第46卷第1期 中国科学技术大学学报 Vol. 46, No. 1
2016年1月 JOURNAL OF UNIVERSITY OF SCIENCE AND TECHNOLOGY OF CHINA Jan. 2016

文章编号:0253-2778(2016)1-0047-09

基于大规模流式车牌识别数据的即时伴随车辆发现

朱美玲[1,2],王雄斌[2,3],张守利[1,2],刘晨[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. 北方工业大学大规模流数据集成与分析技术北京市重点实验室,北京 100144;
3. 北方工业大学云计算研究中心,北京 100144)

摘要:提出了一种基于流式大规模车牌识别数据集的伴随车辆(伴随车辆是指在一段持续的时间内一起移动的车辆组群)即时发现方法,可实现即时发现疑似伴随车辆并将其按伴随概率排序.该方法充分利用了云基础设施的并行计算能力,基于整数划分思想建立并行发现的负载均衡模型,优化了伴随车辆的发现性能,可用于对时间敏感的交通应用场景,如发现并监控运钞车等特殊车辆的跟踪车辆等.实验证明,该方法能够有效处理大规模的流式车牌识别数据,并实时地输出发现结果.

关键词:伴随车辆;车牌识别数据;流数据;即时性;点伴随

中图分类号:TP311.5 **文献标识码**:A doi:10.3969/j.issn.0253-2778.2016.01.007

引用格式:ZHU Meiling, WANG Xiongbin, ZHANG Shouli, et al. Instant traveling companion discovery based on large scale streaming ANPR data[J]. Journal of University of Science and Technology of China, 2016,46(1):47-55.

朱美玲,王雄斌,张守利,等. 基于大规模流式车牌识别数据的即时伴随车辆发现[J]. 中国科学技术大学学报,2016,46(1):47-55.

Instant traveling companion discovery based on large scale streaming ANPR data

ZHU Meiling[1,2], WANG Xiongbin[2,3], ZHANG Shouli[1,2], LIU Chen[2,3], HAN Yanbo[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: Traveling companions are object groups that move together in a period of time. To quickly identify traveling companions from a special kind of streaming traffic data, called automatic number plate recognition (ANPR) data, a framework and several algorithms were presented to discover companion vehicles, which can instantly detect suspicious companion vehicles with their probabilities when they pass through monitoring cameras. The framework can be used in many time-sensitive scenarios like taking surveillance on suspect trackers for specific vehicles. Experiments show that the proposed approach can process streaming ANPR data directly and discover companion vehicles in nearly real time.

Key words: traveling companion; ANPR data; stream data; instant; moment companion

收稿日期:2015-08-27;**修回日期**:2015-09-29

基金项目:北京市自然科学基金重点项目(4131001),北京市属高等学校创新团队建设与教师职业发展计划(IDHT20130502),北方工业大学"人才强校计划"青年拔尖人才培育计划资助.

作者简介:朱美玲,女,1987年生,博士生.研究方向:服务计算,大规模流数据关联分析. E-mail: meilingzhu2006@126.com

通讯作者:韩燕波,博士/教授. E-mail:yhan@ict.ac.cn

0 引言

目前，车辆发现问题逐渐受到国内外学者的关注，成为热点研究问题[1-4]. 该问题的研究目标是发现一段时间内在一起移动的车辆组群. 解决该问题的技术已经被广泛应用到各种领域. 例如，社交网络、交通管理、科学研究以及军事侦查[4].

对于特定的应用，伴随车辆的发现对时间是敏感的. 例如，为了保证安全，安保人员需要实时监视某些特殊车辆(如运钞车)的伴随车辆情况，以确保没有可疑车辆跟踪. 然而当前大多数研究工作都是针对二维欧几里得空间上的静态数据集展开的，无法针对实时监控数据直接输出即时结果[4].

本文基于一种具有流式特征的交通实时监控数据(车牌识别数据)，实现伴随车辆的即时发现. 当前，我国许多城市都在道路路口安装了交通监控摄像头，这些摄像头能够不断采集过往车辆的照片，从中识别并提取相应的车牌信息，如车牌号以及车辆通过该路口的时间等. 这些信息借助监控设备和无线网络以流数据的形式被传输到交通管理部门的中央数据中心，用于进一步的分析和挖掘.

相对于已有的伴随车辆发现方法[2-6]，本文的方法可在车辆通过监控摄像头时，即时发现该车辆的疑似伴随车辆集合，并对集合中的每辆车计算其伴随概率，以按照概率对伴随车辆进行排序. 本文的贡献包括以下两点：第一，本文提出了一种点伴随(moment companion, MC)的数据结构用来记录车辆在给定时刻的伴随车辆信息. 点伴随随着车辆经过监测点而陆续生成，不同车辆的点伴随又可以相互关联形成一张动态演化的有向图. 基于这个图，可以做很多有价值的分析，如计算车辆的伴随概率. 第二，本文提出了一种并行优化算法. 为了应对大规模流数据，设计了可以动态调整的数据分块策略. 该策略基于整数划分思想，将相同监测点下的数据划分到同一分块中，避免了计算伴随车辆时不同工作节点之间的数据交换. 实验表明，本文的方法对于处理大规模的流式车牌识别数据，可以保持较低的延迟.

1 相关工作

目前，有许多研究者对伴随车辆发现很感兴趣，但大多数的研究工作主要集中在二维欧几里得空间上的静态的数据集. 文献[7]试图找到两个连续的时间戳内有大部分重叠的移动对象. 文献[1]提出概念 flock 来定义移动对象组群，这些移动对象在 k 个连续的时间戳内在一个固定大小的盘内移动. 文献[2]提出了概念 convoy，convoy 是 flock 的扩展，是基于密度的空间聚类. 文献[3]提出了 swarm，放宽了移动对象聚类的时间约束；更多人开始关注大规模轨迹数据的研究. 文献[6]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移动对象组群聚类的性能. 文献[8]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. 文献[9]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. 值得注意的是，文献[6,8]更多关注算法的设计而不是并行化.

上述研究方法尚不能适用于流数据的处理，而在许多交通领域，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. 文献[5]提出了一个框架，用来在流式轨迹数据中发现伴随车辆，并以增量的方式输出结果. 文献[10-11]针对流式轨迹数据提出了一种基于密度的聚类算法，尝试实时地发现轨迹组群. 所有的这些相关工作都针对的是流式 GPS 数据并且取得了很大的进步，但是他们并没有关注于算法本身的并行化. 本文提出了一个计算框架，以流式车牌识别数据集为研究数据，即时地识别发现伴随车辆.

2 问题定义和分析

图 1 是一个车牌识别数据集的例子. 如图所示，交通监控摄像头分布在各道路路口. 每个摄像头都可以看作一个流数据源，它们不断地采集过往车辆的照片并生成车牌识别数据. 显然，这些数据产生的速率取决于车辆通过监控摄像头的频率. 在高峰时期，数据记录的生成速率可以达到每秒一个. 值得注

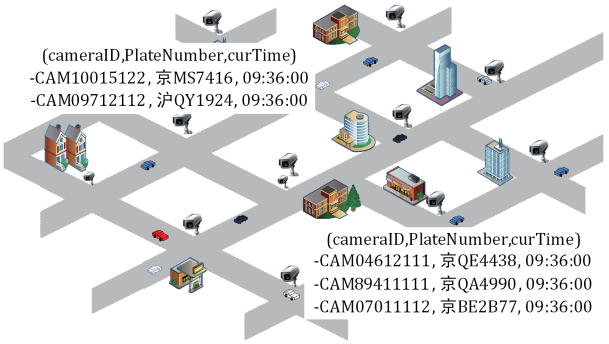


图 1 车牌识别数据集示例

Fig. 1 ANPR dataset examples

意的是，每条车牌识别数据记录仅包含一辆车的信息. 车牌识别数据记录的详细定义如下所示.

定义 2.1　车牌识别数据记录. 一条车牌识别数据记录 r 可以由一个三元组 $r=(c, p, t)$来表示，其中 c 代表摄像头的 ID，p 代表车牌号，t 是时间戳. 一条车牌识别数据记录 $r=(c, p, t)$表示车牌号为 p 的车辆在 t 时刻经过摄像头 c. 下面是几条真实的车牌识别数据记录.

(CAM10015122，京 MS7416，2013-1-1 09:36:00)
(CAM09712112，沪 QY1924，2013-1-1 09:36:00)
(CAM04612111，京 QE4438，2013-1-1 09:36:00)

图 2 展示了本文希望达到的效果. 当某车辆(如 v_1)经过某监控摄像头时，用本文的方法可以立刻识别出该车的伴随车辆，根据伴随车辆与给定车辆(即 v_1)之间的伴随概率对伴随车辆进行排序，并即时输出排序结果. 图 2 中，当车辆 v_1 在时刻 T_1 经过摄像头 C_1 时，v_1 的伴随车辆为 v_2 v_4 v_5，组成一组有序数组：<v_2：67%，v_4：67%，v_5：33%>；当车辆 v_1 在时刻 T_2 经过摄像头 C_2 时，v_1 的伴随车辆变为 v_2 v_5 v_6，对应的输出为：<v_2:75%，v_4:50%，v_5:50%，v_6:25%>；当车辆 v_1 在时刻 T_3 经过摄像头 C_3 时，输出为：<v_2:80%，，v_5:60%，v_4:40%，v_7:20%，v_6:20%>.

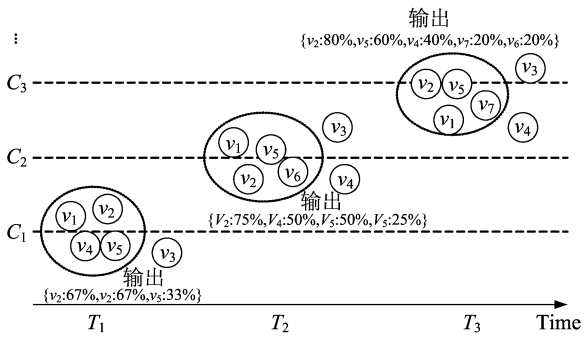


图 2　伴随车辆的即时发现

Fig. 2　Instant discovery of travelling companion

通过上述场景，本文总结了伴随车辆的即时发现的几点要素：

(Ⅰ) 相邻性

伴随车辆就是一起移动的车辆组群，也就是说，伴随车辆之间应该足够接近. 对于 GPS 数据，人们可以随时获得车辆的位置信息；然而车牌识别数据却不能满足这种要求. 一方面是因为监控摄像头采集车辆照片时存在时间延迟，另一方面是因为监控摄像头的位置是固定的；因此，本文将在一定时间间隔内通过同一监控摄像头的车辆视作伴随车辆.

(Ⅱ) 数据规模性

本文要解决的问题是伴随车辆的即时发现，庞大的数据规模是本文面临的挑战之一. 中国的大城市安装的交通监控摄像头的数量均超过 5 000 台. 在交通高峰期，照片的拍摄速率可达到每秒一张. 假设大城市每天的高峰时刻可以持续 4 小时，那么产生的车牌识别数据记录能够达到 1.44 亿条；因此，为了解决此问题，本文设计了并行算法来处理规模庞大的数据集.

(Ⅲ) 动态性

本文的研究数据是具有流式特征的车牌识别数据，无法预测下一时刻到达的数据以及数据的到达率，因此需要一个动态的数据分块策略，使并行算法能够更好地适应数据特征和到达速率的变化.

3　伴随车辆即时发现

3.1　伴随车辆即时发现框架

图 3 展示了本文提出的框架的基本原理. 如图所示，每个监控摄像头都可以作为一个流数据源，能够不间断地产生流式车牌识别数据. 接收到摄像头的流数据之后，本文的计算框架会即时生成相应的点伴随；然后将点伴随互相关联成有向图，并在有向图上计算车辆的伴随概率；最后输出排序后的伴随车辆.

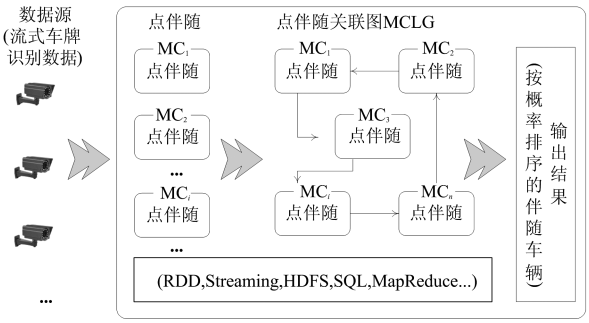


图 3　框架的基本原理

Fig. 3　The rationales of discovery framework

点伴随是本框架中的核心数据结构. 点伴随记录了车辆在某时刻通过某监控摄像头时的所有伴随车辆. 下面给出点伴随的具体定义：

定义 3.1　点伴随. 已知车辆 v 和交通监控摄像头 C，$pt(v, C)$为车辆 v 经过摄像头 C 的时刻，Δt 为时间阈值，车辆 v 在时刻 $pt(v, C)$的点伴随可以定义为如下形式：

$$\mathrm{MC}(v,C,pt(v,C))=\{v' \mid pt(v,C)-\Delta t \leqslant pt(v',C) \leqslant pt(v,C)\}$$

点伴随记录了给定车辆 v 在给定时间段内的所有伴随车辆信息. 时间阈值 Δt 是判断两辆车是否是点伴随的关键因素：车辆 v 在给定时刻 $pt(v, C)$ 的伴随车辆实际上就是在时间区间 $[pt(v, C)-\Delta t, pt(v, C)]$ 内经过交通监控摄像头 C 的所有车辆.

监控摄像头采集的车牌识别数据可以生成大量的点伴随，这些点伴随又相互关联，从而形成一张有向图，本文称之为点伴随关联图. 随着点伴随的增加，点伴随关联图也会不断地发展演化. 基于点伴随关联图，可以做很多有价值的分析，如通过挖掘相关的历史点伴随，计算伴随关系的概率.

本文基于 Spark Streaming 框架设计并实现了并行算法处理对大规模、动态变化的流式车牌识别数据. Spark Streaming 是 Spark 的核心 API，它支持高吞吐量、容错的实时流数据处理. 弹性分布式数据集 RDD (resilient distributed dataset) 是 Spark 的一个核心抽象，它是一个跨集群节点的数据集合，可以并行操作. 在设计算法时，本文还使用了 Spark 提供的两种操作类型：Transformations 和 Actions. Transformations 可以在已有的数据集中创建新的数据集. Actions 在对数据集进行计算之后向驱动程序返回结果值. 表 1 列举了本文算法中所用到的主要操作. 有关 Spark 的详细信息请参考文献[14].

表 1 算法中所用到的主要操作

Tab. 1 Main Spark streaming operations used in our framework

	操作	描述
transformation	window	设置滑动窗口的大小和间隔
	collect	将流数据转换为数组
	parallelize	设置并行度
	filter	通过定义布尔函数筛选返回值为真的数据
action	print	将数据输出至控制台
	foreach	以 RDD 形式处理输出数据
	save As Text Files	将数据存至 HDFS 文件系统

3.2 点伴随的发现

图 4 描述了生成点伴随的过程. 本框架包含一个主节点和若干工作节点. 主节点创建一个滑动窗口，接收流式车牌识别数据并缓存下来以进行下一步处理. 主节点同时把缓存的车牌识别数据集分割成块，把任务分配给空闲的工作节点. 工作节点接收到车牌识别数据块后即开始计算点伴随，计算结果存储在 Hadoop 分布式文件系统中.

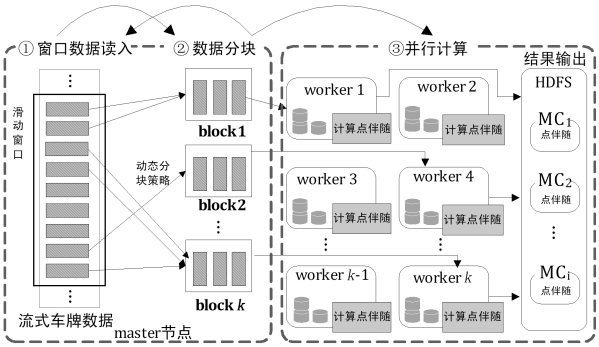


图 4 点伴随的生成过程

Fig. 4 The process of discovering moment companions

为保障节点的负载均衡，本文设计了一个动态的分块策略，使来自同一交通监控摄像头的车牌识别数据尽可能分到同一块中，且每个数据分块大小尽量相同. 根据定义 3.1，对于流式车牌识别数据，伴随车辆在某时刻区间内经过同一监测点下. 即潜在的伴随车辆仅仅出现在相同的摄像头下. 而 Spark、MapReduce 等当下流行的并行计算框架采用的分块策略往往是只保障数据块大小相等而忽略数据本身的放置方式，因此默认的数据分块策略很可能会将来自同一摄像头的数据记录划分到不同的块中，即将潜在的伴随车辆划分到不同的数据块中. 当主节点按照这样的分块策略将数据分块调度到工作节点上并启动计算任务后，工作节点在计算伴随车辆时要进行数据传输与交换，从而给系统带来不必要的负担及开销. 为了解决上述问题，本文提出一种基于数据源的数据分块策略，并将其转换为典型的整数划分问题[12-13]进行求解. 该策略放宽了数据块大小相等的约束，而侧重于数据的放置. 定义 3.2 给出了该数据分块策略的形式化定义.

定义 3.2 基于数据源的分块策略. 给定一个车牌识别数据集 S，基于数据分块策略 sp 是一个分块函数 $sp(S)=\{S_1, S_2, \cdots, S_k\}$，它满足：

(Ⅰ) $S=\bigcup_{i=1,\cdots,k} S_i$；

(Ⅱ) $S_i \cap S_j = \phi, i, j = 1, \cdots, k \wedge i \neq j$；

(Ⅲ) 已知两条数据 $r=(C, p, t)$ 和 $r'=(C', p', r')$，如果 $r.C = r'.C'$，那么 $\exists\ S_i$，使得 $r \in$

$S_i \wedge r' \in S_i$)；

(Ⅳ) $\min(\sum_{i,j=1..k} |S_i|-|S_j|)$，$|S_i|$ 是 S_i 的大小.

为了求解基于数据源的分块策略，本文把它转换成典型的整数划分问题. 整数划分问题就是如何把一个整数数组 L 分成 k 个子数组，使得每个子数组的元素之和大小尽量相等. 例如，可以把一个整数组{3，1，1，2，2，1}分成两个子数组 L_1 = {1，1，1，2} 和 L_2 = {2，3}，L_1 和 L_2 的元素和都是 5. 显然，无法找到更合适的划分方式，使得两个子数组中的元素之和的最大值小于 5. 定义 3.3 借鉴了文献[13]，对整数划分问题进行了形式化描述.

定义 3.3　整数划分问题. 给定一个全部为正整数的数组 L = {l_1，l_2，…，l_n}，设 sum(L)为 L 的元素之和，将 L 划分为 k 个子数组 L_1，L_2，…，L_k，使得 max{sum(L_1)，sum(L_2)，…，sum(L_k)}最小.

首先根据摄像头 ID 对车牌识别数据记录进行分块. 这样的分块方式既保证了来自不同摄像头的车牌识别数据记录被分到不同的块中，也保证了来自同一摄像头的车牌识别数据记录不会被分到不同的分块中. 然后以每个分块的大小值作为元素，构造一个整数数组. 针对这个整数数组，通过定义 3.3 将前文所述的数据划分策略转换成整数划分问题.

整数划分问题是一个 NP 难问题[12]，文献[13]给出了一个时间复杂度为 $O(kn^2)$ 的 次优解，算法 3.1 展示了详细的解决方案. 算法 3.1 是一种递归的整数划分问题的解决方法. 它首先设定了求解问题的边界条件(lines 1-9)，然后计算各种可能情况的最小开销 $M[i][j]$ 和相应的分配器的位置 $D[i][j]$(lines 10-22). $M[i][j]$ 表示将数组的前 i 个元素分成 j 个分块的最小开销，$D[i][j]$ 用来记录分配器的位置信息.

算法 3.1　整数划分[13]

Function：IntP

Input：List L={ l_1，l_2，…，l_n}，size of L= n，block number k；

Output：cost matrixM，divider matrix D；

```
1  Initial matrix M, D, initial a list P, initialint cost, i,j,x
2  P[0]=0
3  for (i=1; i<=n; i++)
4  P[i]=P[i-1]+L[i-1]
5  M[i][1] = P[i]
6  for end
7  for (j=1; j<=k; j++)
8  M[1][j] = L[0]
9  for end
10 for (i=2; i<=n; i++)
11 for (j=2; j<=k; j++)
12 M[i][j] = POSITIVE_INFINITY
13 for (x=1; x<=(i-1); x++)
14 cost = max(M[x][j-1], P[i]-P[x])
15 if (M[i][j] > cost)
16 M[i][j] = cost
17 D[i][j] = x
18 if end
19 for end
20 for end
21 for end
22 return M, D
```

算法 3.2　点伴随的发现

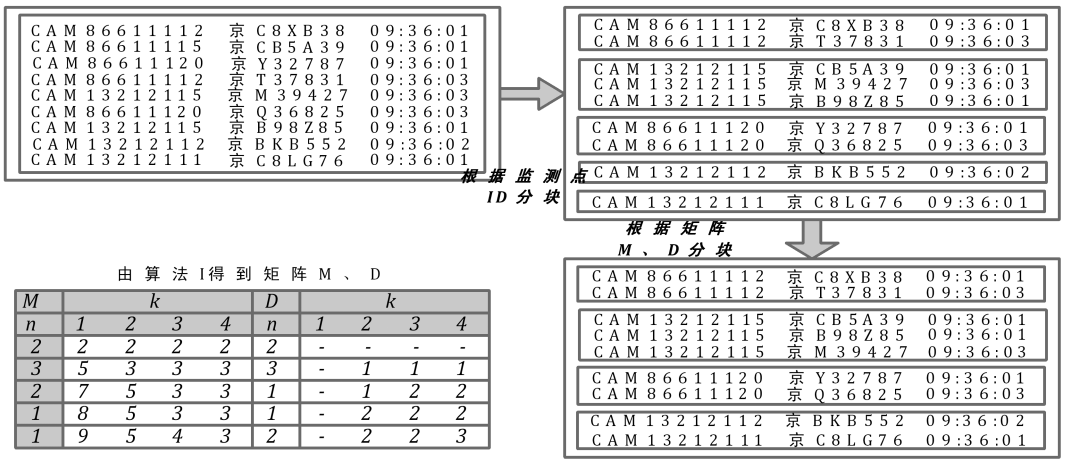
Function：DMC

Input：Δt be the time threshold，S be the ANPR dataset contains data records in the sliding window；

Output：a set of records of companion vehicles ofv；

```
1  window(). foreach()
2  Q←RDD. collect()
3  partitionQ by sp strategy based on IntP algorithm
4  Initial a listR, outputSet
5  for each block Block
6  Initial aJavaPairRDD blockRDD←parallelize(Block)
7  for each record r_i=(v_i, c_i, t_i) in Block
8  if Block contains enough data of r_i and never discover moment companion of r_i before
9  outputSet . add(blockRDD. filter(). collect())
10 R. add(r_i)
11 if end
12 for end
13 for end
14 Q. removeAll(R)
15 window(). foreach() end
16 Initial aJavaRDD outputRDD←parallelize(outputSet)
17 outputRDD. saveAsTextFiles
```

图 5 是一个简单的例子，描述了如何利用算法 3.1 对车牌识别数据集进行分块. 假设需要将数据集分为 4 块(k=4). 首先，根据每条数据记录的摄像头 ID(cameraID)把原始数据集分为 5 块；然后，计算每个分块包含数据记录的数目，并构造一个整数数组 L={2，3，2，1，1}；最后利用算法 3.1 生成矩阵 $\boldsymbol{M}$ 和 $\boldsymbol{D}$，并根据 $\boldsymbol{M}$ 和 $\boldsymbol{D}$ 把上述 5 个分块划分成 4 个新的数据块.



M	k				D	k			
n	1	2	3	4	n	1	2	3	4
2	2	2	2	2	2	-	-	-	-
3	5	3	3	3	3	-	1	1	1
2	7	5	3	3	1	-	1	2	2
1	8	5	3	3	1	-	2	2	2
1	9	5	4	3	2	-	2	2	3

图 5 利用基于数据源的划分策略对车牌识别数据集进行划分示例

Fig. 5 Example of Partition an ANPR Dataset with Source-based Partition Strategy

基于数据源的分块策略，本文提出了基于 Spark Streaming 流框架执行的点伴随发现算法. 算法 3.2 是点伴随发现算法的伪代码. 首先，算法 3.2 通过滑动窗口接收输入的流数据并将数据缓存在 Q 中 (lines 1-2)；然后，根据基于数据源的分块策略划分缓存在 Q 中的数据(line 4)；再利用工作节点并行计算点伴随(lines 6-12)；主节点把不需要继续参与计算的数据从 Q 中移除(line 14)；最后计算结果被存储到 HDFS 中(lines 16-17).

3.3 伴随概率计算

每当有车辆经过交通监控摄像头时，算法 3.2 都会计算出该车辆在该时刻的点伴随. 把这些点伴随关联起来就形成了一个有向图，本文称之为点伴随关联图(MC linkage graph, MCLG). 图 6 是点伴随关联图的示例. 点伴随由 PlateNumber, cameraID, curTime 和 CompanionVehicle 四个属性构成：PlateNumber 代表车牌号，是一辆车的标识符；这辆车被序列号为 cameraID 的监控摄像头在 curTime 时刻捕获；CompanionVehicle 是有效时间区间内经过摄像头 cameraID 的所有车辆的车辆信息.

点伴随关联图的每个节点都是一个点伴随，记录了给定车辆的相关信息. 节点之间由有向边连接. 图 6 中从 MC_1 到 MC_2 的存在有向边的充分必要条件是 MC_1 的给定车辆是 MC_2 给定车辆的伴随车辆，且 MC_1 的 curTime 比 MC_2 的 curTime 早，也就是说，MC_1. PlateNumber 属于 MC_2. CompanionVehicle.

定义 3.4 点伴随关联图. 点伴随关联图 MCLG = (V, E)是一个有向图，其中 V 是点伴随

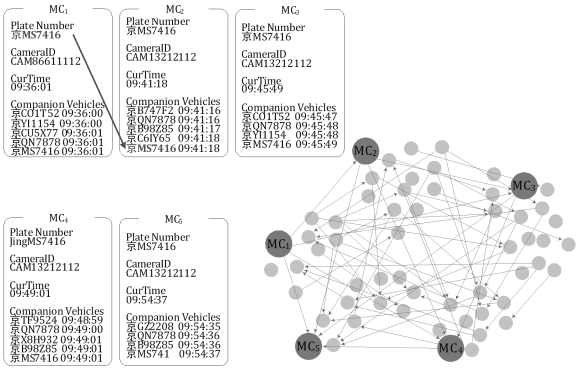


图 6 点伴随关联图(MCLG)示例

Fig. 6 An example of MC linkage graph

的集合，E 是有向边的集合.

$$\forall e \in E, e = (x, y) \text{ iff } x, y \in V \wedge x.\text{PlateNumber} \in y.\text{CompanionVehicle} \wedge x.\text{curTime} < y.\text{curTime}.$$

显然，根据定义 3.4，利用广度优先搜索算法可以在点伴随关联图中找到任何一辆车辆的全部的历史点伴随. 对于计算伴随概率来说，距离当前时刻越近的历史点伴随具有越高的有效性，因而本文让用户设定时间阈值，确定有效时间范围，在该时间范围内搜寻历史点伴随，并把这些历史点伴随作为计算伴随概率的基础. 为此，本文设置了一个由用户赋值的参数 δ_t，以给定车辆 v 最近的 curTime 作为参照时间，在时刻区间[curTime− δ_t , curTime]中搜索车辆 v 的所有历史点伴随，根据频率理论计算伴随概率.

以图 6 中车辆“京 MS7416”作为例子，显然车辆“京 MS7416”最近一次被监控摄像头采集信息的时间是 curTime = 09:54:37，如果设 δ_t 为 20 min，

则只需要在[09:34:37, 09:54:37]时间区域内搜集点伴随,于是可以得到有效的历史点伴随的集合为{MC_1, MC_2, MC_3, MC_4, MC_5}. 当用户将 δ_t 设置为 10 min 时,有效的历史点伴随的集合变为{MC_3, MC_4, MC_5}.

概率论支持人们根据事件发生的频率计算事件的发生概率[15]. 定义 3.5 给出了事件发生概率的计算公式.

定义 3.5　事件 E_i 的发生概率. 已知事件样本空间 Ω,事件 $E_i \in \Omega$, 事件 E_i 的出现概率如下:

$$P(E_i)=\frac{N_{E_i}}{N_\Omega}.$$

式中, N_Ω 表示 Ω 中所有事件的发生次数的和, N_{E_i} 是事件 E_i 的发生次数.

在本文的场景中,对于已知车辆 v_1 和任意其他车辆 v_2 ,可以定义 $\Omega=\{E_1,E_2\}$,其中 E_1 表示车辆 v_1 和车辆 v_2 在某时刻伴随出现; E_2 则表示车辆 v_1 和车辆 v_2 在任何时刻都不伴随. 根据定义 3.5,本文给出 v_2 成为 v_1 伴随车辆的概率的定义.

定义 3.6　伴随概率 v_1, v_2 为两辆车, δ_t 是由用户预定义的时间阈值,curTime 是 v_1 在点伴随关联图中最近的被采集时刻,设在时间区间[curTime-δ_t , curTime]内车辆 v_1 的全部历史点伴随集合为 $S_{MC}(v_1,\delta_t,\text{curTime})=\{MC_1,MC_2,\cdots,MC_n\}$,则 v_2 成为 v_1 的伴随车辆的概率可以被定义为:

$$P(E_1)=\frac{N_{E_1}}{|S_{MC}|}.$$

N_{E_1} 是在集合 S_{MC} 中 v_2 是 v_1 的伴随车辆的次数, $|S_{MC}|$ 是集合 S_{MC} 的大小.

图 7 是随着时间的推移和点伴随关联图的演化,车辆之间伴随概率的变化示例. 根据定义 3.6 和图 6,设 δ_t 为 15 min ,搜索车辆“京 MS7416”的伴随车辆. 图 7 展示了从时刻 T_1 到时刻 T_5 车辆“京 MS7416”的所有伴随车辆的概率. MC_1 是车辆“京 MS7416”在 T_1(09:36:01)的点伴随. 显然该车的历史时刻伴随为: $S_{MC}(v_1,15\ \text{min},09:36:01)=\{MC_1\}$; 相应的可以得到:

$S_{MC}(v_1,15\text{min},09:41:18)=\{MC_1, MC_2\}$; $S_{MC}(v_1,15\text{min},09:45:49)=\{MC_1, MC_2, MC_3\}$; $S_{MC}(v_1,15\text{min},09:49:01)=\{MC_1, MC_2, MC_3, MC_4\}$; $S_{MC}(v_1,15\text{min},09:54:37)=\{MC_2, MC_3, MC_4, MC_5\}$.

此外,图 7 中,以到车辆“京 MS7416”和“京 B98Z85”在点伴随 MC_4 的伴随关系为例,令事件 E_1 表示车辆“京 MS7416”和“京 B98Z85”在某时刻伴随,下面对如何计算伴随概率进行描述:

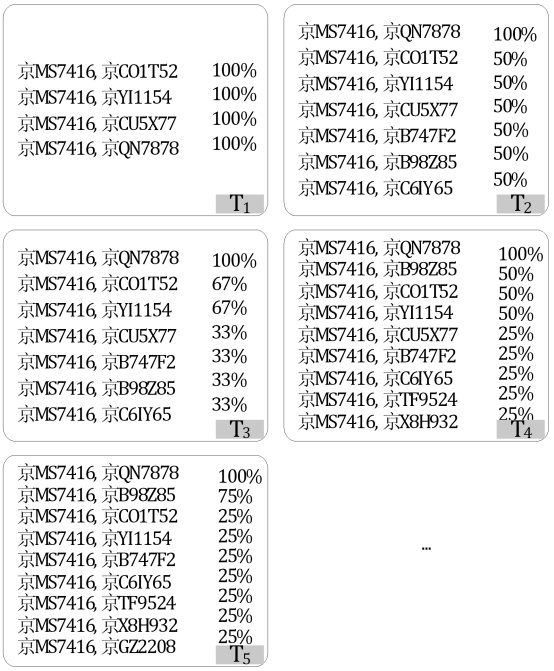


图 7　伴随车辆的概率变化示例
(以 T_1, T_2, T_3, T_4, T_5 为序)

Fig. 7　Example of the probability of each companion vehicle changes (in order of T_1, T_2, T_3, T_4, T_5)

$$S_{MC}=\{MC_1,MC_2,MC_3,MC_4\},$$
$$N_{E_1}=2,\ |S_{MC}|=4.$$

因此 $P(E_1)=\frac{2}{4}=50\%$.

4　实验设计

4.1　实验步骤

为了解决流式车牌识别数据的伴随车辆即时发现,本文设计了两个指标来检验时间延迟.

定义 4.1　处理时间. 对于一条车牌识别数据 r,令 $t_s(r)$ 和 $t_e(r)$ 分别代表 r 的到达时间和处理 r 的结束时间,则 r 的处理时间 $t_{pt}(r)$ 可以被定义为:

$$t_{pt}(r)=t_e(r)-t_s(r).$$

定义 4.2　总处理时间. 对于车牌识别数据集 $R=\{r_1, r_2, \cdots, r_m\}$,定义总的处理时间为:

$$t_{tpt}=\sum\nolimits_i t_{pt}(r_i),i=1,\cdots,m.$$

本文的实验是基于中国某大城市的真实车牌识别数据集完成的,选取从 2012-12-06 17:57:53 到 2013-01-04 22:03:54 时间段内所有车牌识别数据. 为了验证数据规模和处理时间之间的关系,本文从

车牌识别数据集中提取数万到数百万条数据记录，转换成流数据，分别在1、3、5个节点上运行算法3.2，实验过程中算法3.2的时间阈值Δt取为2 min.

实验在拥有5个节点的集群上进行. 所有节点均为装有CentOS-6.4的虚拟机，jdk版本为1.70. 集群的详细信息见表2. 集群所使用的Spark版本为1.1.0，Hadoop版本为2.3.0. 所有的结果存储在HDFS上. 集群信息如表2所示.

表2 集群配置信息

Tab. 2 Configuration of our cluster

节点角色	CPU	核数	内存
主节点	Intel Xeon E312xx	6	6G
工作节点1	Intel Xeon E312xx	6	6G
工作节点2	Intel Xeon E312xx	3	3G
工作节点3	Intel Xeon E312xx	3	3G
工作节点4	Intel Xeon E312xx	3	3G

4.2 实验结果

为评估本文的算法，主要从两个方面进行分析.

(Ⅰ)数据的到达速率和处理时间

图8描绘了实验选取的1 160 921条车牌识别数据的到达速率，从图中可以发现，最大的到达速率大概可以达到140条每秒，而最小的数据到达数率可以低至40条每秒，整个数据集的到达速率呈现波形.

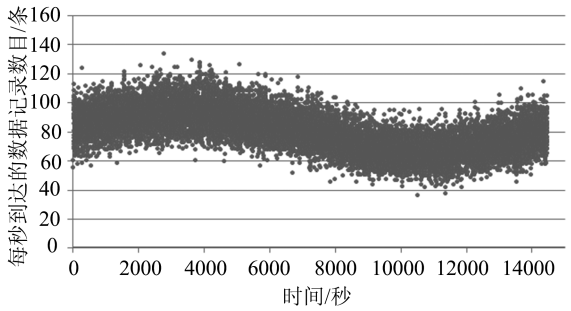
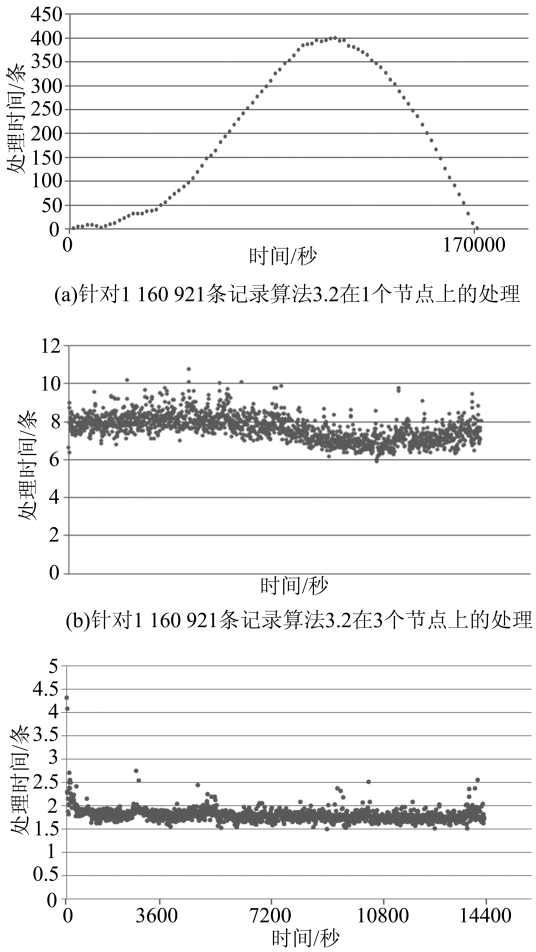


图8 1 160 921条车牌识别数据记录的到达率

Fig. 8 Data arrival rate of 1,160,921 ANPR data records

图9描述了当Δt =2min时，算法3.2分别在1个、3个、5个节点上执行时的处理时间. 对于1个节点的情况，本文选择的是主节点这一台机器(同时充当主节点和工作节点)；对于3个节点的情况，本文选择主节点、工作节点2和3三台机器进行实验；对于5个节点的情况，本文选取了集群包含的全部5台机器. 如图9(a)所示，单节点的处理时间呈现抛物线形. 随着流式车牌识别数据的到达，处理时间先是快速增长，随后又迅速下降. 我们认为造成这种现象的原因是单个节点在一开始就无法有效处理大规模的流数据. 随着越来越多的数据到达，延迟进一步扩大，因此单节点并不足以解决本文的问题. 图9(b)描绘了在3个节点上执行算法3.2的处理时间变化曲线. 可以发现图9(b)的曲线与图8的曲线相似，且数据的处理情况明显优于单节点. 图9(c)刻画了五个节点的处理时间变化情况，其曲线要比9(b)的曲线光滑. 根据这三张图可以得出结论：节点越多，越能有效处理流数据.



(a)针对1 160 921条记录算法3.2在1个节点上的处理

(b)针对1 160 921条记录算法3.2在3个节点上的处理

(c)针对1 160 921条记录算法3.2在5个节点上的处理

图9 1 160 921条数据记录的处理时间，Δt=2分钟

Fig. 9 Processing time of algorithm3.2 with 1 160 921 records, Δt=2min

(Ⅱ)数据规模和总处理时间

表3和图10刻画了数据规模和总处理时间之间的关系. 与图9得到的结论一致，从表3和图10可以得到如下结论：使用更多的节点可以提高处理流数据的性能. 当数据量足够大时，单节点的处理时间上升的非常迅速，这说明单节点并不能有效地应对本文要解决的问题. 另外，5个节点的总处理时间比3个节点下的总处理时间少了大约25%.

表 3　不同数据规模下的总处理时间

Tab. 3　Total processing times with different data scales

数据记录数目	处理时间/s		
	1 个节点	3 个节点	5 个节点
10 269	114.23	82.65	22.18
104 409	1265.15	825.30	198.90
1,160 921	171677.85	9055.06	2728.51

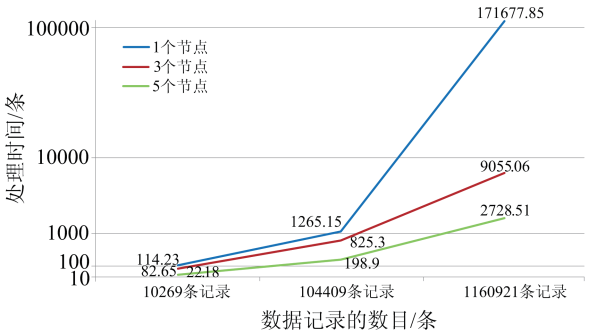


图 10　总处理时间

Fig. 10　Total processing time

5　结论

本文提出了一个计算框架以及若干算法用来即时发现伴随车辆. 本文的计算框架以流式车牌识别数据记录为输入,并且实现了流式车牌数据的并行处理. 本文的主要贡献是当车辆通过交通监控摄像头时,可以即时识别发现可疑的伴随车辆,并且按照伴随概率进行排序. 为了保障算法的性能,本文采用了一种基于数据源的分块策略来动态划分数据. 实验结果表明,本文的方法能够实时地输出排序后的伴随车辆. 我们将在以后的工作中考虑更多形式的交通数据如 GPS 数据. 考虑多种数据形式能够扩展计算框架的能力、拓展计算框架的应用范围.

参考文献(References)

[1] Gudmundsson J M, van Kreveld M. Computing longest duration flocks in trajectory data[C]// Proceedings of the 14th Annual ACM International Symposium on Advances in Geographic Information Systems. Arlington, USA: ACM Press, 2006: 35-42.

[2] 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): 1068-1080.

[3] Li Z H, Ding B L, Han J W, et al. Swarm: Mining relaxed temporal moving object clusters accurate discovery of valid convoys from moving object trajectories [C]// Proceedings of International Conference on Very Large Data Base. Springer-Verlag, 2010: 723-734.

[4] Tang L A, Zheng Y, Yuan J, et al. A framework of traveling companion discovery on trajectory data streams[J]. ACM Transactions on Intelligent Systems and Technology, 2013, 5(1): 992-999.

[5] Tang L A, Zheng Y, Yuan J, et al. On discovery of traveling companions from streaming trajectories[C]// Proceedings of the International Conference on Data Engineering. Arlington, USA: IEEE Press, 2012: 186-197.

[6] Zheng Y, Yuan N J, Zheng K, et al. On discovery of gathering patterns from trajectories[J]. nternational Conference on Data Engineering, 2013, 26(8): 242-253.

[7] Kalnis P, Mamoulis N, Bakiras S. On discovering moving clusters in spatio-temporal data [C]// Proceedings of the 9th International Conference on Advances in spatial and temporal databases. Springer, 2005: 364-381.

[8] Zhang J M, Li J L, Wang S G, et al. On retrieving moving objects gathering patterns from trajectory data via spatio-temporal graph [C]// IEEE International Congress on Big Data. Anchorage, USA: IEEE Press, 2014: 390-397.

[9] Yoo J S, Boulware D, Kimmey D. A parallel spatial co-location mining algorithm based on MapReduce [C]// IEEE International Congress on Big Data. Anchorage, USA: IEEE Press, 2014: 25-31.

[10] Yu Y W, Wang Q, Wang X D. Continuous clustering trajectory stream of moving objects [J]. Communications, 2013, 10(9): 120-129.

[11] Yu Y W, Wang Q, Wang X D, et al. Online clustering for trajectory data stream of moving objects [J]. Computer Science and Information Systems, 2013, 10(3): 1293-1317.

[12] Mertens S. The Easiest Hard Problem: Number Partitioning [A]// Computational Complexity and Statistical Physics. 2003: 125-140.

[13] S. S. Skiena. The Algorithm Design Manual[Z]. 2ed, Springer, 2008: 294-298.

[14] Kemmerer B. SPARK [EB/OL]. http://spark.apache.org/ last retrieved at 2015/1/10.

[15] Cosmides L, Tooby J. Are humans good intuitive statisticians after all? Rethinking some conclusions from the literature on judgment under uncertainty[J]. Cognition, 1996, 58(1): 1-73.