Postprint: Research on Aircraft Trajectory Pattern Mining Based on Trajectory Clustering

Authors: Guo Wei, Huifeng Tang

Date: 2018-12-13T00:00:00+00:00

Abstract

Trajectory patterns represent relatively stable flight patterns of aircraft during certain time periods or within specific regions, and are of significant importance for understanding and determining target behavior over time or within particular areas. Addressing the characteristics of target trajectories, we design and implement a trajectory clustering method based on line segment density, built upon point density-based clustering algorithms. This method employs the Minimum Description Length principle to partition a target' s historical trajectory into multiple segments, performs clustering of flight trajectories by computing similarities between these segments, and finally utilizes a sweep line algorithm to generate the target' s trajectory pattern. Experimental results demonstrate that the proposed method can fairly accurately discover trajectory patterns of aircraft targets from large-scale trajectory data.

Full Text

Preamble

Vol. 37 No. 2
Application Research of Computers (ChinaXiv Cooperative Journal)
Accepted Paper

Research on Trajectory Pattern Mining Based on Trajectory Clustering

Guo Weia, Tang Huifengb
(a. Graduate School; b. Data & Target Engineering College, Information Engineering University of Strategic Support Force, Zhengzhou 450001, China)

Abstract: Trajectory pattern represents a relatively stable flight mode of an aircraft within a certain time period or region, which is of great significance for understanding and judging target behavior. Aiming at the characteristics

of target trajectories, this paper designs and implements a trajectory clustering method based on line segment density, building upon point density clustering algorithms. The method employs the Minimum Description Length (MDL) principle to segment historical trajectories into multiple trajectory segments, clusters flight trajectories by calculating similarity between segments, and finally generates target trajectory patterns using a sweep line algorithm. Experiments demonstrate that this method can accurately extract aircraft trajectory patterns from large-scale trajectory data.

Key words: trajectory pattern; trajectory clustering; MDL principle; line segment density; sweep line algorithm

0 Introduction

Advances in aviation technology have driven tremendous growth and prosperity in the aviation industry, yet the increasing number of various aircraft has created significant regulatory challenges. While aircraft have brought convenience to humanity, monitoring these maneuverable aerial targets has become a thorny problem. Although modern society' s improved information capabilities and remote sensing technology have simplified real-time aircraft position acquisition, analyzing the behavior of certain aircraft—particularly non-commercial flights with flexible and variable flight processes—remains largely dependent on controllers' knowledge and experience, making full automation difficult to achieve.

Against the backdrop of rapidly expanding flight data volumes and insufficient real-time analytical processing capabilities, discovering frequent trajectory patterns and understanding flight characteristics from massive information is crucial for applications including aircraft flight control, trajectory anomaly detection and early warning, and hotspot region identification. Trajectory mining has long been widely applied in meteorology, ecology, and other fields. In nature, animal migration paths, hurricane trajectories, and ocean current movements often exhibit certain regularities, with their routes reflecting common motion trends to some extent. Inspired by this, researchers have applied such pattern discovery methods to broader domains. For instance, in military applications, analyzing group target movement routes over time can reveal activity patterns and predict operational intentions. Similarly, these methods can be applied to aircraft trajectory pattern mining to discover flight characteristics and patterns hidden within massive trajectory datasets.

Clustering is one of the commonly used methods in target trajectory pattern mining. Among numerous trajectory clustering approaches, this paper selects the density-based DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm and improves upon it according to aircraft flight data characteristics, implementing an aircraft trajectory clustering algorithm based on line segment density. This method clusters historical flight trajectory data to extract frequent trajectory patterns from massive historical flight data of specific

targets. Based on this foundation, target attribute and behavioral characteristics can be fully mined, laying a solid groundwork for applications such as aerial target control, trajectory anomaly detection, and hotspot region discovery.

---

1 Trajectory Data

Aircraft trajectory data typically refers to aerial target movement data obtained via satellites, radar, or other sensors. In recent years, rapid development in mobile communication and location-aware technologies has made acquiring real-time position information of various mobile targets increasingly simple, causing trajectory data volumes to grow exponentially. How to mine required knowledge and information from massive trajectory data within a relatively short time has become a new challenge in data mining for the big data era.

According to the definition of data mining, mobile trajectory pattern mining can be defined as the process of extracting potential, frequently occurring, and valuable trajectory sequences from massive, heterogeneous, and noisy mobile trajectory sequences. Mobile trajectory data refers to position data generated after a target' s geographical location changes within a certain spatiotemporal environment. This information, ordered chronologically, constitutes the target' s trajectory data. Based on different data collection methods, trajectory data can be categorized as location-sampling-based, time-sampling-based, or event-triggered trajectory data.

The data used in this paper is time-sampling-based trajectory data obtained through multi-source fusion. Specifically, flight data consists of a series of temporally ordered multidimensional data points. In principle, each data point includes information such as point time, acquisition time, longitude, latitude, altitude, speed, movement direction, acquisition method, positioning error, and weather conditions. However, due to differences in sampling methods, instability in sensor operation and data transmission processes, and various uncontrollable external factors, actual acquired flight data is often incomplete, with some attributes missing or erroneous, resulting in non-uniform sampling intervals and precision in the fused data.

To ensure research reliability, data preprocessing is required before trajectory clustering to eliminate obviously erroneous data points through calculation combined with prior knowledge, completing cleaning and noise reduction steps (specific methods are not detailed here). The data format after preprocessing is shown in [Figure 1: see original paper].

---

2 Clustering Method Selection

Clustering is a common technique for partitioning data into meaningful or useful groups (clusters), widely applied in industries and disciplines that process large data volumes. Since aircraft generate massive trajectory data during high-speed flight, treating each target' s every flight trajectory as a separate entity is impractical. Therefore, in practice, massive trajectory data of targets must be clustered to obtain typical flight trajectories. After clustering, target flight patterns can be more easily discovered from general trajectory modes, and flight characteristics can be extracted.

There are many clustering methods, including partitioning methods, hierarchical methods, density-based methods, grid-based methods, and model-based methods. In trajectory clustering, either entire trajectories or sub-trajectories can be clustered. Most existing trajectory clustering algorithms treat target trajectories as a whole to analyze and obtain common characteristics. However, such algorithms encounter many problems. For example, multiple trajectories may contain many similar sub-trajectories, but because the overall trajectories differ, they are classified into different categories, causing common feature information to be overlooked. As shown in [Figure 2: see original paper], T1~T4 are four different trajectories with a common portion (the boxed area), yet under the whole-trajectory clustering concept, these four trajectories are considered completely different, and their common portion is ignored.

The actual application addressed in this paper involves analyzing specific targets' trajectories in certain regions to discover frequent trajectory patterns. Extracting sub-trajectories and discovering their common features is more meaningful. Therefore, this paper adopts an approach that segments original trajectories into multiple sub-trajectories for clustering analysis. Specifically, the Minimum Description Length (MDL) principle is first used to segment original trajectories into sub-trajectories. Then, DBSCAN is improved for trajectory clustering characteristics, and a trajectory clustering algorithm based on line segment density is designed and implemented. This density-based clustering method can discover clusters of arbitrary shapes and is insensitive to noise points, generating clusters of any shape according to actual conditions. It is more suitable for line segment clustering and easier for discovering features hidden in different flight trajectories of the same target from local segments. Finally, target trajectory patterns are generated from the obtained trajectory clusters.

---

3 Algorithm Description

Traditional density-based clustering uses a center-based approach, where the density of a point in a dataset is obtained by counting points within a certain distance from it. DBSCAN is a simple yet effective density-based clustering algorithm for finding high-density regions separated by low-density areas. It also explains many important concepts in density-based clustering, such as neighbor-

hood, point density, core points, direct density reachability, density reachability, and density connectivity, which are referenced in subsequent algorithm improvements and implementations.

The classic DBSCAN algorithm performs point clustering with relatively simple implementation. Point density depends on a specified Euclidean distance. This center-based approach can classify points as core points (points in dense regions), border points (points on dense region edges), or noise points (points in sparse regions). In trajectory clustering, however, the clustering objects are not points but trajectory segments. Therefore, DBSCAN concepts must be appropriately adjusted for trajectory clustering scenarios. Before trajectory clustering, definitions for core line segments, general line segments, and noise line segments are introduced:

a) **Core line segment**: If the number of line segments within a given distance of a line segment exceeds a threshold, that line segment belongs to the core line segment category. Core line segments are located inside clusters, with segment neighborhoods determined by distance calculation functions and specified distance parameters.

b) **General line segment**: General line segments are not core line segments but lie within the neighborhood of a core line segment. A general line segment may fall within the neighborhoods of multiple core line segments.

c) **Noise line segment**: Noise line segments are those that are neither core nor general line segments.

With these definitions, the improved line segment density-based algorithm can be described as follows: The algorithm segments target trajectories into multiple sub-trajectories, each consisting of several line segments. After dividing multiple trajectories into trajectory segments, these segments are clustered. Any two sufficiently close line segments are grouped into the same cluster. Line segments whose neighboring line segments exceed a threshold within a given domain are identified as core line segments. General line segments are assigned to the cluster containing the core line segment, while noise line segments are discarded. The algorithm' s greatest advantage is its ability to discover common sub-trajectories from large-scale trajectory data, thereby obtaining similar paths.

Clustering sub-trajectories involves trajectory segmentation. Since target trajectories consist of chronologically ordered points, the simplest segmentation method uses line segments between every two adjacent points as sub-trajectories. While this segmentation theoretically best represents the original trajectory, it introduces problems: when points are dense, sub-trajectories become too short, making common pattern discovery more difficult, and excessive segmentation leads to excessive computational load. Therefore, a more appropriate trajectory segmentation method is needed.

Minimum Description Length is an important concept in information theory and computer science. Simply put, it means describing the most content with

the fewest symbols, which aligns with this paper' s segmentation requirement: representing a complete trajectory with the fewest segments while preserving its features.

Line segment density-based trajectory clustering operates in three stages: segmentation, clustering, and reconstruction. The segmentation step uses the MDL principle to represent original target trajectories in segmented form. The clustering stage employs a density-based line segment clustering algorithm to cluster similar target trajectory segments. The reconstruction stage generates target trajectory patterns from the obtained trajectory segment clusters.

Assume all n trajectories are in the same set TRA = {T_1, T_2, T_3, ···, T }. Each trajectory consists of a series of multidimensional data points p. If trajectory T contains *len* points, then T = {p_1, p_2, p_3, ···, plen }, and the trajectory pc_1pc_2 ···pck ($1 < c_1 < c_2 <$ ck $<$ *len*) is called a sub-trajectory of T .

During algorithm implementation, each trajectory is first segmented into a set of trajectory segments C = {C_1, C_2, ···, C } representing that trajectory, where C ($0 < i < n$) is the segmented line segment of trajectory T . A line segment p p ($i < j$) is a line segment after trajectory segmentation. If the distance between line segments is less than a certain threshold, they are grouped into the same cluster. Each trajectory segment is eventually identified as a core line segment, general line segment, or noise line segment as the algorithm progresses. Finally, representative trajectories are extracted from the obtained trajectory clusters to generate target trajectory patterns. A representative trajectory consists of a series of points and is a simulated trajectory representing the target' s main flight characteristics. The complete algorithm pseudocode is as follows:

```
Input: Original target trajectory set TRA={T1, T2, T3, ···, Tn}
Output: Target pattern trajectory set
step1 foreach T in TRA
        execute sub-trajectory using MDL principle
        get a line segment set D
step2 foreach line segment L in D
        execute line segment cluster using DBSCAN
step3 execute representative trajectory
```

---

4 Algorithm Principles

Based on aircraft trajectory characteristics during flight, this paper designs and implements a line segment density-based trajectory clustering algorithm on the foundation of DBSCAN. The algorithm comprises three stages: first, using the MDL principle to segment original trajectories, enabling segmented representations to be as concise as possible without distortion, simplifying computation and reducing time and space consumption; second, designing and employing a combined-distance-based line segment similarity discrimination function to clus-

ter trajectory segments, merging similar segments into the same cluster; third, generating representative trajectories from each trajectory cluster as target trajectory patterns.

4.1 Trajectory Segmentation and Representation Based on MDL Principle

This section describes how to segment trajectories while preserving their original appearance as much as possible. The ideal segmentation identifies points where trajectory features change significantly, called characteristic points. For trajectory T = $p_1p_2p_3\cdots$plen , a series of characteristic points {pc_1, pc_2, pc_3, ⋯, pcn} ($c_1 < c_2 < c_3 < \cdots <$ cn) are selected to segment T . After segmentation, every two adjacent characteristic points are connected as a line segment, dividing T into *n*-1 line segments. [Figure 3: see original paper] illustrates the trajectory segmentation process, where solid lines represent the original trajectory, dashed lines represent the segmented characteristic trajectory, and p_1, p_3, p_5 are characteristic points.

During segmentation, two factors must be considered: accuracy and conciseness. Accuracy requires the segmented trajectory to differ as little as possible from the original trajectory. Conciseness requires the number of segmented trajectory segments to be as few as possible, meaning points with obvious trajectory feature changes should be selected. However, selecting too few characteristic points compromises segmentation accuracy. Since trajectory segmentation cannot simultaneously satisfy both requirements optimally, the best balance must be sought. For this purpose, this paper employs the Minimum Description Length (MDL) principle, widely used in information theory, to segment original target trajectories.

In MDL principle, the minimum length comprises two parts: *L(H)* and *L(D|H)*. *L(H)* represents the information length occupied by the hypothesis, while *L(D|H)* represents the information length needed to represent the target based on the hypothesis. When applying MDL principle to trajectory segmentation, *L(H)* and *L(D|H)* have corresponding meanings: *L(H)* represents the sum of lengths of trajectory segments obtained by segmenting the original trajectory according to characteristic points, and *L(D|H)* represents the difference between the original trajectory and segmented trajectory segments, determined by a distance function. This paper' s algorithm uses a distance function represented by the sum of three distances between the original trajectory and segmented trajectory segments: perpendicular distance, angular distance, and parallel distance, with specific definitions and calculation methods explained in the next section. Examples of *L(H)* and *L(D|H)* are shown in [Figure 4: see original paper], where p_1 and p_4 are assumed characteristic points.

As seen from the formula definitions, *L(H)* represents segmentation conciseness, while *L(D|H)* represents accuracy. During characteristic point selection, *L(H)* increases with the number of segmented trajectory segments, and *L(D|H)*

increases with the deviation of segmented trajectories from the original trajectory.

To simplify calculation when finding trajectory segmentation points (characteristic points), a locally optimal approach replaces global optimization. Assuming p is a characteristic point, $L\text{par}(i)$ represents the sum of *L(H)* and *L(D/H)*; when p is an ordinary data point, $L\text{nopar}(i)$ represents the original trajectory length. When determining whether p is a characteristic point, $L\text{par}(i)$ and $L\text{nopar}(i)$ are compared. If $L\text{par}(i) \leq L\text{nopar}(i)$, p can be used as a segmentation point. Moreover, trajectory segmentation achieves optimal effect when the sum of *L(H)* and *L(D/H)* is minimized. The algorithm pseudocode is as follows:

```
Input: Original target trajectory T={p1,p2,p3,p4,⋯,pleni}
Output: Trajectory segmentation point set C
d dd
add p1 into the C
start=1,length=1
while(start+length$\leq$leni)
    i=start+length
    if (Lpar(start,i)<Lnopar(start,i))
        Add pi into C
        startIndex=i
        length=1
    length++
add pleni into C
```

4.2 Similarity Discrimination Function Based on Combined Distance

Before clustering segmented trajectory segments, several distance functions used in the clustering process are introduced. The similarity discrimination function based on combined distance consists of three combined distances: perpendicular distance, parallel distance, and angular distance.

As shown in [Figure 5: see original paper], assuming two *d*-dimensional trajectory line segments L = s e and L = s e , where s , e , s , e are the start and end points of the two segments, and p and p are the projection points of s and e on L , respectively. l_1 is the Euclidean distance between s and p , and l_2 is the Euclidean distance between e and p . The perpendicular distance between L and L is defined as shown in Equation (1). Assuming p and p are projection points of s and e on L , l_1 represents the distance from p to s , and l_2 represents the distance from p to s , the parallel distance between L and L is defined as shown in Equation (2). Assuming |L | represents the length of L , |L | represents the length of L , and represents the angle between L and L (calculable via vector methods), the angular distance between L and L is defined as shown in Equation (3).

The definitions and calculation methods of perpendicular distance, parallel distance, and angular distance are now complete. The similarity discrimination

function between two line segments can be obtained by combining these three distances. Assuming *dist*(L , L) represents the distance discrimination function between L and L , it is defined as shown in Equation (4). A larger *dist*(L , L) value indicates lower similarity between L and L .

```
The algorithm pseudocode is shown below.
Input: Trajectory segment set D={L1,L2⋯Ln}; parameters  and MinLns1, MinLns2.
Output: Trajectory set O={C1,C2⋯Cn}.
step1 mark all the line segments in D as unclassified
    foreach L in D
        if (L is unclassified)
            compute N(L)
            if(|N(L)|$\geq$MinLns1)
                assign clusterID to  X N(L)
                insert N(L)-{L} into queue Q
step2 while(Q$\neq$ )
        Compute N(M)//M is one of a segment in Q
        if(|N(M)|$\geq$MinLns1)
            foreach Y in N(M)
                if (Y is unclassified)
                    assign clusterID to Y
        mark M as noise
        Remove M from the queue Q
The definitions and calculation methods of perpendicular distance, parallel distance, and an
```

4.3 Line Segment Density-Based Trajectory Clustering

For descriptive convenience, in addition to the previously mentioned core line segments, border line segments, and noise line segments, three definitions used in the trajectory clustering process must be introduced. Assuming *D* represents the set of all segmented trajectory segments, and L , L represent any trajectory segment in *D*:

a) **-neighborhood**: The region within a given distance of a trajectory segment is called its -neighborhood, denoted as *N* .

b) **MinLns**: The threshold number of surrounding trajectory segments for determining whether a trajectory segment is a core trajectory segment. If the number of trajectory segments within a trajectory segment' s -neighborhood is greater than or equal to MinLns, that trajectory segment is a core trajectory segment.

c) **Direct density reachability**: In set *D*, if L is within L 's -neighborhood and L is a core trajectory segment, then L is directly density-reachable from L .

d) **Density reachability**: In set *D*, if there exist trajectory segments L_1, L_2, ⋯, L where L $= L_1$ and L $=$ L , and L is directly density-reachable from

L $_1$ $(1 < k \leq n)$, then trajectory segment L is density-reachable from L .

e) **Density connectivity**: In set D, if trajectory segment L is density-reachable from both L and L , then L and L are density-connected.

The purpose of the line segment density-based trajectory clustering algorithm is to find the maximum set O of density-connected trajectory segments in trajectory set D. Implementation requires setting three parameters: proximity threshold and quantity thresholds MinLns1 and MinLns2. Before algorithm execution, all segmented trajectory segments are marked as unclassified. As the algorithm progresses, these segments are marked as different types, assigned to clusters, or marked as noise. For convenience, the algorithm can be divided into three stages:

a) Calculate the -neighborhood of each unmarked trajectory segment. If trajectory segment L is marked as a core trajectory segment, the algorithm proceeds to the second part to calculate and obtain its -neighborhood.

b) Calculate all directly density-reachable trajectory segments in this region and add them to the cluster formed by core trajectory segments. If a newly added trajectory segment has not been evaluated as a core trajectory segment, it is added to queue Q for pending judgment. Parameter MinLns1 is used in this evaluation process.

c) Calculate the number of original trajectories containing trajectory segments in each cluster. If this number is less than threshold MinLns2, the cluster is filtered out.

```
The algorithm pseudocode is shown below.
Input: Trajectory segment set D={L1,L2⋯Ln}; parameters  and MinLns1, MinLns2.
Output: Trajectory set O={C1,C2⋯Cn}.
step1 mark all the line segments in D as unclassified
    foreach L in D
        if (L is unclassified)
            compute N(L)
            if(|N(L)|$\geq$MinLns1)
                assign clusterID to  X N(L)
                insert N(L)-{L} into queue Q
step2 while(Q$\neq$ )
        Compute N(M)//M is one of a segment in Q
        if(|N(M)|$\geq$MinLns1)
            foreach Y in N(M)
                if (Y is unclassified)
                    assign clusterID to Y
        mark M as noise
        Remove M from the queue Q
clusterID ++
mark L as noise
step3 foreach C in O
```

```
if |C|<MinLns2
    Remove C from the set O of clusters
```

4.4 Feature Trajectory Generation

After clustering trajectories, several trajectory clusters are obtained. The final algorithm step generates feature trajectories representing overall motion trends from these clusters. Feature trajectories consist of a series of points obtained through the sweep line algorithm.

The sweep line algorithm proceeds as follows: For the trajectory segment set obtained in the previous step, a line segment perpendicular to the trend of trajectories within the set is generated in the trajectory coordinate system—this is called the sweep line. By calculating the number of intersection points between each trajectory segment and the sweep line, the algorithm determines whether to generate feature trajectory points at corresponding positions. If the number of intersection points exceeds a set threshold, the mean coordinates of these intersection points are calculated as the coordinates of feature trajectory points. Finally, these feature trajectory points are connected to form the feature trajectory of that trajectory set. At the algorithm' s conclusion, feature trajectories from all trajectory segment sets are combined to constitute the target' s trajectory pattern.

An algorithm example is shown in [Figure 6: see original paper], where black solid lines represent trajectory segment sets, dashed lines represent sweep lines moving in the arrow direction, and the intersection threshold is assumed to be 3. Generated feature points are marked as black solid dots, and the trajectory formed by connecting these feature points serves as the set' s feature trajectory.

---

5 Experimental Simulation and Results Analysis

This section verifies algorithm effectiveness through simulation experiments. It first introduces the experimental environment and data, then processes preprocessed trajectory data and parameters, and finally visualizes and analyzes the results.

5.1 Experimental Environment

Hardware Environment: Windows 7 operating system, Intel Core-i7 processor (3.6 GHz), 8 GB RAM.
Programming Language: Java.
Development Environment: Eclipse Neon, JDK 1.8.

The experimental verification used flight trajectory data from Type A aircraft in the first quarter of 2017 (Note: Due to data confidentiality, target names, data, and results have been processed). The dataset contained 157 trajectories

with 23,035 data points. Before experiments, data was cleaned to maximize completeness and reliability.

During experiments, parameters , MinLns1, and MinLns2 required configuration. Initial small-scale data tests during program design revealed that these three parameters significantly affected results. When was fixed, MinLns1 values negatively correlated with the number of generated trajectory clusters; when MinLns1 was fixed, values positively correlated with cluster numbers. MinLns2 also affected the final number of trajectory clusters. Since too many or too few clusters make it difficult to objectively and accurately extract target feature trajectories, parameter settings must be reasonably adjusted based on actual conditions and empirical knowledge.

Many parameter optimization algorithms exist for density-based clustering methods. This paper selected the relatively easy-to-understand and implement Sum of Squared Error (SSE) method, combining it with manual experience to select parameters that generate feature trajectories best representing target flight patterns while maintaining relatively small SSE. Through multiple trials and optimization, was finally set to 0.45, and both MinLns1 and MinLns2 were set to 10.

5.3 Program Execution and Results Analysis

Parallel distance is not used here because, in general, aircraft original trajectories and segmented trajectories are relatively similar. To reduce excessive computational load when trajectories are long and data volumes are large, parallel distance is ignored.

Since experimental data is location information, visualization technology is employed to display data geographically on maps for more intuitive demonstration. The open-source Python toolkits Matplotlib and Basemap were used for data visualization.

[Figure 7: see original paper] shows target trajectories before and after data preprocessing. Comparison clearly reveals numerous erroneous points in preprocessed trajectories. These errors result from non-uniform fused data due to different collection methods. During preprocessing, obviously erroneous data points are eliminated. Therefore, data preprocessing is essential; otherwise, it would affect subsequent trajectory clustering accuracy.

[Figure 8: see original paper] displays target trajectory patterns obtained after complete trajectory clustering of original trajectories. The figure shows that the algorithm successfully extracted three trajectory patterns from historical trajectories. Experimental results were verified by domain experts and found to conform to prior domain knowledge, accurately characterizing Type A aircraft trajectory patterns and activity regularities. The results achieved experimental objectives and proved the algorithm' s rationality.

6 Conclusion

This paper designed and implemented a line segment density-based trajectory clustering method for aircraft trajectory pattern discovery. The method comprises three stages: original trajectory segmentation, trajectory segment clustering, and feature trajectory generation. The segmentation stage uses the Minimum Description Length criterion to divide original target trajectories into segments. The clustering stage employs an improved density-based algorithm to cluster segmented trajectories. The feature trajectory generation stage uses a sweep line algorithm to extract feature trajectories as target trajectory patterns. Finally, experimental simulations verified the method' s rationality and practicality. Beyond aircraft trajectory pattern mining, with appropriate domain-specific improvements, this method also holds significant value for sea-air target control, target trajectory prediction, and hotspot region discovery. Future work will focus on algorithm parameter optimization for different scenarios and target trajectory prediction based on motion trends.

References

[1] Yuan Guan, Xia Shixiong, Zhang Lei. Trajectory clustering algorithm based on structural similarity [J]. Journal of Communications, 2011, 32 (9): 103-110.

[2] Han Jiawei, Kamber M, Pei Jian. Data mining: concept and technology [M]. Fan Ming, Meng Xiaofeng, Trans. 3rd ed. Beijing: Machinery Industry Press, 2012: 1-54.

[3] Apostolakis J. An introduction to data mining [M]// Data Mining in Crystallography. Berlin: Springer, 2009: 1-35.

[4] Liu Dayou, Chen Huiling, Qi Hong, et al. Advances in spatio-temporal data mining [J]. Computer Research and Development, 2013, 50(2): 225-239.

[5] An Jianrui. MapReduce-based algorithm for mining user motion trajectory sequence patterns [D]. Zibo: Shandong University of Technology, 2016.

[6] Pang-Ning Tan, Michael Steinbach, Vipin Kumar. Introduction to Data Mining [M]. Boston: Addison-Wesley Longman Publishing Co. Inc., 2005.

[7] Yu Qianqian. Research on DBSCAN algorithm based on data partition [D]. Wuxi: Jiangnan University, 2013.

[8] Lee J G, Han Jiawei, Whang K Y. Trajectory clustering: a partition-and-group framework [C]//Proc of ACM SIGMOD International Conference on Management of Data. 2007: 593-604.

[9] Xing Dongli, Zhao Meihong, Chen Wencheng. Density-based DBSCAN algorithm [J]. Computer Engineering and Applications, 2007, 43(20): 216-221.

[10] Ankerst M, Breunig M M, Kriegel H P. OPTICS: ordering points to identify the clustering structure [J]. ACM SIGMOD Record, 1999, 28(2): 49-60.

[11] Shamos M I, Hoey D. Geometric intersection problems [C]//Proc of Symposium on Foundations of Computer Science. Piscataway, NJ: IEEE Press, 1976: 208-215.

[12] Han J, Kamber M. Data mining: concepts and techniques [J]. Data Mining Concepts Models Methods & Algorithms, 2006, 5(4): 1-18.

[13] Li Lei, Zheng Jinna, Wang Xinhua. Study on the conversion and making methods of digital mapping geographic map [J]. Geological Survey and Research, 2013, 36(4): 318-323.

Note: Figure translations are in progress. See original paper for figures.