IET Intelligent Transport Systems

Research Article

Approach to discovering companion patterns based on traffic data stream

ISSN 1751-956X
Received on 1st October 2017
Revised 25th April 2018
Accepted on 7th August 2018
E-First on 9th October 2018
doi: 10.1049/iet-its.2018.5166
www.ietdl.org

Meiling Zhu[1,2,3] ✉, *Chen Liu*[2,3], *Yanbo Han*[2,3]

[1]*School of Computer Science and Technology, Tianjin University, Tianjin, People's Republic of China*
[2]*Beijing Key Laboratory on Integration and Analysis of Large-Scale Stream Data, North China University of Technology, Beijing, People's Republic of China*
[3]*Institute of Data Engineering, North China University of Technology, Beijing, People's Republic of China*
✉ *E-mail: meilingzhu2006@126.com*

Abstract: A companion of moving objects is an object group that move together in a period of time. Platoon companions are a generalised companion pattern, which describes a group of objects that move together for time segments, each with some minimum consecutive duration of time. This study proposes a method that can instantly discover platoon companions from a special kind of streaming traffic data, called automatic number plate recognition data. Compared to related approaches, the authors transform the companion discovery into a frequent sequence mining problem. The authors propose a data structure, platoon tree (PTree), to record discovered platoon companions. To reduce the cost of tree traversal during mining platoon companions, they utilise the last two together-moving objects of a group to update PTree. Finally, a lot of experiments have been carried out to show the efficiency and effectiveness of the proposed approach.

1 Introduction

In recent years, the volume of spatio-temporal data for recording movement patterns of moving objects has grown exponentially. Discovering such patterns from spatio-temporal data is very meaningful. A group of objects that moves together within a period of time is called a vehicle companion pattern, which is one of the common movement patterns. Companion pattern discovery has become a hot topic [1–10] and is applicable in many aspects of life such as transportation management, protecting special kinds of vehicles and military surveillances [7].

Nowadays, most of the research on discovering companion patterns focu on the global positioning system (GPS) data [1–10]. However, GPS data is generated by GPS devices installed on vehicles. Vehicles without or with turn-off GPS device will generate no GPS data. In some special occasions, such as tracking of vehicles taken by escaping criminals, they often turn off or remove the GPS device to avoid being captured. Existing studies based on GPS data are not suitable for these scenarios.

A special kind of traffic data, which is called automatic number plate recognition (ANPR) data, is streamed from a data centre of traffic management department which receives raw data transmitted by the cameras deployed on roads which continuously takes pictures of passing vehicles. ANPR data is very suitable for the around-the-clock and wide-range monitoring of vehicles without requiring pre-installing or turning on any extra devices. Presently, the number of cameras installed in tier 1 or tier 2 city in China exceeds 5000 and continues to increase. It leads to the total number of ANPR records in each day beyond 144 million. Eventually, the total data volume can get into multi-petabyte scale per annum.

Typical works on companion pattern discovery based on GPS data contain *flock* [1–3], *convoy* [4, 5], *swarm* [6], *travelling companion* [7–9] and *platoon* [10]. All of the above works tried to discover vehicle groups travelling together within an area under a series of timestamps. However, they focused on different temporal and spatial constraints. On temporal constraints, *flock* and *convoy* focused on consecutive timestamps, while *swarm* and *travelling companion* paid attention to non-consecutive timestamps. *Platoon* is a more general pattern. The timestamp series of *platoon* pattern consists of several segments of consecutive timestamps. Any adjacent timestamp in timestamp series does not exceed maximum time interval threshold. Selecting different length thresholds of segments will generate different patterns as mentioned above. Assume that the total timestamp number of a *platoon* is *len*, and the length threshold of segments is δ_{cc}. When $\delta_{\text{cc}} > len/2$, the temporal constraint of the *platoon* is the same with that of the *flock* and *convoy*; when $\delta_{cc} = 1$, the temporal constraint of the *platoon* is the same with that of the *swarm* and *travelling companion*. Besides, on spatial constraint, the *flock* focused on a disc area and other works extended the constraint into the density-reachable area. Previously we studied on how to effectively discover *travelling companion* patterns on ANPR data. In [11], we borrowed ideas from frequent itemset mining algorithm, like Apriori algorithm, to mine companion vehicles from historical ANPR dataset and apply it into a scenario of carpooling recommendations. Then, in [12, 13], we tried to discover *travelling companion* patterns from live ANPR data stream. To achieve this goal, we firstly proposed a concept called moment companion, which is a companion pattern for a vehicle that is recorded when passing through a camera. The moment companion is the vehicle together with other vehicles that follow it in a short while. Then, we transformed the companion vehicle discovering into a frequent sequence mining problem. There were shortcomings of *travelling companion* patterns discovered in our previous works that it would send companion vehicles under consecutive or non-consecutive timestamps to users indistinguishably. However, they create different values in applications. The *platoon* pattern can consummate the shortcoming. Users can get different patterns by setting length threshold δ_{cc} of segments on top of the above analysis.

Based on our previous work [12, 13], we use frequent sequence mining to discover *platoon* patterns. Due to lack of temporal and spatial constraints, traditional frequent sequence mining algorithms cannot solve our problem. Hence, we introduce customised temporal and spatial constraints into frequent sequence mining to solve our problem. Our contributions include: (i) we redefine *platoon* pattern under ANPR data; (ii) we propose an algorithm named Platoon Discovery on ANPR Data Stream (PANDA), which is a frequent sequence mining algorithm with the following constraints: firstly, each element in a frequent sequence occurs closely in time; secondly, any adjacent elements in the frequent sequence span over no more than predefined time interval in these sequences containing it; thirdly, a frequent sequence can be divided into several segments (i.e. subsequences). Each segment is consecutive in these sequences containing it. (iii) We optimise the algorithm to handle ANPR data stream efficiently. We propose a data structure, platoon tree (PTree), to record discovered *platoon* patterns. By the optimisation skill, we reduce the tree traversal cost when updating the PTree. A lot of experiments have been carried out to verify our algorithm.

2 Related works

2.1 Companion pattern discovery

Many researchers have put their interests on companion pattern study. They proposed various definitions of companion patterns and mining algorithms under different cases. In chronological order, typical work includes *flock* [1–3], *convoy* [4, 5], *swarm* [6], *travelling companion* [7–9], *platoon* [10] and so on. Table 1 compares the above companion patterns with the one in this study.

Some researchers focus on discovering moving clusters [14–20]. Their goal is to find clusters of objects with similar moving patterns or behaviours. Kalnis *et al.* proposed the first study to automatically extract moving clusters from large spatial datasets [14]. Li *et al.* clustered the moving objects by micro clustering [15]. Both current and near future positions of moving objects are considered during clustering. Kriegel *et al.* clustered the moving objects by fuzzy distance functions [16]. Jensen *et al.* discovered moving object clusters incrementally within a period of time [17]. Wang *et al.* defined continuous behaviour patterns as a concise representation of popular migration routes and underlying sequential behaviours during migration [18]. They developed a candidate generation and refinement framework to discover such patterns. Silva *et al.* focused on discovering mobility patterns online and maintaining pattern evolution by tracking sub-trajectories of moving objects at each time window [19].

More recently, researchers begin to pay attention to a large-scale trajectory. Zheng *et al.* developed a set of techniques to improve the performance of discovering gathering patterns over static large-scale trajectory databases [20]. Yoo *et al.* tried to leverage the MapReduce framework to achieve higher spatial data processing efficiency. It also proposed a partition strategy to avoid spatial relationships missing [21]. Zhang *et al.* focused on effectively and efficiently discovering the gathering patterns from the high volume of trajectory data. They utilised an improved R-tree based density clustering algorithm to index moving objects and clusters and a spatio-temporal graph to retrieve patterns [22, 23]. Xian *et al.* paid attention to both batch and streaming fashion of gathering patterns parallel discovery [24].

However, most of the studies above are designed to work on static datasets on 2D Euclidean space. They cannot effectively handle streaming data. In recent years, more and more studies began to process traffic data stream. Besides the above-mentioned works [7–9, 19, 20, 24], Yu *et al.* studied on a density-based clustering algorithm for trajectory data stream and tried to discover trajectory clusters in real time [25, 26]. All these related work aimed at processing GPS data stream and provide some foundations for our study.

2.2 Frequent sequence mining

Mining frequent sequences from databases are one of the classic topics in data mining and have been well studied. Previous studies about mining frequent sequences can be classified into two categories, including apriori-based algorithms and projection-based pattern growth algorithms [27]. Typical work of the former category includes AprioriAll [28], AprioriSome [28], generalised sequential patterns (GSP) [29], sequential pattern discovery using equivalence classes (SPADE) [30], sequential pattern mining (SPAM) [31] and so on. The shortage of apriori-based algorithms is to generate the large scale of candidate subsequences. Typical projection-based pattern growth algorithms include FreeSpan [32], PrefixSpan [33], CloSpan [34], BIDE [35] and so on. Projection-based pattern growth algorithms employ the divide and conquer strategy to construct projection database and greatly reduce the efforts of candidate subsequence generation. Otherwise, some researchers have focused their interests on mining frequent sequences with constraints. Pinto *et al.* defined the concept of multi-dimensional sequential pattern, and proposed an algorithm to discover them [36]. Different from the traditional sequence pattern, this pattern contains several attributes as well as a sequence. Pei *et al.* summarised the constraints in frequent sequence mining [37]. However, neither of the constraints discussed the temporal constraint in this study. Chueh went into more details on mining frequent sequence with time intervals between every pair of successive itemsets, which is the so-called gap constraint [38].

To improve the efficiency, some researchers parallelised these mining algorithms. Demiriz proposed a parallel sequence mining algorithm, webSPADE, to analyse the click streams found in site web logs [39]. Guralnik *et al.* studied a variety of distributed memory parallel algorithms which is able to minimise the overheads [40]. Ma *et al.* proposed a distributed memory parallel algorithm to mine closed frequent sequences [41]. Each processor mined local closed frequent sequences independently, which significantly reduced communication time cost. Qiao *et al.* proposed a trajectory patterns mining algorithm with three optimisation techniques, including prefix projection, parallel formulation, and candidate pruning [42]. Yu *et al.* parallelised BIDE algorithm by MapReduce framework [43]. Kessl proposed an algorithm for mining frequent sequences by static load balancing based on the probabilistic model [44].

Besides, mining frequent patterns over data streams has also attracted much attention. Some methods are proposed to compute the exact results of recent frequent patterns over data streams [45–50]. Chang *et al.* proposed SeqStream algorithm to mine closed frequent sequence in a sliding window for arriving data records [45]. The algorithm transformed the original data sequence database into an inverse sequence database to facilitate the removal of expired data. Also, it utilised a core data structure, called inverse closed sequence tree (IST), to keep closed sequential patterns in the inverse sequence database of the current sliding window. Besides it, IncSpan [46] is proposed to discover frequent sequences on the data stream, by using semi-frequent nodes in a prefix tree. By a

Table 1 Comparison of several companion vehicles discovery methods

Pattern	Temporal constraint	Spatial constraint	Dataset	Solution
flock [1–3]	consecutive	disc	GPS	matrix analysis; weighted directed graph; clustering and intersection
convoy [4, 5]	consecutive	density reachable	GPS	trajectory similarity
swarm [6]	not consecutive	density reachable	GPS	frequent item mining
travelling companion [7–9]	not consecutive	density reachable	GPS	clustering and intersection
travelling companion(our previous work) [11–13]	time threshold Δt	non-consecutive cameras	ANPR	frequent sequence mining with temporal constraint
platoon [10]	consecutive/non consecutive	density reachable	GPS	frequent item mining and frequent sequence mining
platoon (in this study)	time threshold Δt	consecutive/non-consecutive cameras	ANPR	frequent sequence mining with multiple constraints

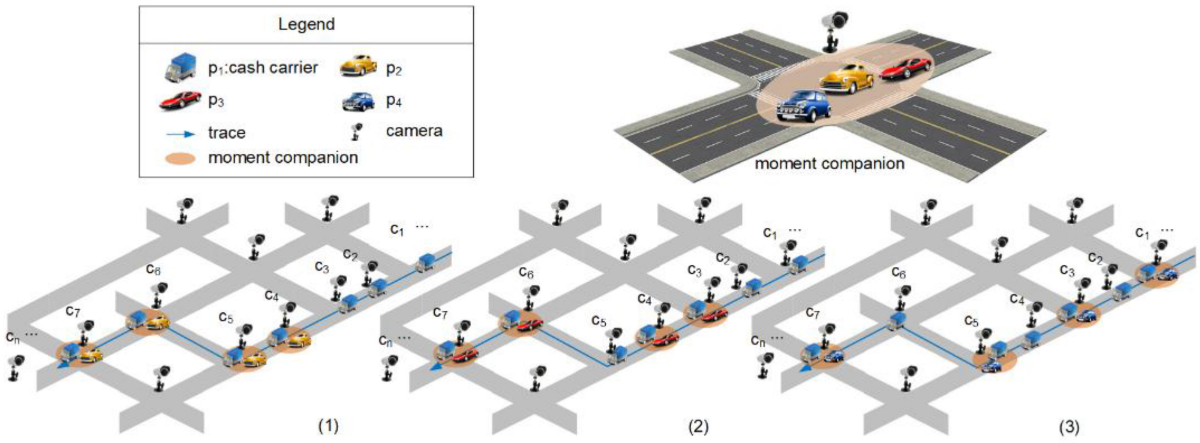

Fig. 1 *Examples of platoon patterns based on ANPR data stream*

tree structure, mining in multiple streams (MILE) [47] utilises the knowledge of existing frequent sequences to avoid redundant data scanning and learns from the prior knowledge of the data distribution in the data stream to enhance the efficiency. IncSPAM [48] utilises a tree structure in which the algorithm needs only one scan at each timestamp. CISpan [49] builds a tree upon both the new data and the previously affected data and then merges it with the previous tree together to build a new tree for the updated data. StreamCloSeq [50] also saves discovered frequent sequences in a tree structure. To improve the performance, it prunes the unpromising search spaces by the information of the previous sliding window and filters out the non-closed prefixes.

The typical methods mentioned above the lay foundation of our research. This study transforms companion pattern discovery into frequent sequence mining problem with special temporal constraint. However, neither of the existing works takes the temporal constraint in our study into consideration. Hence, we learn from these typical methods and our previous work to develop an effective projection-based pattern growth algorithm to achieve our goal. We optimise the algorithm according to the features of the input data stream. In the future, we plan to parallelise it to enhance the performance.

3 Problem definition

Some vehicles (e.g. cash carriers and *t*-axis) are more exposed to criminals' attention in recent years. Criminals usually follow them for a while to reach a suitable, secluded spot to commit a robbery. Being able to detect suspicious trackers over traffic data stream and alert the tracked driver in time can help to prevent such crimes. Fig. 1 shows some examples of *platoon* patterns.

Under ANPR data, a *platoon* pattern is a vehicle group passing through a series of cameras together. A series of cameras means a set of cameras, which consists of several sub-segments within which the cameras are geographically adjacent. Taking Fig. 1 (2) as an example, the *platoon* pattern is a vehicle group passing camera c_3, c_4, c_6 and c_7 together in sequence, in which c_3 and c_4 are geographically adjacent and c_6 and c_7 are geographically adjacent. Selecting the different minimal number of cameras in a sub-segment with geographic proximity will generate different patterns, as sub pictures of Fig. 1 show. Related definitions of *platoon* patterns are listed below:

Definition 1: (ANPR data record): An ANPR data record is a three-tuple, which can be denoted as $r = (c, p, t)$, where c is a camera ID, p is a vehicle plate number, and t is a timestamp. It means a vehicle p passes through a camera c at time t.

Definition 2: (moment companion): Let Δt be the time threshold, a moment companion for camera c at time t can be defined as: $MC(c, t, \Delta t) = \{r{\cdot}p \mid r{\cdot}c = c \wedge t - \Delta t \le r{\cdot}t \le t\}$, where $r = (c, p, t)$ is an ANPR data record, and $|MC(c, t, \Delta t)| > 1$.

A moment companion is a vehicle group, which passes through a camera during a short period.

Definition 3: (platoon companion pattern): Let Δt be the time threshold, $\delta_{\max}$ be maximal time interval threshold, δ_{cc} be the consecutive camera number threshold, and δ_{tc} be the total camera number threshold, (P,C,T) where P is a vehicle set, C is a camera sequence and T is an ordered timestamp sequence, is called a *platoon companion pattern* (short for *platoon*), if it satisfies the following conditions:

(i) $|P| > 1$, $|C| = |T|$, and $|C| \ge \delta_{tc}$;

(ii) for each camera $c_i \in C$ and $t_i \in T$, $P \subseteq MC(c_i, t_i, \Delta t)$;

(iii) for any adjacent cameras c_i and c_{i+1} in C, $|t_i - t_{i+1}| \le \delta_{\max}$;

(iv) $C = <C_1, C_2, \ldots, C_k>$, C_j $(j = 1, 2, \ldots, k)$ is a sub-sequence of consecutive cameras, and $|C_j| \ge \delta_{cc}$.

4 Platoon discovery on ANPR data stream

4.1 Problem analysis

Before introducing our algorithm, we list some related concepts to frequent sequence mining. A *sequence* in a sequence set D is associated with an identifier, called a *sid*. A sequence s' is called a *consecutive subsequence* of another sequence s if every two adjacent elements in s' are also adjacent elements in s. A *support* of a sequence is the number of it is contained in D. A *support* of a sequence in a sequence set D is the number of occurrences this particular sequence happened in D. A sequence becomes *frequent* if its support exceeds a pre-specified minimum support threshold in D. A frequent sequence with length k is called *k-frequent sequence*. It becomes *closed* if it has no super-sequence with the same support in the sequence set D. A *projection database* of sequence s in D is defined as $D_s = \{\alpha \mid \eta \in D, \eta = \beta \diamond \alpha\}$ (β is the minimum prefix of η containing s).

According to our previous work [11–13], an ANPR dataset can be transformed into a sequence set through grouping by vehicle plate numbers. Such a sequence under the ANPR data can also be viewed as a representation of a vehicle trajectory. In such a sequence set D, each sequence is associated with a vehicle plate number as its *sid*. A moment companion can be viewed as a one-frequent sequence occurring in a time period Δt in D. Also, a *platoon* pattern is the same as a frequent sequence, which consists of several consecutive camera subsequences in D, each element of which is a one-frequent sequence occurring in Δt in D. Table 2 shows a running example. Assuming that $\Delta t = 10$ s, c_1 occurs in p_1 and p_2 in 10 s and forms a one-frequent sequence within period 10 s, it indicates that vehicles p_1 and p_2 pass through c_1 in 10 s and constitutes a moment companion. A frequent sequence $\langle c_2, c_4, c_7, c_8\rangle$ is contained in p_3 and p_4. In this frequent sequence, $\langle c_2, c_4\rangle$ and $\langle c_7, c_8\rangle$ are two consecutive subsequences of p_3 and p_4. It implies that cameras c_2 and c_4 are adjacent and cameras c_7 and c_8 are adjacent geographically. Hence, (P, C, T) can be regarded as a *platoon* pattern under the case of $\delta_{cc} = 2$, where $P = \{p_3, p_4\}$, $C =$

Table 2 Sample of a sequence set on ANPR dataset in a sliding window

sid	t_1 = 9:35:00	t_2 = 9:35:03	t_3 = 9:35:05	t_4 = 9:40:20	t_5 = 9:40:23	t_6 = 9:40:26	t_7 = 9:45:22	t_8 = 9:45:28	t_9 = 9:45:29	t_{10} = 9:48:32	t_{11} = 9:48:39	t_{12} = 9:52:31	t_{13} = 9:52:38
p_1	—	c_1	—	c_2	—	—	c_4	—	—	c_5	—	—	c_6
p_2	c_1	—	—	—	—	c_3	—	c_4	—	—	c_6	c_7	—
p_3	c_2	—	—	—	—	c_4	c_6	—	—	c_7	—	—	c_8
p_4	—	c_2	—	—	c_4	—	—	—	c_5	—	c_7	c_8	—
p_5	—	c_3	—	c_5	—	—	—	—	c_9	—	c_{12}	c_{11}	—
p_6	c_3	—	—	—	—	c_5	—	c_9	—	c_{12}	—	—	c_{13}
p_7	—	—	c_3	—	c_5	—	—	c_7	—	c_{10}	—	—	c_{11}
p_8	—	c_4	—	—	—	c_6	c_8	—	—	—	c_{10}	c_{12}	—
p_9	c_4	—	—	c_6	—	—	—	—	c_8	c_9	—	—	c_{12}
p_{10}	c_5	—	—	—	—	c_7	—	c_{10}	—	c_{13}	—	c_{17}	—
p_{11}	—	—	c_5	—	c_7	—	c_{10}	—	—	—	c_{15}	—	c_{16}
p_{12}	—	c_5	—	c_7	—	—	—	—	c_{11}	—	c_{14}	c_{15}	—

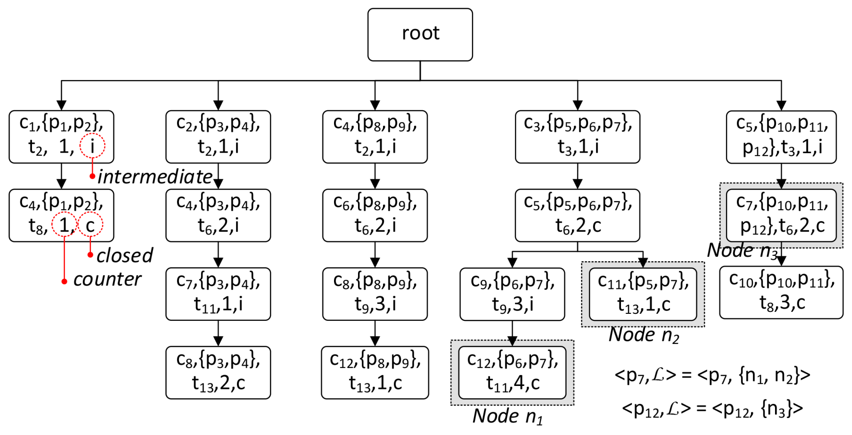

Fig. 2 *Example of PTree based on Table 2*

$\langle c_2, c_4, c_7, c_8\rangle$, and $T = \langle t_2, t_6, t_{11}, t_{13}\rangle$. However, existing algorithms cannot discover such frequent sequences since they do not consider the time interval in a one-frequent sequence (temporal constraint) and the location among elements in the sequence set (spatial constraint).

4.2 (Δt,1)-frequent sequence and PTree

We propose a new concept, $(\Delta t, 1)$-frequent sequence, for discovering *platoon* patterns. Traditionally, a frequent sequence with length 1 is called one-frequent sequence. In this paper, a 1-frequent sequence occurring in the time period Δt in a sequence set D is called a $(\Delta t, 1)$-frequent sequence. A $(\Delta t, 1)$-frequent sequence can be denoted as $c:(P, t, \Delta t)$, where c is the one-frequent sequence, P identifies the sequences containing c within Δt, and t is the latest occurrence timestamp of c in P within Δt. A $(\Delta t, 1)$-frequent sequence $c:(P, t, \Delta t)$ is equated to a moment companion $MC(c, t, \Delta t)$.

Our previous work proposed a data structure, a rooted prefix tree, to record discovered *travelling companion* patterns over the ANPR stream [13]. Each node of the tree contains a camera id, a vehicle plate number set, a time interval, a node type and so on. The first three items form a moment companion. It is an inverse tree, which the upper node contains newer information. In this study, we propose PTree based on our previous work to mine *platoon* patterns.

Definition 4: (PTree): PTree is a tree structure defined below:

(i) It consists of one root labelled as “null” and a set of item prefix subtrees as the children of the root.
(ii) Each node n in a PTree consists of six fields: *nid* (a node identifier), *cameraID* (a camera identifier), P (a plate number set), t (the latest timestamp of P when passing *cameraID*), *counter* (camera number in the last sub-segment P passed, in which the cameras are geographically consecutive), and *type* (a node type, closed or intermediate, it is closed when the child nodes have smaller plate number set with n, or when n is a leaf node).
(iii) PTree has a reference list, which is called *PRef*. *PRef* consists of a set of key value pairs $p, \mathscr{L}$, where p is a vehicle plate number and $\mathscr{L}$ is a set of node references. Any $n_i \in \mathscr{L}$ is the deepest closed node containing p in each branch with n_i. $t \in [t_{\text{cur}}, t_{\text{cur}} - \delta_{\max}]$ (t_{cur} is the current time).

PTree is a tree structure recording all *platoon* patterns. A node n of a PTree corresponds to a sequence that starts from the root to node n, and the sequence is denoted by s_n. Fig. 2 shows an example of a PTree based on Table 2.

4.3 PANDA algorithm

PANDA algorithm is developed to mine *platoon* patterns by maintaining a PTree. The straightforward idea of maintaining PTree is as follows: once a new ANPR data record r arrives, a new moment companion MC related to r is generated. It tries to compare each node with MC to extend, update this node or build a new depth-1 node under the root node.

However, there are two shortcomings of the straightforward idea. Firstly, a PTree may have a lot of depth-1 nodes, and most of them are redundant. This is because, at peak hours, as vehicles successively pass through a camera within a small time interval, it will continuously generate new moment companions which have a lot of overlaps between them. Moreover, most of the depth-1 nodes are useless as they will not constitute a *platoon* pattern since the vehicle groups probably split up for different destinations. Secondly, the size of a PTree will increase sharply with new data arriving. The tree traversal cost for comparison between new

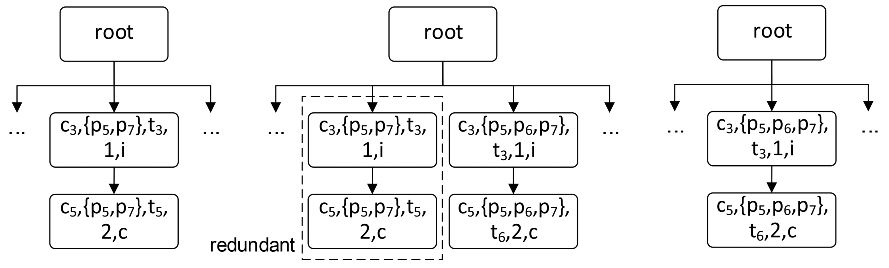

Fig. 3 *Example of redundant nodes*

generated moment companions and nodes on the tree becomes a huge challenge. It will significantly reduce the efficiency of PANDA in handling ANPR data stream. Our previous work laid a foundation for solving the above problems. From this work, we can gain the following conclusion. Given a new data record r arriving at time t, D_r^+ will contain all newly emerging *platoon* patterns related to the arrival of r, where $D_r^+ = \{r' \mid r' \cdot v \in MC(c, t, \Delta t)\}$. The conclusion indicates that when a vehicle passes through a camera, a new data record r is generated. Assume that r leads to a new moment companion $MC(r.c, r.t, \Delta t)$. Any newly generated or updated *platoon* pattern related to $r.p$ is contained in trajectory streams of vehicles in $MC(r.c, r.t, \Delta t)$.

Given a platoon pattern (P, C, T) occurred/updated due to the new data record r, P and the last element in C and T is a part of $MC(r.c, r.t, \Delta t)$ and is not contained in current PTree. However, P, the second last element of C and T should by contained by a closed node in current PTree. In summary, the new *platoon* pattern corresponds to a frequent camera sequence. The suffix of the sequence with length 2 (i.e. 2-suffix) can greatly help us to update PTree. Discovering the 2-suffix is a much easier task than tree traversal.

This idea can help address the two shortcomings concluded above. Firstly, it can avoid adding useless depth-1 nodes under the root. According to experiments in our previous work [13], for a group of vehicles, moving together under a few cameras are probably occasional. Adding a new depth-1 node under the root node once, a new vehicle group moving together under one camera occurs is unnecessary. In this study, only after the new vehicle group moves together under two cameras within the maximal time interval threshold (definition 3), it will be added under root node as a branch with two nodes, such as the leftmost subtree in Fig. 2. This can be done easily by the 2-suffix corresponding to the new vehicle group.

Secondly, it can reduce the tree traversal. If a new occurrence of a vehicle group under a camera within Δt is an extension of an existing *platoon* pattern, the existing part is obviously contained by PTree. The latest moment companion in the existing *platoon* pattern must be stored in a closed node. Therefore, we design a reference list PRef, which consists of a set of key value pairs $p, \mathscr{L}$, for PTree. Owing to PRef, we can easily locate the latest moment companion of the existing *platoon* pattern in PTree by matching nodes with the prefix of the 2-suffix corresponding to the vehicle group.

Based on the above, we propose a PANDA algorithm. Our algorithm uses a sliding window to handle ANPR data stream. Window size should be no less than $\delta_{\max} + \Delta t + 1$ s. The reason is two adjacent timestamps in a *platoon* pattern is required to span over no more than $\delta_{\max}$, due to the maximal time interval constraint in definition 3. Besides, a smaller window requires lower computational complexity, and a larger window will generate patterns that are more likely to violating maximal time interval constraint, therefore we set the window size to be $\delta_{\max} + \Delta t + 1$ s.

Denote the above new *platoon* pattern (P, C, T) as $C = \langle c_1, c_2, \ldots, c_k \rangle$ and $T = \langle t_1, t_2, \ldots, t_k \rangle$. Once r arrives in a window, PANDA firstly discovers the 2-suffix of the frequent camera sequence corresponding to (P, C, T), i.e. $(P, \langle c_{k-1}, c_k \rangle, \langle t_{k-1}, t_k \rangle)$.

PANDA utilises *PRef* to determine which node should be extended. It searches *PRef* to find $\langle r.p, \mathrm{L} \rangle$ and compares P, c_{k-1} and t_{k-1} with each node linked in L. If each vehicle in group P passed camera c_{k-1} recorded in the node under the same time with that in P, c_{k-1} and t_{k-1}, we call they are matched up. In this case, the PANDA tries to add a child node under the node storing P, c_k and t_k. Otherwise, PANDA compares the nearest closed ancestor node of the current node with P, c_{k-1} and t_{k-1}. The recursion is terminated until the timestamp of the node is no larger than t_k. When more than one node matches up with P, c_{k-1}, and t_{k-1}, PANDA tries to add a child node under the one with the largest depth. If there are multiple matched nodes with the largest depth, PANDA will select one of them randomly to extend.

After PANDA decides to add a child node under some node, it firstly checks whether the cameras in the two nodes have geographical proximity. In the case that the two cameras are not adjacent, the counter of the added node is set to be 1. The child node should be added under the nearest consecutive camera segment, the length of which is not less than δ_{cc} (consecutive camera number threshold). PANDA can search such a segment easily by the counter in a node. It will add the child node under such a segment when the time interval is no more than $\delta_{\max}$ (maximal time interval threshold). On the other hand, for consecutive cameras, the counter of the added node is its parent's counter increased by 1. After that, PANDA updates the *type* of the added node's parent and *PRef*.

If there is no matched node in L and its closed ancestors, PANDA adds a subtree of root consisting of the depth-1 node with P, c_{k-1} and t_{k-1} and depth-2 node with P, c_k and t_k. It checks and deletes redundant nodes with every addition of a new node. For example, as Table 2 shows, when vehicle p_7 passed camera c_5 at t_5, the PTree contains a branch as the left part in Fig. 3 shows. After vehicle p_6 passed camera c_5 at t_6, PANDA will add a new branch under root, as the middle part shows. In this case, the former branch is redundant and should be removed as the right part of Fig. 3 shows. Additionally, to prevent PTree from huge size, PANDA drops a subtree, if the timestamp in each leaf node of this subtree is $\delta_{\max}$ earlier than the current time. Note that, if there is no 2-suffix of the frequent camera sequence corresponding to the new moment companion, PANDA does not update the PTree (see Fig. 4).

Fig. 4 is the pseudo code of our PANDA algorithm. In Algorithm 1, parMiner(S') is a function to discover *platoon* patterns in the sequence set S'. Firstly, parMiner generates the new $-(\Delta t, 1)$ frequent sequence in S' related to new record r, denoted as f_r. Let the *sid* set corresponding to f_r be P_r. parMiner collects all $(\Delta t, 1)$-frequent sequence in the sequence set corresponding to P_r. It intersects P_r with each rest collected $(\Delta t, 1)$-frequent sequence and keeps all distinct vehicle groups containing $r.p$. For each distinct vehicle group, parMiner finds the last two collected $(\Delta t, 1)$-frequent sequences containing the group. It is the 2-suffix of the frequent camera sequence corresponding to a new *platoon* pattern related to r mentioned above.

5 Experiment

5.1 Experiment setup

Parameters: We do experiments to measure the effects and efficiency of PANDA algorithm. Our algorithm involves several key parameters Δt, $\delta_{\max}$, δ_{cc} and δ_{tc}. They should be pre-assigned before running the algorithm. The parameters' values are defined

```
Algorithm I PANDA
Input:  D: ANPR data stream in sliding window;
        $\Delta t$: time threshold;
        $\delta_{max}$: maximal time interval threshold;
        $\delta_{cc}$: consecutive camera number threshold;
        $\delta_{tc}$: total camera number threshold;
Output: updated PTree;
1.  group $D$ by plate number to generate a sequence set $S$;
2.  for each new record $r$
3.    generate its moment companion $MC$;
4.    get sequence set $S'$ related to $MC$;
5.    $NewPlatoons$ = parMiner($S'$); // mine new platoon patterns in $S'$
    related to $r$
6.    for each ($P$, $C$, $T$) in $NewPlatoons$, where $C = <c_1, c_2, \ldots, c_k>$, $T =$
    $<t_1, t_2, \ldots, t_k>$
7.      initialize node list $MatchedNodes$;
8.      $Ns = PRef$.get($r.p$);
9.      for each node $n$ in $Ns$
10.       match($n, P, c_{k-1}, t_{k-1}$);
11.     if $MatchedNodes$ is non-empty
12.       get a node $l$ with largest depth in $MatchedNodes$;
13.       if $c_{k-1}$ and $c_k$ are not consecutive and $l.counter < \delta_{cc}$
14.         $n'.counter$=1
15.         node $ancestor$ = getNearestSeg($l$);
16.         if $|parent.t - t_k| \leq \delta_{max}$
17.           add a child node $n'$ under $ancestor$;
18.       else
19.         add a child node $n'$ under $l$;
20.         $n'.counter = n'$.getparent().$counter$+1;
21.       $n'.P=P$, $n'.cameraID=c_k$, $n'.t=t_k$, $n'.type=closed$;
22.       update $type$ in $l$ and $<r.p, L>$ in $PRef$;
23.     else
24.       add a child node $n'$ under root;
25.       add a child node $n''$ under $n'$;
26. remove redundant nodes and expired subtrees;
27. return PTree;

28. match($n, P, c_{k-1}, t_{k-1}$)
29. if $n$ matches with $P$, $c_{k-1}$ and $t_{k-1}$
30.   if $|t_{k-1} - t_k| \leq \delta_{max}$
31.     add $n$ into $MatchedNodes$;
32.   break;
33. else if $n.t > t$
34.   match($n.getNearestClosedAncestor()$, $P$, $c_{k-1}$, $t_{k-1}$);
35. else
36.   break;

37. getNearestSeg($n$)
38. if $n.getParent().counter < \delta_{cc}$
39.   $getNearestSeg(n.getParenet())$;
40. else
41.   return $n.getParent()$;
```

Fig. 4 *Algorithm 1. PANDA*

based on the experiences gained from our previous work. Table 3 lists the values of each parameter selected in our experiments.

Baselines: To test the effectiveness of our algorithm, we compare our algorithm with three state-of-the-art algorithms on discovering *flock*/*convoy* pattern [3], *swarm* pattern [6] and *travelling companion* pattern [7]. There are two proposed algorithms to discover travelling companions, which generate the same results according to their experiments [7]. We select the first one in our experiment.

Datasets: The following experiments use a real ANPR dataset in Beijing, China. The dataset contains vehicle information from 21 December 2012 06:00:00 to 30 December 2012 20:00:00. Totally, 1040 cameras, 54,316,820 ANPR data records and 4,669,229 vehicles are involved. We simulate each dataset as a stream. The time interval between two adjacent data records is set in accordance with real-time intervals between two snapshots taken by cameras.

Environments: The experiments are performed on a PC with four Intel Core i7-7700k CPUs 4.2G Hz and 8.00 GB RAM. The operating system is Windows 7 Ultimate. All the algorithms are implemented in Java with JDK 1.8.0.

Table 3 Parameter settings

Parameters	Explanations	Values
Δt	time threshold of a moment companion	20, 40,…, 120 s
δ_{cc}	consecutive camera number threshold	1, 2, *len* (real total camera number)
δ_{tc}	total camera number threshold	2, 3,…, 10
δ_{max}	maximal time interval threshold	30 min
window size	window size	$\delta_{max} + \Delta t + 1$

5.2 Effectiveness

5.2.1 Effectiveness of PANDA algorithm: In this section, we test the effectiveness of our PANDA algorithm. As mentioned above, parameters Δt, δ_{cc} and δ_{tc} are the main factors, which can affect the results of PANDA. We firstly test the impact of these parameters on the discovered *platoon* patterns. On our real ten days ANPR dataset, we consider each day as a subset and input them as a stream into PANDA.

Definition 5: (average number): Given datasets R_1, R_2,…, R_m within the same time period, PANDA discovers n_i *platoon* patterns on R_I ($i = 1, 2,\ldots, m$). The average number on R_1, R_2,…, R_m is denoted as $\text{avg}(R_1, R_2,\ldots, R_m) = \sum_i n_i/m$.

According to our previous work, to guarantee the correctness of our algorithm, the sliding distance is set to be 1 s. According to the previous introduction, when $len/2 < \delta_{cc} \leq len$ (*len* is the real total camera number of a *platoon* pattern), PANDA will discover *flock*/*convoy* patterns. In this case, we set $\delta_{cc} = len$ in our experiments. On the other hand, $\delta_{cc} = 1$ will enable PANDA to discover *swarm*/*travelling companion* patterns. We also select $\delta_{cc} = 2$ to verify common types of *platoon* patterns. Besides, to avoid double counting on discovered patterns, we only consider platoon patterns corresponding to closed nodes in PTree. These patterns are maximal, i.e. they are not sub-patterns of any in the vehicle group, camera sequence or time sequence.

Fig. 5 shows the final results. Each sub picture in it illustrates results under different values of Δt. In each sub-picture, the average number of discovered *platoon* patterns drops exponentially as parameter δ_{tc} increases. We infer that companion behaviour under low δ_{tc} is probably occasional. Hence, the average number, in this case, is large. The figure shows that a *platoon* pattern can contain ten cameras and more. When δ_{tc} reaches 9, the average number of *platoon* patterns is around single-digit. However, *platoon* patterns under larger δ_{tc} attract more attention from users. A smaller δ_{cc} normally corresponds to the larger volume of patterns as a smaller δ_{cc} means the constraint is looser. For example, $\delta_{cc} = 1$ equals to *swarm*/*travelling companion* pattern, which has the weakest constraint on camera location among the three types of *platoon* patterns.

On the other hand, the difference between three types ($\delta_{cc} = 1$, $\delta_{cc} = 2$ and $\delta_{cc} = len$) of *platoon* pattern changes with the growth of δ_{tc}. The number of patterns that can be generated under $\delta_{cc} = 1$ is more than what can be generated by the other two values (i.e. 2 and *len*). When $\delta_{tc} \leq 8$, the difference between pattern number under $\delta_{cc} = 2$ and $\delta_{cc} = len$ becomes larger; when $8 < \delta_{tc} \leq 10$, the difference trends to be 0.

Overall, with the growth of Δt, the average number of discovered *platoon* number increases linearly. Alternatively, the average number drops significantly with the increasing of δ_{tc}. With the rise of δ_{cc}, the difference between pattern number under $\delta_{cc} = 2$ and $\delta_{cc} = len$ firstly becomes larger, then smaller.

5.2.2 Comparative effectiveness of different approaches: In this part, we compare the effectiveness of our algorithm with the three baseline methods using the 10 days of ANPR datasets

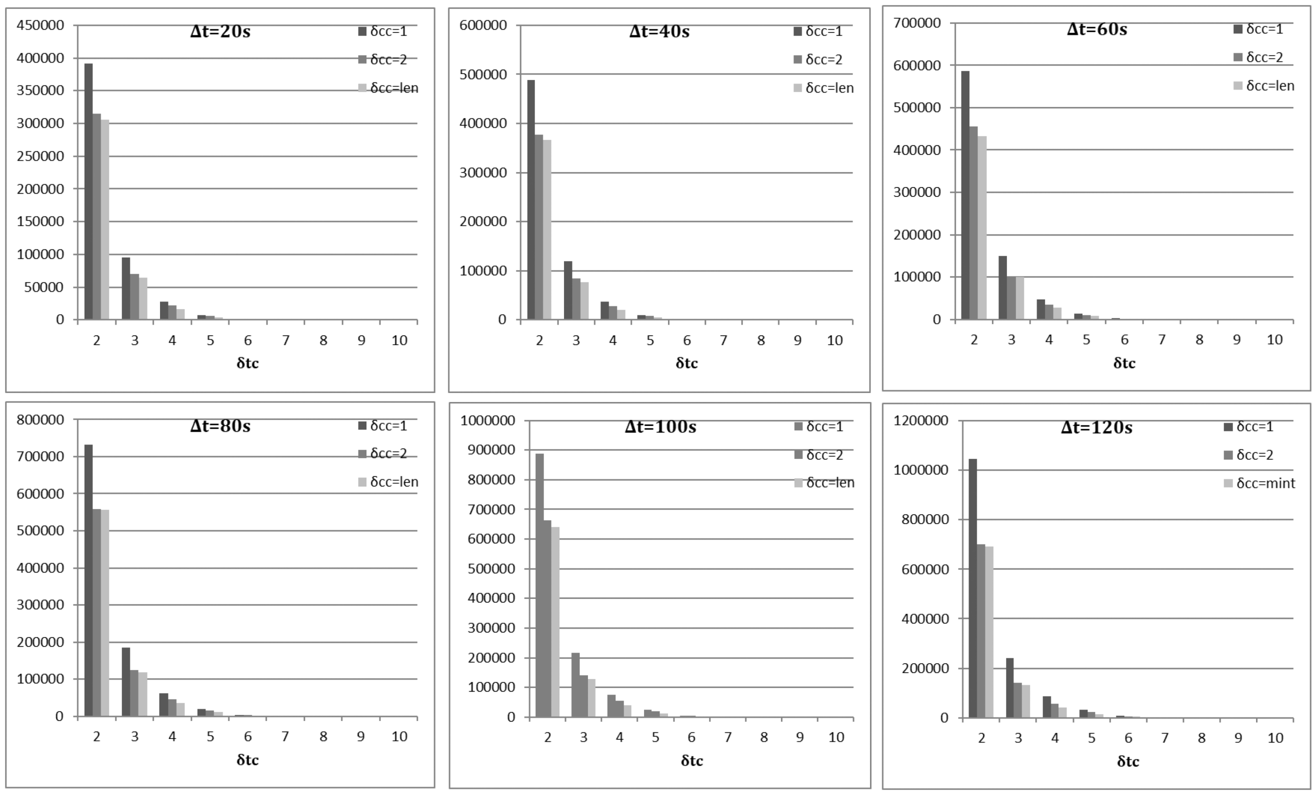

Fig. 5 *Average numbers per day of discovered platoon patterns under different Δt, δ_{cc} and δ_{tc}*

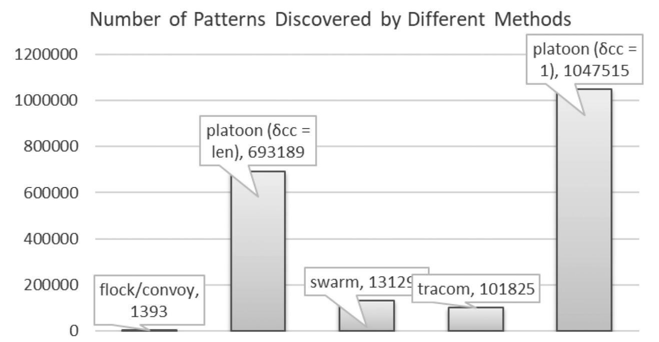

Fig. 6 *Average number per day of flock/convoy, swarm, travelling companion (short for TraCom) and platoon patterns*

integrated with the camera location dataset. According to the baseline methods, we set the distance threshold to be 300 m. On the other hand, we input the ANPR datasets into PANDA algorithm with $\Delta t = 60$ s, $\delta_{\rm cc} = 1$, and $\delta_{\rm tc} = 2$ and $\Delta t = 60$ s, $\delta_{\rm cc} = len$, and $\delta_{\rm tc} = 2$, respectively. Experimental results are shown in Fig. 6.

As Fig. 6 shows, the discovered *flock*/*convoy* patterns have the smallest number, which is 1393. It is because a *flock*/*convoy* pattern has a stronger temporal constraint than a *swarm* pattern and a *travelling companion* pattern. A group of vehicles have to travel together for a series of consecutive timestamps to form a *flock*/*convoy* pattern. On the other hand, a *swarm* and a *travelling companion* pattern is a group of vehicles travelling together for a series of consecutive or non-consecutive timestamps. Besides, the discovered *swarm* patterns are a little more than *travelling companion* patterns. We look into the two kinds of patterns to seek the reason, which is the *swarm* discovering algorithm adopts distance-based clustering method [6] and *travelling companion* discovering algorithm utilises the density connected clustering method [7]. The former method will depart two cameras if their distance is larger than the pre-specified value. However, the latter one may group them together if there exist another camera *c* and the distance between any camera and the camera *c* is less than the pre-specified value. Owing to this difference, a pattern discovered by *TraCom travelling companion* discovering algorithm may be divided into two separate patterns by the *swarm*-discovering algorithm.

Fig. 5 shows that the number of discovered patterns increases when the value of $\delta_{\rm cc}$ decreases. The range of $\delta_{\rm cc}$ is from 1 to *len*. Under the scenario, $\delta_{\rm cc} = 1$ or $\delta_{\rm cc} = len$, our PANDA algorithm discovers the most patterns than the other three baseline algorithms. Thus, we infer that our algorithm will generate the most patterns no matter what value $\delta_{\rm cc}$ is set at. We compare all patterns generated by PANDA with the patterns generated by the three baseline methods, we then find out that any pattern that is discoverable by the latter can also be discovered by PANDA. The reason is that there is more than one camera at each road crossing to monitor vehicles from different directions. Cameras at the same road crossing locate closely to each other so that they are grouped together by the baseline methods. Thus, the baseline methods frequently detect vehicles that pass different cameras on the same road crossing at the same time. That is why each baseline method can detect partial patterns over the ANPR dataset.

5.3 Efficiency

In this section, we analyse the impact of different Δ*t* on the efficiency of PANDA.

Definition 6: (average latency): Given an ANPR data stream *S*, let the current length of *S* be *N*. Denote the time for discovering *platoon* patterns for each record r_i in *S* as t_i. The average latency of our algorithm on handling *S* can be defined as $t_{\rm avg}(S) = \sum_i t_i / N$.

In the experiment to test efficiency, we test the average latency with $\delta_{\rm cc} = 1$, $\delta_{\rm tc} = 2$, $\delta_{\rm max} = 30$ min under different values of Δ*t* from 20, 40,…, 240 s. Less $\delta_{\rm cc}$ means weaker constraint, which leads to more *platoon* patterns. Similarly, less $\delta_{\rm tc}$ will also generate more patterns. In this experiment, we try to test PANDA's average latency under different values of Δ*t* while maintaining a large PTree with $\delta_{\rm cc} = 1$, $\delta_{\rm tc} = 2$, $\delta_{\rm max} = 30$ min.

We run our algorithm 20 times for different Δ*t*. Each time our algorithm runs for 2 h from 8:00:00 to 10:00:00 on each day by continuously receiving arriving ANPR data records and instantly generate discovered patterns. We will compute the latency value

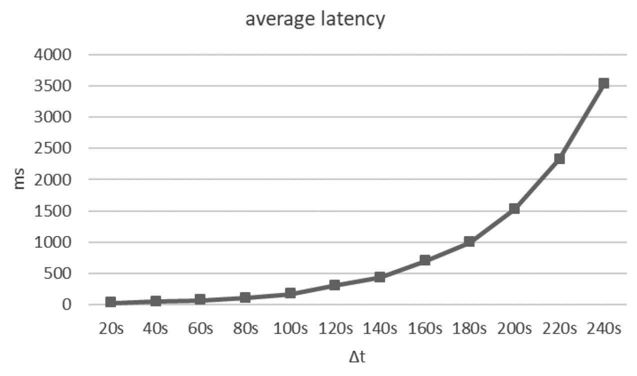

Fig. 7 *Average latency under different values of δt*

based on definition 6. Finally, the average latency values are shown in Fig. 7.

The experiment results show that our algorithm can handle the real ANPR data stream when $\Delta t \leq 180$ s instantly. It is because when $\Delta t \leq 180$ s, the average latency of our algorithm is no more than 1 s, which is the minimum time interval between two ANPR data records. Under $\Delta t > 180$ s, the average latency increases significantly. Under these situations, unprocessed input ANPR data may pile up in the PANDA. In the future work, we will try to optimise our algorithm to solve this issue.

Besides, we compared our algorithm with three state-of-the-art algorithms on the efficiency. They show lower average latency than our algorithm. Owing to Fig. 6, the three baseline algorithms discovered many fewer patterns than our algorithm. In this case, their lower average latency cannot prove that the three algorithms have better efficiency than ours.

6 Conclusion

This study proposed a PANDA algorithm to instantly discover *platoon* companions from a special kind of streaming traffic data, called ANPR data. We transform the *platoon* companion discovery into a frequent sequence-mining problem with temporal and spatial constraints. A tree structure, PTree with a reference list, is designed to record discovered *platoon* companions. To optimise tree traversal during updating a PTree, we utilise the 2-suffix of a frequent sequence corresponding to a *platoon* pattern. Experiment results verify that our algorithm can effectively and efficiently handle a real ANPR data stream when $\Delta t \leq 180$ s.

7 Acknowledgments

This work was supported by the National Natural Science Foundation of China (no. 61672042), models and methodology of data services facilitating dynamic correlation of big stream data; the Beijing Natural Science Foundation (no. 4172018), building stream data services for spatio-temporal pattern discovery in cloud computing environment; the program for youth backbone individual, supported by the Beijing Municipal Party Committee Organisation Department, research of instant fusion of multi-source and large-scale sensor data.

8 References

[1] Laube, P., Imfeld, S.: 'Analyzing relative motion within groups of trackable moving point objects'. Proc. Second Int. Conf. on Advances in Geographic Information Systems (GIScience), Boulder, CO, USA, September 2002, pp. 132–144

[2] Gudmundsson, J., Van Kreveld, M.: 'Computing longest duration flocks in trajectory data'. Proc. 14th Annual ACM Int. Symp. on Advances in Geographic Information Systems (ACM GIS), Arlington, Virginia, USA, November 2006, pp. 35–42

[3] Vieira, M.R., Bakalov, P., Tsotras, V.J.: 'On-line discovery of flock patterns in spatio-temporal data'. Proc. 17th ACM Int. Symp. on Advances in Geographic Information Systems (ACM SIGSPATIAL GIS), Seattle, Washington, USA, November 2009, pp. 286–295

[4] Jeung, H., Shen, H.T., Zhou, X.: 'Convoy queries in spatio-temporal databases'. Proc. IEEE 24th Int. Conf. Data Engineering (ICDE), Cancun, Mexico, April 2008, pp. 1457–1459

[5] Jeung, H., Yiu, M. L., Zhou, X., *et al.*: 'Discovery of convoys in trajectory databases', *Proc. VLDB Endowment*, 2008, **1**, (1), pp. 1068–1080

[6] Li, Z., Ding, B., Han, J., *et al.*: 'Swarm: mining relaxed temporal moving object clusters', *Proc. VLDB Endowment*, 2010, **3**, (1), pp. 723–734

[7] Tang, L.A., Zheng, Y., Yuan, J., *et al.*: 'On discovery of traveling companions from streaming trajectories'. Proc. IEEE 28th Int. Conf. on Data Engineering (ICDE), Washington, DC, USA, April 2012, pp. 186–197

[8] Tang, L.A., Zheng, Y., Yuan, J., *et al.*: 'A framework of traveling companion discovery on trajectory data streams', *ACM Trans. Intell. Syst. Technol.*, 2013, **5**, (1), pp. 1–34

[9] Nautiyal, A., Lal, R.P.: 'Time-efficient discovery of moving object groups from trajectory data'. Proc. Innovations in Computer Science and Engineering (ICICSE), Singapore, June 2017, pp. 185–192

[10] Li, Y., Bailey, J., Kulik, L.: 'Efficient mining of platoon patterns in trajectory databases', *Data Knowl. Eng.*, 2015, **100**, pp. 167–187

[11] Han, Y., Wang, G., Yu, J., *et al.*: 'A service-based approach to traffic sensor data integration and analysis to support community-wide green commute in China', *IEEE Trans. Intell. Transp. Syst.*, 2016, **17**, (9), pp. 1–10

[12] Zhu, M., Liu, C., Wang, J., *et al.*: 'A service-friendly approach to discover traveling companions based on ANPR data stream'. Proc. IEEE Int. Conf. on Services Computing (SCC), San Francisco, CA, USA, June 2016, pp. 171–178

[13] Liu, C., Wang, X., Zhu, M., *et al.*: 'Discovering companion vehicles from live streaming traffic data'. Proc. 17th Asia-Pacific Web Conf. (APWeb), Suzhou, China, September 2016, pp. 116–128

[14] Kalnis, P., Mamoulis, N., Bakiras, S.: 'On discovering moving clusters in spatio-temporal data'. Proc. Ninth Int. Symp. on Spatial and Temporal Databases (SSTD), Santorini Island, Greece, August 2005, pp. 364–381

[15] Li, Y., Han, J., Yang, J.: 'Clustering moving objects'. Proc. 10th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD), Seattle, Washington, USA, August 2004, pp. 617–622

[16] Kriege, H.P., Pfeifle, M.: 'Density-based clustering of uncertain data'. Proc. 11th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD), Chicago, Illinois, USA, August 2005, pp. 672–677

[17] Jensen, C.S., Lin, D., Ooi, B.C.: 'Continuous clustering of moving objects', *IEEE Trans. Knowl. Data Eng.*, 2007, **19**, (9), pp. 1161–1174

[18] Wang, Y., Luo, Z., Takekawa, J., *et al.*: 'A new method for discovering behavior patterns among animal movements', *Int. J. Geogr. Inf. Sci.*, 2015, **30**, (5), pp. 929–947

[19] Silva, T.L.C.D., Zeitouni, K., Macêdo, J.A.F.D., *et al.*: 'A framework for online mobility pattern discovery from trajectory data streams'. IEEE Int. Conf. on Mobile Data Management (MDM), Porto, Portugal, June 2016, pp. 365–368

[20] Zheng, K., Zheng, Y., Yuan, N.J., *et al.*: 'Online discovery of gathering patterns from trajectories', *IEEE Trans. Knowl. Data Eng.*, 2014, **26**, (8), pp. 1974–1988

[21] Yoo, J.S., Boulware, D., Kimmey, D: 'A parallel spatial co-location mining algorithm based on map reduce'. Proc. IEEE Int. Congress on Big Data (BigData Congress), Anchorage, AK, USA, June 2014, pp. 25–31

[22] Zhang, J., Li, J., Wang, S., *et al.*: 'On retrieving moving objects gathering patterns from trajectory data via spatio-temporal graph'. Proc. IEEE Int. Congress Big Data (Big Data), Anchorage, AK, USA, June 2014, pp. 390–397

[23] Zhang, J., Li, J., Liu, Z., *et al.*: 'Moving objects gathering patterns retrieving based on spatio-temporal graph', *Int. J. Web Serv. Res.*, 2016, **13**, (3), pp. 88–107

[24] Xian, Y., Liu, Y., Xu, C.: 'Parallel gathering discovery over big trajectory data'. IEEE Int. Conf. on Big Data (Big Data), Washington, DC, USA, December 2016, pp. 783–792

[25] Yu, Y., Wang, Q., Wang, X.: 'Continuous clustering trajectory stream of moving objects', *China Commun.*, 2013, **10**, (9), pp. 120–129

[26] Yu, Y., Wang, Q., Wang, X., *et al.*: 'Online clustering for trajectory data stream of moving objects', *Comput. Sci. Inf. Syst.*, 2013, **10**, (3), pp. 1293–1317

[27] Mooney, C.H., Roddick, J.F.: 'Sequential pattern mining: approaches and algorithms', *ACM Comput. Surv.*, 2013, **45**, (2), pp. 94–111

[28] Agrawal, R., Srikant, R.: 'Mining sequential patterns'. Proc. 1995 IEEE 11th Int. Conf. on Data Engineering (ICDE), Taipei, Taiwan, March 1995, pp. 3–14

[29] Srikant, R., Agrawal, R.: 'Mining sequential patterns: generalizations and performance improvements'. Proc. 5th Int. Conf. Extending Data Base Technology (EDBT), Avignon, France, March 1996, pp. 3–17

[30] Zaki, M.J.: 'SPADE: an efficient algorithm for mining frequent sequences', *Mach. Learn.*, 2001, **42**, (1/2), pp. 31–60

[31] Ayres, J., Flannick, J., Gehrke, J., *et al.*: 'Sequential pattern mining using a bitmap representation'. Proc. 8th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD), Edmonton, Alberta, Canada, July 2002, pp. 429–435

[32] Han, J., Pei, J., Mortazavi-Asl, B., *et al.*: 'Freespan: frequent pattern-projected sequential pattern mining'. Proc. 6th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD), Boston, MA, USA, August 2000, pp. 355–359

[33] Pei, J., Han, J., Mortazavi-Asl, B., *et al.*: 'Prefixspan: mining sequential patterns efficiently by prefix-projected pattern growth'. Proc. IEEE 17th Int. Conf. Data Engineering (ICDE), Heidelberg, Germany, August 2001, pp. 215–224

[34] Yan, X., Han, J., Afshar, R.: 'Clospan: mining closed sequential patterns in large databases'. Proc. Third SIAM Int. Conf. Data Mining (SDM), San Francisco, CA, USA, May 2003, pp. 166–177

[35] Wang, J., Han, J., Li, C.: 'Frequent closed sequence mining without candidate maintenance', *IEEE Trans. Knowl. Data Eng.*, 2007, **19**, (8), pp. 1042–1056

[36] Pinto, H., Han, J., Pei, J., *et al.*: 'Multi-dimensional sequential pattern mining'. Proc. 10th Int. Conf. on Information and Knowledge Management (CIMK), Atlanta, Georgia, USA, January 2001, pp. 81–88

[37] Pei, J., Han, J., Wang, W.: 'Constraint-based sequential pattern mining: the pattern-growth methods', *J. Intell. Inf. Syst.*, 2007, **28**, (2), pp. 133–160

[38] Chueh, H.E.: 'Mining target-oriented sequential patterns with time-intervals', *Int. J. Comput. Sci. Inf. Technol.*, 2010, **2**, (4), pp. 113–123

[39] Demiriz, A.: 'webSPADE: a parallel sequence mining algorithm to analyze web log data'. Proc. IEEE Int. Conf. on Data Mining (ICDM), Maebashi, Japan, December 2002, pp. 755–758

[40] Guralnik, V., Karypis, G.: 'Parallel tree-projection-based sequence mining algorithms', *Parallel Comput.*, 2004, **30**, (4), pp. 443–472

[41] Ma, C., Li, Q.: 'Parallel algorithm for mining frequent closed sequences'. Proc. Int. Workshop on Autonomous Intelligent Systems: Agents and Data Mining (AIS-ADM), Petersburg, Russia, June 2005, pp. 184–192

[42] Qiao, S., Tang, C., Dai, S., *et al.*: 'Partspan: parallel sequence mining of trajectory patterns'. Proc. 5th Int. Conf. on Fuzzy Systems and Knowledge Discovery (FSKD), Jinan, Shandong, China, October 2008, pp. 363–367

[43] Yu, D., Wu, W., Zheng, S., *et al.*: 'BIDE-based parallel mining of frequent closed sequences with mapreduce'. Proc. 12th Int. Conf. on Algorithms and Architectures for Parallel Processing (ICA3PP), Fukuoka, Japan, November 2012, pp. 177–186

[44] Kessl, R.: 'Probabilistic static load-balancing of parallel mining of frequent sequences', *IEEE Trans. Knowl. Data Eng.*, 2016, **28**, (5), pp. 1299–1311

[45] Chang, L., Wang, T., Yang, D., *et al.*: 'SeqStream: mining closed sequential patterns over stream sliding windows'. Proc. 8th IEEE Int. Conf. on Data Mining (ICDM), Pisa, Italy, February 2008, pp. 83–92

[46] Cheng, H., Yan, X., Han, J.: 'IncSpan: incremental mining of sequential patterns in large database'. Proc. 10th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD), Seattle, Washington, USA, August 2004, pp. 527–532

[47] Chen, G., Wu, X., Zhu, X.: 'Sequential pattern mining in multiple streams'. Proc. IEEE 5th Int. Conf. Data Mining (ICDM), Houston, Texas, USA, November 2005, pp. 585–588

[48] Ho, C.C., Li, H.F., Kuo, F.F., *et al.*: 'Incremental mining of sequential patterns over a stream sliding window'. Workshops Proc. IEEE 6th Int. Conf. on Data Mining (ICDM - Workshops), Hong Kong, China, January 2006, pp. 677–681

[49] Yuan, D., Lee, K., Cheng, H., *et al.*: 'CISpan: comprehensive incremental mining algorithms of closed sequential patterns for multi-versional software mining'. Proc. 8th SIAM Int. Conf. Data Mining (SDM), Atlanta, Georgia, USA, April 2008, pp. 84–95

[50] Gao, C., Wang, J., Yang, Q.: 'Efficient mining of closed sequential patterns on stream sliding window'. Proc. IEEE 11th Int. Conf. Data Mining (ICDM), Vancouver, BC, Canada, January 2011, pp. 1044–1049