Mining Frequent Trajectory Patterns from GPS Tracks

Gang Chen
College of Computer Science and Technology, Zhejiang University
Hangzhou, China
zjucg@zju.edu.cn

Baoquan Chen
Shenzhen Institutes of Advanced Technology
Chinese Academy of Sciences, China
baoquan.chen@gmail.com

Yizhou Yu
University of Illinois at Urbana-Champaign and Zhejiang University
Hangzhou, China
yyz@illinois.edu

***Abstract*—As recent advances and wide usage of mobile devices with positioning capabilities, trajectory database that captures the historical movements of populations of moving objects becomes important. Given such a database that contains many *taxi* trajectories, we study a new problem of discovering *frequent sequential patterns*. The proposed method comprises two phases. First, we cluster the stay points of taxis to get *collocation patterns* for passengers. Then, for each pattern instance, we use an efficient graph-based searching algorithm to mine the frequent trajectory patterns, which utilizes the adjacency property to reduce the search space. The performance evaluation demonstrates that our method outperforms the Apriori-based and PrefixSpan-based methods.**

Keywords- sptiotemporal data; clustering; frequent trajectory patterns; graph-based searching

I. INTRODUCTION

With the recent advances and wide usage of location-detection devices (FRID, GPS, etc.), huge amounts of trajectory data are captured every day [1]. As a result, mining implicit and useful patterns from such databases has gained much interest in the past few years [2] [3] [4] [5] [6]. In many applications, the movements of objects obey frequently repeated patterns [7]. Finding the frequent trajectory can help us to analyze and predict the regular patterns of objects. For example, for traffic, sending passengers from place A to place B, many taxi drivers may choose almost the same series of streets because it may be a cost-effective way. In ecology, analyzing animals' regular routes can help us better understand their behavior regularities or detect abnormal phenomenon, especially if some animals' trajectory patterns change abruptly.

The discovery of sequential patterns from transactional database has attracted lots of interest since Agrawal et al. introduced the problem [10]. In this paper, we are interested in finding sequential patterns from GPS tracks, which are the spatiotemporal records of taxi trajectories. To achieve the mining goal, several challenges need to be addressed. First, the GPS trajectory database is not directly applicable for finding frequent repeated trajectories because of the fuzziness and redundancies of the data. Since we do not expect a taxi to visit *exactly* the same location in each frequent pattern instance, the patterns are not rigid but fuzziness exists, which allows one trajectory differ slightly from another in the pattern. This approximate nature of spatiotemporal patterns increases the complexity of mining tasks. Second, each trajectory is a long spatiotemporal sequence, without any predefined segmentation contributing to a pattern. Thus, we must use a particular abstract method to identify internal segmentation, without overlap between them. Third, in conventional frequent sequential mining algorithms, due to support counting and pattern extension, the transactional database needs to be scanned many times. Another challenge is how to exploit and utilize the adjacency property of spatiotemporal data to accelerate the mining process.

To summarize, the main contributions of this paper are: (i) We model a new frequent pattern mining problem from spatiotemporal data of taxi GPS tracks, based on the passenger collocation pattern extraction and approximate definitions of spatial regions. (ii) We use a clustering method to extract collocation patterns for passengers from the GPS tracks. (iii) We propose an efficient algorithm for discovering frequent trajectory patterns, which can localize support counting and exploit adjacency property to reduce the search space. The rest of the paper is organized as follows: Section 2 describes the preliminary concepts and the problem definitions. Section 3 presents the proposed algorithm in detail. Section 4 illustrates the experiments and performance evaluation of our method. Section 5 concludes this paper.

II. PRELIMINARIES AND PROBLEM DEFINITION

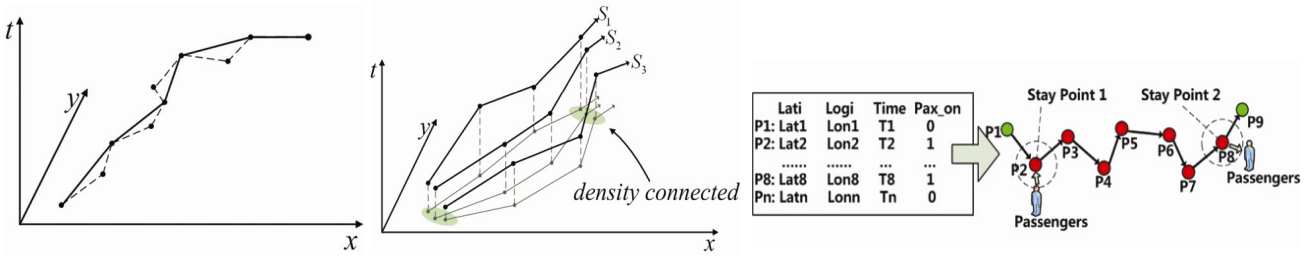
This section formalizes the proposed problem. We start with some preliminary concepts, including distance function and GPS sequence.

A. Preliminary Concepts

1. Distance Function for Line Segments. Given two line segments L_i and L_j, which can be represented by two vectors, $L_i = s_i e_i$ and $L_j = s_j e_j$, we apply the distance function used in [13] to measure the similarity between L_i and L_j. The distance function is composed of three components: (i) the *perpendicular distance* ($d_\perp$), (ii) the *parallel distance* ($d_\parallel$), (iii) *angle distance* (d_θ). The distance between L_i and L_j is defined Formula 1, in which $w_\perp$, $w_\parallel$ and w_θ are the weights of the three components, respectively.

$$dist(L_i, L_j) = w_\perp \cdot d_\perp(L_i, L_j) + w_\parallel \cdot d_\parallel(L_i, L_j) + w_\theta \cdot d_\theta(L_i, L_j) \quad (1)$$

978-1-4244-5392-4/10/$26.00 ©2010 IEEE



(a) A simplified trajectory (*solid line*) (b) An example of a collocation pattern (c) GPS sequence and stay points

Fig.1. Example of definitions

The three components are easily computed using vector operations. For more information about the definitions of the three distance components, refer to the studies described in [13].

2. GPS sequence and stay points. The data collected by the GPS devices equipped in mobile objects is of the GPS log form, which is a sequence of GPS points $P = \{p_1, p_2, ..., p_n\}$. Each point $p_i \in P$ is the form of (x, y, t), indicating that a moving object was or will be at the geographic location with coordinates (x, y) at time t. For each point (x, y, t), an associated attribute status shows whether any passenger is on the taxi at time t. Thus, the GPS subsequence of a taxi with passengers on can be easily got from the whole sequence. In our problem, we refer the first and last points of the subsequence as stay points, which indicate the locations where the passengers get on and off the taxi. An example of GPS log, GPS sequence and stay points is depicted in Fig.1(c).

B. Problem Definition

This section formally models the frequent pattern mining problem from GPS tracks, which includes the collocation pattern discovery for passengers, and the frequent pattern mining based on the collocation pattern.

Given a collection of trajectories $D = \{S_1, S_2, ..., S_n\}$, S_i is a movement history of a taxi. It is of interest to discover groups of passengers that travel with the similar purpose. In other words, we are interested to find the groups of passengers that both get on and get off the taxis on the similar sites. The identification of so-called **collocation patterns** may be used to discover abnormal passenger flow that indicates a special event takes place, or be used to represent intensity of the relationship between different business areas. And further, the discovery of frequent trajectory with a collocation pattern may be used for the scheduling of collective taxis because following these paths may reduce congestion and earn more money.

As a precursor to define the collocation patterns, we need to understand the notion of **density connection** [8]. Given a distance threshold e and a set of points P, the **e-neighborhood** of a point p is given as $NH_e(p) = \{q \in P \mid D(p,q) \le e\}$. Then, given a distance threshold e and an integer m, a point p is directly density reachable from a point q if $p \in NH_e(q)$ and $|NH_e(q)| \ge m$. A point p is said to be density reachable from a point q with respect to e and e if there exists a chain of points $p_1, p_2, ..., p_n$ in set P such that $p_1 = q$, $p_n = q$, and p_{i+1} is directly reachable from p_i .The concept of density connection permits us to use a **density-based clustering** method to capture a group of "connected" points with arbitrary shape and extent. We refer these groups as collocation patterns, as shown in Fig.1(b).

After the collocation pattern discovery, we can easily get a collection of transactional sequences that start and end at the similar places for each corresponding pattern instance. Given such a collection of sequences, we are interested to find the frequent path among them. First, with a trajectory $S_i = \{p_1, p_2, ..., p_n\}$, and a tolerance ε, we apply an abstract method based on line simplification algorithm [14] on it. Fig.1(a) visualizes the simplified trajectory. Notice that each point in a simplified trajectory corresponds to a point in the original trajectory and also the associated timestamp.

Given a GPS sequence S_i and its simplified version S_i', let s_{ij} be a **segment** in S_i and its simplified **representative line segment** in S_i' is $\vec{l}_{ij}$. We say that $\vec{l}_{ij}$ complies with s_{ij} if $dist(p_k, \vec{l}_{ij}) \le \varepsilon$ for $p_k \in s_{ij}$, which is confirmed by the principle of line simplification. Let $B(s_{ij})$ be the **minimum bounding box** of segment s_{ij}. The **central line segment** $\vec{l}_{ij}^{\,c}$ of $B(s_{ij})$ is considered best fit all the points in s_{ij}. Usually, $\vec{l}_{ij}^{\,c} \ne \vec{l}_{ij}$. $B(s_{ij})$ is termed as pattern elements and obviously contains $\vec{l}_{ij}$. In our problem setting, we call $B(s_{ij})$ as spatial regions. From this abstract method, each GPS sequence can be transformed into a list of ordered spatial regions, each region being a bounding box of a segment. Therefore, for GPS sequences, the **sequential pattern** P is an ordered list of spatial regions $r_1, r_2, ..., r_q$, q is the length of pattern P.

The continuous subsequence $s_i s_{i+1} ... s_{i+q-1}$ of is a pattern instance of $P: r_1, r_2, ..., r_q$, if the representative line segment $\vec{l}_{i+j-1}$ for segment s_{i+j-1} is similar to the central line segment $\vec{l}_j^{\,c}$ of region r_j, for each $1 \le j \le q$, that is the similarity between $\vec{l}_j^{\,c}$ and $\vec{l}_{i+j-1}$ is smaller than a user specified threshold η. From this definition, the **support** of P is the number of instances supporting P. Given a support threshold *min_sup*, P is frequent if its support exceeds *min_sup*. The mining problem discussed in this paper is to find frequent patterns from a collection of sequences, which are a pattern instance of collocation patterns, with respect to a simplifying distance tolerance ε, a similarity threshold η, and support threshold *min_sup*.

III. THE PROPOSED METHOD

In this section, we describe how to extract collocation patterns from a database of long taxi GPS sequences and how to discover frequent sequential patterns based the discovered collocation patterns.

A. Discovery Collocation Patterns for Passenger Behavior

Given a database of long taxi GPS trajectories, a simple and direct way for computing collocation patterns is to perform density-connected clustering on the stay points of trajectories and then to extract common trajectories to form collocations. Here, we apply shared nearest neighbor clustering (SNN) algorithm [9] which is an improved version of the density-based clustering method, DBSCAN, to discover collocation patterns.

Such trajectories from the database are long spatiotemporal sequences, for the first level modeling we extract transactional subsequences from the trajectories. For each subsequence, we can get two stay points, one of which is start point, and the other is end point. We then join the two stay point into a four dimensional vector and put it into V . Then, we apply SNN algorithm on V to obtain a set C of clusters. If the cardinality of a cluster is smaller than a threshold, remove it from C . The remaining clusters are referred as collocation patterns and to be used for further processing in the next iteration.

B. Discovering Frequent Singular Patterns

For each collocation pattern instance, our second level modeling is to discover frequent singular patterns from the transnational sequences.

Given a collection of subsequences contributing to a collocation pattern, the line simplification algorithm can be used to convert the original location sequences to line segments list. Our idea is to transform each subsequence S_i into $S_i^{'}$ using such a technique and use the segments as seed to absorb the similar segments to form a spatial region. In addition, in order to localize the support counting, for each spatial region we maintain a trajectory list (T-list), which stores the trajectories that contains the segments in the region. In this way, the fuzziness of spatiotemporal data can be overcome by spatial regions.

Discovering frequent singular patterns from a collection of sequences is a challenging problem, since we should compute the bounding box of each segment and compute the similarity between central line segments. In practice, we use an absorbing way based on the line segments obtained from line simplification. Before all, in order to improve the ratio between "true" and "false" line segments, we firstly remove the line segments that are shorter than a threshold.

Let *Lines* be a set initially containing all the line segments from the simplified sequences. Absorbing algorithm works as follows. Firstly, it prunes the shorter line segments from *Lines*. Then, it selects the line segment with median length as seed for the initial spatial region r . Simultaneously, we construct a trajectory list (T-list) for the spatial region. Then, absorbs other similar line segments from *Lines* through clustering process, described later, and put the trajectory ids into T-list where the trajectories contain the segments corresponding to these line segments. Next, for the line segments in *Lines* not absorbed by r , a new line segment with median length is selected as seed for a new absorbing iteration. Finally, the algorithm terminates after all line segments have been assigned to a region. Given a support threshold *min_sup* , it is easy to find frequent spatial regions through comparing the size of T-list and *min_sup* .

The clustering process computes similarity *sim* between l_i and l_j using the Formula 1. Given a similarity threshold η , l_i is similar to l_j if $sim(l_i, l_j) < \eta$. The overall algorithm is described in Algorithm 1.

Algorithm 1 Singular Pattern Discovery.

Require:
A collection of transactional sequences, S;
Similarity threshold, ε
Ensure:
A set of regions, $R = \{r_1, r_2, ..., r_{num_{clus}}\}$;
1: set *regionId* to be 0;
2: **for** each $l_i \in Lines$ with median length **do**
3: **if** $\vec{l_i}$ is unclassified **then**
4: create a new region $r_{regionId}$ and create a T_list for it;
5: compute $NH_\eta(\vec{l_i})$;
6: assign *regionId* to $\forall \vec{x} \in NH_\eta(\vec{l_i})$;
7: insert $NH_\eta(\vec{l_i}) - \{\vec{l_i}\}$ into a queue Q;
8: **ExpandRegion**(Q, *regionId*, η);
9: **end if**
10: **end for**

11: **ExpandRegion**(Q, *regionId*, η) {
12: **while** $Q \neq \emptyset$ **do**
13: let $\vec{l}$ be the first line segment in Q;
14: compute $NH_\varepsilon(\vec{l})$;
15: **for** each $\vec{x} \in NH_\eta(\vec{l})$ **do**
16: **if** $\vec{x}$ is unclassified **then**
17: assign *regionId* to $\vec{x}$;
18: insert $\vec{x}$ into the queue Q;
19: **end if**
20: **end for**
21: remove $\vec{l}$ from the queue Q;
22: **end while**
23: }

C. Expanding Longer Patterns

After the frequent singular patterns are discovered, each original sequence can be converted into an ordered region series by changing the segments in frequent regions to region ids. The region series demonstrate how the objects move among regions.

A direct and simple way is to perform an Aprioir-based algorithm to mine in a level by level way. However, as demonstrated in many literatures, this approach suffers from the bottleneck that we must scan the original region series many times to generate candidate patterns and count their supports globally. Here, we consider the spatial property of spatiotemporal data. From this nature, a property so-called **connectivity property** exists in the region series, described as follows. Based on this property, we propose an efficient solution to reduce the number of candidate patterns and localize support counting.

(Connectivity Property) *Due to continuity of GPS movement trajectories, a spatial region can only connect to its directly neighborhood regions in the region series.*

This property can help reduce the number of candidates since the patterns are expanding in a local space and many redundant candidates are eliminated. Therefore, we first extract connectivity information from the region series. Then, we construct a connectivity mapping graph (CMG) from the information. The way we extract connectivity information is using frequent 2-patterns (L_2) and employing L_2 to construct CMG. Afterwards, through traversing the CMG in a depth searching manner, we can find all frequent sequential patterns from the region series, since the relation information about regions are all stored in the graph. Different from the Aprioir-like algorithms, our CGS (connectivity mapping graph searching algorithm) algorithm can out of order find all the long frequent sequential patterns, which means that we can find all frequent k-patterns not directly through frequent (k-1)-patterns. This mining feature leads to better performance than the Apriori-like algorithms.

The construction process of CMG is shown as follows. After getting the frequent singular patterns L_1, we join L_1 with itself to produce candidate 2-patterns and then use the information stored in T_lists to calculate their supports to get frequent 2-patterns L_2. For each pattern instance in L_2, say $r_i r_j$, a directed edge from r_i to r_j is added to CMG and the edge weight is the support count of $r_i r_j$.

After the CMG construction finished, the CGS algorithm traverses the CMG in a depth-first search manner to mine all frequent patterns. Based on CMG and connectivity property, the search process can be easily implemented. Let $r_1 r_2 ... r_k$ be a frequent pattern being mined and assume r_k connects to r_i and r_j. So we only get two candidates $r_1 r_2 ... r_k r_i$ and $r_1 r_2 ... r_k r_j$. If the weight from r_k to r_j or is not larger than min_sup, we need not generate the corresponding candidate. And also, having the T_lists, we can localize the support count, which avoid the global database scanning. The algorithm is shown in Algorithm 2 and contains a sub-procedure so-called traverse. Step 1 in algorithm 2 is to start the **traverse** procedure for each frequent singular pattern instance in L_1, and Step 2 is the traverse procedure in which it starts traversing CMG from the last element of *currentFP* and expends it with its neighbors. The procedure is recursively repeated until no more neighbors can be concatenated.

Algorithm 2 Connectivity Mapping Graph Search Algorithm.

Require:
Connectivity mapping graph, G;
Minimum support threshold, min_sup;
Ensure:
All frequent patterns, $allFP$;
/*Step 1*/
set $allFP$ and $currentFP$ to $\emptyset$;
set T_list to $\emptyset$;
for each element $v \in L_1$ **do**
  $currentFP.add(v)$;
  $T\_list = v.T\_list$;
  traverse($currentFP, T\_list, min\_sup, allFP$);
end for
/*Step 2*/
traverse($currentFP, T\_list, min\_sup, allFP$) {
let s be the last element in $currentFP$;
compute direct neighbors N of s in G;
for each neighbor v in N **do**
  if $|T\_list \cap v.T\_list| \geq min\_sup$ **then**
    $currentFP.add(v)$;
    traverse($currentFP, T\_list, min\_sup, allFP$);
  else if $|currentFP| > 1$ **then**
    $allFP.add(currentFP)$;
    $currentFP.clear()$;
  end if
end for
}

IV. EXPERIMENTS AND EVALUATION

In this section, we evaluate our proposed approach with real and synthetic data. We first discuss the data source and data formation. Then, we study the effectiveness and efficiency using these data.

A. Data Source and Data Formation

Real data: The real data is the GPS tracks collected from taxis with GPS equipment, where the taxis are from Shenzhen city in the south of China. Each GPS sequence is the movement location history of a taxi in a day. Every GPS point in the data reports the location, time, and operation status (no passengers or with passengers), where the location is represented by latitude and longitude. The locations in each sequence are sampled every 20 seconds and the record time span is between 15 hours (9:00AM to midnight) and 24 hours, so the sequence length is very long. However, since the taxi stops and device fault exits, the GPS data is redundant and noisy.

Synthetic data: In order to facilitate the evaluation of the performance, we generated a database consisting of long sequences. First, we construct a squared grid space, where each vertex represents a location. To generate a potential frequent pattern, we choose the lower left point in the space as the starting point. To extend the pattern and ensure the general direction of the pattern, we randomly choose a point from the three points in the upper right corner that are adjacent to the last point of the pattern. This process continues until the pattern reaches the upper edge or right edge of the space. Then, the pattern directly extends until it reaches the upper right point. Finally, we use the potential frequent patterns to generate a synthetic database, where each pattern has the same start point and end point.

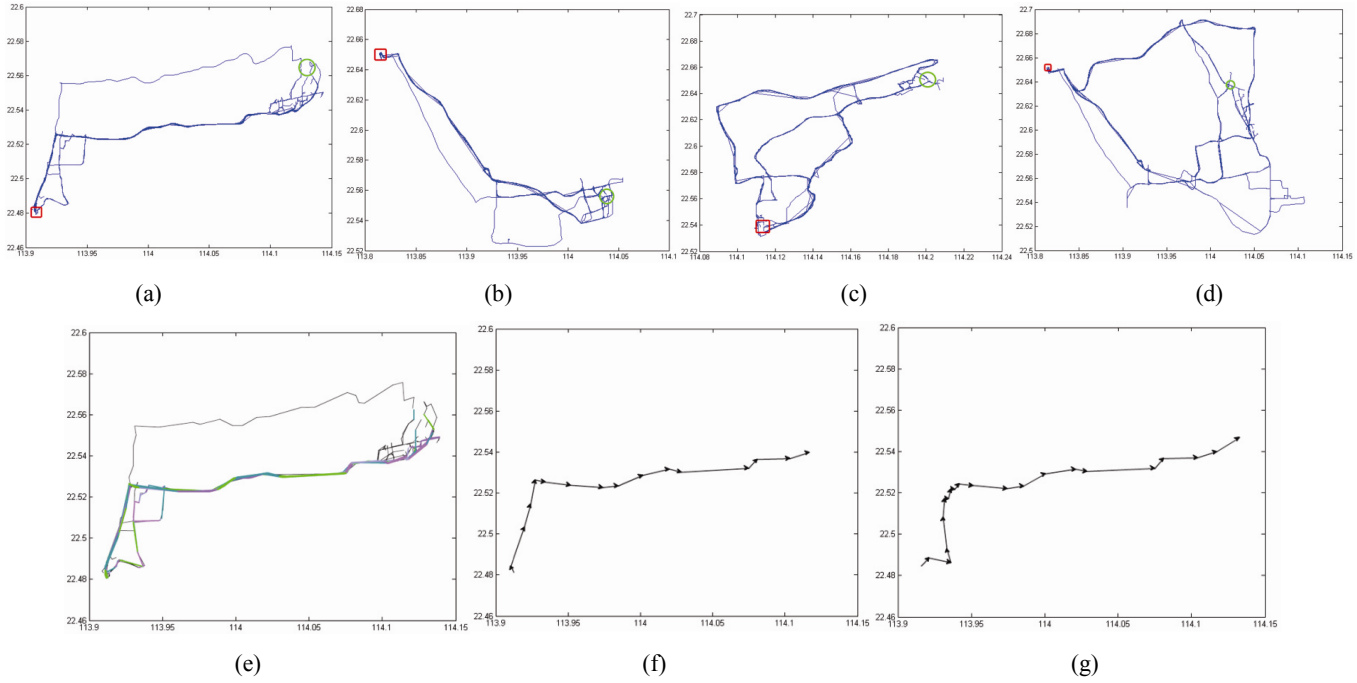


Fig. 2. Collocation pattern and long frequent pattern discovered. (a)-(d) show four collocation pattern instances, where red rectangles mark the start places and green circles mark the end places, and (e) is the simplified and frequent singular pattern discovered of (a), and (e)-(f) are two long frequent patterns discovered.

B. Parameter Setting

In the clustering step for collocation pattern extraction, we use shared nearest neighbor (SNN) clustering, in which the neighbor list size k is the most important parameter as it determines the granularity of clusters. As described in [9], the density-based clustering parameter *MinPts* is a fraction of the neighbor list size k. Hence, we can use the sampling method proposed in [8] to generally determine the density control parameter *MinPts* and then determine the parameter k. In the line segment clustering to find frequent singular pattern in section III-B, we set the similarity threshold $\varepsilon = 5\times10^{-4}$, set the length difference threshold $f = 0.1\times|\vec{l}|$, $\vec{l}$ is the line segment with media length and set the minimum support threshold $min\_sup = 4$.

C. Effectiveness and Efficiency Study

We examine the effectiveness of our study taking as input a database of taxi trajectories, which contains trajectories of 1300 taxis in 12 days. Each GPS sequence in the database is the historical locations of a taxi. Unfortunately, since the fault of devices, much noisy data exits, which makes the locations may change abruptly, so, for the first step, we should denoise the data.

The first experiment is conducted to extract collocation patterns for passengers. As the description in Section II-B, we extract stay points for taxis from the GPS sequences and applied directly SNN algorithm to them, where we set $k = 20$, $MinPts = 120$ and $Eps = 15$. From the result of clustering, we can get the collocation patterns in each of which the trajectory subsequences start and end at the similar locations. For visualization, we show four collocation pattern instances in Fig. Fig. 2(a)-(d)}. In each instance, the start locations are marked by a red rectangle and the end locations are marked by a green circle. Although they start and end at similar places, the chosen routes are different. Our next experiment is to discover frequent path from the subsequences in each instance.

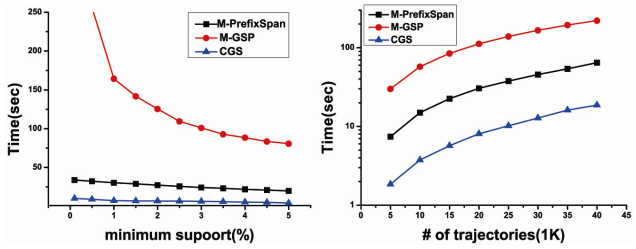
Before frequent pattern discovering, we use line simplification techniques to abstract all the subsequences, where the parameter we set $\varepsilon = 0.001$. After simplification, the singular pattern discovering is performed as the Algorithm1. Fig. 2(e) shows the result of line simplification and singular frequent pattern discovered. For ease to display, we only show the line segments in each region and use different colors to mark different regions, where each color represents a spatial region. Fig. 2(f) &(g) show the two longest frequent pattern discovered by Algorithm 2 for the collocation pattern instance shown in 2(a). For simplicity, only the central line segment of each region is shown, where the arrows are used to indicate the direction changes between different regions.

Next, we evaluate the efficiency of the mining task. For collocation pattern extraction, we used SNN clustering method. The SNN requires $O(n^2)$ time to find the k nearest neighbors for similarity computation. While the original k-nearest neighbor list computation is the bottleneck and the data to process is low dimensional, we use k-d tree to reduce the time complexity to $n*\log(n)$. For singular frequent pattern discovery, the clustering step requires $O(M^2)$, where M is the number of line segments. Although the basic time complexity is $O(M^2)$, we can reduce it to $M*\log(M)$ if we adopt the index technique for moving objects such as TPR-tree [15].

For evaluating the performance of our CGS algorithm, we compare it with the modified PrefixSpan algorithm [11] and modified GSP algorithm [12]. The modified PrefixSpan algorithm finds the frequent 1-pattern only from the first

region of each trajectory in the projected database. And then, each frequent 1-pattern is appended to the prefix to generate a new prefix. For each newly generated prefix, the projected database is built and the mining process is recursively repeated until no more frequent pattern can be found. The GSP algorithm is a level-wise mining method. It first finds all frequent 1-patterns. Next, it generates all candidate 2-patterns by combining a joinable frequent k-pattern and a 1-pattern together. Then, scan the database to count the support for each candidate. The steps are repeated until no more patterns can be found.

Fig. 3(a) shows the execution time versus the minimum support threshold for the CGS, and process 20000 synthetic trajectories. As the minimum support threshold decreases, the execution time of the three algorithms increases because the number of frequent patterns increases substantially. To get the required frequent patterns, the PrefixSpan algorithm needs to generate the projected database many times and the GSP algorithm needs to generate each candidate pattern and calculate its support to scan the entire database. However, the CGS algorithm only extends the pattern and calculates its support in a local space. As a result, the CGS algorithm runs about 3-4 times faster than the PrefixSpan algorithm and 9-50 times faster than the GSP algorithm.



(a) (b)

Fig. 3. Performance evaluation and scalability. (a) is the execution time versus min_sup , and (b) is the scalability of our method.

For scalability, Fig. 3(b) the execution time versus the number of trajectories, where the size of database varies from 5K to 40K. From Fig. 3(b), the execution time of CGS is nearly linear to n .

V. CONCLUSION

In this paper, we modeled a new frequent pattern mining problem from spatiotemporal data of taxi GPS tracks. With the help of the special attribute in the data, we discovered the collocation patterns for passengers, which indicate the groups of passengers with similar start locations and also similar terminations. To find singular frequent patterns, we use a clustering method to abstract the transactional sequences. Based the Connectivity Property of spatiotemporal data and use a structure of T_list, we use an efficient graph-based search method to expand longer frequent patterns, which can substantially reduce candidate patterns and localize support countings.

ACKNOWLEDGMENT

The authors would like to thank the Center for Visual Computing Research in Shenzhen Institutes of Advanced Technology. This work was supported in part by National Natural Science Foundation of China through Grants 60902104 and 60728204, National High-tech R&D Program of China (2009AA01Z302), and Shenzhen Science and Technology Foundation (GJ200807210013A). Part of the data used in the article was provided by Shenzhen Department of Transportation.

REFERENCES

[1] D. Choi, “Personalized local internet in the location-based mobile web search,” Decision Support Systems 43(1)(2007), pp. 31-45.

[2] F. Frentzos, K. Gratsias, Y. Theodoridis, “Trajectory Pattern Mining,” In Proc. SIGKDD, 2007.

[3] M. Hadjieleftheriou, G. Kollios, V.J. Tsotras, and D. Gunopulos, “Efficient Indexing of Sptiotemporal Objects,” In Proc. Eighth Int. Conf. Extending Database Technology, pp. 251-268, 2002.

[4] D. Pfoser, C.S. Jensen, and Y. Theodoridis, “Novel Approaches in Query Processing for Moving Object Trajectories,” Journal of VLDB pp. 395-406, 2000.

[5] Y. Tao and D. Papadias, “MV3R-Tree: A Spatio-Temporal Access Method for Timestamp and Interval Queries,” In Proc. VLDB Conf., pp. 431-440, 2001.

[6] R. Benetis, C. S. Jensen, G. Karciauskas, S. Saltenis, “Nearest Neighbor and Reverse Nearest Neighbor Queries for Moving objects,” IDEAS 2002.

[7] H. Cao, N. Mamoulis, D. W. Cheung, “Mining Frequent Spatio-Temmporal Sequential Patterns,” The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

[8] M. Ester, H. -P. Kriegel, J. Sander, X. Xu, “A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise,” In Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining, Portland, Oregon, pp. 226-231, Aug. 1996.

[9] L. Ertoz, M. Steinbach, and Kumar, “Finding Clusters of Different Sizes, Shapes, and Densities in Noisy, High Dimensional Data,” Technical Report 2002.

[10] R. Agrawal and R. Srikant, “Mining Sequential Patterns,” In Proc. Of Int. Conf. on Data Engineering, pp. 3-14, 1995.

[11] J. Pei, J. Han, B. Mortazavi-Asl, Q. Chen, U. Dayal, M. C. Hsu, “PrefixSpan: Mining Sequential Patterns efficiently by Prefix-Projected Pattern Growth,” In Proc. Int. Conf. on Data Engineering, 2001, pp. 215-224.

[12] R. Srikant, R. Agrawal, “Mining Sequential Patterns: Generalizations and Performance Improvements,” In Proc. 5th Int. Conf. on Extending Database Technology: Advances in Database Technology, 1996, pp. 3-17.

[13] J. -G. Lee, J. Han, and K. -Y. Whang, “Trajectory Clustering: a Partition-and-Group Framework,” In Proc. of SIGMOD, 2007.

[14] D. H. Douglas and T. K. Peucker, “Algorithms for the reduction of the number of points required to represent a digitized line or its or ites caricature,” In the Canadian Cartographcer, Vol. 10, No. 2, pp. 112-122, 1973.

[15] Y. Tao, D. Papadias, and J. Sun, “TPR*-Tree: An optimized spatio-temporal access method for predictive queries,” In Proc. of the Int'l. COnf. on Very Large Data Bases, VLDB, step. 2003.