Journal of Computer Applications
计算机应用,2015,35(11):3203-3207
ISSN 1001-9081
CODEN JYIIDU
2015-11-10
http://www.joca.cn

文章编号:1001-9081(2015)11-3203-05
doi:10.11772/j.issn.1001-9081.2015.11.3203

基于车牌识别大数据的伴随车辆组发现方法

曹　波[1,2*],韩燕波[1,2],王桂玲[2]
(1.山东科技大学 信息科学与工程学院,山东 青岛 266590;
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)
(*通信作者电子邮箱 18612183347@163.com)

摘　要:基于对车牌识别大数据的处理与分析,可以完成伴随车辆组的发现,在涉案车辆追踪等方面具有广泛的应用。然而当前单一机器模式下伴随车辆组发现算法存在时间和空间上处理性能低下等问题。针对此问题,提出了一种伴随车辆组发现方法——FP-DTC 方法。该方法将传统的 FP-Growth 算法利用分布式处理框架 Spark 进行了并行化,并作了相应的改进和优化来更加高效地发现伴随车辆组。实验结果的分析表明,提出的方法能够很好地解决车牌识别大数据上的伴随车辆组发现问题,性能相比采用同样方法的 Hadoop 实现提升了近 4 倍。

关键词:智能交通系统;车牌识别;伴随车辆组;FP-Growth 算法;Spark 并行框架
中图分类号:TP391.41　　**文献标志码**:A

Discovery method of travelling companions based on big data of license plate recognition

CAO Bo[1,2*], HAN Yanbo[1,2], WANG Guiling[2]
(1. *College of Information Science and Engineering, Shandong University of Science and Technology, Qingdao Shandong* 266590, *China*;
2. *Beijing Key Laboratory on Integration and Analysis of Large-scale Stream Data (North China University of Technology), Beijing* 100144, *China*)

Abstract: The discovery of travelling companions based on processing and analysis of the license plate recognition big data has become widely used in many aspects such as the involved vehicle tracking. However, discovery algorithms of travelling companions have poor performance in single machine mode no matter in time and space. To solve this problem, a discovery method of travelling companions named FP-DTC was proposed. This method based on the algorithm of FP-Growth was parallelled by the distributed processing framework- Spark, and had made some improvement and optimization to discover the travelling companions more efficiently. The experimental results show that, this method performs well on the discovery of travelling companions, and achieves an increase of nearly four times than the same algorithm with Hadoop.

Key words: Intelligence Transportation System (ITS); license plate recognition; travelling companions; FP-Growth algorithm; Spark parallel framework

0　引言

随着科技的发展,通过使用传感器、位置捕获和跟踪设备等技术产生了大量的位置相关方面的数据,智能交通系统(Intelligence Transportation Systems, ITS)领域的应用程序开始利用这些交通数据,来记录车辆移动和交通轨迹的动态生成情况[1]。车牌自动识别(Automatic Number Plate Recognition, ANPR)数据是对交通摄像头捕捉到的道路交通数据进行处理生成的数据。ANPR 数据每时每刻都在不停地产生,形成了庞大的数据规模。

现代社会道路监控技术发展的同时,违法犯罪行为与车辆、交通系统的联系也越来越密切。伴随车辆是一个交通术语,是指在一定时间内与追踪车辆以一定概率存在伴随关系的车辆。如果事先知道涉案车辆的车牌号,可以直接通过查询 ANPR 数据找出其伴随车辆,然而实际情况中往往并不知道涉案车辆的车牌号,在这种情况下就需要通过伴随车辆组发现方法从海量的 ANPR 数据中寻找出经常一起出现的伴随车辆,提供给公安机关进行排查。

在涉案车辆追踪服务应用中,可以对海量 ANPR 数据进行分析处理,为公安部门办案中的犯罪嫌疑车辆排查分析提供参考。本文的主要贡献是:1)提出了一种基于并行 FP-Growth 算法的伴随车辆组发现方法——FP-DTC 方法。该方法对关联分析中的 FP-Growth 算法作了并行化的改进和优化,解决了车牌识别大数据处理中涉及到的频繁子集挖掘问题;2)利用云计算环境下的分布式并行处理框架 Spark,实现了该算法。经过实验验证该方法能够很好地处理海量 ANPR 数据,解决了单机模式下的内存不足等问题,在伴随车辆组分析发现上的性能得到了提升。

1　问题的提出

伴随车辆组的发现是从 ANPR 数据集中的不同车辆之间的联系来分析车辆的行驶习惯,通过了解哪些车辆频繁地在

收稿日期:2015-06-17;**修回日期**:2015-07-28。
基金项目:北京市属高等学校创新团队建设与教师职业发展计划项目(IDHT20130502);北京市自然科学基金重点项目(4131001)。
作者简介:曹波(1990-),男,山东莱芜人,硕士研究生,主要研究方向:云计算、大规模数据集成;　韩燕波(1962-),男,河北承德人,教授,博士生导师,博士,CCF 高级会员,主要研究方向:云计算、互联网服务;　王桂玲(1978-),女,山东菏泽人,副教授,博士,CCF 会员,主要研究方向:数据集成、服务组合。

多个监测点共同出现来分析它们之间的相互关系，本质上就是寻找不同车辆之间的关联性或相关性，因此可以使用关联分析方法来解决。点伴随是指在一定的时间范围内共同经过同一监测点的车辆所具有的一种伴随关系，具有点伴随关系的车辆共同组成点伴随组。前面提到伴随车辆是在一定时间内与追踪车辆以一定概率存在伴随关系的车辆，实际场景中这个概率通常指设定的监测点阈值与点伴随车辆共同经过的监测点数目的比值。因此可以通过对点伴随组进行关联分析，找出满足这一概率的频繁子集车辆，即可求解出伴随车辆组，作为涉案车辆重点追踪和排查的对象。

当前的车辆数据越来越多，据统计，中国一个大型城市部署的带车牌识别功能的摄像头可达到5000个，高峰期每个摄像头车牌识别数据的采集频率可达每秒1条，每天的交通高峰折算率按0.33统计，则一天的车辆识别数据记录数将达到1.44亿条，数据量约12 GB[2]。面对如此大量的ANPR数据，利用关联分析方法在单台机器上分析求解伴随车辆组存在大量的计算和存储负担，效率偏低。

目前一些先进的伴随车辆组发现方法及技术被用于全球定位系统(Global Positioning System, GPS)的数据分析[3]，而本文所研究的ANPR数据与GPS数据不同，其记录的位置由于摄像头固定等原因一般都是有限制的，其方法和技术并不完全适用于ANPR数据。文献[4]提出的伴随车辆查询(Accompany Vehicle Discovery, AVD)方法虽然可以适用于ANPR数据的分析，但其采用的滑动时间窗口技术仅在求解点伴随组上提升了效率，最后利用关联分析算法求解伴随车辆时摆脱不了单台机器的计算能力限制。

基于以上两个原因，需要考虑一种新的高效的方法来解决伴随车辆组的发现问题。本文提出的FP-DTC方法，通过使用分布式处理框架Spark实现的并行FP-Growth算法来从车牌识别大数据中更加高效地发现伴随车辆组。

2 伴随车辆组发现方法——FP-DTC方法

计算伴随车辆组，需要综合数天的车牌识别数据进行分析处理，本文采用一种基于多过程并行模式的处理方法(简称为FP-DTC方法)。首先，需要对ANPR数据集进行预处理，过滤掉不符合要求的数据，仅保留计算过程中需要的字段值；然后，将过滤后的数据集按时间先后排序，根据车牌号生成每辆车的车辆轨迹；再根据所得的车辆轨迹计算各监测点下的点伴随组；最后，根据点伴随组求得伴随车辆组。在这一章中将具体介绍这些过程的实现方法。

2.1 交通数据的预处理

ANPR数据集中的每一条记录均包含多个字段，由于所捕获的监测点数据有限导致某些字段的值缺失或者某些字段对于当前的数据分析处理没有任何意义，这样的数据在车辆轨迹判定中很难发挥作用。因此本文方法通过Spark中的过滤函数将数据集并行的处理成只包含〈车牌号，监测点，时间点〉(简写为〈v, s, $time$〉)3个字段的数据集，从而降低参与后续计算的数据规模，提高处理速度。

2.2 车辆轨迹和点伴随组的生成

车辆轨迹是一段时间内车辆所经过的监测点位置序列。对过滤后的数据集先按照车牌号分组，然后根据监测时间先后排序，最终得到在一定日期时间范围内的车辆轨迹。步骤如图1所示。

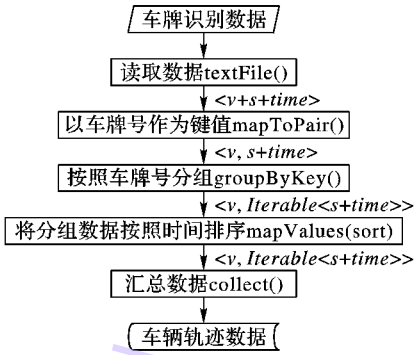

图1 基于Spark产生车辆轨迹的流程

1)使用textFile方法读取ANPR数据集并将其转换为相同格式的弹性分布式数据集(Resilient Distributed Dataset, RDD)形式，具体为HadoopRDD，其包含〈v,s,$time$〉3个字段的信息；2)通过mapToPair方法以车牌号作为键，监测点和时间作为值将RDD从DRDD转换为PairRDD的形式，其格式为〈v,s + $time$〉；3)然后通过groupByKey方法将PairRDD按照键v进行分组，将具有相同键值v的数据放在一起，形成另一种形式的PairRDD，格式为〈v, $Iterable$〈s + $time$〉〉，其中键v不变，值为具有相同键v的一组数据；4)再通过mapValues方法实现对PairRDD中的数据排序的功能，该方法将对同一车牌号下的数据按照时间先后排序。5)最后使用collect方法得出车辆轨迹数据，其格式为 $List$〈v, $Iterable$〈s + $time$〉〉。

本文算法都是基于Spark实现的，而弹性分布式数据集(RDD)是Spark最基本的数据抽象。它是一个容错的、并行的数据结构，可以让用户显式地将数据存储到磁盘和内存中，并能控制数据的分区[5]。同时，RDD还提供了一组丰富的操作可以像在MapReduce中处理数据一样并行地操作数据，如flatMap、filter、mapToPair、groupByKey等操作。

得出车辆轨迹数据后，基于这些轨迹数据对每一个监测点和相同监测点下的每一辆车进行迭代，当满足时间阈值时，将该车辆加入点伴随组G，其数据格式为〈s, v_1:v_2:v_3:…〉。

2.3 伴随车辆组的发现

为了方便地分析求解问题，本文为伴随车辆组作了如下的形式化定义：

设q是点伴随组G的一个子集，δ_{com}为监测点阈值，n_{com}为q中车辆共同经过的监测点数目，当且仅当$n_{com} \geqslant \delta_{com}$时，称$q$中的车辆互为伴随车辆，$q$称为伴随车辆组。

下面以图2为例简单介绍下发现伴随车辆组的过程。

图2中，共有$\{s_1, s_2, \cdots, s_6\}$6个监测点，$\{v_1, v_2, \cdots, v_{10}\}$10辆车，横坐标是车辆经过某监测点的时间，纵坐标是监测点的位置。假定监测点阈值为5，时间阈值为30 min，车辆组$\{v_1, v_2, v_7, v_3, v_4\}$和$\{v_8, v_{10}, v_5\}$在时间阈值内共同经过了同一个监测点$s_1$，则它们共同组成一个点伴随组。从图中可以看出，车辆组$\{v_1, v_2, v_3, v_4\}$、$\{v_5, v_{10}\}$都是此点伴随组的子集，车辆组$\{v_1, v_2, v_3, v_4\}$共同经过了$\{s_1, s_2, s_3, s_5\}$4个监测点，而车辆组$\{v_5, v_{10}\}$共同经过了$\{s_1, s_2, s_3, s_4, s_6\}$5个监测点，所以只有车辆组$\{v_5, v_{10}\}$满足大于等于监测点阈值的条件，在这种情况下，车辆$v_5$, v_{10}共同组成伴随车辆组。

上一节中求出点伴随组后，其子集均为共同经过某一监

测点的车辆或车辆组,根据前面给出的伴随车辆组的形式化定义,要想求得伴随车辆组,需要找出满足共同经过的监测点数目超过监测点阈值的所有点伴随组子集,因此可以使用关联分析算法对点伴随组进行频繁子集挖掘即可,求得的这些点伴随组的子集就是伴随车辆组。目前传统的频繁项集挖掘主要包括两大类算法,基于 Apriori 的挖掘算法和基于模式增长(FP-Growth)类的算法。其中 FP-Growth 算法摆脱了 Apriori 算法必须产生候选项集的方式,提高了数据的挖掘效率[6]。

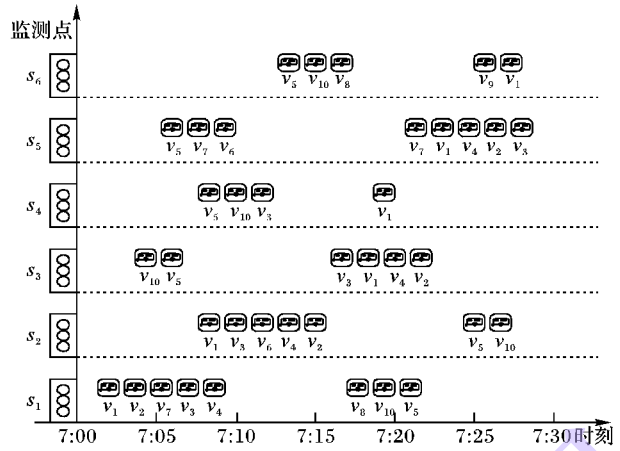

图 2　计算伴随车辆组的例子

传统 FP-Growth 算法的基本思路是:不断地迭代 FP-Tree 的构造和投影过程,对 FP-Tree 进行递归挖掘找出所有的频繁项集。该算法需要扫描两次事务集:第 1 次扫描事务集求出频繁 1 项集,并按照支持度降序排列;第 2 次扫描事务集,对于每个频繁项,构造它的条件投影数据库和投影 FP-Tree;对每个新构建的 FP-Tree 重复这个过程,直到构造的新 FP-Tree为空,或者只包含 1 条路径。当构造的 FP-Tree 为空时,其前缀即为频繁模式;当只包含 1 条路径时,通过枚举所有可能组合并与此树的前缀连接即可得到频繁模式[7]。

由于传统的 FP-Growth 算法对于 FP-Tree 的构造是在内存中进行的,当数据规模很大时,FP 树的内存占用会相当可观,同时 FP-Tree 的构造过程也需要很高的运算性能。本文基于 Spark 框架将 FP-Growth 算法进行了并行化的改进和优化,使其可以根据事务集的规模进行分组,将事务集均衡地分配到每个节点下进行并行计算来提高运算效率。基于 Spark 的并行 FP-Growth 处理计算框架如图 3 所示。

图 3 所示框架展示了算法的 4 个步骤:1)首先通过一个并行计算过程,如 mapToPair、groupByKey 等求出频繁 1 项集,统计事务项频繁度并按其降序排列。2)为了达到负载均衡的效果,并且保证每组相对独立,以便后续处理更方便,要对数据进行平衡分组。通过利用频繁 1 项集的结果建立 Hash 表,按照 Hash 分组策略第 2 次扫描事务集将其分组。假设有 m 个节点,n 个频繁 1 项集,数据分解后的空间复杂度就减小到 $O(n/m)$。3)对分组后的事务集进行一定的并行处理后将其分配到各个节点单独计算各分组的子频繁项集,各节点从条件 FP-Tree 分单分支和多分支两种情况进行本地递归挖掘频繁项集。4)最后对各个节点的频繁子集进行汇总。其伪代码如算法 1 所示。

算法 1　基于点伴随组生成伴随车辆组 genFrequentSet。

输入　点伴随组 G,监测点阈值 δ_{com};

输出　伴随车辆组数据集 Q。

```
1)  freqset_1 = FPGStepOne(G, δcom);
2)  FPGStepTwo(G, freqsubset_1, δcom)
3)    DRDD = SparkConf.textFile(G);
4)    mapToPair(DRDD)
5)    groupByKey(s, Gi)
6)    //将事务分组到每个节点
7)    List(Gi) = Grouping(G, freqset_1)
8)    //在各个节点下运行本地 FP-Growth 算法
9)    LocalFPTree(Gi, δcom, null)
10)     // 构建项头表
11)     headerTable = buildHeaderTable(Gi, δcom)
12)     //构建 FP-Tree
13)     buildFPTree(Gi, headerTable)
14)       for each(gj) in Gi
15)         sortByHeaderTable(gj, headerTable)
16)         addNodes(TreeNode, gj, headerTable)
17)       end for
18)     end buildFPTree
19)   //递归以求子 FP-Tree
20)   LocalFPTree(Gj, δcom, item)
21)   Qi.add(Iterable(freqSubset));
22)   end LocalFPTree
23) end FPGStepTwo
24) Q = CollectFromEachNode(Qi)
25) return Q;
```

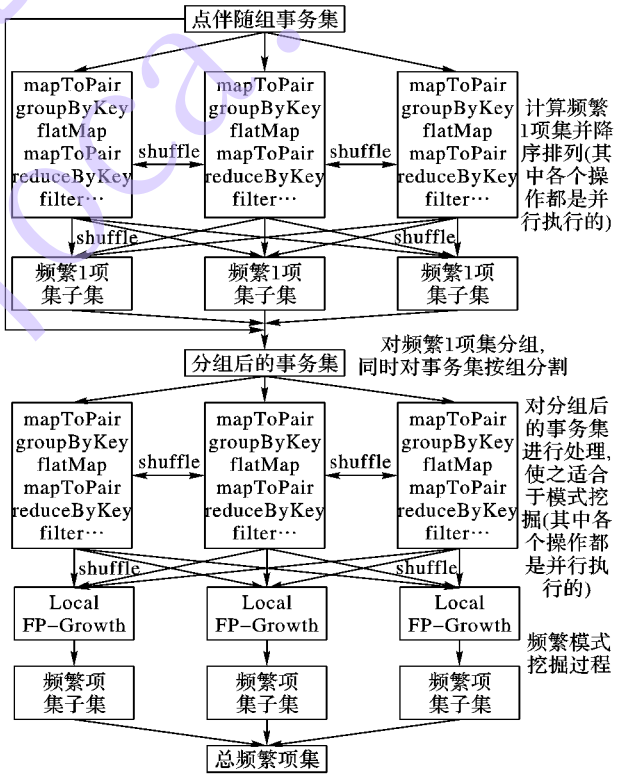

图 3　并行 FP-Growth 处理计算框架

3　实验与分析

为了有效验证本文提出的利用分布式并行处理框架 Spark 实现的 FP-Growth 算法来发现伴随车辆组方法的有效性,搭建了基于 Spark 集群的实验环境并进行了多组实验。

3.1　实验环境与数据

本文的 Spark 集群采用基于 Yarn 的资源调度模式,由 5

台装载 CentOS release 6.4 系统,Spark-1.1.0 以及 Hadoop-2.3.0软件的服务器搭建而成,内存主节点配置 6 GB,从节点机器配置为 3 GB,其他硬件配置均相同。实验中采用的数据为北京市 2012 年 11 月 13 日到 11 月 19 日 7 天全天采集到的真实车牌识别数据,每天的数据记录约 970 万余条,涉及约 230 万辆车,1 794 个道路监测点,实验中的所有数据均存储在同一个 HDFS 集群中。

3.2 实验结果与分析

1)性能测试与分析。

本文方法通过在相同规模数据与硬件配置环境下,测试单机算法与并行算法在 Spark 集群中单个计算节点下的执行时间来说明采用分布式计算的必要性,通过测试 FP-DTC 方法在不同的并行计算框架下的执行时间来评估该算法的性能。如测试 10 min、20 min、…、60 min 时间之内的交通数据分别在单机算法和并行算法框架下,以及分别在 Hadoop 和 Spark 框架下执行 5 次的平均时间。

表 1 展示了单机 FP-Growth 算法和并行 FP-DTC 算法 Spark 集群单节点下的实验结果对比。

表 1 单台机器处理实验结果

数据集规模/min	单机算法/s	并行算法/s
10	32	51
20	75	83
30	99	134
40	内存不足	155
50	内存不足	180
60	内存不足	202

从表 1 中可以看到,当输入数据规模很小时,并行算法处理效率低于单机 FP-Growth 算法,这是因为 Spark 集群启动和分配任务时需要消耗一定的时间和资源,且在总运行时间中占据很大的比例。随着数据规模的增长,单机算法内存消耗增大,剩余内存不足以支撑计算任务;而并行算法由于具有良好的并行框架 Spark 的支持,进行适当的内部资源调度,最终能够完成计算任务。这充分说明了在单一机器模式下不足以处理和分析车牌识别大数据,利用集群并行处理数据是很有必要的。

图 4 展示了在两种不同的并行计算框架下实现的 FP-DTC 算法的性能比较。从图中可以看出,Spark 实现的算法执行时间明显短于用 Hadoop 实现的算法,性能相比提升了近 4 倍。这是由于 Spark 框架的每一次迭代都是基于内存计算的,而 Hadoop 则需要频繁地读写磁盘,耗费了大量的时间。还可以看到随着数据规模的增大,前者增长幅度明显小于后者。因此通过使用 Spark 框架实现该 FP-DTC 方法能够很好地解决车牌识别大数据上的伴随车辆组发现问题。

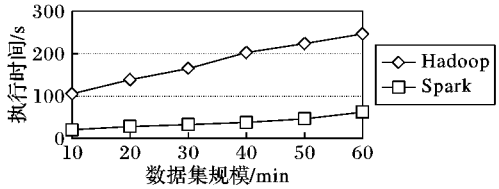

图 4 不同框架下的性能对比

2)关键参数影响测试与分析。

在计算伴随车辆组的过程中,有两个参数阈值的调整对结果具有很大的影响,分别是时间阈值 δ_t 和监测点阈值 δ_{com}。

时间阈值 δ_t 是产生点伴随组的基础。首先设定数据集的时间范围为 30 min,监测点阈值为 4,然后依次将 δ_t 设置为 1 min,2 min,3 min,…,10 min,计算算法执行 5 次的平均时间。

图 5 展示了不同时间阈值下算法的执行性能。随着时间阈值的增大,算法的执行时间成本相应的增加,这是因为点伴随组的数量规模也在不断增加,此时可以通过扩展集群节点的方法降低程序执行时间。

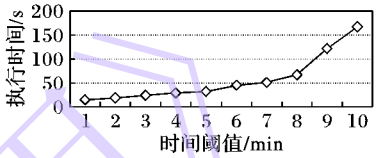

图 5 不同时间阈值下的处理性能

监测点阈值 δ_{com} 是计算伴随车辆组的基础。为了测试其对结果的影响,设定数据集的时间范围为 30 min,时间阈值为 5 min,监测点阈值 δ_{com} 为 3,4,5,6,7,8。从图 6 可以看出算法的执行时间随着监测点阈值的增大而减小,这是因为每一次迭代计算的数据规模都变小了。

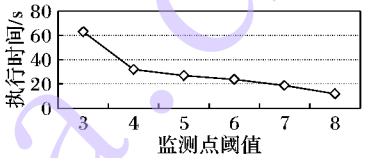

图 6 不同监测点阈值下的处理性能

4 相关工作

虽然伴随车辆组发现一直是智能交通领域的一个研究热点,然而,基于交通大数据的集成与分析研究尚处于起始阶段,本文总结了当前的相关工作如下:

1)伴随车辆组发现。近年来,移动对象的伴随组发现与查询成为移动对象数据管理领域的研究热点[8]。很多研究者也提出了许多的计算方法,如 Grid-based、Euclidean distance、动态时间归整(Dynamic Time Warping, DTW)等方法[9-11],但在最初的研究中它们很多都缺乏对移动对象轨迹时间属性的考虑以及偏侧重于对全球定位系统(GPS)数据的分析处理,并不适合位置可测但车辆对象不定的车牌识别数据,而文献[4]中的伴随车辆查询(AVD)方法虽然适用于车牌识别数据,但是同以上方法一样并没有从并行化的角度来考虑算法的性能问题。

2)大数据的处理框架。随着数据量的不断增加,越来越多的并行编程框架被用在加速大数据的处理之中,如 Hadoop 框架、Dryad 框架、Storm 框架等,但是这些框架并不适合用来进行迭代式计算和交互式计算。Spark 是开源的类 Hadoop MapReduce 的通用的并行计算框架,拥有 Hadoop MapReduce 所具有的优点,不同于 MapReduce 的是,job 的中间输出和结果可以保存在内存中,从而不再需要读写 HDFS,因此 Spark 能更好地适用于数据挖掘与机器学习中需要迭代的算法。

3)关联分析领域的工作。当前数据挖掘领域的很多算

法都已经实现了并行化,对于并行的关联规则挖掘,Agrawal 等在文献[12]中提出了计数分布(Count Distribution,CD)、候选分布(Candidate Distribution,CaD)和数据分布(Data Distribution,DD)算法,但是这些算法对于节点数量的扩展不具有很好的支持,算法在实际生产环境中还需要考虑多节点所带来的节点故障、网络通信故障等复杂问题。

5　结语

本文论述了在面对海量交通数据时如何利用分布式并行数据处理框架 Spark 来分析处理数据,并利用并行 FP-DTC 方法来求解伴随车辆组。本文提出了一种基于多过程并行模式的处理方法,分别从车辆轨迹生成、计算点伴随组以及产生伴随车辆组 3 个过程展开叙述,最后通过实验证明,该方法适用于大规模交通数据的分析与应用。本文还有许多需要改进的地方,比如求解伴随车辆组时采用更加高效的求解频繁子集的算法等。

参考文献:

[1] NOREIKIS M, BUTKUS P, NURMINEN J K. In-vehicle application for multimodal route planning and analysis[C]// Proceedings of the 2014 IEEE 3rd International Conference on Cloud Networking. Piscataway: IEEE, 2014:350 - 355.

[2] LU S, ZHAO Z, HAN Y. A query method of the similarity of the vehicle trajectory[J]. Computer and Digital Engineering, 2014, 42(9):1565 - 1569.(卢帅, 赵卓峰, 韩燕波. 一种车辆移动对象相似轨迹查询算法[J]. 计算机与数字工程, 2014, 42(9): 1565 - 1569.)

[3] TANG L A, ZHENG Y, YUAN J, *et al*. On discovery of traveling companions from streaming trajectories[C]// Proceedings of the 2012 IEEE 28th International Conference on Data Engineering. Piscataway: IEEE, 2012: 186 - 197.

[4] FANG A, LI X, MAN S, *et al*. A discovery algorithm of travelling companions based on association rule mining[J]. Computer Applications and Software, 2012, 29(2): 94 - 96.(方艾芬, 李先通, 蔺世明, 等. 基于关联规则挖掘的伴随车辆发现算法[J]. 计算机应用与软件, 2012, 29(2): 94 - 96.)

[5] ZAHARIA M. An architecture for fast and general data processing on large clusters, UCB/EECS-2014-12 [R/OL]. [2015-01-22]. http://www. eecs. berkeley. edu/Pubs/TechRpts/2014/EECS-2014-12. html.

[6] AGRAWAL R, IMIELINSKI T, SWAMI A. Database mining: a performance perspective[J]. IEEE Transactions on Knowledge and Data Engineering, 1993, 5(6):914 - 925.

[7] AGRAWAL R, IMIELINSKI T, SWAMI A. Mining assocaition rules between sets of items in large databases[C]// Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data. New York: ACM, 1993:207 - 216.

[8] DING R, MENG X, YANG N. An efficient query method of similar trajectories of moving objects[J]. Computer Science, 2003, 30(10): 386 - 403.(丁锐, 孟小峰, 杨楠. 一种高效的移动对象相似轨迹查询方法[J]. 计算机科学, 2003, 30(10): 386 - 403.)

[9] UEHARA K, SEKI K, JINNO R. Parallel distributed trajectory pattern mining using MapReduce[C]// Proceedings of the 2012 IEEE 4th International Conference on Cloud Computing Technology and Science. Piscataway: IEEE, 2012: 269 - 273.

[10] CHAN K P, FU A C. Efficient time series matching by wavelets[C]// Proceedings of the 15th International Conference on Data Engineering. Piscataway: IEEE, 1999:126 - 133.

[11] KEOGH E J, PAZZANI H J. Scaling up dynamic time warping for datamining applications[C]// Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM, 2000:285 - 289.

[12] AGRAWAL R, SHAFER J C. Parallel mining of association rules[J]. IEEE Transactions on Knowledge and Data Engineering, 1996, 8(6):962 - 969.

(上接第 3111 页)

[4] KARLIN S. A first course in stochastic processes[M]. New York: Academic Press, 2014:16 - 22.

[5] LU D. Research on construction of connected dominating sets in wireless sensor network[D]. Suzhou: Soochow University, 2014: 33 - 46.(鲁登月. 无线传感器网络中连通支配集的构造算法研究[D]. 苏州:苏州大学, 2014: 33 - 46.)

[6] YIM S J, CHOI Y H. Fault-tolerant event detection using two thresholds in wireless sensor networks[C]// Proceedings of the 15th IEEE Pacific Rim International Symposium on Dependable Computing. Piscataway: IEEE,2009: 331 - 335.

[7] XUE W, LUO Q, WU H. Pattern-based event detection in sensor networks[J]. Distributed and Parallel Databases, 2012, 30(1): 27 - 62.

[8] PIAO D, MENON P G, MENGSHOEL O J. Computing probabilistic optical flow using Markov random fields[C]// Proceedings of the 4th International Conference on Computational Modeling of Objects Presented in Images, LNCS 8641. Berlin: Springer, 2014: 241 - 247.

[9] WANG X R, LIZIER J T, OBST O, *et al*. Spatiotemporal anomaly detection in gas monitoring sensor networks[C]// Proceedings of the 5th European Conference on Wireless Sensor Networks. Berlin: Springer-Verlag, 2008: 90 - 105.

[10] HUANG T, MA X, JI X, *et al*. Online detecting spreading events with the spatio-temporal relationship in water distribution networks[M]// Proceedings of the 9th International Conference on Advanced Data Mining and Applications. Berlin: Springer, 2013: 145 - 156.

[11] DAI F, WU J. An extended localized algorithm for connected dominating set formation in Ad Hoc wireless networks[J]. IEEE Transactions on Parallel and Distributed Systems, 2004, 15(10): 908 - 920.

[12] ROSSMAN L A. EPANET 2 Users manual[J]. Laboratory Office of Research and United States Environmental Protection Agency, 2000, 19(1):115 - 118.

[13] LI H, LIU J. New progress of study of water quality monitoring sensors[J]. Transducer and Microsystem Technologies, 2012, 31(3): 11 - 14.(黎洪松, 刘俊. 水质检测传感器研究的新进展[J]. 传感器与微系统, 2012, 31(3): 11 - 14.)