软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2017,28(6):1498−1515 [doi: 10.13328/j.cnki.jos.005220]
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于车牌识别流数据的车辆伴随模式发现方法*

朱美玲[1,2,3], 刘 晨[2,3], 王雄斌[2,3], 韩燕波[2,3]

[1](天津大学 计算机科学与技术学院,天津 300072)

[2](大规模流数据集成与分析技术北京市重点实验室(北方工业大学),北京 100144)

[3](北方工业大学 云计算研究中心,北京 100144)

通讯作者: 朱美玲, E-mail: meilingzhu2006@126.com

摘 要: 针对伴随车辆检测这一新兴的智能交通应用,在一种特殊的流式时空大数据——车牌识别流式大数据(ANPR)下,重新定义了 Platoon 伴随模式,提出 PlatoonFinder 算法,即时地在车牌识别数据流上挖掘 Platoon 伴随模式.主要贡献包括:第一,将 Platoon 伴随模式发现问题映射为数据流上的带有时空约束的频繁序列挖掘问题,与传统频繁序列挖掘算法仅考虑序列元素之间位置关系不同,该算法能够在频繁序列挖掘的过程中有效处理序列元素之间复杂的时空约束关系;第二,该算法融入了伪投影等性能优化技术,针对数据流的特点进行了性能优化,能够有效应对车牌识别流式大数据的速率和规模,从而实现车辆Platoon伴随模式的即时发现.通过在真实车牌识别数据集上的实验分析表明:PlatoonFinder 算法的平均延时显著低于经典的 Aprior 和 PrefixSpan 等频繁模式挖掘算法,也低于真实情况下交通摄像头的车牌识别最小时间间隔.因此,所提出的算法可以有效地发现伴随车辆组及其移动模式.

关键词: 流式时空大数据;大数据分析;伴随模式;频繁序列挖掘

中图法分类号: TP311

中文引用格式: 朱美玲,刘晨,王雄斌,韩燕波.基于车牌识别流数据的车辆伴随模式发现方法.软件学报,2017,28(6):1498−1515. http://www.jos.org.cn/1000-9825/5220.htm

英文引用格式: Zhu ML, Liu C, Wang XB, Han YB. Approach to discover companion pattern based on anpr data stream. Ruan Jian Xue Bao/Journal of Software, 2017,28(6):1498−1515 (in Chinese). http://www.jos.org.cn/1000-9825/5220.htm

Approach to Discover Companion Pattern Based on ANPR Data Stream

ZHU Mei-Ling[1,2,3], LIU Chen[2,3], WANG Xiong-Bin[2,3], HAN Yan-Bo[2.3]

[1](School of Computer Science and Technology, Tianjin University, Tianjin 300072, China)

[2](Beijing Key Laboratory on Integration and Analysis of Large-Scale Stream Data (North China University of Technology), Beijing 100144, China)

[3](Cloud Computing Research Center, North China University of Technology, Beijing 100144, China)

Abstract: Companion vehicle discovery is a newly emerging intelligent transportation application. Aiming at it, this paper redefines the Platoon companion pattern over a special type of spatio-temporal data stream, or ANPR (automatic number plate recognition data). Accordingly, a PlatoonFinder algorithm is also proposed to mine Platoon companions over ANPR data stream instantly. First, Platoon discovery problem is transformed into frequent sequence mining problem with customized spatio-temporal constraints. Compared to traditional frequent sequence mining algorithms, this new algorithm can effectively handle complex spatio-temporal relationships among

* 基金项目: 国家自然科学基金(61672042); 北京市市委组织部、北京市优秀人才培养资助,青年骨干个人项目; 北方工业大学“人才强校计划”青年拔尖人才培育计划

Foundation item: National Natural Science Foundation of China (61672042); Program for Youth Backbone Individual, Beijing Municipal Party Committee Organization Department; Training Plan of Top Young Talent in North China University of Technology

收稿时间: 2016-05-07; 修改时间: 2016-07-15; 采用时间: 2016-12-22; jos 在线出版时间: 2017-02-20

CNKI 网络优先出版: 2017-02-20 15:14:42, http://www.cnki.net/kcms/detail/11.2560.TP.20170220.1514.029.html

sequence elements rather than their positions. Second, the new algorithm also integrates several optimization techniques such as pseudo projection to greatly improve the efficiency. It can efficiently deal with high speed and large scale ANPR data stream so as to instantly discover Platoon companions. Experiments show that the latency of the algorithm is significantly lower than classic frequent pattern mining algorithms including Apriori and Prefixspan. Furthermore, it is also lower than the minimum time interval between any two real ANPR data records. Hence, the proposed algorithm can discover Platoon companions effectively and efficiently.

Key words: big spatio-temporal data stream; big data analytics; companion pattern; frequent sequence mining

智能交通系统(intelligent transportation system,简称 ITS)是软件工程技术的典型应用领域.随着大数据时代的到来,智能交通应用必然会产生重大变化,新型交通应用系统的研发和部署,也必将对大数据软件工程技术的发展起到极大促动.目前,各种交通信息采集技术广泛应用于城市交通,持续地产生海量且实时的时空数据.这些海量数据是交通的规划、管控、预测、引导的信息源和基础,同时也带来了新的问题和困境:海量实时时空数据的潜在价值还没有得到有效的分析、挖掘和应用.为此,本文重点关注一类典型的交通大数据——车牌识别流式大数据(automatic number plate recognition data,简称 ANPR)的实时分析和挖掘方法.车牌识别数据源自城市道路路口安装的交通摄像头,提供了全天候的车辆行驶位置监视能力.本文将在车牌识别流式大数据基础上,瞄准伴随车辆即时检测这一新兴的智能交通应用,聚焦于伴随车辆检测的核心算法和软件实现.

1 引 言

当前,记录对象移动模式的时空数据在规模上迅速膨胀.在时空数据中发现移动对象的移动模式具有重要的研究价值.车辆是日常生活中最常见的移动对象之一.在一段时间内,一起移动的车辆组群所具有的移动模式通常被称为车辆伴随模式.近年来,车辆伴随模式的发现逐渐成为国内外学者关注的热点研究问题[1−8],相关的研究成果已经被广泛应用到智能交通等领域[7].

当前,移动车辆伴随模式的发现研究大多基于 GPS 数据[1−8].GPS 数据由安装在车辆上的 GPS 设备按照固定的时间间隔产生并回传.没有安装 GPS 设备或者设备没有开启的车辆不会产生 GPS 数据.在某些特殊场合下,如车辆跟踪、犯罪嫌疑人出逃等,嫌疑人通常会关闭甚至拆掉 GPS 设备以防止被公安机关发现并拦截.这使得基于 GPS 数据实现的车辆伴随模式发现方法无法满足上述场景的需求.

与 GPS 数据不同,车牌识别数据源自城市道路路口安装的交通摄像头,这些摄像头能够不间断采集过往车辆的信息,包括车牌号码和采集时间.这些信息以流数据的形式传输到交通管理部门的数据中心,用于进一步的分析和挖掘.车牌识别数据提供了全天候的车辆行驶位置监视能力,是交通大数据的一类典型代表.当前,我国大型城市安装的交通摄像头已经超过 5 000 台,且数量仍在持续增加.摄像头采集车辆信息的速率取决于摄像头的拍摄频率,最高可以达到每秒采集一次,即,高峰时期流式车牌识别数据的速率可达 5 000 条每秒.假设每台摄像头每天采集车辆信息的平均速率为高峰期的 1/3,则每天采集数据的总数可以达到 1.44 亿条,一年的数据量可以达到 PB 级.

基于 GPS 数据,现有研究工作已经抽象并总结了多种典型的车辆伴随模式,包括 Flock[1−3],Convoy[4,5],Swarm[6],Traveling Companion[7],Platoon[8]等.这些模式的目的都是为了发现一段时间内一起移动的车辆群组,但是不同模式对于伴随车辆的时空约束定义有较大不同.在时间约束上,Flock 和 Convoy 模式要求伴随车辆经过的时间点具有连续性;Swarm 和 Traveling Companion 对时间点的连续性没有要求;Platoon 模式要求车辆组群在若干时间段内一起移动.每个时间段由不少于δ个连续的时间点组成,且时间点总数不少于 k.Platoon 模式实质上是一种比上述几种模式更为通用的伴随模式,δ的不同取值可以得到上述几种不同的伴随模式.例如,设 Platoon 模式实际经过了 l 个时间点,当$\delta>l/2$ 时,Platoon 模式的时间约束与 Flock/Convoy 一致,当$\delta=1$ 时, Platoon 模式的时间约束与 Swarm/Traveling Companion 一致[8].此外,在空间约束方面,Flock 要求车辆处于圆盘形的地理区域,而其他工作将这一约束放松至密度可达.

本文的前期工作是基于历史车牌识别数据和流式车牌识数据开展了 Traveling Companion 伴随模式的发现研究.首先,基于历史车牌识别数据,通过借鉴频繁项集挖掘经典算法 Apriori 算法设计了基于 Spark 的并行伴

随车辆发现算法,并将其应用于社区拼车[9].随后,我们又尝试从流式车牌识别数据中即时发现伴随车辆[10].为实现这一目标,我们提出了点伴随的概念,用于描述一起通过单个摄像头的车辆组群,并设计了点伴随检测的并行优化算法.

在前期工作的基础上,本文聚焦于车牌识别数据流上的 Platoon 伴随模式的发现问题,将其映射为带有时空约束的频繁序列挖掘问题,并提出了相应的伴随车辆发现算法 PlatoonFinder,以在车辆通过交通摄像头时即时发现并输出车辆的 Platoon 伴随模式.算法的主要贡献包括:

(1) 基于车牌识别流式大数据的特点,本文在借鉴相关工作的基础上重新定义了 Platoon 伴随模式;

(2) 与传统的频繁序列挖掘算法相比,PlatoonFinder 算法在计算频繁序列时考虑了自定义的时空约束:首先,要求频繁序列的每个元素在序列数据库中以足够接近的时间出现;第二,要求频繁序列由若干子序列组成,每个子序列的元素在序列数据库中相邻出现;

(3) 本文一方面压缩了参与计算的序列数据库,另一方面在构建数据库时融入了伪投影技术,从而降低了计算复杂度,提升了算法的性能.实验结果表明,PlatoonFinder 算法的平均延时明显低于真实情况下车牌识别数据之间的最小时间间隔.

2 相关工作

2.1 伴随模式发现

研究人员进行了大量针对伴随模式的研究与探索,提出了不同条件下的伴随模式定义及挖掘算法.以提出的时间为顺序,典型的工作包括 Flock[1−3],Convoy[4,5],Swarm[6],Traveling Companion[7],Platoon[8]等. Flock 和 Convoy 定义了一种强时空约束的伴随模式.Flock 要求车辆组群在不少于 k 个连续时间点上处于一个足够小的圆形地理空间内(k 可称为时间点总数阈值).Convoy 将 Flock 的空间约束扩展为密度可达的地理空间. Swarm 和 Traveling Companion 可以视作同一种伴随模式[7].该模式放宽了 Convoy 的时间约束,要求车辆组群在 k 个时间点(不一定连续)上处于密度可达的地理空间内.因此,如果一组车辆组群的移动模式符合 Flock 模式,则一定符合 Convoy 模式,也一定符合 Swarm 和 Traveling Companion 模式.Platoon 伴随模式也要求车辆组群在 k 个时间点上处于密度可达的地理空间.具体的说,Platoon 模式是指车辆组群在若干个长度至少为 δ(δ为正整数)的连续时间点下处于某种地理位置区域内,且时间点的总和不少于 k 个.不妨设 Platoon 模式实际的时间点总数为 l,满足 $l\geqslant k$.Platoon 伴随模式是更通用的模式,用户可以通过设定δ值得到上述伴随模式.当$\delta>l/2$,得到的是 Convoy 模式;而当$\delta=1$时,得到的是 Swarm/Traveling Companion 模式[8].此外,Platoon 伴随模式可以根据设定的阈值细化出不同于上述各种的模式.例如 $1<\delta<l/2$ 时,可以得到局部时间连续的伴随模式.因此,Platoon 伴随模式既可以根据不同场景挖掘 Convoy 模式和 Swarm/Traveling Companion 模式,也可以根据不同需求挖掘出局部时间连续模式和全局时间连续模式.这些是 Flock,Convoy,Swarm,Traveling Companion 等任何一个模式无法实现的.

表 1 对比了上述 5 种伴随模式的相关工作及本文的工作.

Table 1 Comparison of several companion vehicles discovery methods

表 1 几种伴随模式发现方法的对比分析

模式	时间约束	空间约束	数据集	解决方法
Flock[1]	连续	圆盘	GPS	矩阵分析
Flock[2]	连续	圆盘	GPS	有向加权图
Flock[3]	连续	圆盘	GPS	聚类和求交集
Convoy[4]	连续	密度可达	GPS	轨迹相似度计算
Convoy[5]	连续	密度可达	GPS	轨迹相似度计算
Swarm[6]	不连续	密度可达	GPS	有重叠项的频繁项集挖掘
Traveling companion[7]	不连续	密度可达	GPS	聚类和求交集
Platoon[8]	可连续可不连续	密度可达	GPS	频繁项集和频繁序列挖掘
Platoon(本文)	给定时间阈值Δt	伴随的交通摄像头可相邻可不相邻	ANPR	针对时空数据的带有时空约束的频繁序列挖掘

除了上述伴随模式发现工作,还有相当一部分研究人员专注于移动对象的聚类[11−14].Kalnis 试图找到两个连续的时间戳内有大部分的重叠移动对象[11].Li 等人采用微聚类(micro-clustering),在对移动对象进行聚类时既考虑了移动对象的当前位置又考虑了短期内未来的位置[12].Kriegel 等人利用了模糊距离函数进行移动对象的聚类[13].Jensen 等人提出了一种增量式的方法在一段时间内对移动对象进行聚类,减少了计算开销[14].

此外,近年来,更多人开始关注大规模轨迹数据的研究.Zheng 使用一系列的技术手段,力求提高在大规模静态轨迹数据集中对移动对象组群聚类的性能[15].Zhang 提出一种聚类检索算法,通过查找一个由移动对象组群组成的时空图来检索移动对象的聚类模式[16].Yoo 尝试使用 MapReduce 框架来达到更优的处理效率,同时也提出了一种划分策略来避免数据之间关系的丢失[17].

然而,上述的大部分研究方法针对静态 GPS 数据设计,无法直接应用于流数据的处理.流数据具有实时到达、不可预知、没有边界等特点,流数据的这些特性导致了流数据难以存储且对实时性要求很高.显然,静态数据方法无法满足这些需求.但在交通领域的应用中,数据通常都是以流的形式提供的.因此,近年来,越来越多的学者开始研究流式交通数据的处理.除了 Tang 提出的流式轨迹数据上 Traveling Companion 模式的发现框架[7]外,Yu 针对流式轨迹数据研究了一种基于密度的聚类算法,尝试实时地发现轨迹组群[18,19].本文聚焦于一种特殊的流式交通数据——流式车牌识别数据,根据前文的分析,这种数据的到达率可以到达 5 000 条/秒甚至更高,且高峰时期每台交通摄像头可以每秒产生一条数据.因此,本文面临的问题是如何在如此高的数据到达速率下即时发现伴随车辆,即:每当车辆通过交通监控摄像头时,本文的方法能够快速发现该车辆的伴随车辆,并将结果输出出来.根据流式车牌识别数据的特性,本文方法的延时应该低于真实数据之间的最小时间间隔(1s),以避免因数据堆积导致的延时急剧增加.而目前,所有的相关工作都针对的是流式 GPS 数据并且取得了很大的进步,虽然无法解决本文的问题,但为本文的研究提供了坚实的基础.

2.2 频繁序列挖掘

频繁序列挖掘算法可以分为两类:基于 Apriori 的算法(Apriori-based algorithms)和基于投影数据的算法(projection-based algorithms).基于 Apriori 的典型算法包括 AprioriAll[20],AprioriSome[20],GSP[21],SPADE[22],SPAM[23]等,这类算法的缺点是需要构建规模庞大的候选集;基于投影数据的典型算法包括 FreeSpan[24],PrefixSpan[25],CloSpan[26]和 BIDE[27]等,这类算法通过构造投影数据库而非候选集,显著提升了算法的性能.此外,研究人员开始关注带有约束的频繁序列挖掘问题.Helen 等人提出了多维频繁序列的概念(multi-dimensional sequential pattern),并提出了相应的挖掘算法[28].多维频繁序列数据库与传统序列数据库的区别在于,前者除了有 SID 和序列之外还包含若干属性.Pei 等人归纳了频繁序列挖掘中存在的 7 个约束,包括项约束(如用户只对某些项感兴趣)、长度约束(用户对频繁序列的长度约束)、超集约束(是否存在频繁序列$\alpha \sqsupseteq \beta$)、聚合约束(如频繁序列的各项和、平均值、最大最小值、方差等的约束)、正则表达式约束(对某些项的条件约束)、持续时间约束(频繁序列的各项必须在某个时间段内出现)以及间隔约束(频繁序列相邻项的时间间隔约束)[29].而本文涉及的时间约束并没有包含在以上 7 种约束之内.另外,Chueh 深入研究了带有时间间隔约束的频繁序列挖掘问题,并提出了相关算法[30].

为了提高算法性能,部分学者对频繁序列挖掘算法进行了并行优化[31−36].Demiriz 基于 SPADE 提出了分析网站点击流的并行化算法 webSPADE[31].Guralnik 等人提出了基于分布式内存的频繁序列挖掘并行算法[32].Ma 等人提出了一种基于分布式内存系统的闭频繁序列挖掘并行算法 PFCSeq[33].每个处理程序独立挖掘本地的闭频繁序列,处理器之间交互的减少,显著降低了算法的时间消耗.Qiao 等人提出了一种轨迹序列挖掘算法 PartSpan,该算法使用了前缀投影技术、并行技术和候选集剪枝技术降低计算开销[34].Yu 等人利用 MapReduce 框架将 BIDE 算法并行化[35].Kessl 提出了一种基于概率模型的静态负载均衡并行算法挖掘频繁序列[36].

以上工作都是频繁序列挖掘领域的典型工作,为本文的算法奠定了良好的基础.本文的定理 1 即是根据 Clospan 算法[27]的公共前缀引理(common prefix)和偏序引理(partial order)而得.然而,以上工作并没有涉及本文提出的时空约束:没有考虑频繁序列各元素在序列中的相邻位置关系,没有考虑频繁序列各元素在不同序列中出现的时间间隔.因此,本文在借鉴已有工作的基础上设计了能够挖掘带有时空约束的频繁序列算法.该算法为

基于车牌识别流式大数据的 Platoon 伴随模式发现而设计,在未来的工作中,我们将进一步研究带有时空约束的频繁序列挖掘通用算法,使之得到更广泛的应用.此外,在未来的工作中,我们还将尝试利用并行等优化手段进一步提高算法的性能.

3 问题定义及分析

近年来,针对运钞车、出租车等特殊车辆的罪案时有发生.犯罪分子通常会跟踪这类车辆至隐蔽的地点并实施抢劫.当车辆经过交通摄像头时,即时检测其可疑跟踪车辆并向被跟踪车辆发出警报,可以有效防止此类案件的发生.如图 1 的 3 幅子图所示,运钞车依次经过交通摄像头 $c_1,c_2,c_3,c_4,c_5,c_6,c_7,\ldots,c_n$.每当运钞车经过摄像头时,即时发现其伴随车辆,是检测运钞车的可疑跟踪车辆的有效手段.

Platoon 模式要求车辆组群在若干时间段内一起移动.每个时间段由不少于δ个连续的时间点组成,且时间点总数不少于 k.在车牌识别数据下,移动车辆在不同时间点下的数据对应了不同的交通摄像头.因此,Platoon 模式实际经过的时间点总数代表了实际经过的交通摄像头,不妨设为 l 个.这 l 个摄像头按时间顺序排列,且由若干位置相邻的摄像头序列组成.因此,δ实际上是位置相邻的摄像头子序列的长度阈值.图 1 描绘了 3 组不同的伴随车辆,为了便于理解,将 3 组车辆的移动方式展现在不同的子图中.根据相关工作的分析,不妨设车辆伴随模式经过的交通摄像头总数阈值为 4(k=4,即伴随车辆至少经过 k 个交通摄像头),l 是移动模式实际经过的交通摄像头数量.图 1(1)刻画了运钞车 p_1 和车辆 p_2 在 4 个相邻交通摄像头下伴随;图 1(2)刻画了运钞车 p_1 和车辆 p_3 在 2 个长度为 2 的相邻交通摄像头下伴随;图 1(3)刻画了运钞车 p_1 和车辆 p_4 在 4 个不相邻交通摄像头下伴随.如果 δ 取值为 1,图 1(1)~图 1(3)所示的 3 组伴随车辆都将被返回给用户;如果 δ 取值为 2,图 1(1)、图 1(2)所示两组伴随车辆将被返回给用户;如果 δ 取值为大于 l/2 的任意整数,图 1(1)所示伴随车辆将被返回给用户.

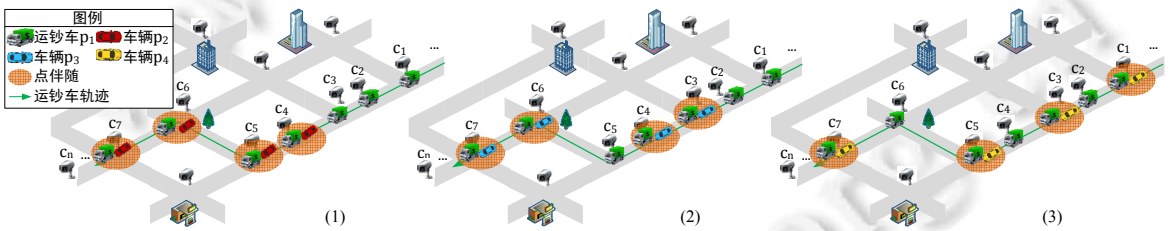


Fig.1 Examples of Platoons based on ANPR data stream

图 1 基于车牌识别流式大数据(ANPR)的 Platoon 伴随模式示意图

定义 1(车牌识别数据记录,ANPR). 一条车牌识别数据记录 r 可以由三元组 r=(c,p,t)来表示,其中,c 代表交通摄像头 ID,p 代表车牌号码,t 是时间戳.一条车牌识别数据记录 r=(c,p,t)表示车牌号码为 p 的车辆在 t 时刻经过摄像头 c.

示例 1:(CAM14115111,京 OA89**,08:20:54),(CAM14115111,京 QE63**,08:20:59)是两条真实车牌识别数据记录,其中,CAM14115111 是交通摄像头 ID.这两条数据代表着车辆京 OA89**和京 QE63**先后在时间 08:20:54 和 08:20:59 经过了 ID 为 CAM14115111 的交通摄像头.

目前,基于 GPS 数据,大量研究工作已经给出了各种伴随模式的定义.与车牌识别数据不同,GPS 数据是从连续变化的时间、经纬度、速度等方面来描述车辆的信息.大部分研究工作都利用了 GPS 数据的经纬度属性来计算伴随模式,如基于密度的聚类[3,6−8]和距离计算[2,4,5].还有部分工作甚至利用了 GPS 数据的速度信息和角度信息进行计算[1,4,5].然而由前文的分析及定义 1 可知:车牌识别数据是由分布在各个路口的交通摄像头自动产生的,并不包含经纬度、速度以及角度等信息.因此,适用于 GPS 数据的方法无法直接应用到车牌识别数据中.为此,本文借鉴前期工作中提出的点伴随概念,利用基于时间和摄像头的聚类给出了车牌识别数据集上的 Platoon 伴随模式的定义.

定义 2(点伴随,moment companion). 已知Δt 为时间阈值,交通摄像头 c 在时刻 t 下点伴随可以被定义为如

下形式:

$$MC(c,t,\Delta t)=\{r.p|r.c=c\wedge t-\Delta t\leqslant r.t\leqslant t\}.$$

示例 2:以图 1(3)为例,交通摄像头 c_1 下关于某时间点存在点伴随$\{(c_1,p_2,t_2),(c_1,p_1,t_1)\}$,其中,$t_2$ 是车辆 p_2 经过交通摄像头 c_1 的时间,$t_2-t_1\leqslant\Delta t$.同理可知,交通摄像头 c_3,c_5 和 c_7 下也存在包含车辆 p_1 和 p_2 的点伴随.

定义 3(Platoon 伴随模式,platoon companion pattern). 已知车辆数量阈值δ_{veh},位置相邻的交通摄像头长度阈值δ和交通摄像头总数阈值δ_{cam},时间阈值Δt,(P,C,T)如果满足以下条件,则称其为车牌识别数据上的 Platoon 伴随模式:

(1) $|P|\geqslant\delta_{veh},|C|\geqslant\delta_{cam},|C_i|\geqslant\delta$;

(2) 每个(P,C_i,T)满足:$\forall c_{i,j}\in C_i$,存在 $MC(c_{i,j},t_{i,j},\Delta t)$,满足 $P\subseteq MC(c_{i,j},t_{i,j},\Delta t)$.

其中,$P=\{p_1,p_2,\ldots,p_m\}$是车牌号码集合;$C=\langle C_1,C_2,\ldots,C_k\rangle$是交通摄像头序列,子序列 $C_i=\langle c_{i,1},c_{i,2},...,c_{i,n_i}\rangle$ 是一组地理位置相邻的交通摄像头;$T=\langle T_1,T_2,\ldots,T_k\rangle$是时间戳序列,子序列 $T_i=\langle t_{i,1},t_{i,2},...,t_{i,n_i}\rangle$,且满足:

$$t_{1,1}<...<t_{1,n_1}<...<t_{i,1}<...<t_{i,n_i}<...<t_{k,1}<...<t_{k,n_k}.$$

定义 4(封闭 platoon 伴随模式,closed platoon companion pattern). 假设(P,C,T)是 Platoon 伴随模式,如果$\nexists C',T',C\sqsubset C',T\sqsubset T'$,使得$(P,C',T')$是 Platoon 伴随模式,则$(P,C,T)$是封闭 Platoon 伴随模式.

为了避免重复计算,本文在车牌识别流式大数据下即时发现封闭 Platoon 伴随模式.

4 Platoon 伴随模式发现算法

4.1 预备知识

频繁序列挖掘为本文的算法奠定了基础,本节将简要介绍频繁序列挖掘的相关概念及经典算法.

设 $I=\{i_1,i_2,\ldots,i_m\}$是项的集合,则 I 的子集称为项集.序列 s 是一个有序项集数组,即 $s=\langle q_1,q_2,\ldots,q_n\rangle(q_i\subseteq I,i=1,2,\ldots,n)$.序列数据库 D 是序列的集合,其中每个序列都配有唯一的标识符,记为 SID.序列 s 的支持度是指序列数据库 D 中包含 s 的序列个数.如果序列 s 在 D 中的支持度超过了预定义的最小支持度,s 称为频繁序列.如果序列数据库 D 中不存在频繁序列 s',$s\sqsubset s'$($\sqsubset$,$\sqsubseteq$用于连接序列,表示前者是后者的子序列;同理,$\sqsupset$,$\sqsupseteq$表示后者是前者的子序列),且 s'与 s 在 D 中具有相同支持度,则频繁序列 s 称为闭频繁序列(closed frequent sequence).序列 s 在 D 中的投影数据库记为 $D_s=\{\eta|s'\in D,s'=\alpha\Diamond\eta\}$,其中,$s\sqsubseteq\alpha$且$\nexists\alpha',s\sqsubseteq\alpha'\sqsubseteq\alpha$.

频繁序列挖掘算法可以分为基于 Apriori 的算法(Apriori-based algorithms)和基于投影数据的算法(projection-based algorithms)[37].基于投影数据的算法采用了分而治之的策略,极大地降低了基于 Apriori 的算法构造候选子序列时产生的时空消耗.这类算法挖掘频繁序列的主要步骤如图 2 所示.

1) 遍历序列数据库 D,生成所有的 1-频繁序列(即,长度为 1 的频繁序列);

2) 对每一个 1-频繁序列 s^1,生成其在 D 的投影数据库 D_{s^1};

3) 对每一个投影数据库 D_{s^1} 重复上述步骤,直至不存在 1-频繁序列为止.

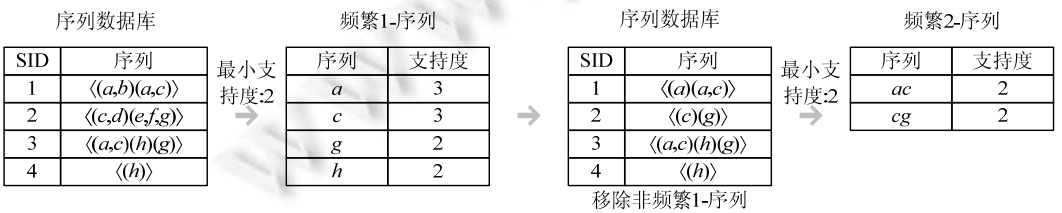


Fig.2 An example of basic steps of projection-based algorithms

图 2 基于投影数据的频繁序列挖掘算法的基本步骤示例

在车牌识别数据集 R 中,任意车辆 p_i 的行驶轨迹都对应了一条交通摄像头序列 $s_{p_i}=\langle c_{i,1},c_{i,2},...,c_{i,n}\rangle$:对任意

元素 $c_{i,j}$,存在 $r_j \in R, r_j.p=p_i, r_j.c=c_{i,j}$,且元素 $c_{i,j}$ 和 $c_{i,j+1}$ 对应的车牌数据满足 $r_j.t<r_{j+1}.t$.易知,$c_{i,1},c_{i,2},\ldots,c_{i,n}$ 是车辆 p_i 依次经过的交通摄像头,$r_1.t,r_2.t,\ldots,r_n.t$ 是经过时间.因此,任意给定的车牌识别数据集 R 都可以映射成序列数据库 $D=\{s_{p_i}\}$,车牌号码 p_i 是序列 s_{p_i} 的 *SID*.

基于上述分析,本文将要求解的 Platoon 伴随模式发现问题映射为车牌识别数据流上的频繁序列挖掘问题.近年来,频繁序列挖掘工作,尤其是针对流数据的挖掘工作取得了许多进展[38−41].然而,现有的频繁序列挖掘算法并不能解决本文问题,这是因为现有的频繁序列挖掘算法仅考虑了序列中元素的出现顺序,而 Platoon 伴随模式则要求序列中元素要满足预定义的时空约束,例如,伴随车辆通过同一摄像头的时间应该足够接近.图 3 直观地展示了这一区别.如图所示:从传统频繁序列的角度来看,假设最小支持度为 2,则$\langle c_1,c_2,c_4,c_5\rangle$和$\langle c_1,c_7,c_3,c_4\rangle$都是频繁序列.而 Platoon 伴随模式的约束更为复杂:如图 3 所示的 Platoon 伴随模式,当δ_{veh}=2,Δt=30s,δ=2,δ_{cam}=4 时,模式(P_1,C_1,T_1),$P_1=\{p_1,p_2\}$,$C_1=\langle c_1,c_2,c_4,c_5\rangle$,$T_1=\langle$09:30:01,09:34:01,09:42:01,09:47:01$\rangle$是 Platoon 伴随模式,而模式(P_2,C_2,T_2),$P_2=\{p_3,p_4\}$,$C_1=\langle c_1,c_7,c_3,c_4\rangle$,$T_1=\langle$09:30:01,09:35:00,09:40:00,09:51:00$\rangle$却不是.这是因为 c_1,c_7 和 c_4 出现在车辆 p_3 和 p_4 序列的时间差分别为 59 秒、59 秒和 8 分 59 秒(小于Δt=30s).实际上,如果δ=3 则模式(P_1,C_1,T_1)也不再是 Platoon 伴随模式.原因如下:根据图 3 所示的交通摄像头的地理位置关系,$\langle c_1,c_2\rangle$和$\langle c_4,c_5\rangle$是两条长度为 2 的相邻摄像头序列;而δ=3 约束了相邻摄像头序列的长度不小于 3.

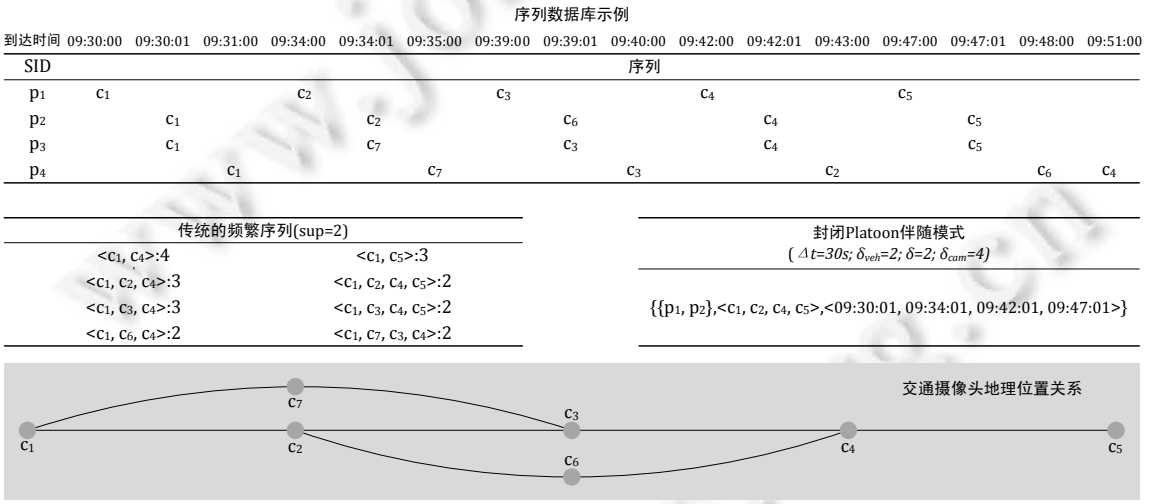


Fig.3 Comparison between Platoon discovering and traditional frequent sequence mining

图 3 Platoon 伴随模式挖掘和传统频繁序列挖掘问题的区别

4.2 PlatoonFinder算法

为了简明清晰地阐述算法,提前定义一些符号.记当前窗口内车牌识别数据产生的序列数据库为 *curD*.当新数据(c_0,p_0,t_0)到达,如果 *curD* 已有车辆 p_0 的序列,则将新数据附加到该序列尾部;否则,在 *curD* 中创建车辆 p_0 的新的序列.记加入新数据 $r_0=(c_0,p_0,t_0)$后的序列数据库为 $curD_+$.记序列数据库:

$$D_+^{r_0}=\{s \mid s\in curD_+ \wedge \exists r, r.c=c_0, t_0-\Delta t \leqslant r.t \leqslant t_0, r\in s \wedge |s| \geqslant \delta_{cam}\},$$
$$D_+=\bigcup D_+^{r_0}.$$

设Δt=2s,δ=2,δ_{cam}=3,图 4 举例说明了上述符号指代的内容,加粗的数据项是新到达的数据记录.

图 5 描述了本文提出的 PlatoonFinder 伴随模式发现算法的基本原理.

窗口中的车牌识别数据记录

(c_1,p_1,09:22:59),(c_3,p_1,09:27:01),(c_6,p_1,09:32:00),(c_8,p_1,09:36:01),(c_{10},p_1,09:40:25),**(c_{11},p_1,09:45:01)**
(c_8,p_2,09:36:00),(c_{11},p_2,09:45:00)
(c_1,p_3,09:23:00),(c_2,p_3,09:27:00),(c_5,p_3,09:32:01),(c_8,p_3,09:36:01),(c_{10},p_3,09:40:26),(c_{12},p_3,09:45:00)
(c_1,p_4,09:22:59),(c_3,p_4,09:27:00),(c_6,p_4,09:31:59),(c_8,p_4,09:35:59),(c_{10},p_4,09:40:24),(c_{11},p_4,09:44:59)
(c_1,p_5,09:23:00),(c_2,p_5,09:27:01),(c_5,p_5,09:32:00),(c_7,p_5,09:35:59),(c_9,p_5,09:40:26),(c_{11},p_5,09:45:00)
(c_1,p_6,09:23:01),(c_2,p_6,09:26:59),(c_4,p_6,09:32:01),(c_7,p_6,09:36:01),(c_9,p_6,09:40:25),(c_{11},p_6,09:44:59)
(c_1,p_7,09:23:00),(c_2,p_7,09:27:01),(c_5,p_7,09:31:59),(c_7,p_7,09:36:00),(c_9,p_7,09:40:24)
(c_1,p_8,09:23:01),(c_3,p_8,09:26:59),(c_6,p_8,09:32:01),(c_8,p_8,09:36:00),(c_{10},p_8,09:40:26),**(c_{12},p_8,09:45:01)**

curD

	09:22:59	09:23:00	09:23:01	...	09:26:59	09:27:00	09:27:01	...	09:31:59	09:32:00	09:32:01	...	09:35:59	09:36:00	09:36:01	...	09:40:24	09:40:25	09:40:26	...	09:44:59	09:45:00	09:45:01
p_1	c_1			...			c_3	...		c_6		...			c_8	...		c_{10}		...			
p_2				...				...				...		c_8		...				...			c_{11}
p_3		c_1		...		c_2		...			c_5	...			c_8	...			c_{10}	...			
p_4	c_1			...		c_3		...	c_6			...	c_8			...	c_{10}			...	c_{11}		
p_5		c_1		...			c_2	...		c_5		...	c_7			...			c_9	...			c_{11}
p_6			c_1	...	c_2			...			c_4	...			c_7	...		c_9		...	c_{11}		
p_7		c_1		...			c_2	...	c_5			...		c_7		...	c_9			...			
p_8			c_1	...	c_3			...			c_6	...		c_8		...			c_{10}	...			

curD+

p_1	c_1			...			c_3	...		c_6		...			c_8	...		c_{10}		...			**c_{11}**
p_2				...				...				...		c_8		...				...		c_{11}	
p_3		c_1		...		c_2		...			c_5	...			c_8	...			c_{10}	...		c_{12}	
p_4	c_1			...		c_3		...	c_6			...	c_8			...	c_{10}			...	c_{11}		
p_5		c_1		...			c_2	...		c_5		...	c_7			...			c_9	...		c_{11}	
p_6			c_1	...	c_2			...			c_4	...			c_7	...		c_9		...	c_{11}		
p_7		c_1		...			c_2	...	c_5			...		c_7		...	c_9			...			
p_8			c_1	...	c_3			...			c_6	...		c_8		...			c_{10}	...			**c_{12}**

D+

p_1	c_1			...			c_3	...		c_6		...			c_8	...		c_{10}		...			**c_{11}**
p_3		c_1		...		c_2		...			c_5	...			c_8	...			c_{10}	...		c_{12}	
p_4	c_1			...		c_3		...	c_6			...	c_8			...	c_{10}			...	c_{11}		
p_5		c_1		...			c_2	...		c_5		...	c_7			...			c_9	...		c_{11}	
p_6			c_1	...	c_2			...			c_4	...			c_7	...		c_9		...	c_{11}		
p_8			c_1	...	c_3			...			c_6	...		c_8		...			c_{10}	...			**c_{12}**

Fig.4 Examples of sequence database symbols (Δt=2s, δ=2, δ_{cam}=3)

图 4 序列数据库符号示例(Δt=2s,δ=2,δ_{cam}=3)

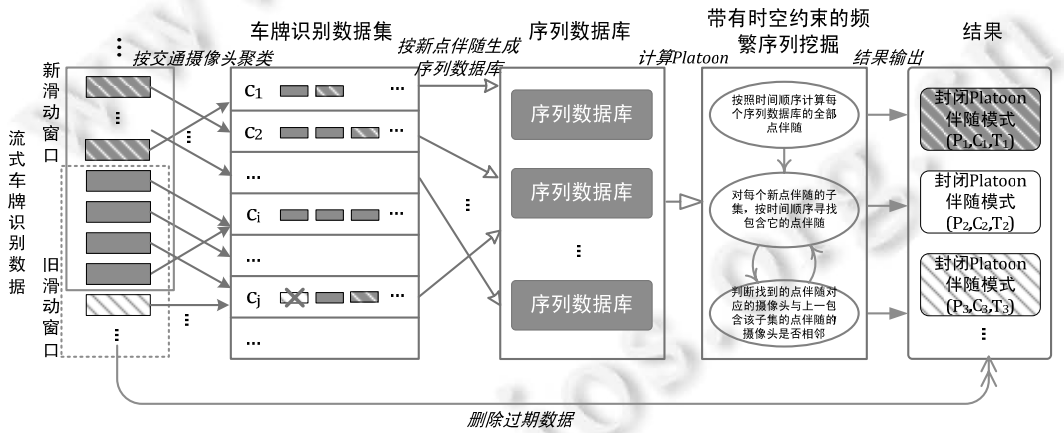


Fig.5 Rationales of PlatoonFinder algorithm

图 5 PlatoonFinder 算法基本原理

算法利用滑动窗口接受实时到达的车牌识别数据,并对数据进行预处理,得到按交通摄像头聚类的车牌识别数据集.假设任意新到达的数据 r_0 产生了新点伴随 MC_0.根据新点伴随 MC_0,生成对应的序列数据库 $D_+^{\prime 0}$.在序列数据库 $D_+^{\prime 0}$ 中挖掘带有时空约束的频繁序列.最后,根据新旧数据的更替更新缓存的结果.在序列数据库 $D_+^{\prime 0}$ 中挖掘带有时空约束的频繁序列的思路如下:首先,按照从大到小的时间顺序找出序列数据库 $D_+^{\prime 0}$ 中所有互不包含的点伴随,对每个点伴随 MC_0 的子集按照时间顺序寻找包含它的点伴随,当找到一个包含它的点伴随 MC_i 时,判断上一个包含它的点伴随 MC_{i-1} 对应的元素与 MC_i 对应的元素是否相邻,并根据判断结果更新结果;然后继续寻找包含 MC 的点伴随.以图 4 为例,r_1=(c_{11},p_1,09:45:01)和 r_2=(c_{12},p_8,09:45:01)是新到达的数据,并且产生了两组新的点伴随 MC_1={p_1,p_4,p_5,p_6}和 MC_2={p_3,p_8},p_2 由于轨迹长度小于δ_{cam} 而被抛弃.MC_1 对应的序列数据库

$D_+^n=\{s_{p_1},s_{p_4},s_{p_5},s_{p_6}\}$.在 D_+^n 中挖掘带有时空约束的频繁序列过程如图 6 所示.

时间	点伴随
09:45:01	c_{11}:{p_1, p_4, p_5, p_6}
09:40:26	c_9:{p_5, p_6}
09:40:25	c_{10}:{p_1, p_4}
09:36:01	c_8:{p_1, p_4}
09:36:01	c_7:{p_5, p_6}
09:32:00	c_6:{p_1, p_4}
09:27:01	c_3:{p_1, p_4}
09:27:01	c_2:{p_5, p_6}
09:23:01	c_1:{p_1, p_4, p_5, p_6}

点伴随	点伴随的超集	判断
{p_1, p_4}	c_1:{p_1, p_4, p_5, p_6}	加入结果集
	c_3:{p_1, p_4}	c_3和c_1在$D_+^{T_1}$ 中相邻出现，加入结果集，继续寻找包含{p_1, p_4}的点伴随
	c_6:{p_1, p_4}	c_6和c_3相邻出现，加入结果集，继续寻找包含{p_1, p_4}的点伴随
	c_8:{p_1, p_4}	c_8和c_6相邻出现，加入结果集，继续寻找包含{p_1, p_4}的点伴随
	c_{10}:{p_1, p_4}	c_{10}和c_8相邻出现，加入结果集，继续寻找包含{p_1, p_4}的点伴随
	c_{11}:{p_1, p_4, p_5, p_6}	c_{11}和c_{10}相邻出现，加入结果集，寻找结束；结果集符合阈值δ和δ_{cam}
{p_5, p_6}	c_1:{p_1, p_4, p_5, p_6}	加入结果
	c_2:{p_5, p_6}	c_2和c_1相邻出现，加入结果集，继续寻找包含{p_5, p_6}的点伴随
	c_7:{p_5, p_6}	c_7和c_2不相邻出现，结果集符合阈值δ，加入结果集，继续寻找
	c_9:{p_5, p_6}	c_9和c_7相邻出现，加入结果集，继续寻找包含{p_5, p_6}的点伴随
	c_{11}:{p_1, p_4, p_5, p_6}	c_{11}和c_9相邻出现，加入结果集，寻找结束；结果集符合阈值δ和δ_{cam}
{p_1, p_4, p_5, p_6}	c_1:{p_1, p_4, p_5, p_6}	加入结果集
	c_{11}:{p_1, p_4, p_5, p_6}	c_{11}和c_1不相邻出现，结果集不符合阈值δ，删除已添加结果c_1:{p_1, p_4, p_5, p_6}，添加c_{11}:{p_1, p_4, p_5, p_6}至结果集，寻找结束；结果集不符合阈值δ和δ_{cam}，删除结果集

结果集
(P,C,T)=({p_1, p_4, p_5, p_6},{c_{11},c_{10},c_8,c_6,c_3,c_1},{09:45:01,09:40:25,09:36:01,09:32:00,09:27:01,09:23:01})
(P,C,T)=({p_5, p_6},{c_{11},c_9,c_7,c_2,c_1},{09:45:01, 09:40:26, 09:36:01, 09:27:01, 09:23:01})

Fig.6 Eampleof mining frequent sequences with spatio-temporal constraints

图 6 带有时空约束的频繁序列挖掘过程示例

前文叙述了挖掘带有自定义时空约束频繁序列的思路.为了保障 PlatoonFinder 算法能够高效处理流数据,本文提出了以下 3 个定理并融合了伪投影技术以优化整个计算过程.

定理 1. 已知存在封闭 Platoon 伴随模式(P,C,T),当新数据到达时,如果$\exists C',T'$使得 $C\sqsubset C',T\sqsubset T'$且$(P,C',T')$是封闭 Platoon 伴随模式,则$(P,C,T)$不再封闭,也不会再成为封闭 Platoon 伴随模式.

根据定义 4,可容易推知定理成立.

定理 2. 序列数据库 $curD_+$可以收缩至 D_+,包含的 Platoon 伴随模式不变.

证明:由定义 3 可知:当新数据 $NR=\{r|r.t=t_n\}$到达时,对于每一个新产生的 Platoon 伴随模式(P,C,T),$P=\{p_1,p_2,\ldots,p_m\}$,$C=\langle c_1,c_2,\ldots,c_n\rangle$,$T=\langle t_1,t_2,\ldots,t_n\rangle$,必须满足 $P\subseteq MC(c_n,t_n,\Delta t)$.记 P 中任意车辆 p_i 在交通摄像头 c_n 下的经过时间为 t_i^n,则满足 $t_n-t_i^n\leqslant\Delta t$.因此,车辆 p_i 的序列包含于 D_+.由此可知,新的伴随模式(P,C,T)一定存在于 D_+. □

根据上述证明可知:只有当新数据 r 产生了新的点伴随 $MC(r.c,r.t,\Delta t)$,且$|MC(r.c,r.t,\Delta t)|\geqslant\delta_{veh}$,才有可能生成新的 Platoon 伴随模式.基于此,本文提出了定理 3.

定理 3. 已知点伴随 $MC(r.c,r.t,\Delta t)$是新数据 r 到达后产生的新点伴随,且$|MC(r.c,r.t,\Delta t)|\geqslant\delta_{veh}$,该数据产生的新的 Platoon 伴随模式存在于序列数据库 $D_+^r=\{s\mid s\in D_+\wedge s.sid\in MC(r.c,r.t,\Delta t)\}$ 中.

证明:设模式(P,C,T)是新数据 r 到达后产生的 Platoon 伴随模式,不妨设 $P=\{p_1,p_2,\ldots,p_m\}$,$C=\langle c_1,c_2,\ldots,c_n\rangle$,$T=\langle t_1,t_2,\ldots,t_n\rangle$.根据定理 2,$(P,C,T)$存在于序列数据库 D_+.由于(P,C,T)是新数据 r 到达后产生的 Platoon 伴随模式,故而 $P\subseteq MC(c_n,t_n,\Delta t)$且 $r.c=c_n,r.t=t_n$.因此,(P,C,T)存在于 $D_+^r=\{s\mid s\in D_+\wedge s.p\in MC(r.c,r.t,\Delta t)\}$ 中. □

定理 3 阐明了如何进一步缩减关于新数据 r 的序列数据库,以及如何基于点伴随的定义在构造序列数据库时考虑时间约束Δt.

此外,在构建序列数据库 D_+^r 时,本文借鉴了频繁序列挖掘领域伪投影数据库构建原理.伪投影序列数据库是指当序列数据库在内存中时,通过指针在已有序列数据库上确定投影序列数据库,而非在内存中建立真实的

投影序列数据库.如图4所示:序列数据库D_+中,每个数据项都具有*SID*和时间戳两个属性.借鉴伪投影数据库构建原理,本文的算法利用*SID*和时间戳从D_+确定序列数据库D_+^r.

算法1是*InsertData*()的伪代码.*InsertData*()方法主要计算新数据产生的新点伴随,并调用*InitFreqMiner*()进行下一步计算.具体过程是:*InsertData*()维护了一个以交通摄像头ID为*key*值的哈希表,通过该哈希表,对点伴随按交通摄像头进行聚类.该哈希表的*value*值按时间存储了该交通摄像头下互不包含的点伴随.当新数据到达时,*InsertData*()将新数据加到该哈希表中(第1行~第3行).根据前面的定理,*InsertData*()利用维护的哈希表找到新形成的点伴随,则有可能生成新的 Platoon 伴随模式.因此,对于每一个新生成的点伴随,*InsertData*()计算其包含的车辆集合P_r(第4行、第5行).接下来根据定理2和定理3,针对新到达的数据*r*计算序列数据库D_+^r,并启动线程调用*FreqMiner*()进行下一步计算.具体过程如下:利用伪投影技术得到序列数据库D_+^r(第6行).将序列数据库D_+^r中包含的时间戳按降序排列得到*T*(第7行).根据D_+^r得到根据*T*排序的的点伴随哈希表*MCS*(第8行).该哈希表的*key*值是时间,*value*值存储了该时间下的全部点伴随,只保留超集点伴随(第9行).即:如果点伴随$MC_i \subseteq MC_j$,去掉点伴随MC_i.调用方法*FreqMiner*()(第10行).最后,根据定理1从结果集中删除不可能成为封闭Platoon伴随模式的已有模式(第12行).

算法1. *InsertData*(r,D_+,δ_{veh},Δt,δ,δ_{cam}).

Input: r, D_+, δ_{veh}, Δt, δ, δ_{cam};

Output: *platMap*,更新的Platoon伴随模式集合;

1. for each 新数据 *r*
2. 将*r*按照交通摄像头聚类,加入hash表*CamMCs*; //*CamMCs*数据结构:⟨*cameraID*,⟨*time*,*ANPRs*⟩⟩
3. for end
4. for each *CamMCs*中修改过的*value*值
5. 得到新点伴随所包含的车辆集合P_r;
6. 利用指针构建$D_+^r=\{\eta \mid \eta \in D_+ \wedge \eta.p \in P_r\}$; //即利用伪投影序列数据库构建原理确定$D_+^r$
7. 根据D_+^r得到降序排列的时间戳序列*T*;
8. 得到基于*T*和D_+^r的点伴随*MCS*=⟨*time*,*MC*s⟩ //*time*是*T*中的时间点,*MCs*是各交通摄像头在*time*时刻的点伴随集合,点伴随包含的元素个数不小于δ_{veh}
9. *MCS*←*MCS*中全部的超集点伴随; //频繁序列时间约束
10. 调用*FreqMiner*(*MCS*,*T*);
11. for end
12. 删除重复的Platoon伴随模式 //定理1

算法2是方法*FreqMiner*()的伪代码.*FreqMiner*()主要负责在序列数据库D_+^r上挖掘封闭的Platoon伴随模式.具体过程是:*FreqMiner*()维护了一个保存挖掘结果的 Map 数据结构*platMap*以及一个用于判断是否满足δ的Map数据结构*platIndex*.*platMap*的*key*值是车辆集合;*value*值是该车辆集合一起经过的摄像头ID和时间.*platIndex*的*key*值是车牌集合,*value*是初始值为0的正整数,用于存储该车辆集合最近驶过的相邻交通摄像头个数(如$P=\{p_1,p_2\}$,P在交通摄像头⟨c_3,c_4,c_6,c_7,c_8⟩下伴随,其中,c_3,c_4是相邻交通摄像头,c_6,c_7,c_8是相邻交通摄像头,则*value*先等于2后等于3).首先,记算法1得到按从大到小的时间顺序排列的点伴随集合*MCS*中最新的点伴随为MC_{new}.对*MCS*中任意点伴随MC_{sub}满足$MC_{sub} \subseteq MC_{new}$,计算Platoon伴随模式(第1行).在*platIndex*中新加键值对⟨MC_{sub},0⟩(第2行).根据时间顺序,依次比较*MCS*中每个点伴随*MC*与MC_{sub}的关系.对于每个$MC_{sub} \subseteq MC$,如果*platMap*中*key*为MC_{sub}的*value*中最后一个点伴随对应的交通摄像头与该*MC*的交通摄像头相邻,将*MC*对应的时间和摄像头加入*platMap*,*platIndex*中对应的*length*加1(第3行~第7行).否则,对*platMap*关于键*MC*的值进行判断:如果上一组位置连续的摄像头数量不少于δ,将*MC*对应的时间和摄像头加入*platMap*,将*platIndex*中对应的*length*设为1(第8行~第11行);否则删除上一组位置连续的摄像头,并将*MC*对

应的时间和摄像头加入 *platMap*,将 *platIndex* 中对应的 *length* 为设 1(第 12 行~第 15 行).检查 *platIndex*,将 *length* 小于δ的点伴随在 *platMap* 的的最后一组相邻摄像头删除(第 20 行~第 22 行).检查 *platMap*,将 *value* 长度小于δ_{cam}的键值对从 *platMap* 中移除(第 25 行~第 27 行).返回 *platMap*(第 30 行).

算法 2. *FreqMiner*(*MCS*,*T*).

Input: *MCS*, *T*;

Output: *platMap*.

1. for each *MCS* 中的点伴随 $MC_{sub} \subseteq MC_{new}$ //频繁序列空间约束
2. *platIndex*←⟨MC_{sub},0⟩;
3. for each *MCS* 中的点伴随 *MC*
4. if $MC_{sub} \subseteq MC$
5. if *MC* 对应的摄像头与上一个包含 MC_{sub} 的点伴随对应的摄像头相邻出现在序列数据库 D_+^r
6. 将 *MC* 对应的时间和摄像头加入 *platMap*;
7. *platIndex.put*(MC_{sub},*platIndex.get*(MC_{sub})+1);
8. else
9. if *platIndex.get*(MC_{sub})$\geqslant\delta$
10. 将 *MC* 对应的时间和摄像头加入 *platMap;*
11. *platIndex.put*(MC_{sub},1);
12. else
13. 删除 *platMap* 中 *key* 为 MC_{sub} 的值的最后一组相邻摄像头;
14. 将 *MC* 对应的时间和摄像头加入 *platMap*;
15. *platIndex.put*(MC_{sub},1);
16. if end
17. if end
18. if end
19. for end
20. for each *platIndex* 中的⟨*MC*,*length*⟩
21. if *length*<δ
22. 删除 *platMap* 中 *MC* 对应值的最后一组相邻摄像头;
23. if end
24. for end
25. for each *platMap* 的⟨*MC*,*value*⟩
26. if *value* 的长度小于δ_{cam}
27. 从 *platMap* 中删除该⟨*MC*,*value*⟩;
28. if end
29. for end
30. 返回 *platMap*;

最后,PlatoonFinder 算法调用了方法 *removeData*.对于过期数据 *r*,*removeData* 先将 *r* 从窗口中移除.如果已发现的封闭 Platoon 伴随模式(*P*,*C*,*T*),$P=\{p_1,\ldots,p_m\}$,$C=\langle c_1,\ldots,c_n\rangle$,$T=\{t_1,\ldots,t_n\}$涉及该数据,则有 $r.p\in\{p_1,\ldots,p_m\}$,$r.c=c_1$,$r.t\leqslant t_1$.对 *r* 进行如下操作:将 *r* 加入封闭 Platoon 伴随模式(*P*,*C*,*T*)的过期数据集 *E*;如果(*P*,*C*,*T*)的过期数据集 *E* 满足 $E.P=\{p_1,\ldots,p_m\}$,且存在 $r_0\in E$ 使得 $E\subseteq MC(r_0.c,r_0.t,\Delta t)$,清空 *E*,更新(*P*,*C*,*T*).

4.3 PlatoonFinder算法实现

本文将实现的算法嵌入了百度地图 API,可以在地图上实时呈现 PlatoonFinder 算法发现的伴随模式.用户

界面提供了算法的参数输入,分别包括点伴随时间阈值Δt 交通摄像头个数阈值δ及δ_{cam}.

用户调用 PlatoonFinder 算法后,用户界面将发现的 Platoon 伴随模式以车辆组群行驶轨迹的形式在百度地图上展现给用户,并将详细信息列在地图的右侧.

图 7 是嵌入百度地图 API 后 PlatoonFinder 算法的实现效果图.图中的摄像头标识代表了分布在道路上的交通摄像头,椭圆标识代表了交通摄像头下的点伴随,箭头代表了车辆组群的行驶轨迹.图 7 是在Δt=60s,δ=2,δ_{cam}=4 时发现的伴随模式.上述参数要求将一起驶过不少于 4 个交通摄像头,且局部相邻的交通摄像头不少于 2 个的伴随车辆展现在百度地图上.

Fig.7 Implementation of PlatoonFinder algorithm

图 7 PlatoonFinder 算法实现示意图

5 实验结果及分析

5.1 实验设置

本文的目标在于近实时地发现车牌识别流式大数据中的 Platoon 伴随模式.本节主要从有效性和效率方面对本文提出的 PlatoonFinder 算法进行验证.

- 参数设置

表 2 列出了实验使用的参数,其中,l 代表伴随模式的实际交通摄像头个数.

Table 2 Parameter settings

表 2 参数设置

参数名称	说明	取值
Δt	点伴随的时间约束;	Δt=20s,40s,60s,80s,100s,120s;
δ_{veh}	Platoon 伴随模式的最小车辆数量;	δ_{veh}=2;
δ	Platoon 伴随模式的相邻交通摄像头序列最小长度;	δ=1,2,l;
δ_{cam}	Platoon 伴随模式的交通摄像头序列最小总长度;	δ_{cam}=2,3,4,5,6,7,8,9,10;

- 对比算法

本文选取了 Apriori 算法和经典的基于投影的频繁序列挖掘算法 PrefixSpan 作为对比算法.我们在前期工作中,通过借鉴频繁项集挖掘经典算法 Apriori 算法设计了基于 Spark 的并行伴随车辆发现算法.本文将该算法作为对比算法之一.此外,本文将 Spark Mlib 自带的 PrefixSpan(http://spark.apache.org/mllib/)算法作为另一个对

比算法.本文利用滑动窗口机制,将上述算法实现为基于 Spark 框架的处理流数据的实时算法.

- 环境配置

本节实验的硬件环境为一台独立的电脑以及一个 5 个节点的集群.独立电脑的 CPU 为四核 Intel(R) Core(TM)i5-2400,3.10GHz;内存 4GB;硬盘 500GB.集群上每个节点核数为 2,拥有 4G 内存 150G 硬盘.独立电脑的软件环境为 Windows7 操作系统,Eclipse4.4.0 集成 jdk1.8.0.集群的软件环境如下:所有节点均为装有 CentOS-6.4 的虚拟机,jdk 版本为 1.8.0.集群所使用的 Spark 版本为 1.5.0,Hadoop 版本为 2.6.0.所有的结果存储在 HDFS 上.所有算法使用 Java 及 Scala 语言开发完成.

5.2 有效性验证

本节将对 PlatoonFinder 算法的有效性进行验证,并对实验结果进行分析.由前文介绍可知,Δt,δ和δ_{cam} 是算法的主要参数,不同的取值将会对算法结果产生不同的影响.为此,本节首先测试了这些参数的取值将会对算法最终发现的伴随模式数量产生何种影响,从而为算法的实际应用提供相应指导.下面介绍实验数据集和指标.

- 数据集

本节实验所用数据集是北京市真实车牌识别数据集,该数据集包含了从 2012-12-21 06:00:00 到 2012-12-30 20:00:00 共 10 天的 1 040 个交通摄像头产生的车牌识别数据.这 10 天数据由 8 天工作日和 2 天周末组成.数据集共存储了 54 316 820 条数据记录,包含了 4 669 229 辆车的相关信息.本文在该数据集上选取了若干子集进行实验.前期工作[10]对车牌识别数据的分析结果显示,每天的车牌识别数据可以分为 4 个部分:第 1 部分从 00:00:00 到 07:00:00,这一部分的数据量十分稀少;第 2 部分从 07:00:00 到 12:30:00,这一部分包含了早高峰;第 3 部分从 12:30:00 到 19:30:00,这一部分包含晚高峰,第 3 部分与第 2 部分包含的数据量基本持平;第 4 部分从 19:30:00 到 23:59:59,这一部分数据量介于第 2 部分(第 3 部分)与第 1 部分之间.不失一般性,本文在 10 天真实车牌识别数据集的每个部分各选取了 2 小时的车牌识别数据,共 80 个子数据集.并将选取的 80 个子数据集按原有速率还原为流数据接入 PlatoonFinder 算法.

定义 5(平均数量). 已知存在车牌识别数据集 $R_1,R_2,\ldots,R_k$.对每个数据集 R_i,PlatoonFinder 算法发现的 Platoon 伴随模式数量为 n_i,则 PlatoonFinder 算法在全部数据集 $R_1,R_2,\ldots,R_k$ 上发现的 Platoon 伴随模式平均数量为 $\bar{n}=\sum_{i=1}^{k}n_i/k$.

本文所有实验(包括效率验证)均将窗口大小设置为 30min.同时,为了尽可能保障不丢失伴随模式,本文的所有实验(包括效率验证)均将窗口的滑动距离设置为 1s.对于每个选取的子数据集,实验调用 PlatoonFinder 算法计算 Platoon 伴随模式.对结果四舍五入保留整数.

图 8 是有效性实验结果柱状图.如图所示:在不同的δ取值下(δ=1,δ=2,δ=l),PlatoonFinder 算法发现的 Platoon 伴随模式的平均数量随着δ_{cam}的增加都快速下降.图 8 还显示:当δ_{cam}增加时,不同的δ取值下,Platoon 伴随模式的平均数量之间的差异也发生相应的变化.首先,δ=1 时,平均数量明显多于其他两种模式;其次,当$\delta_{cam}\leqslant 8$ 时,δ=2 较δ=l 下的平均数量逐渐增加;当 $8<\delta_{cam}\leqslant 10$ 时,两种δ取值下的平均数量又趋于相等.通过分析δ_{cam}=9,10 时,δ=l 的详细实验结果发现,造成这一现象的原因如下:当算法 PlatoonFinder 发现并保留了封闭 Platoon 伴随模式(P,C,T),随着新数据的到达,发现并保留了新的封闭 Platoon 伴随模式(P,C',T'),且满足 $C\sqsubset C'$,$T\sqsubset T'$.根据定义 4,在对保留的实验结果进行检验时,(P,C,T)将被丢弃.而当δ=l 时,相同车辆组群产生的多个 Platoon 伴随模式不会合并.此外,当δ_{cam}较大时,在相同大小的窗口内包含的δ=l 下的 Platoon 伴随模式会减少.以上两点导致了 $8<\delta_{cam}\leqslant 10$ 时,δ=2 和δ=l 下的平均数量趋于相等.此外,进一步观察图 8 的实验结果,在参数δ和Δt 固定的情况下,随着δ_{cam}的增加,本方法发现的伴随车辆数量呈指数型下降.对于每一组δ和Δt 在不同δ_{cam}取值下的伴随模式进行分析发现,伴随车辆数量的下降规律如下:随着δ_{cam} 的增加,伴随模式数量的下降趋势呈现先减缓又加速再减缓的总体趋势.计算相邻δ_{cam}取值(例如δ_{cam}=2 和δ_{cam}= 3)下伴随模式数量的下降程度发现,伴随模式数量的平均下降程度分别为 77.67%,67.60%,68.83%,80.45%,87.354%,78.20%,80.03%和 75.91%.推断在δ_{cam} 取值较小时,车辆偶然伴随的可能性较高,因此,下降程度的第 1 次减缓可能是由于偶然性因素降低而造成的.当δ_{cam}>5,模式数量的下降程

度开始加速,推断是由于伴随区间对伴随模式数量的影响超过了偶然性因素的影响.δ_{cam}从 6 增加到 7 时下降程度最大,出现了拐点.当δ_{cam}>7 时,模式数量的下降程度再次开始减缓,推断当伴随区间足够长时,车辆具有更高可能性继续保持这种伴随的状态.

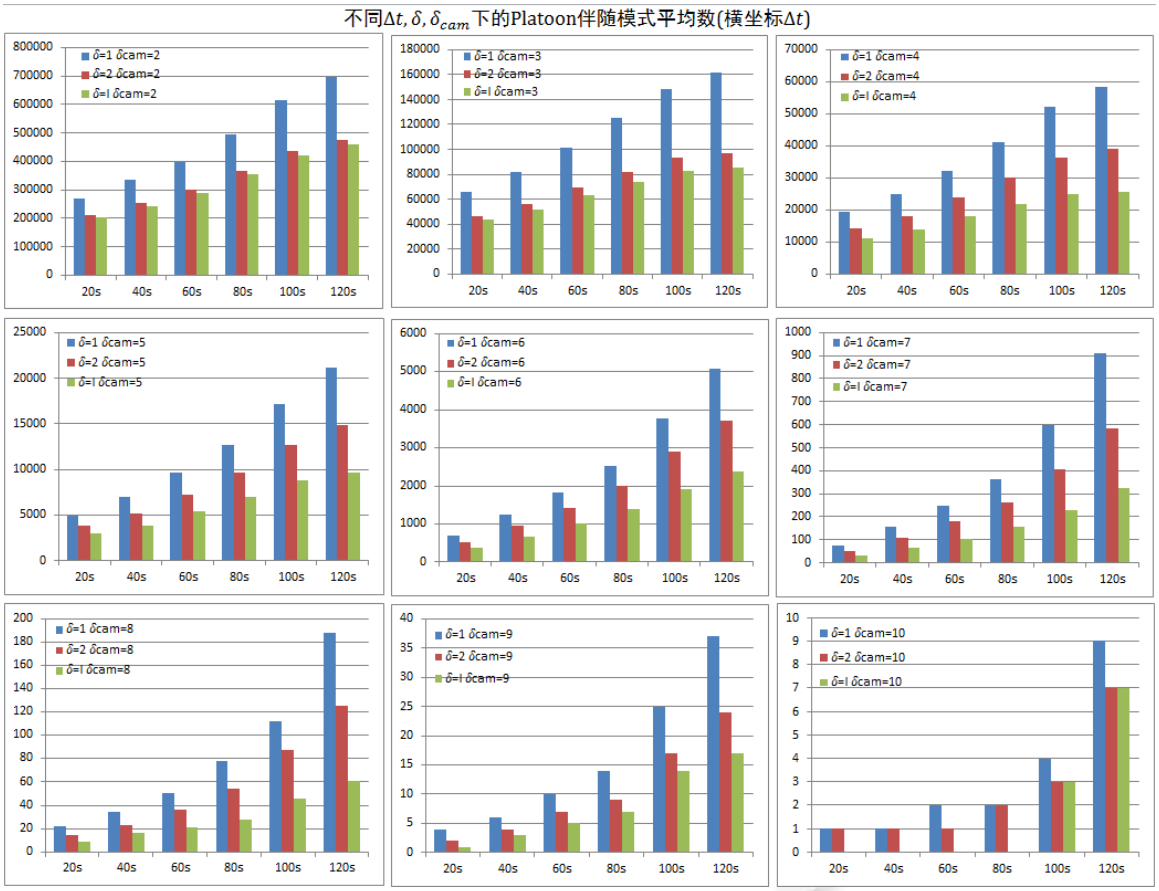


Fig.8 Viaration of average number under different Δt, δ, δ_{cam}

图 8 平均数量随参数Δt,δ,δ_{cam}变化趋势图

本节实验得出了以下结论:

(1) PlatoonFinder 算法发现的 Platoon 伴随模式数量与Δt 呈指数型上升;

(2) PlatoonFinder 算法发现的 Platoon 伴随模式数量与δ_{cam} 呈指数型下降.

5.3 效率验证

本节验证了 PlatoonFinder 算法的效率,并对实验结果进行了分析,测试了不同的阈值Δt 以及不同数据到达率对算法延时的影响.

定义 6(流数据的到达速率). 流数据的到达速率是指单位时间内的数据到达数量,记为 r_a.通常取 1s 为单位时间,即为 1s 内到达的车牌识别数据数量.

定义 7(平均延时). 设当前的滑动窗口为 *curWin*,窗口的滑动距离为 d_s.*curWin* 滑动 d_s 的长度到达下一窗 *nextWin*.算法在 *nextWin* 上计算得到伴随模式的时间为 t_i.设窗口的数量为 m,则算法的平均延时定义为

$$\overline{t}_{lat}=\sum\nolimits_i t_i/m.$$

首先,本文在有效性验证使用的 80 组真实车牌识别数据子集上测试δ=1,δ_{cam}=2 时,Δt=20s,40s,60s,80s,100s,

120s下,算法PlatoonFinder的平均延时,并绘制成图9.随后,为了验证吞吐量对平均延时的影响,本文在真实车牌识别数据的基础上共模拟了80组时长为2小时的数据集:数据到达率为10条/秒、100条/秒、1000条/秒、10000条/秒的数据集各20组.将这些数据集以流数据的形式接入算法,计算平均延时并绘制成图10.所有结果均保留两位小数.

图9是在不同的Δt取值下,PlatoonFinder算法平均延时实验结果柱状图.如图9所示,算法PlatoonFinder的平均延时随着Δt的增加呈指数型上升.当$\Delta t \leqslant 100$s时,算法的平均延时保持在1 000ms以下.而当Δt=100s时,算法的平均延时达到1 528.57ms.由于车牌识别数据的最小间隔时间为1s,图9的实验结果证明了当$\Delta t \leqslant 100$s时,算法PlatoonFinder能够在车辆通过摄像头时即时发现Platoon伴随模式.在这种条件下,算法对新到达的数据进行的Platoon伴随模式计算时不需要排队等待.而当$\Delta t \geqslant 120$s时,算法PlatoonFinder计算高峰时期(车牌识别数据到达率为每秒1条)的Platoon伴随模式时可能存在数据堆积现象.因此,本文的未来工作将聚焦于算法的并行优化,力求在$\Delta t \geqslant 120$s的条件下即时发现的Platoon伴随模式.

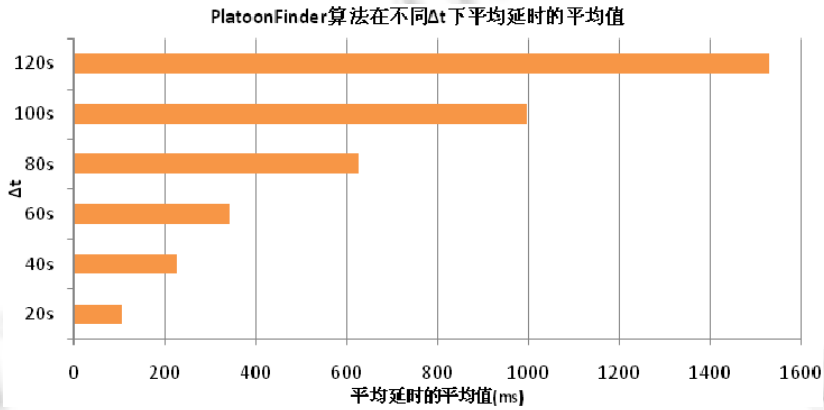


Fig.9 Viaration of average latency under different Δt (δ=1, δ_{cam}=2)

图9 不同Δt下,平均延时的平均值变化趋势图(δ=1,δ_{cam}=2)

图10是在从10条/秒到10 000条/秒不同流数据到达速率下我们的算法和Apriori算法、PrefixSpan算法的平均延时对比.

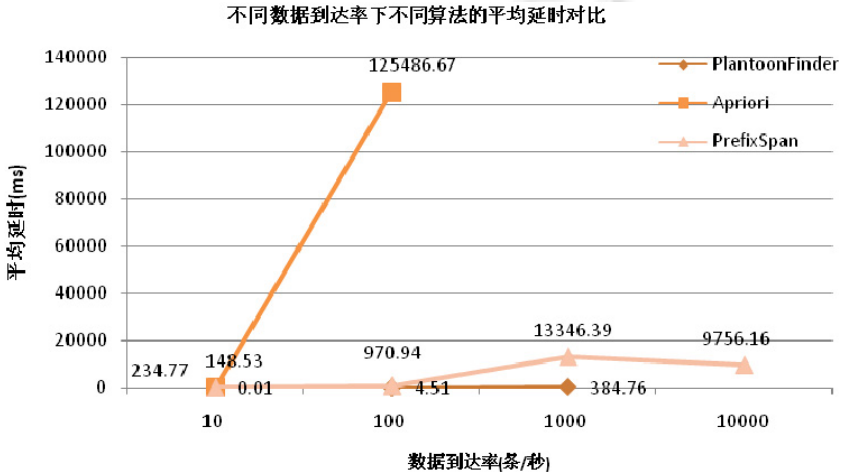


Fig10 Comparison of average latency of PlatoonFinder, Apriori and PrefixSpan under different arrival rate r_a

图10 不同数据到达速率r_a下PlatoonFinder、Apriori和PrefixSpan的平均延时对比

随着数据到达速率的增加,3 种算法的平均延时均大幅上升.其中,当到达速率为 10 条/时,3 种算法的延时均小于真实车牌数据的最小时间间隔 1s,在这种情况下,3 种算法都可以满足处理流数据的实时性要求.当到达速率上升至 100 条/秒时,3 种算法的平均延时都大幅上升.PlatoonFinder 和 PrefixSpan 算法的平均延时仍然低于 1s,而 Apriori 算法的平均延时已经高达 125s,无法满足实时处理流数据的需求.当到达速率上升至 1 000 条/秒时,Apriori 算法崩溃,这是由 Apriori 算法十分消耗内存而 Spark 框架是基于内存的计算框架而导致的.同时,PrefixSpan 算法的平均延时达到 133s,不足以处理流数据;而本文算法的平均延时为 384.76ms,仍然小于真实车牌数据的最小时间间隔 1s,可以满足实时性需求.当到达速率到达 10 000 条/秒时,本文算法崩溃,意味着处于单机环境的本文算法无法处理如此高速率的流数据.同时,基于 Spark 框架的 PrefixSpan 算法的平均延时却明显降低至 97s.这证明了 Spark 并行框架提升了 PrefixSpan 算法面对高吞吐量流数据的处理能力.基于此实验结果,在未来的工作中,我们考虑将本文算法并行化,以提升其面对高速流的处理能力.

6 总 结

本文针对一种特殊的流式时空大数据——车牌识别流式大数据,瞄准伴随车辆即时检测这一新兴的智能交通应用,提出了伴随车辆检测的核心算法.本文将 Platoon 伴随模式发现问题映射为数据流上的带有时空约束的频繁序列挖掘问题.为了有效应对车牌识别流式大数据的速率和规模,本文算法针对数据流的特点进行了性能优化,并融入了伪投影等性能优化技术,从而实现车辆 Platoon 伴随模式的即时发现.实验证明:在真实车牌识别数据集下,本文算法有效地发现伴随车辆组及其移动模式;然而当数据到达速率上升至 10 000 条/秒时,我们的算法已经无法应对如此高速的流数据.因此在未来的工作中,我们计划并行化本文的算法,以保障高速率的流数据实时处理.

References:

[1] Laube P, Imfeld S. Analyzing relative motion within groups of trackable moving point objects. In: Proc. of the 2nd Int'l Conf. on Advances in Geographic Information Systems (GIScience). Berlin: Springer-Verlag, 2002. 132−144. [doi: 10.1007/3-540-45799-2_10]

[2] Gudmundsson J, Van Kreveld M. Computing longest duration flocks in trajectory data. In: Proc. of the 14th Annual ACM Int'l Symp. on Advances in Geographic Information Systems (ACM-GIS). New York: Association for Computing Machinery, 2006. 35−42. [doi: 10.1145/1183471.1183479]

[3] Vieira MR, Bakalov P, Tsotras VJ. On-Line discovery of flock patterns in spatio-temporal data. In: Proc. of the 17th ACM Int'l Symp. on Advances in Geographic Information Systems (ACM SIGSPATIAL). New York: Association for Computing Machinery, 2009. 286−295. [doi: 10.1145/1653771.1653812]

[4] Jeung H, Shen HT, Zhou XF. Convoy queries in spatio-temporal databases. In: Proc. of the IEEE Int'l Conf. on Data Engineering (ICDE). Washington: IEEE Computer Society, 2008. 1457−1459. [doi: 10.1109/ICDE.2008.4497588]

[5] Jeung H, Yiu ML, Zhou XF, Jensen CS, Shen HT. Discovery of convoys in trajectory databases. Proc. of of the VLDB Endowment, 2008,1(1):1068−1080. [doi: 10.14778/1453856.1453971]

[6] Li ZH, Ding BL, Han JW, Kays R. Swarm: Mining relaxed temporal moving object clusters. Proc. of of the VLDB Endowment, 2010,3(1):723−734. [doi: 10.14778/1920841.1920934]

[7] Tang LA, Zheng Y, Yuan J, Han JW, Leung A, Peng WC, Porta TL. A framework of traveling companion discovery on trajectory data streams. ACM Trans. on Intelligent Systems & Technology, 2013,5(1):992−999. [doi: 10.1145/2542182.2542185]

[8] Li YX, Bailey J, Kulik L. Efficient mining of platoon patterns in trajectory databases☆. Data and Knowledge Engineering, 2015, 100(PA):167−187. [doi: 10.1016/j.datak.2015.02.001]

[9] Han YB, Wang GL, Yu J, Liu C, Zhang ZM, Zhu ML. A service-based approach to traffic sensor data integration and analysis to support community-wide green commute in China. IEEE Trans. on Intelligent Transportation Systems, 2015,17(9):1−10. [doi: 10.1109/TITS.2015.2498178]

[10] Zhu ML, Liu C, Wang JW, Wang XB, Han YB. Instant discovery of moment companion vehicles from big streaming traffic data. In: Proc. of the Int'l Conf. on Cloud Computing and Big Data (CCBD). Piscataway: IEEE, 2015. 73−80. [doi: 10.1109/CCBD.2015.65]

[11] Kalnis P, Mamoulis N, Bakiras S. On discovering moving clusters in spatio-temporal data. In: Proc. of the 9th Int'l Conf. on Advances in Spatial and Temporal Databases (SSTD). Berlin: Springer-Verlag, 2005. 364−381. [doi: 10.1007/11535331_21]

[12] Li YF, Han JW, Yang J. Clustering moving objects. In: Proc. of the 10th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining (KDD). New York: Association for Computing Machinery, 2004. 617−622. [doi: 10.1145/1014052.1014129]

[13] Kriegel HP, Pfeifle M. Density-Based clustering of uncertain data. In: Proc. of the ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining (KDD). New York: Association for Computing Machinery, 2005. 672−677. [doi: 10.1145/1081870.1081955]

[14] Jensen CS, Lin D, Ooi BC. Continuous clustering of moving objects. IEEE Trans. on Knowledge and Data Engineering, 2007,19(9): 1161−1174. [doi: 10.1109/TKDE.2007.1054]

[15] Zheng K, Zheng Y, Yuan NJ, Shang S. Online discovery of gathering patterns from trajectories. In: Proc. of the 29th Int'l Conf. on Data Engineering (ICDE). Washington: IEEE Computer Society, 2013. 242−253. [doi: 10.1109/ICDE.2013.6544829]

[16] Zhang JM, Li JL, Wang SG, Liu ZH, Yuan Q, Yang FC. On retrieving moving objects gathering patterns from trajectory data via spatio-temporal graph. In: Proc. of the IEEE Int'l Congress on Big Data (BIGDATACONGRESS). Piscataway: IEEE, 2014. 390−397. [doi: 10.1109/BigData.Congress.2014.64]

[17] Yoo JS, Boulware D, Kimmey D. A parallel spatial co-location mining algorithm based on mapreduce. In: Proc. of the IEEE Int'l Congress on Big Data (BIGDATACONGRESS). Piscataway: IEEE, 2014. 25−31. [doi: 10.1109/BigData.Congress.2014.14]

[18] Yu YW, Wang Q, Wang XD. Continuous clustering trajectory stream of moving objects. China Communications, 2013,10(9): 120−129. [doi: 10.1109/CC.2013.6623510]

[19] Yu YW, Wang Q, Wang XD, Wang H, He J. Online clustering for trajectory data stream of moving objects. Computer Science and Information Systems, 2013,10(3):1293−1317. [doi: 10.2298/CSIS120723049Y]

[20] Agrawal R, Srikant R. Mining sequential patterns. In: Proc. of the IEEE Int'l Conf. on Data Engineering (ICDE). Washington: IEEE Computer Society, 1995. 3−14.

[21] Srikant R, Agrawal R. Mining sequential patterns: Generalizations and performance improvements. In: Proc. of the Advances in Database Technology (EDBT), Vol.1507. Berlin: Springer-Verlag, 1996. 1−17. [doi: 10.1007/BFb0014140]

[22] Zaki MJ. SPADE: An efficient algorithm for mining frequent sequences. Machine Learning, 2001,42(1):31−60. [doi: 10.1023/A:1007652502315]

[23] Ayres J, Flannick J, Gehrke J, Yiu T. Sequential pattern mining using a bitmap representation. In: Proc. of the 8th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining (KDD). New York: Association for Computing Machinery, 2002. 429−435. [doi: 10.1145/775047.775109]

[24] Han JW, Pei J, Mortazavi-Asl B, Chen Q, Dayal U, Hsu MC. FreeSpan: Frequent pattern-projected sequential pattern mining. In: Proc. of the ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining (KDD). New York: Association for Computing Machinery, 2000. 355−359. [doi: 10.1145/347090.347167]

[25] Pei J, Han J, Mortazavi-Asl B, Pinto H, Chen Q, Dayal U, Hsu MC. PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth. In: Proc. of the 17th Int'l Conf. on Data Engineering (ICDE). Washington: IEEE Computer Society, 2001. 215−224. [doi: 10.1109/ICDE.2001.914830]

[26] Yan XF, Han JW, Afshar R. CloSpan: Mining closed sequential patterns in large datasets. In: Proc. of the 2003 SIAM Int'l Conf. on Data Mining (SDM). Philadelphia: SIAM, 2003. 166−177. [doi: 10.1137/1.9781611972733.15]

[27] Wang JY, Han JW, Li C. Frequent closed sequence mining without candidate maintenance. IEEE Trans. on Knowledge and Data Engineering, 2007,19(8):1042−1056. [doi: 10.1109/TKDE.2007.1043]

[28] Helen P, Han JW, Pei J, Wang K, Chen QM, Dayal U. Multi-Dimensional sequential pattern mining. In: Proc. of the 10th Int'l Conf. on Information and Knowledge Management (CIMK). New York: Association for Computing Machinery, 2001. 81−88. [doi: 10.1145/502585.502600]

[29] Pei J, Han JW, Wang W. Constraint-Based sequential pattern mining: The pattern-growth methods. Journal of Intelligent Information Systems, 2007,28(2):133−160. [doi: 10.1007/s10844-006-0006-z]

[30] Chueh HE. Mining target-oriented sequential patterns with time-intervals. Int'l Journal of Computer Science & Information Technolo, 2010,2(4):113−123. [doi: 10.5121/ijcsit.2010.2410]

[31] Demiriz A. webSPADE: A parallel sequence mining algorithm to analyze Web log data. In: Proc. of the IEEE Int'l Conf. on Data Mining (ICDM). Piscataway: IEEE, 2002. 755−758. [doi: 10.1109/ICDM.2002.1184046]

[32] Guralnik V, Karypis G. Parallel tree-projection-based sequence mining algorithms☆. Parallel Computing, 2004,30(4):443−472. [doi: 10.1016/j.parco.2004.03.003]

[33] Ma CX, Li QH. Parallel algorithm for mining frequent closed sequences. In: Proc. of the Workshop on Autonomous Intelligent Systems: Agents and Data Mining (AIS-ADM). Berlin: Springer-Verlag, 2005. 184−192. [doi: 10.1007/11492870_15]

[34] Qiao SJ, Tang CJ, Dai SC, Zhu MF, Peng J, Li HJ, Ku YC. PartSpan: Parallel sequence mining of trajectory patterns. In: Proc. of the 5th Int'l Conf. on Fuzzy Systems and Knowledge Discovery (FSKD). Washington: IEEE Computer Society, 2008. 363−367. [doi: 10.1109/FSKD.2008.33]

[35] Yu DJ, Wu W, Zheng SH, Zhu ZX. BIDE-Based parallel mining of frequent closed sequences with mapreduce. In: Proc. of the 12th Int'l Conf. on Algorithms and Architectures for Parallel Processing (ICA3PP). Berlin: Springer-Verlag, 2012. 177−186. [doi: 10.1007/978-3-642-33065-0_19]

[36] Kessl R. Probabilistic static load-balancing of parallel mining of frequent sequences. IEEE Trans. on Knowledge and Data Engineering, 2016,28(5):1299−1311. [doi: 10.1109/TKDE.2016.2515622]

[37] Mooney CH, Roddick JF. Sequential pattern mining: Approaches and algorithms. ACM Computing Surveys, 2013,45(2):94−111. [doi: 10.1145/2431211.2431218]

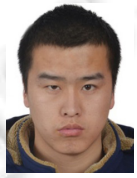
[38] Chang JH, Lee WS. Finding recent frequent itemsets adaptively over online data streams. In: Proc. of the 9th ACM Int'l Conf. on Knowledge Discovery and Data Mining (SIGKDD). New York: Association for Computing Machinery, 2003. 487−492. [doi: 10.1145/956750.956807]

[39] Leung CS, Khan QI. DSTree: A tree structure for the mining of frequent sets from data streams. In: Proc. of the 6th IEEE Int'l Conf on Data Mining (ICDM). Piscataway: IEEE, 2006. 928−932. [doi: 10.1109/ICDM.2006.62]

[40] Li J, Maier D, Tufte K, Papadimos V, Tucker PA. No pane, no gain: Efficient evaluation of sliding-window aggregates over data streams. SIGMOD Record, 2005,34(1):39−44. [doi: 10.1145/1058150.1058158]

[41] Mozafari B, Thakkar H, Zaniolo C. Verifying and mining frequent patterns from large windows over data streams. In: Proc. of the 24th Int'l Conf on Data Engineering (ICDE). Washington: IEEE Computer Society, 2008. 179−188. [doi: 10.1109/ICDE.2008.4497426]

朱美玲(1987－),女,辽宁营口人,硕士,主要研究领域为服务计算,大数据.



王雄斌(1992－),男,主要研究领域为服务计算,大数据.

刘晨(1980－),男,博士,副研究员,CCF 专业会员,主要研究领域为服务计算,大数据.

韩燕波(1962－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为服务计算,大数据.