An Approach to Instant Discovering Companion Vehicles from Live Streaming ANPR Data

Meiling Zhu[1,2,3], Chen Liu[2,3], Jianwu Wang[4], Yanbo Han[2,3]

[1] School of Computer Science and Technology, Tianjin University, China
[2] Beijing Key Laboratory on Integration and Analysis of Large-Scale Stream Data, North China University of Technology, China
[3] Cloud Computing Research Center, North China University of Technology, China
[4] Department of Information Systems, University of Maryland, U.S.A
meilingzhu2006@126.com, liuchen@ncut.edu.cn, jianwu@umbc.edu, hanyanbo@ncut.edu.cn

Abstract

Companions of moving objects are object groups that move together in a period of time. This paper proposes to instantly discover companion vehicles from a special kind of streaming sensor data, called Automatic Number Plate Recognition (ANPR) data. Compared to related approaches, we transform the companion discovery into a frequent sequence mining problem. We make several improvements on top of our previous work, including one scan and tree traversal reduction, to optimize the performance of our previous approach and accelerate the process of discovering companion vehicles. Finally, extensive experiments are done to show efficiency and effectiveness of our approach.

Keywords: Companion vehicles, ANPR data stream, Moment companion, Traveling companion, Frequent sequence mining

1 Introduction

To monitor the running of the real world, widely deployed sensors today keep generating a large volume of streaming sensor data. Instant analyses of them can create lots of interesting value-added applications that we never imagined before. In this paper, we put our focus on a special kind of streaming sensor data, which is called Automatic Number Plate Recognition (ANPR) data. It is generated by the cameras deployed on roads through continuously taking pictures of passing vehicles.

Compared with GPS (Global Positioning System) data, ANPR data provides another source to study the movement patterns of vehicles in a large city. Companion vehicles are a useful movement pattern, which means vehicles that move together in a period of time. Today, timely discovering companion vehicles over traffic data stream has gradually attracts more attentions of researchers [1-8]. Such ability is very useful in many time sensitive scenarios like solving crimes, detecting suspicious trackers and pursuing escaped criminals. For example, vehicles such as bank cash carriers and taxies are more exposed to criminals' attentions in recent years. Criminals usually follow them for a while before reaching a suitable, secluded spot to commit a robbery. Detecting suspicious trackers over traffic data stream and alert the tracked driver in time can help to prevent such crimes.

However, lots of researchers study on how to discover companion vehicles based on historical or real-time GPS dataset [1-8]. GPS data is generated from GPS devices installed on vehicles and sent back within a fixed frequency. Vehicles without GPS device or turning such devices off will not generate enough data to make further analyses. For example, in some special occasions like vehicle tracking and criminal escaping, suspects often turn the GPS device off or even remove it to avoid being captured. Obviously, GPS data is not a good choice to be counted on when meeting requirements in such occasions.

To make useful complements to current researches, our previous works introduce a different kind of traffic big data, called as ANPR (Automatic Number Plate Recognition) data [9-11]. Compared with GPS data, ANPR data comes from Traffic Enforcement Cameras, which are a special kind of sensors installed at most road crossings in most cities in China. These cameras continuously take pictures of passing vehicles with approximate one second interval at rush hour. Vehicle information, e.g., plate number and passing time, is automatically recognized and transmitted to a data center of traffic management department in form of data stream. ANPR data provides the capacity of around-the-clock and wide-range monitoring of vehicles. Presently, the number of cameras installed in a big city in China exceeds 5,000 and continues to increase. It leads to the total number of ANPR records

*Corresponding Author: Meiling Zhu; E-mail: meilingzhu2006@126.com
DOI: 10.3966/160792642018121907005

in each day beyond 144 million. Eventually, the total data volume can get into multi-petabyte scale per annum.

In our previous works, we have designed several algorithms to effectively discover companion vehicles on ANPR data. In [9], we borrow 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 [10-11], we try to discover companion vehicles from live ANPR data stream. To reach this goal, we firstly propose a concept called Moment Companion, which records a snapshot of companion vehicles for a vehicle when it passes through a camera. Then, we transform the discovery of companion vehicles into a frequent sequence mining problem. However, we met performance bottlenecks when making guarantees about the instantaneity on discovering companion vehicles over ANPR data stream. Our previous experiments show that the latency of our previous algorithm under some case can reach 1500ms and even more on real ANPR data stream, which is higher than the minimum value of time interval between two real ANPR data records (1 second). It means there are probabilities that an obvious delay will be emerged with such latencies keeping accumulated.

In this paper, we make lots of efforts to consummate our previous work [11]. We focus on how to improve the performance of our algorithm when handling ANPR data stream. The main contributions include: (1) Borrowing ideas from [12], we revise our window model to receive and handle incoming ANPR data stream. With this new window model, we can greatly decrease the times of scanning data records of our algorithm. And the error is guaranteed not to exceed a user specified parameter. (2) We create a list Vlist of references of nodes on IST$^+$ (Improved Inverse Closed Sequence Tree) to reduce the cost of tree traversal. Based on Vlist, we only need to visit the nodes containing those accompanying vehicles with the new arriving ones instead of every node on IST+. (3) More experiments are done to show efficiency and effectiveness of our improved algorithms. We compare the performance between our previous algorithm and the improved algorithm on a real ANPR dataset and several simulated datasets with data arrival rate 1000, 2000, to 5000 records per second. The experimental results show that the performance of our algorithm is improved more than 70% and 59% on the real dataset and simulated datasets respectively.

2 Problem Analysis

Figure 1 shows the effects of our algorithm, which tries to instantly compute companions of a given vehicle passing through a camera. As this figure shows, cameras are spread over road crossings. Each camera is a source of a data stream, and continuously generates ANPR data records.

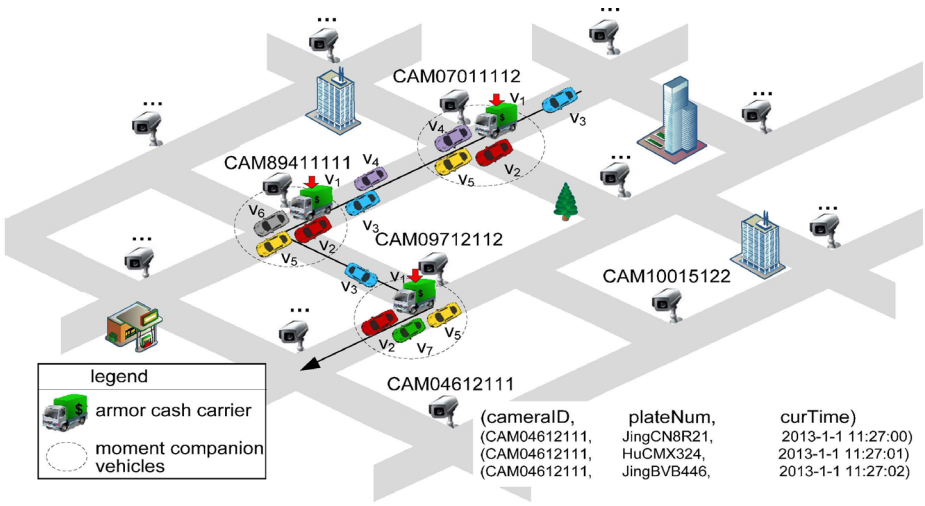

Figure 1. The illustration of discovering traveling companions

An ANPR data record $r = (c, v, t)$ means a vehicle v passes through a camera c at time t. For example, (CAM04612111, JingCN8R**, 2013-1-1 11:27:00) is a real ANPR data record.

We try to discover companion vehicles in accordance with traveling companion pattern proposed by Tang [7]. However, the definition is given based on GPS data. In [7], based on GPS data, a traveling companion pattern is defined as a clustering of moving objects traveling together within a duration no shorter than pre-specified value. Specifically, the authors learn from the classic density-based clustering [13] to group the moving objects in each snapshot.

However, ANPR data is very different. Firstly, it is essentially discrete on location. Each ANPR data record is generated from a camera fixed on the road. The involved location information is the fixed camera location. Thus, all ANPR data records are distributed at

cameras on roads. Secondly, the generation frequency of ANPR data is not constant. It depends on frequency with which vehicles pass through a camera. And each camera generates one data record per second at most. Hence, existing methods developed such like density-based clustering are not suitable for ANPR data.

The concepts of Moment Companion and Traveling Companion have been defined in our previous works. Their definitions are shown as follows:

Definition 1 (moment companion). Let Δt be the time threshold, a moment companion for camera c at time t_2 can be defined as: $MC(c, t_2, \Delta t) = (V, c, t_1, t_2)$, where $V = \{r.v \mid t_1 \le r.t \le t_2 \wedge t_2\text{-}t_1 \le \Delta t\}$, $r = (c, v, t)$ is an ANPR data record, and $|MC(c, t_2, \Delta t)| > 1$.

Example. In Figure 1, given vehicle v_1, $\{v_2, v_4, v_5\}$, $\{v_2, v_5, v_6\}$ and $\{v_2, v_5, v_7\}$ are all moment companions when v_1 passing through camera CAM07011112, CAM89411111 and CAM09712112 successively.

Definition 2 (traveling companion). Given an ANPR data stream $S = \{r_1, r_2, \ldots, r_i, \ldots\}$ and three thresholds time threshold Δt, vehicle number threshold δ_{veh}, moment companion ratio threshold δ_{mc} ($0<\delta_{mc}<1$), a traveling companion is a sequence of moment companions: $<MC_k, MC_{k-1}, \ldots, MC_1>$, in which:

(1) $k \ge \delta_{mc} * N$, where k is the length of the traveling companion and N is the current length of S;

(2) $|MC_i.V| \ge \delta_{veh}$, where $i = 1, 2, \ldots, k$;

(3) $\forall\, i, j = 1, 2, \ldots, k$, $MC_i.V = MC_j.V$;

(4) $\forall\, i = 1, 2, \ldots, k\text{-}1$, $MC_{i+1}.t_1 > MC_i.t_2 + \Delta t$.

Example. In Figure 1, $TC_1 = <MC_2, MC_1>$ and $TC_2 = <MC_3, MC_2, MC_1>$ are both traveling companions, where MC_1(CAM07011112, t_1, Δt) = (V, CAM07011112, t_1), MC_2(CAM89411111, t_2, Δt) = (V, CAM89411111, t_2), and MC_3(CAM09712112, t_3, Δt) = (V, CAM09712112, t_3), and each moment companion has $V = \{v_1, v_2, v_5\}$.

3 Rationales of Our Algorithm

3.1 Running Example

Table 1 shows a simple example of an ANPR dataset. We divide this table into two parts. The first part is from column t_{11} to t_1. It contains historical data records and stores them in a window. The column t_{12} are data records newly arrived. We establish a sequence for each vehicle by its passing cameras and corresponding timestamp. In order to give a clear exposition of our algorithm, all examples below in this paper are based on the data scenario in Table 1 under Δt = 10s.

Table 1. A sample of an inverse sequence database on ANPR dataset (running example)

	t_{12}= 9:48:39	t_{11}= 9:48:38	t_{10}= 9:48:31	t_9= 9:45:29	t_8= 9:45:28	t_7= 9:45:26	t_6= 9:40:28	t_5= 9:40:23	t_4= 9:40:20	t_3= 9:35:06	t_2= 9:35:03	t_1= 9:35:00
v_1		c_4			c_3			c_2			c_1	
v_2	c_4			c_3			c_2			c_1		
v_3						c_3			c_2			c_1
v_4			c_6									
v_5		c_6			c_4							
v_6	c_6			c_4			c_3			c_2		
v_7		c_7				c_5		c_4			c_3	
v_8			c_7	c_5					c_4	c_3		
v_9	c_7				c_5			c_3				

3.2 Rationales

As our previous work shows, we borrow ideas of frequent sequence to solve our problem. For a given time threshold Δt, a traveling companion represents a group of vehicles (V) passes each camera c in time interval $[t\text{-}\Delta t, t]$. It means the series of passed cameras is contained by the sequence of each vehicle in V in the sequence set like Table 1 shows. Hence, the series of passed cameras is a frequent sequence in the sequence set; V is a set of sequence id, each of which corresponding to a sequence containing the series of passed cameras. Therefore, we can discover traveling companions by mining frequent sequences, each element of which occurs in the sequence set within time period Δt.

Our algorithm adopts sliding window mechanism to receive input ANPR data stream. To guarantee error not to exceed a user specified parameter $\in$, the sliding window contains $\lceil 1/\in \rceil$ data records and its following Δt data records. We generate a new moment companion MC for each new arrival data record. And MC is compared with the IST$^+$ by a reference list Vlist, which is designed for facilitating tree traversal. If the vehicle set involved in the moment companion is contained completely by a node on IST$^+$, we increase its frequency by one and insert a node storing the moment companion into IST$^+$. If the vehicle set is partly contained by a node, we split the node and its descendants, and increase the frequency of each split node. Then, we also insert a node. Additionally, if there is no node containing the vehicle set completely, we create a new node for the new moment companion and set the frequency to be 1. We also prune the IST$^+$ by deleting nodes at window boundaries. Figure 2 shows the framework of our algorithm.

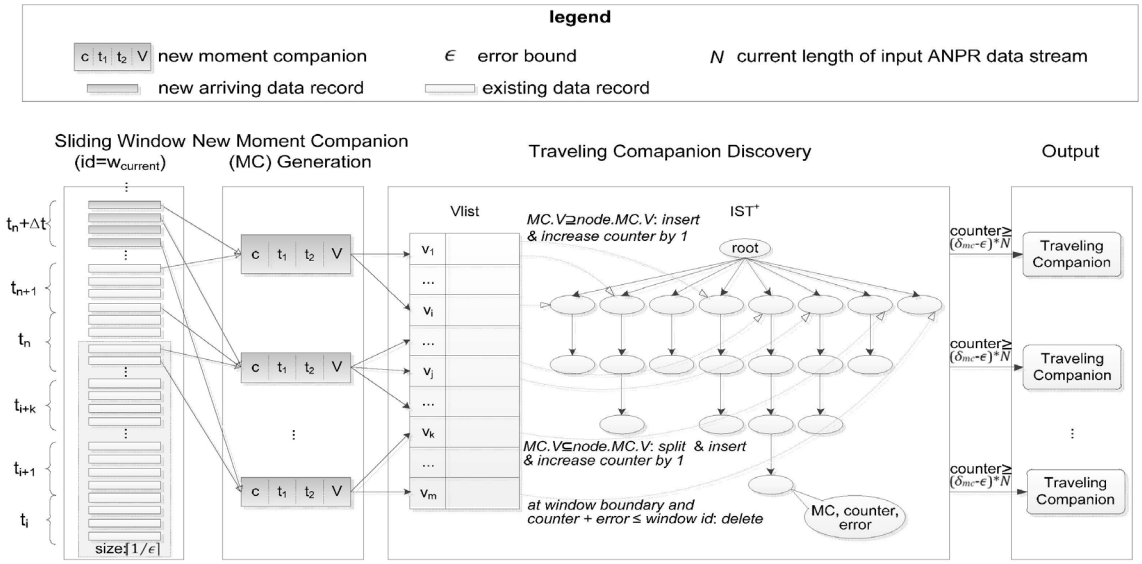

Figure 2. The framework of our algorithm

4 Traveling Companion Discovery on ANPR Data Stream

4.1 IST$^+$ Tree

To record discovered traveling companions, our previous work proposed a new data structure, which is called IST$^+$. It is a variation of the IST (Inverse Closed Sequence Tree) data structure which is initially proposed in SeqStream [14]. An IST is used to keep closed frequent sequential patterns in the sequence database of current sliding window. A node n (containing a data item and its frequency) of an IST corresponds to a sequence that starts from the root node to node n, and the sequence is denoted by s_n. The root node of an IST is a NULL node, which represents an empty sequence ϕ. Except for the root node, nodes of an IST can be divided into following three types.

- c-node (closed node): If s_n is a closed sequential pattern in D', n is a c-node.
- t-node (termination node): n is a t-node if 1) there exists a frequent sequence β in D' such that $\beta \sqsupset s_n$ and D'_β; 2) it does not have any t-node ancestor.
- i-node (intermediate node): If s_n is frequent, n is neither a c-node nor a t-node, and n has no t-node ancestor, then n is an i-node.

Based on IST, we design IST$^+$ to represent traveling companions by taking the temporal constraint into account. Like IST, IST$^+$ is a rooted prefix tree as well. Based on IST and our previous work, we give the formal definition of IST$^+$.

Definition 3 (IST$^+$, improved inverse closed sequence tree). IST$^+$ is a tree structure defined below.

(1) It consists of one root labeled as "null" and a set of item prefix subtrees as the children of root.

(2) Each node n in the item prefix subtree consists of four fields: *item* (*n.MC*, a moment companion), *counter* (*n.counter*, an integer), *error* (*n.error*$_{max}$, an integer) and *node type* (*n.type*), where *item* registers which moment companion this node represents; *counter* records the frequency of *n.V* since it is added into the tree; *error* determines the maximum possible error in *n.counter*; *node type* flags whether the sequence from root to this node is closed or not.

The node type (*n.type*) of an IST$^+$ node is in accordance with IST tree. A difference between IST$^+$ and IST tree is that IST$^+$ tree doesn't keep any live t-nodes. If a c-node becomes a t-node, it will be abandoned immediately.

The performance bottleneck of our previous algorithm is the traversal cost on the IST$^+$ when a new moment companion is generated. To solve this problem, we design a reference list of nodes on IST$^+$ to facilitate tree traversal based on the following theorem proposed in our previous work.

Theorem 1. Given a new data record r arriving at time t, database D_r^+ contains all newly emerging traveling companions related to the arrival of r, where $D_r^+ = \{r' \mid \exists c, \Delta t, s.t., r.v, r'.v \in MC(c, t, \Delta t).V\}$.

According to the theorem, we maintain a separate hashmap of references to the depth-1 nodes, which is called Vlist. Each element in Vlist corresponds to one vehicle. It keeps all depth-1 nodes on IST$^+$ which involves the vehicle. When a new moment companion occurs, our algorithm gets all depth-1 nodes containing any vehicle in the moment companion by Vlist. Figure 3 presents the Vlist following our running example.

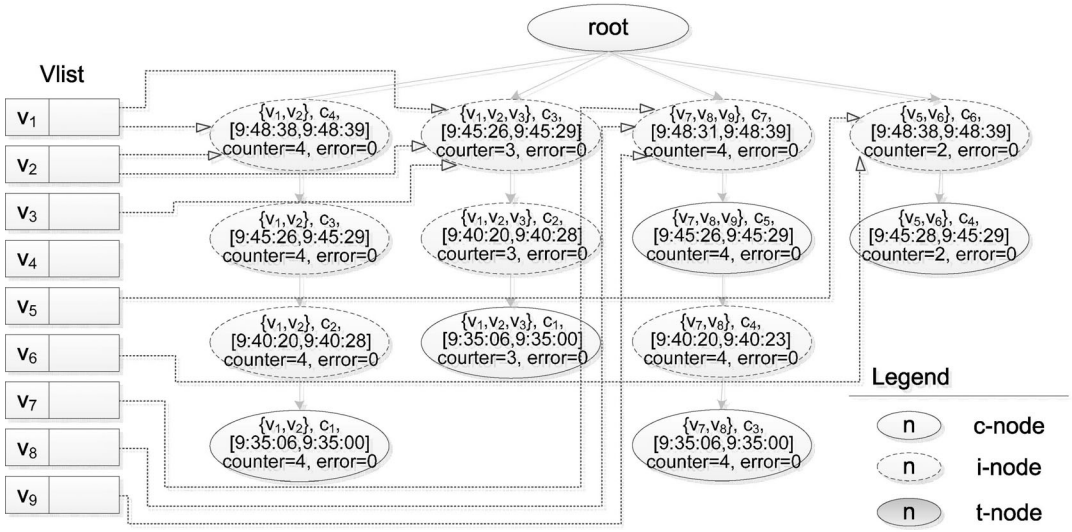

Figure 3. The vlist in our running example

For each new data record, our algorithm generates a new moment companion *MC* and compares *MC* with each depth-1 node on the IST$^+$ gained by Vlist. If a depth-1 node on the tree involves or partly involves the vehicle set in *MC*, we increase the counter of the involved vehicles by one. There are four cases of the comparison results.

Case 1: $MC.V = n.MC.V$. In this case, we increase the *counter* by one in node *n*. For each descendant node n_d of *n* satisfying $n_d.MC.V = MC.V$, we also increase its *counter* by one. At the same time, we insert a new node *n''* between root node and *n* such that $n.MC = MC$ and $n''.counter = n.counter$, $n''.error_{max} = n.error_{max}$.

Case 2: $MC.V \subset n.MC.V$. In this case, we split node *n'* from *n*, where $n'.MC = MC$. For each descendant node n_d of *n* satisfying $MC.V \subset n_d.MC.V$, split n'_d from n_d, where $n'_d.MC.V = MC.V$. Then we increase the *counter* by one in each split node and insert a node *n''* with $n''.MC = MC$ between the root node and node *n'*.

Case 3: $MC.V \supset n.MC.V$. In this case, we generate a moment companion *MC'* from *MC*, where $MC'.V = n.MC.V$. Herein, we process this *MC'* in the way of case 1.

Case 4: $|MC.V \cap n.MC.V| \geq \delta_{veh}$, $|MC.V - n.MC.V| > 0$ and $|n.MC.V - MC.V| > 0$. In this case, we generate a moment companion *MC'* from *MC*, where $MC'.V = MC.V \cap n.MC.V$. Herein, we process this *MC'* in the way of case 2.

After whole traversal, if there is no depth-1 node involving the vehicle set of *MC* completely, we need create a new node *m* with $m.MC = MC$, $m.counter = 1$ and $m.error_{max} = w_{current} - 1$.

Furthermore, a depth-1 node may not need to be compared with every moment companion generated at same timestamp. The following theorem elaborates the foundation.

Lemma 1. Any two newly emerging moment companions MC_i and MC_j at same timestamp satisfy that $MC_i.V \cap MC_j.V = \phi$.

Theorem 2. Given a new arriving data record *r*, *MC* is the emerging moment companion related to data record *r*. If $|n.MC.V - MC.V| < \delta_{veh}$, node *n* needs no comparison with other newly emerging moment companions, where $n.MC.V - MC.V = \{v | v \in n.MC.V \wedge v \notin MC.V\}$.

Proof. We prove this theorem in two cases.

Firstly, $|n.MC.V - MC.V| = 0$. In this case, we get $n.MC.V \subseteq MC.V$. According to lemma 1, there is no any other new moment companion containing $n.MC.V$. Thus, depth-1 node *n* needs no comparison with other new moment companions.

Secondly, $0 < |n.MC.V - MC.V| < \delta_{veh}$. In this case, we get $MC.V \cap n.MC.V \neq \phi$. According to lemma 1, there is no any other new moment companion containing $n.MC.V$. And there either will be no any other new moment companion *MC'* satisfying that $MC'.V \supseteq n.MC.V \wedge |MC'.V \cap n.MC.V| \geq \delta_{veh}$, because $MC'.V \cap n.MC.V \subseteq n.MC.V - MC.V$. Hence, depth-1 node *n* needs no comparison with other new moment companions in this case either. Proof is over.

After traversing the depth-1 nodes on IST$^+$ in the above cases, we need to check consistency on the processing results. It is because that a new moment companion may have same overlapped vehicle sets with different depth-1 nodes. Under this situation, more than one subtrees will be generated which keeps same vehicles. These subtrees save repetitive information. We merge them to keep the maximal one on the tree.

We prune IST$^+$ at each sliding window boundary. The strategy of deletion is as follows. For each node *n* on IST$^+$, if $n.counter + n.error_{max} \leq w_{current}$, node *n* needs to be deleted. In this paper, we adopt a lazy deletion strategy. It means we only prune the nodes which are affected by the new moment companions as

well as their descendants.

4.2 Sliding Window

Our algorithm is improved to compute traveling companion over ANPR data stream in one pass, satisfying approximation guarantee. Thus, the user has to specify error parameter $\in e(0,1)$ such that $\in \ll \delta_{mc}$. Incoming ANPR data stream is received by the sliding window presented below. Denote N as the current length of incoming ANPR data stream. Let $w_1 = \lceil 1/\in \rceil$, and the timestamp of the $num*w_1$th (num = 0, 1, …) data record as t. Each sliding window contains w_1 data records and data records in the following time scope $[t, t+\Delta t]$. Denote the window size as w. And set the sliding distance to be w_1. Each sliding window is labeled with a window id, which starts from 1. The current window id is denoted as $w_{current}$ such that $w_{current} = \lceil N/\lceil 1/\in \rceil \rceil$. The overlapped portion between two adjacent windows in this paper is much smaller than that in our previous work, since the sliding distance is set to be 1 second in our previous work. It indicates that our new sliding window model saves space by avoiding storing repetitive data. Figure 4 illustrates the sliding window model used in this paper.

Finally, the improved algorithm outputs the traveling companions with no less than $(\delta_{mc}-\in)*N$ moment companions. In fact, *counter* in a node refers to the frequency of vehicle set *n.MC.V* since it is added into the tree. It is an approximate value. We denote its true frequency in the whole ANPR data stream seen so far by f_{true}. We guarantee the accuracy of our algorithm according to a classic frequent item mining algorithm on stream, which is called Lossy Counting [12].

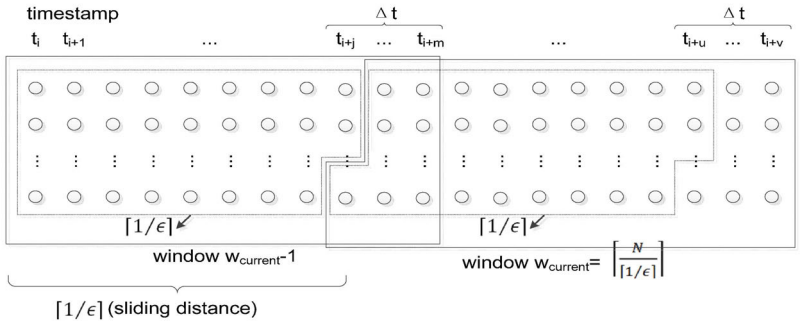

Figure 4. Sliding window model (N is the current length of incoming ANPR data stream)

Theorem 3. For any node n on IST$^+$, we have that $n.counter \le f_{true} \le n.counter + \in N$.

Proof. Firstly, according to the sliding window model, whenever deletions occur, $w_{current} \le \in N$.

Secondly, whenever a node n gets deleted, $f_{true} \le w_{current}$. We prove it by induction. Base case: $w_{current} = 1$. A node will be deleted only if $n.counter = 1$. In this case, $f_{true} = 1$. Obviously, $f_{true} \le w_{current}$. Induction step: Assume that $f_{true} \le w_{current}$ holds under $w_{current} = k$-1. Considering a node m which is deleted under $w_{current} = k$, the vehicle set $m.MC.V$ is added into the IST$^+$ in the window with id $m.error_{max}$ +1. Some nodes for the vehicle set may be deleted at the boundary of window $m.error_{max}$ (i.e., window $m.error_{max}$ gets full). By the induction, when this deletion occurred, the true frequency of the vehicle set is no more than $m.error_{max}$. Furthermore, $m.counter$ is the true frequency of the vehicle set since it was added into the tree. Therefore, the true frequency of this vehicle set on the whole input stream is at most $m.counter + m.error_{max}$. Combined with the deletion strategy $counter + error_{max} \le w_{current}$, we get that $f_{true} \le w_{current}$.

Thirdly, if a vehicle set does not appear in the IST$^+$, then $f_{true} \le \in N$.

Based on the above, for any node n on IST$^+$, we have that $n.counter \le f_{true} \le n.counter + \in N$.

If $n.error_{max} = 0$, then $n.counter = f_{true}$. Otherwise, the vehicle set may be deleted in the first $n.error_{max}$ windows. Because the true frequency of the vehicle set when the last deletion happened, is at most $n.error_{max}$. Therefore, $f_{true} \le m.counter + m.error_{max}$. Since $m.error_{max} \le w_{current} - 1 \le \in N$, we conclude that $n.counter \le f_{true} \le n.counter + \in N$. Proof is over.

The above theorem indicates that our algorithm will output no false negatives. The vehicle set in each traveling companion will have true frequency at least $(\delta_{mc}-\in)*N$ in whole ANPR data stream.

5 Experiment

5.1 Experiment Setup

Parameters. We do experiments to measure the effects and efficiency of our algorithm. Our algorithm involves several key parameters Δt, δ_{veh}, δ_{ms} and $\in$. They should be preassigned before running the algorithm. Generally, δ_{veh} is set to 2 as a companion should contain at least two vehicles. And $\in$ is set to be ten percent of δ_{ms}. The values of the rest two parameters depend on our experiences from our previous work. Table 2 lists the values of each parameter selected in our experiments.

Table 2. Parameter settings

Parameter	Explanation	Value
Δt	Temporal Constraint on Moment Companion;	Δt = 20s, 40s, …, 180s;
δ_{veh}	Vehicle Number Threshold on Traveling Companion;	δ_{veh} = 2;
δ_{mc}	Moment Companion Ratio Threshold on Traveling Companion	δ_{mc} = 0.002‰, 0.003‰,…, 0.01‰;

Baselines. To verify the effects of our algorithm, we compare our algorithm with three state of the art algorithms on discovering *flock*/*convoy* pattern [1-5], *swarm* pattern [6] and *traveling companion* pattern [7]. Authors proposed two algorithms to discover traveling companions, which generate same results according to their experiments [7]. We select the first one in our experiment. On the other hand, we compare our algorithm with our previous algorithm in [11] to verify the improvement on performance.

Datasets. The following experiments use a real ANPR dataset in Beijing, China. The dataset contains vehicle information from 2012-11-02 00:00:00 to 2012-11-08 23:59:59. Totally 1040 cameras and 30,518,191 ANPR data records are involved. We have a camera location dataset which records the latitude and longitude of each camera involved in our ANPR dataset. Besides, to verify the ability of handling high speed data stream, we simulate five ANPR datasets with data arrival of 1000, 2000, …, 5000 records per second based on the real dataset. We simulate each dataset as a stream. The time interval between two adjacent data records is in accordance with real intervals when they were shot by cameras.

Environments. The experiments are done on a PC with four Intel Core i5-2400 CPUs 3.10G Hz and 4.00 GB RAM. The operating system is Windows 7 Ultimate. All the algorithms are implemented in Java with JDK 1.8.0.

5.2 Effectiveness

In this part, we firstly evaluate the effects of our algorithm. To evaluate the effects of our algorithm, we run the baseline methods on 7 days of ANPR datasets integrated with the camera location dataset. According to baseline methods, we set the distance threshold to be 300 meters. On the other hand, we input the unchanged ANPR datasets into CompanionMiner algorithm with Δt = 60s, δ_{ms} = 0.002‰, and δ_{veh} = 2. Experimental results are shown in Figure 5.

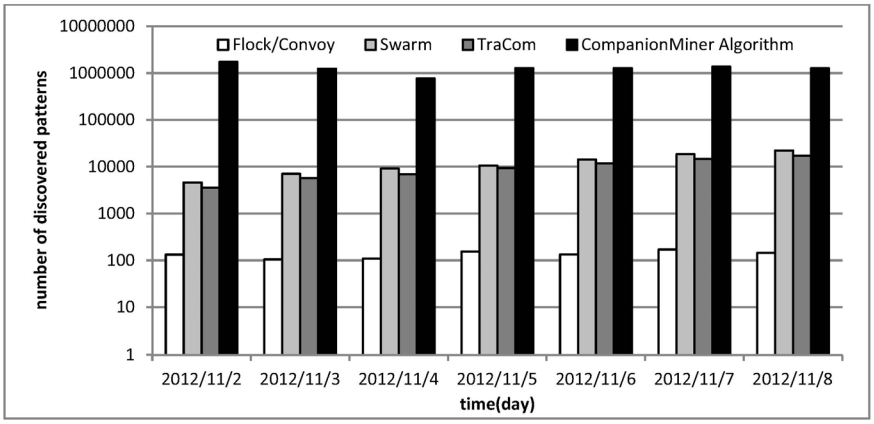

Figure 5. Comparison of effects among different methods

As Figure 5 shows, *flock*/*convoy* algorithm discovered the least patterns; *Swarm* algorithm find a little more results than *TraCom* algorithm; our algorithm found the most results. We firstly analyze the results to conclude that each pattern discovered by any baseline method is contained by our algorithm. There can be more than one camera at each road crossing to monitor vehicles from different directions. Cameras at same road crossing locate closely to be grouped together by baseline methods. Thus, baseline methods can detect vehicles that pass different cameras at same road crossing at the same time frequently. That is the reason that each baseline method can detect patterns over ANPR dataset. However, the detected patterns are equal to the patterns with Δt = 0s under our algorithm, which reduce the number of companion vehicles significantly. Secondly, although the numbers of detected patterns by *swarm* algorithm is more than that of *TraCom* algorithm, each pattern reported by *swarm* algorithm is contained by the results of *TraCom*. The reason is that *swarm* algorithm adopts distance based clustering method and *TraCom* algorithm utilizes density connected clustering method. The former method will depart two cameras if their distance is larger than pre-specified value. But the latter one may group them together if there exists another camera *c* and the distance between any camera and the camera *c* is less than pre-specified value. On top of this difference, a group of companion vehicles discovered by *TraCom* may be divided by *swarm*

algorithm.

5.3 Efficiency

We further verify our algorithm's performance in two ways in this section. We give the following definition.

Definition 4 (average latency). Let t_i is the time our algorithm consumes to discover traveling companions for the ith input data record. Assume that N is the current length of the ANPR data stream, the average latency of our algorithm can be defined as $\bar{t}_{lat} = \sum_i t_i / N$.

Firstly, we compare the performance of our algorithm with our previous work [11] under different values of Δt through the following experiments. Actually, we also do experiments to verify the performance under different δ_{ms} values. However, the experiment results show that δ_{ms} doesn't have obvious impacts on performance. We run our algorithm 20 times for different Δt with δ_{ms} = 0.001‰. Each execution lasts for 2 hours from 8:00:00 to 10:00:00 on each day to continuously receive arriving ANPR data records and instantly output companion results under different values of Δt. For each execution, we will compute the latency value based on Definition 4. Finally, the average latency values are shown in Figure 6.

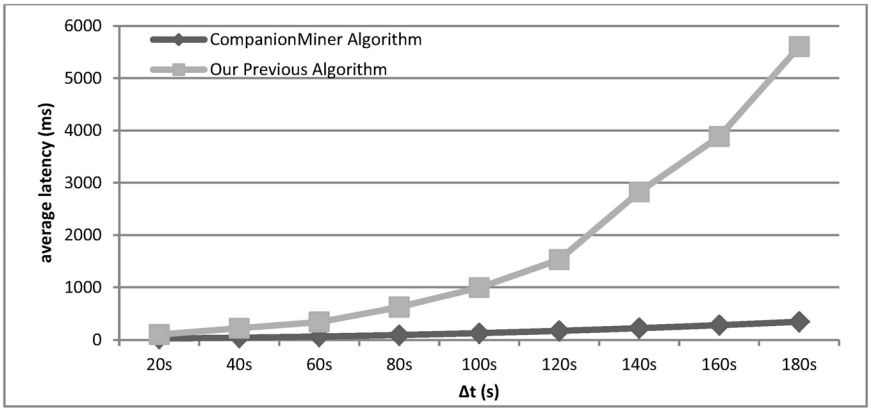

Figure 6. Average latency under different Δt between CompanionMiner and our previous algorithm (δ_{mc} = 0.001‰)

Secondly, we compare the efficiency of CompanionMiner algorithm with our previous work in [11] under different data arrival rate. We simulate five ANPR datasets with data arrival rate of 1000, 2000, …, 5000 records per second. Each dataset spans two hours. We run CompanionMiner algorithm and our previous algorithm on each simulated dataset with Δt = 60s, δ_{ms} = 0.001‰ and compute the average latency. The experiment results are presented in Figure 7.

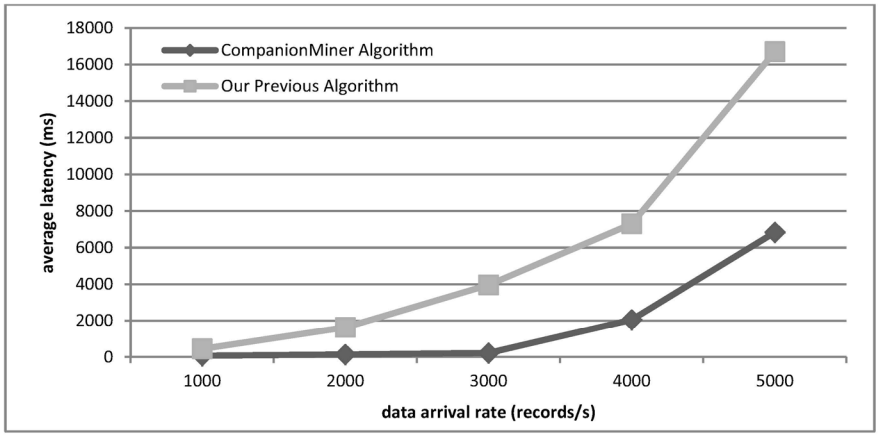

Figure 7. Average latency under data arrival rate between CompanionMiner and our previous algorithm

Figure 6 shows the average latency of our previous work increases exponentially from Δt = 20s to 180s. When $\Delta t \leq$ 100s, the average latency is less than 1000 ms. The minimum value of time interval between two ANPR data records is 1 second. It means our previous algorithm can instantly discover companions with $\Delta t \leq$ 100s when vehicles passing through camera. However, when $\Delta t >$ 100s, the latency increases sharply. On the other hand, CompanionMiner algorithm in this paper shows stable average latency with the growth of Δt. When Δt reaches 180s, the average latency of our algorithm in this paper is around 348.68 ms, which is

still lower than 1 second. On the other hand, as Figure 7 shows, the latency of each algorithm increases exponentially. However, the previous algorithm has a greater curve than the improved one.

Based on the experiments in our previous work, large Δt will generate more traveling companions. Therefore, the experiment results verify that, CompanionMiner algorithm improves the performance on handling data stream with larger traveling companion patterns and higher data arrival rate. Specifically, the performance of CompanionMiner algorithm on the real dataset is improved more than 70% and less than 93%. The percentage increases with the growth of Δt. At the same time, the performance on simulated datasets is improved from 59% to 94%. The percentage firstly increases till data arrival rate reaches 3000 records per second and then falls down. It is because that our previous algorithm can only keep running for the 29 minutes and 26 minutes input data with data arrival of 4000 and 5000 respectively. The average latency under these two cases is less than the truth, which reduce the percentage.

6 Related Works

6.1 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*, *convoy*, *swarm*, *traveling companion*, *platoon* and so on. *flock* is a group of moving objects moving in a disc of a fixed size for k consecutive timestamps [1-3]. *convoy* is an extension of *flock*, where spatial clustering is based on density [4-5]. *swarm* is proposed to enable the discovery of interesting moving object clusters with relaxed temporal constraint [6]. *traveling companion* has same constraints with *swarm* [7]. If a group of vehicles is a *flock* pattern, it must be a *convey* pattern. Furthermore, it is a *swarm* pattern. Besides, a new type of patterns, *platoon*, is proposed to describe object clusters that stay together for time segments, each with some minimum consecutive duration of time [8]. Table 3 compares the above companion patterns with the one in this paper.

Table 3. Comparison of several companion vehicles discovery methods

Pattern	Temporal Constraint	Spatial Constraint	Dataset	Solution
flock [1]	consecutive	disc	GPS	matrix analysis
flock [2]	consecutive	disc	GPS	weighted directed graph
flock [3]	consecutive	disc	GPS	clustering and intersection
convoy [4]	consecutive	density reachable	GPS	trajectory similarity
convoy [5]	consecutive	density reachable	GPS	trajectory similarity
swarm [6]	not consecutive	density reachable	GPS	frequent item mining
traveling companion [7]	not consecutive	density reachable	GPS	clustering and intersection
platoon [8]	consecutive/non consecutive	density reachable	GPS	frequent item mining and frequent sequence mining
traveling companion (in this paper)	time threshold Δt	not consecutive cameras	ANPR	frequent sequence mining with temporal constraint

Some researchers focus on moving cluster discovery [15-18]. Their goal is to find clusters of objects with similar moving patterns or behaviors. Kalnis et al. proposed the first study to automatic extract *moving clusters* from large spatial datasets [15]. Li et al. clustered the moving objects by micro clustering [16]. Both current and near future positions of moving objects are considered during clustering. Kriegel et al. clustered the moving objects by fuzzy distance functions [17]. Jensen et al. discovered moving object clusters incrementally within a period of time [18].

More recently, researchers begin to pay attention to 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 [19]. Zhang et al. proposed a gathering retrieving algorithm to retrieve gathering pattern by searching a spatio temporal graph composed of the moving object clusters [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].

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 framework to incrementally discover travelling companion among streaming trajectories proposed by Tang et al. [7], Yu et al. studied on a density based clustering algorithm for trajectory data stream and tried to discover trajectory clusters in real time [22]. Dow et al. designed and implemented a moving context-aware and location-based paratransit system for providing services according to user's demands and expectations [23]. They captured and analysed user's moving activities by using the built-in accelerometer of a smart device to provide real-time services. All these related work

aimed at processing GPS data stream and provide some foundations for our study.

6.2 Frequent Sequence Mining

Mining frequent sequences from databases is one the classic topic in data mining and has 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 [24]. Typical work of the former category includes AprioriAll [25], AprioriSome [25], GSP [26], SPADE [27], SPAM [28] and so on. The shortage of Apriori based algorithms is to generate large scale of candidate subsequences. Typical work of projection based pattern growth algorithms includes FreeSpan [29], PrefixSpan [30], CloSpan [31], BIDE [32] 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 put their interests on mining frequent sequence with constraints. Pinto et al. defined the concept of *multi-dimensional sequential pattern*, and proposed an algorithm to discover them [33]. Different from traditional sequence pattern, this pattern contains several attributes as well as a sequence. Pei et al. summarized the constraints in frequent sequence mining, including item constraint, length constraint, super pattern constraint, aggregate constraint, regular expression constraint, duration constraint and gap constraint [34]. However neither of the constraints discussed the temporal constraint in this paper. 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 [35].

To improve the efficiency, some researchers parallelized these mining algorithms. Demiriz proposed a parallel sequence mining algorithm, webSPADE, to analyze the click streams found in site web logs [36]. Guralnik et al. studied a variety of distributed memory parallel algorithms which is able to minimize the overheads [37]. Ma et al. proposed a distributed memory parallel algorithm to mine closed frequent sequences [38]. 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 optimization techniques, including prefix projection, parallel formulation, and candidate pruning [39]. Yu et al. parallelized BIDE algorithm by MapReduce framework [40]. Kessl proposed an algorithm for mining frequent sequences by static load balancing based on probabilistic model [41].

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 [14, 42-46]. Chang et al. proposed SeqStream algorithm to mine closed frequent sequence in a sliding window for arriving data records [14]. The algorithm transformed original data sequence database into inverse sequence database to facilitate the removal of expired data. And it utilized a core data structure, called IST, to keep closed sequential patterns in the inverse sequence database of current sliding window. Besides it, IncSpan [42] is proposed to discover frequent sequences on data stream, by using semi frequent nodes in a prefix tree. By a tree structure, MILE [43] utilizes the knowledge of existing frequent sequences to avoid redundant data scanning and learns from the prior knowledge of the data distribution in data stream to enhance the efficiency. IncSPAM [44] utilizes a tree structure *PS-tree* in which the algorithm needs only one scan at each timestamp. CISpan [45] 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 [46] 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 filter out the non-closed prefixes.

The typical methods mentioned above lay foundation of our research. However, neither of them takes the temporal constraint in our paper into consideration. Hence, we learn from these typical methods and our previous work, and design an algorithm to mine closed frequent sequences with temporal constraint. We optimize the algorithm according to the features of input data stream. In the future, we plan to parallelize it to enhance the performance.

7 Conclusion

In this paper, we improved our previous work to further discover companion vehicles over a special kind of streaming sensor data in one pass, which is called ANPR data. Experimental results show that, over real ANPR data stream, our algorithm can achieve much lower latency. However, our algorithm still cannot handle high speed input data stream. When data arrival comes to 4000 records per second, the latency reaches 2 seconds, which is more than the minimum value of time interval between two ANPR data records (1 second). In the future, we plan to parallelize our algorithm to further enhance the performance.

Acknowledgements

Funding: This work was supported by National Natural Science Foundation of China (Grant No. 61672042), "Models and Methodology of Data Services Facilitating Dynamic Correlation of Big Stream Data;" The Program for Youth Backbone Individual, supported by Beijing Municipal Party

Committee Organization Department, "Research of Instant Fusion of Multi-Source and Large-scale Sensor Data;" Training Plan of Top Young Talent in North China University of Technology, "An Incremental Approach to Instant Discovery of Data Correlations among Multi-Source and Large-scale Sensor Data."

References

[1] P. Laube, S. Imfeld, Analyzing Relative Motion Within Groups of Trackable Moving Point Objects, *Proceedings of the 2nd International Conference on Geographic Information Science (GIScience)*, Boulder, CO, 2002, pp. 132-144.

[2] J. Gudmundsson, M. Van Kreveld, Computing Longest Duration Flocks in Trajectory Data, *Proceedings of the 14th Annual ACM International Symposium on Advances in Geographic Information Systems (ACM GIS)*, Arlington, VA, 2006, pp. 35-42.

[3] M. R. Vieira, P. Bakalov, V. J. Tsotras, On-Line Discovery of Flock Patterns in Spatio-Temporal Data, *Proceedings of the 17th ACM International Symposium on Advances in Geographic Information Systems (ACM SIGSPATIAL GIS)*, Seattle, Washington, 2009, pp. 286-295.

[4] H. Jeung, H. T. Shen, X. Zhou, Convoy Queries in Spatio-Temporal Databases, *Proceedings of the IEEE 24th International Conference on Data Engineering (ICDE)*, Cancun, Mexico, 2008, pp. 1457-1459.

[5] H. Jeung, M. L. Yiu, X. Zhou, C. S. Jensen, H. T. Shen, Discovery of Convoys in Trajectory Databases, *Proceedings of the VLDB Endowment*, Vol. 1, No. 1, pp. 1068-1080, August, 2008.

[6] Z. Li, B. Ding, J. Han, R. Kays, Swarm: Mining Relaxed Temporal Moving Object Clusters, *Proceedings of the VLDB Endowment*, Vol. 3, No. 1, pp. 723-734, September, 2010.

[7] L. A. Tang, Y. Zheng, J. Yuan, J. Han, A. Leung, C. C. Hung, W. C. Peng, On Discovery of Traveling Companions from Streaming Trajectories, *Proceedings of the IEEE 28th International Conference on Data Engineering (ICDE)*, Washington, DC, 2012, pp. 186-197.

[8] Y. Li, J. Bailey, L. Kulik, Efficient Mining of Platoon Patterns in Trajectory Databases, *Data and Knowledge Engineering*, Vol. 100, pp. 167-187, November, 2015.

[9] Y. Han, G. Wang, J. Yu, C. Liu, Z. Zhang, M. Zhu, A Service-Based Approach to Traffic Sensor Data Integration and Analysis to Support Community-Wide Green Commute in China, *IEEE Transactions on Intelligent Transportation Systems*, Vol. 17, No. 9, pp. 2648-2657, September, 2016.

[10] M. Zhu, C. Liu, J. Wang, X. Wang, Y. Han, A Service-friendly Approach to Discover Traveling Companions Based on ANPR Data Stream, *Proceedings of IEEE International Conference on Services Computing (SCC)*, San Francisco, CA, 2016, pp. 171-178.

[11] C. Liu, X. Wang, M. Zhu, Y. Han, Discovering Companion Vehicles from Live Streaming Traffic Data, *Proceedings of 18th Asia-Pacific Web Conference (APWeb)*, Suzhou, China, 2016, pp. 116-128.

[12] G. S. Manku, R. Motwani, Approximate Frequency Counts over Data Streams, *Proceedings of the VLDB Endowment*, Vol. 5, No. 12, pp. 1699-1699, August, 2012.

[13] M. Ester, H. P. Kriegel, J. Sander, X. Xu, A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise, *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD)*, Portland, OR, 1996, pp. 226-231.

[14] L. Chang, T. Wang, D. Yang, H. Luan, Seqstream: Mining Closed Sequential Patterns over Stream Sliding Windows, *Proceedings of 8th IEEE International Conference on Data Mining (ICDM)*, Pisa, Italy, 2008, pp. 83-92.

[15] P. Kalnis, N. Mamoulis, S. Bakiras, On Discovering Moving Clusters in Spatio-Temporal Data, *Proceedings of 9th International Symposium on Spatial and Temporal Databases (SSTD)*, Angra dos Reis, Brazil, 2005, pp. 364-381.

[16] Y. Li, J. Han, J. Yang, Clustering Moving Objects, *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, Seattle, Washington, 2004, pp. 617-622.

[17] H. P. Kriegel, M. Pfeifle, Density-Based Clustering of Uncertain Data, *Proceedings of 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, Chicago, IL, 2005, pp. 672-677.

[18] C. S. Jensen, D. Lin, B. C. Ooi, Continuous Clustering of Moving Objects, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 19, No. 9, pp. 1161-1174, September, 2007.

[19] K. Zheng, Y. Zheng, N. J. Yuan, S. Shang, X. Zhou, Online Discovery of Gathering Patterns from Trajectories, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 26, No. 8, pp. 1974-1988, August, 2014.

[20] J. Zhang, J. Li, S. Wang, Z. Liu, Q. Yuan, F. Yang, On Retrieving Moving Objects Gathering Patterns from Trajectory Data via Spatio-Temporal Graph, *Proceedings of the IEEE International Congress on Big Data (BigData Congress)*, Anchorage, AK, 2014, pp. 390-397.

[21] J. S. Yoo, D. Boulware, D. Kimmey, A Parallel Spatial Co-Location Mining Algorithm Based on Mapreduce, *Proceedings of the IEEE International Congress on Big Data (BigData Congress)*, Anchorage, AK, 2014, pp. 25-31.

[22] Y. Yu, Q. Wang, X. Wang, H. Wang, J. He, Online Clustering for Trajectory Data Stream of Moving Objects, *Computer Science and Information Systems*, Vol. 10, No. 3, pp. 1293-1317, June, 2013.

[23] C. R. Dow, P. Y. Lai, J. H. Ye, A Moving Context-aware and Location-based Paratransit System, *International Journal of Internet Protocol Technology*, Vol. 9, No. 1, pp. 34-43, January, 2015.

[24] C. H. Mooney, J. F. Roddick, Sequential Pattern Mining: Approaches and Algorithms, *ACM Computing Surveys*, Vol. 45, No. 2, pp. 94-111, February, 2013.

[25] R. Agrawal, R. Srikant, Mining Sequential Patterns, *Proceedings of the 1995 IEEE 11th International Conference on Data Engineering (ICDE)*, Taipei, Taiwan, 1995, pp. 3-14.

[26] R. Srikant, R. Agrawal, Mining Sequential Patterns: Generalizations and Performance Improvements, *Proceedings*

of 5th International Conference on Extending Data Base Technology (EDBT), Avignon, France, 1996, pp. 3-17.

[27] M. J. Zaki, SPADE: An Efficient Algorithm for Mining Frequent Sequences, *Machine Learning*, Vol. 42, No. 1/2, pp. 31-60, January, 2001.

[28] J. Ayres, J. Flannick, J. Gehrke, T. Yiu, Sequential Pattern Mining Using a Bitmap Representation, *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, Edmonton, Alberta, Canada, 2002, pp. 429-435.

[29] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, M. C. Hsu, Freespan: Frequent Pattern-Projected Sequential Pattern Mining, *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, Boston, MA, 2000, pp. 355-359.

[30] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, M. C. Hsu, Prefixspan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth, *Proceedings of the IEEE 17th International Conference on Data Engineering (ICDE)*, Heidelberg, Germany, 2001, pp. 215-224.

[31] X. Yan, J. Han, R. Afshar, Clospan: Mining Closed Sequential Patterns in Large Datasets, *Proceedings of the 3th SIAM International Conference on Data Mining (SDM)*, San Francisco, CA, 2003, pp. 166-177.

[32] J. Wang, J. Han, C. Li, Frequent Closed Sequence Mining without Candidate Maintenance, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 19, No. 8, pp. 1042-1056, August, 2007.

[33] H. Pinto, J. Han, J. Pei, K. Wang, Q. Chen, U. Dayal, Multi-Dimensional Sequential Pattern Mining, *Proceedings of 10th International Conference on Information and Knowledge Management (CIMK)*, Atlanta, GA, 2001, pp. 81-88.

[34] J. Pei, J. Han, W. Wang, Constraint-based Sequential Pattern Mining: The Pattern-growth Methods, *Journal of Intelligent Information Systems*, Vol. 28, No. 2, pp. 133-160, April, 2007.

[35] H. E. Chueh, Mining Target-Oriented Sequential Patterns with Time-Intervals, *International Journal of Computer Science & Information Technology*, Vol. 2, No. 4, pp. 113-123, August, 2010.

[36] A. Demiriz, webSPADE: A Parallel Sequence Mining Algorithm to Analyze Web Log Data, *Proceedings of IEEE International Conference on Data Mining (ICDM)*, Maebashi, Japan, 2002, pp. 755-758.

[37] V. Guralnik, G. Karypis, Parallel Tree-projection-based Sequence Mining Algorithms, *Parallel Computing*, Vol. 30, No. 4, pp. 443-472, April, 2004.

[38] C. Ma, Q. Li, Parallel Algorithm for Mining Frequent Closed Sequences, *Proceedings of International Workshop on Autonomous Intelligent Systems: Agents and Data Mining (AIS-ADM)*, Petersburg, Russia, 2005, pp. 184-192.

[39] S. Qiao, C. Tang, S. Dai, M. Zhu, J. Peng, H. Li, Y. Ku, Partspan: Parallel Sequence Mining of Trajectory Patterns, *Proceedings of 5th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, Jinan, Shandong, China, 2008, pp. 363-367.

[40] D. Yu, W. Wu, S. Zheng, Z. Zhu, BIDE-based Parallel Mining of Frequent Closed Sequences with Mapreduce, *Proceedings of 12th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP)*, Fukuoka, Japan, 2012, pp. 177-186.

[41] R. Kessl, Probabilistic Static Load-balancing of Parallel Mining of Frequent Sequences, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 28, No. 5, pp. 1299-1311, May, 2016.

[42] H. Cheng, X. Yan, J. Han, Incspan: Incremental Mining of Sequential Patterns in Large Database, *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, Seattle, Washington, 2004, pp. 527-532.

[43] G. Chen, X. Wu, X. Zhu, Sequential Pattern Mining in Multiple Streams, *Proceedings of the IEEE 5th International Conference on Data Mining (ICDM)*, Houston, TX, 2005, pp. 585-588.

[44] C. C. Ho, H. F. Li, F. F. Kuo, S. Y. Lee, Incremental Mining of Sequential Patterns over A Stream Sliding Window, *Workshops Proceedings of the IEEE 6th International Conference on Data Mining (ICDM - Workshops)*, Hong Kong, China, 2006, pp. 677-681.

[45] D. Yuan, K. Lee, H. Cheng, G. Krishna, Z. Li, X. Ma, Y. Zhou, J. Han, CISpan: Comprehensive Incremental Mining Algorithms of Closed Sequential Patterns for Multi-Versional Software Mining, *Proceedings of the 8th SIAM International Conference on Data Mining (SDM)*, Atlanta, GA, 2008, pp. 84-95.

[46] C. Gao, J. Wang, Q. Yang, Efficient Mining of Closed Sequential Patterns on Stream Sliding Window, *Proceedings of the IEEE 11th International Conference on Data Mining (ICDM)*, Vancouver, BC, Canada, 2011, pp. 1044-1049.

Biographies

Meiling Zhu is currently a Ph.D. candidate at the School of Compute Science and Technology of Tianjin University, China. Her research interests include Services Computing, Streaming Data Integration and Analysis.

Chen Liu received his Ph.D. degree in computer science and technology from the Chinese Academy of Sciences in 2007. Now, he is an Associate Professor at Research Center for Cloud Computing in North China University of Technology. His research interests include data integration, service modeling, service composition, and so on.

Jianwu Wang got his Ph.D. degree from Institute of Computing Technology, Chinese Academy of Sciences in 2007. He works at University of Maryland, U.S. His research interests include Big Data, Scientific Workflow, Distributed Computing, Service-Oriented Computing, End-User Programming. He has published 50 papers with more than 400 citations.

Yanbo Han holds a Ph.D. in computing science from the Technical University of Berlin in German. He is also the director of Beijing Key Laboratory on Integration and Analysis of Large-scale Streaming Data. His research interests include streaming data processing, cloud computing, dependable distributed systems, business process collaboration and management.