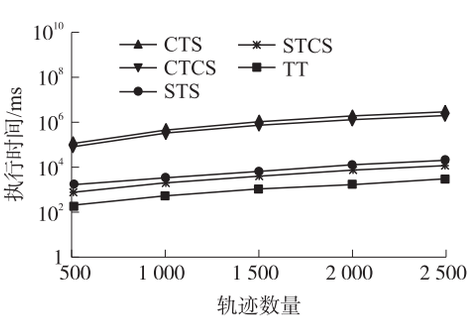


图 4　DBSCAN 的时间性能

Fig. 4　Runtime of DBSCAN

3 轨迹存储方法的应用

按照挖掘任务的不同,常见的时空轨迹数据挖掘可以分为时空模式挖掘、时空聚类、时空分类和时空异常检测四大类[8]. 虽然轨迹数据挖掘涵盖的算法众多,但是从数据访问方式的角度观察,本文提出的3种轨迹存储方法基本可以解决这些算法的存储问题(详见表5),具体分析如下.

(1)在轨迹模式挖掘中,由于伴随模式、聚集模式等算法关注给定时刻中移动对象的空间位置关系,因而时间属性优先的轨迹存储方法比较适合. 而频繁模式、周期模式更关注轨迹序列的变化规律,因此原序保持的轨迹存储方法更适合.

(2)轨迹聚类按照处理对象的不同可以分为空间点聚类和子轨迹聚类. 显然,空间点聚类适合采用时间属性优先的轨迹存储方法;而子轨迹聚类的对象主要是轨迹段,因而应该采用空间属性优先的轨迹存储方法.

(3)在轨迹分类问题中,涉及数据访问的主要任务是特征提取. 通常此任务采用轨迹逐条遍历的方式进行处理. 由此可见,原序保持的轨迹存储方法适合轨迹分类的特征提取任务.

(4)轨迹的异常检测主要通过度量轨迹段间的相似度来发现异常轨迹段,进而发现异常轨迹,因此空间属性优先的轨迹存储方法比较适合此类研究问题.

表5 常见轨迹数据挖掘算法的轨迹存储方法

Table 5 Trajectory storage methods for classical trajectory data mining algorithms

轨迹数据挖掘分类	典型的研究工作	推荐的轨迹存储方法
轨迹模式挖掘	伴随模式:Convoy[13]	时间属性优先的轨迹存储方法
	聚集模式:Gathering[14]	时间属性优先的轨迹存储方法
	频繁模式:T-pattern[15]	原序保持的轨迹存储方法
	周期模式:STPMine1[16]、STPMine2[16]	原序保持的轨迹存储方法
轨迹聚类	空间点聚类:DBSCAN[17]	时间属性优先的轨迹存储方法
	子轨迹聚类:TRACLUS[10]	空间属性优先的轨迹存储方法
轨迹分类	基于时空特征的分类模型[11]	原序保持的轨迹存储方法
异常检测	异常轨迹检测:TRAOD[18]	空间属性优先的轨迹存储方法

4 结束语

按照轨迹数据访问方式的不同,本文提出了3种轨迹数据的存储方法,分别为原序保持的轨迹存储方法、空间属性优先的轨迹存储方法和时间属性优先的轨迹存储方法. 大量实验表明,只有根据轨迹数据挖掘算法的数据访问方式选择适合的存储方法,才能显著提升轨迹数据挖掘算法的运行效率.

[参考文献]

[1] MEDIANO M R,CASANOVA M A,DREUX M. V-Trees a storage method for long vector data[C]//Proceedings of 20th International Conference on Very Large Data Bases. Santiago de Chile,Chile:Morgan Kaufmann,1994:321-330.

[2] CHAKKA V P,EVERSPAUGH A,PATEL J M. Indexing large trajectory data sets with SETI[C]//First Biennial Conference on Innovative Data Systems Research. Asilomar,CA,USA:www.cidrdb.org,2003.

[3] BOTEA V,MALLETT D,NASCIMENTO M A,et al. PIST:an efficient and practical indexing technique for historical spatio-temporal point data[J]. GeoInformatica,2008,12(2):143-168.

[4] MAUROUX P C,WU E,MADDEN S. TrajStore:an adaptive storage system for very large trajectory data sets[C]//Proceedings of the 26th International Conference on Data Engineering. Long Beach,California,USA:IEEE Computer Society,2010:109-120.

[5] Wang H Z,Zheng K,Xu J J,et al. SharkDB:an In-memory column-oriented trajectory storage[C]//Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. Melbourne,Victoria,Australia:ACM,2015:1 099-1 104.

[6] ZHENG Y. Trajectory data mining:an overview[J]. ACM transactions on intelligent systems and technology,2015,6(3):29:1-29:41.

[7] 吉根林,赵斌. 时空轨迹大数据模式挖掘研究进展[J]. 数据采集与处理,2015,30(1):47-58.

[8] 吉根林,赵斌. 面向大数据的时空数据挖掘综述[J]. 南京师大学报(自然科学版),2014,37(1):1-7.
[9] 吉根林,孙鸿艳,赵斌. 时空轨迹群体运动模式挖掘研究进展[J]. 南京航空航天大学学报,2016,48(5):615-624.
[10] LEE J G,HAN J W,WHANG KY. Trajectory clustering:a partition-and-group framework[C]//Proceedings of the ACM SIGMOD International Conference on Management of Data. Beijing,China:ACM,2007:593-604.
[11] ZHENG Y,CHEN Y,LI Q,et al. Understanding transportation modes based on GPS data for web applications[J]. Acm transactions on the Web,2010,4(1):495-507.
[12] PATEL D. Incorporating duration and region association information in trajectory classification[J]. Journal of location based services,2013,7(4):246-271.
[13] JEUNG H,YIU M L,ZHOU X F,et al. Discovery of convoys in trajectory databases[J]. Proceedings of the VLDB endowment,2008,1(1):1 068-1 080.
[14] ZHENG K,ZHENG Y,YUAN N J,et al. On discovery of gathering patterns from trajectories[C]//29th IEEE International Conference on Data Engineering. Brisbane,Australia:IEEE Computer Society,2013,242-253.
[15] GIANNOTTI F,NANNI M,PINELLI F,et al. Trajectory pattern mining[C]//Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,San Jose,California,USA:ACM,2007:330-339.
[16] CAO H P,MAMOULIS N,CHEUNG D W. Discovery of periodic patterns in spatiotemporal sequences[J]. IEEE transactions on knowledge and data engineering,2007,19(4):453-467.
[17] ESTER M,KRIEGEL H P,SANDER J,et al. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise[C]//Proceedings of the Second International Conference on Knowledge Discovery and Data Mining(KDD-96). Portland,Oregon,USA:AAAI Press,1996:226-231.
[18] LEE J G,HAN J W,LI X L. Trajectory outlier detection:a partition-and-detect framework[C]//Proceedings of the 24th International Conference on Data Engineering. Cancun,Mexico,2008:140-149.

[责任编辑:陆炳新]