THE JOURNAL OF NAVIGATION (2020), **73**, 342–363. © The Royal Institute of Navigation 2019
doi:10.1017/S037346331900064X

Vessel Trajectory Online Multi-Dimensional Simplification Algorithm

Yuan-qiang Zhang[1,2], Guo-you Shi[1], Song Li[2] and Shu-kai Zhang[3]

[1](*Navigation College, Dalian Maritime University, Dalian 116026, China*)
[2](*Faculty of Maritime and Transportation, Ningbo University, Ningbo 315211, China*)
[3](*Merchant Marine College, Shanghai Maritime University, Shanghai, 201306, China*)
(E-mail: shiguoyoudmu@163.com)

Facilitated by the establishment of terrestrial networks and satellite constellations of Automatic Identification System (AIS) receivers, large quantities of spatial and temporal information that trace ships' paths have been collected. The exponential increase in the amount of AIS data has caused expensive and time-consuming transmission, calculation and storage problems. Using appropriate trajectory simplification methods in a timely manner to compress redundant information while minimising the loss of importation information is important. To minimise the simplification error, this paper proposes an online multi-dimensional simplification algorithm for AIS trajectory streaming data. The simplification algorithm takes into account position, direction and speed preservation. Finally, a comparison experiment with other algorithms is made to examine the effectiveness of this algorithm. The results indicate that the proposed online multi-dimensional simplification algorithm can effectively preserve a ship's motion state, including its position, speed and course.

KEYWORDS

1. AIS Trajectories. 2. Trajectory simplification. 3. Online algorithm. 4. DOTS algorithm.

Submitted: 30 July 2018. Accepted: 21 June 2019. First published online: 22 August 2019.

1. INTRODUCTION. The Automatic Identification System (AIS) is an automatic tracking system that is used on ships, by Vessel Traffic Services (VTS) and by coastal monitoring organisations for identifying and locating vessels by electronically exchanging data with other nearby ships, AIS base stations and satellites. According to the International Telecommunication Union (ITU) (2014) specification, AIS data include ship-related static and dynamic information. The static information, which is manually entered and is automatically broadcast every six minutes or on request, mainly includes identification information such as ship name, type, call sign, dimensions, etc. The dynamic information, which is from Global Navigation Satellite System (GNSS) information (usually Global Positioning System (GPS)) and is broadcast at a time interval from two seconds to six

https://doi.org/10.1017/S037346331900064X Published online by Cambridge University Press

minutes, depending on the vessels' motion, includes the ship's position (longitude and latitude), timestamp, Course Over Ground (COG), Speed Over Ground (SOG), true heading, etc. Ships fitted with AIS transceivers can be tracked by static AIS base stations located along coastlines or, when out of range of terrestrial networks, through a growing number of satellites that are fitted with special AIS receivers. According to the mandatory requirements of the International Convention for the Safety of Life at Sea (SOLAS), Chapter V: Safety of Navigation, Regulation 19 (International Maritime Organization (IMO), 2014), all ships of 300 gross tonnes and upwards engaged on international voyages, cargo ships of 500 gross tonnes and upwards not engaged on international voyages, as well as passenger ships irrespective of size shall be fitted with an AIS transceiver. The widespread use of AIS base stations and satellites has led to a huge volume of spatial trajectories, each of which is comprised of a sequence of spatio-temporal points. As the points comprise a huge volume of data, and certain points are redundant, in the study of AIS data, such as in outlier detection (Pallotta et al., 2013; Sidibé and Shu, 2017), pattern analysis (Ristic et al., 2008; Wang et al., 2017; Zhang et al., 2018) or the statistical analysis of maritime traffic (Mazaheri et al., 2015; Shu et al., 2013), application systems have to bear expensive data storage costs, high communication loads and time-consuming calculations. This requires trajectory simplification technologies to compress the redundant information while keeping the utility of a trajectory.

To improve the processing speed of AIS data, some data processing methods have been proposed. For example, Aarsæther and Moan (2009) used an image registration technique to group track-lines and produce a simplified ship traffic model. The median tracks of the traffic-groups were used to represent the prevailing features of the traffic lane. For managing AIS data, Altan and Otay (2017) proposed a grid-based data analysis method. The observed area was divided into 13 sectors according to the transit traffic waterway border points of the Strait Of Istanbul. Data at the entrance and the exit of the sectors were analysed to find the traffic characteristics of the area. In the studies of Silveira et al. (2013) and Wu et al. (2017a), to improve the analysis speed of maritime traffic patterns, the study area was split into fixed-edge grids. In addition, Wu et al. (2017b) used multi-threading technology to improve AIS data processing speed. Although the above literature has proposed methods to improve the speed of AIS data processing, they did not involve the deletion of redundant data, which will lead to a lot of repetitive work. The trajectory simplification algorithm proposed in this paper can delete a large amount of redundant data in the original trajectory. It can be used as an important pre-processing process for ship trajectory research and can further improve the speed of AIS data processing.

A number of successful attempts have been made regarding AIS trajectory simplification. Zhang et al. (2016) used an AIS-based statistical minimum ship domain and the DP (Douglas and Peucker, 1973) algorithm distance threshold to compress AIS trajectory data and verify the relationship between the distance threshold and compression rate. Vries and Someren (2012) proposed a Two Stage Piecewise Linear Segmentation (2Stage-PLS) vessel trajectory simplification algorithm based on deterministic error bounds (Gudmundsson et al., 2009). The basic idea of the 2Stage-PLS is that it can compress the trajectory to use the speed error measure and distance error measure based on the DP algorithm, which can provide a good stop retention result. Zhu et al. (2014) proposed the concept of the importance degree, which is used to reflect the richness of a single

https://doi.org/10.1017/S037346331900064X Published online by Cambridge University Press

trajectory point containing the offset distance, speed change and course change according to a pre-set importance degree threshold to decide whether to delete the trajectory point.

There are three main problems in the above three trajectory simplification algorithms. First, they are based on the DP algorithm. The DP algorithm can achieve a good compression rate, but it uses the perpendicular distance between the original point and the simplified trajectory as the error measure; without considering the time dimension, it will lead to a large time error. Second, the vessel trajectory simplification algorithm aims to reduce the size of the vessel trajectory data while obtaining a low simplification error. A position-preserving algorithm that does not limit the directional changes and speed changes will lead to a bigger directional error and speed error between the approximate trajectory and original trajectory. Zhu et al. (2014) considers the current speed change and the directional change in vessel trajectory simplification, but constant slight course alterations or speed alterations will be ignored. Thus, better algorithms should be designed to save the important position, direction and speed information. Last, compression algorithms can be classified as either offline or online algorithms based on whether they require the availability of the full data series. An offline algorithm reduces the size of the trajectory after the trajectory has been fully generated. The online algorithm compresses a trajectory instantly as an object travels (Zheng, 2015). The DP algorithm is a typical offline algorithm. The use of an offline algorithm requires considerable memory space and cannot process and display trajectories in real time. Therefore, an online algorithm is needed.

The contributions can be summarised as follows:

- A new problem that arises with vessel trajectory simplification when using existing algorithms based on position-preservation or direction-preservation alone is studied.
- Based on baseline solutions, preservation is proposed an improved online multi-dimensional simplification algorithm considering position preservation, direction preservation and speed preservation.
- Extensive experiments on real AIS datasets were conducted. The results demonstrate the superior performance of the approach on both the error and compression rates.

This paper is structured as follows. Section 2 introduces line simplification theory and the possible problems. In Section 3, the overall design of the Directed acyclic graph based Online Trajectory Simplification (DOTS) algorithm and how to use *fdr* and *fvr* to optimise the DOTS algorithm without increasing the time complexity such that it has the function of direction preservation and speed preservation is introduced. Section 4 validates the effectiveness of the simplification algorithm. The study's conclusions are presented and future research is outlined in Section 5.

2. TRAJECTORY SIMPLIFICATION.

There are two types of simplification algorithms in the study of trajectory simplification: one type is the position-preserving algorithm, and the other is the direction-preserving algorithm. The speed-preserving algorithm is often integrated into the above two algorithms.

2.1. *Position-preserving algorithm.*

Most studies have concentrated on the position-preserving algorithm. Bertolotto and Zhou (2007), Chen et al. (2009), Pallero (2013) and Shi and Charlton (2013) compressed trajectories based on the DP algorithm, while Keogh et al. (2001) and Liu et al. (2013) realised the online compression function of the DP

https://doi.org/10.1017/S037346331900064X Published online by Cambridge University Press

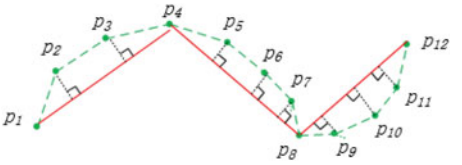


Figure 1. PD error.

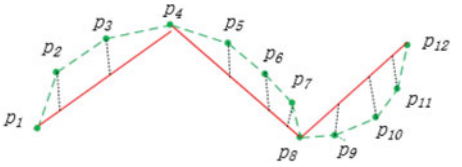


Figure 2. SED error.

algorithm. The algorithm using the Perpendicular Distance (PD) compression trajectory does not take into account the time characteristics of the trajectory. To solve this problem, Meratnia and By (2004); Muckell et al. (2013) and Meng et al. (2017) used Synchronised Euclidean Distance (SED) to compress the trajectory. To improve the speed of the SED computation and ensure the minimum compression error, Chen et al. (2012) and Cao and Li (2017) used the Local Integral Square Synchronised Euclidean Distance (LISSED) to compress the trajectory. The DOTS algorithm of Cao and Li (2017) can minimise the integral Square Synchronised Euclidean Distance (ISSED) under the same compression rate, and the ISSED is the sum of all LISSED errors. The difference between the PD and SED is illustrated in Figures 1 and 2, where the green dashed line is the original trajectory and the red solid line is the simplified trajectory. The PD is the perpendicular distance of the original trajectory point to the simplified trajectory. The SED is the same time position distance between the simplified trajectory and the original trajectory. The LISSED is the SED accumulated sum of each simplified trajectory segment. The ISSED is the LISSED accumulated sum. In general, position-preserving algorithms use the distance between the original trajectory and the simplified trajectory as the criterion to determine whether to delete the trajectory point, and the difference between the various studies is primarily in how to find the best sampling points quickly such that the compression rate is the fastest and the error are the smallest.

The loss of direction changing and speed changing information is the main drawback of the position-preserving algorithm. If the purpose of direction changing is to steer clear of a wreck, buoy or shallow water, the simplified trajectory will be inconsistent with the actual situation. This is a case of over compression and it is difficult to find an appropriate distance threshold to avoid the occurrence of this situation. To solve this problem, the direction-preserving algorithm is added to the position-preserving algorithm when the speed is higher than a pre-set threshold.

2.2. *Direction-preserving algorithm.* There are also a number of studies that compress trajectories from the aspect of direction preservation, and representative algorithms include those by Potamias et al. (2006), Long et al. (2013; 2015), Ke et al. (2017), Wu et al. (2017a) and Deng et al. (2017). Potamias et al. (2006) defined two safe areas where one is formed by the previous two practical points, the other is formed by the previous two sample

https://doi.org/10.1017/S037346331900064X Published online by Cambridge University Press

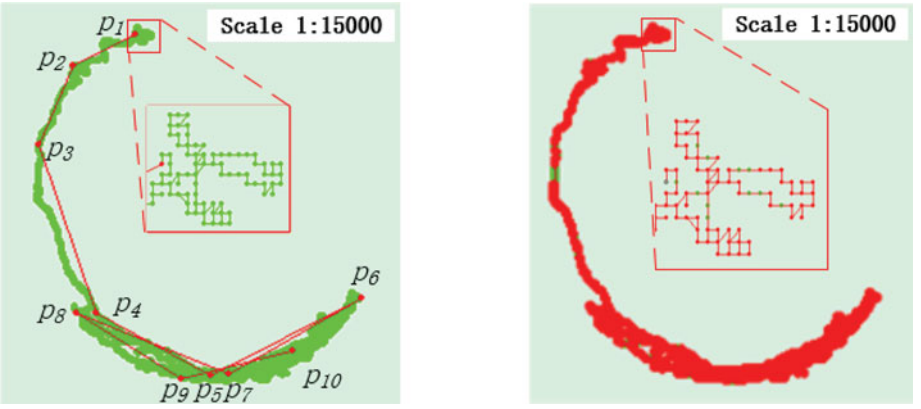


Figure 3. The left diagram shows the compression result based on a position-preserving algorithm. The right diagram shows the compression result based on a direction-preserving algorithm. The green line represents the original trajectory, and the red line represents the simplified trajectory.

points, and only points that are within both safe areas will be deleted. Long et al. (2013; 2015) used the concept of the Feasible Direction Range (FDR) to obtain all the simplified trajectories that meet the requirement of the direction changing threshold, and then returns the solution according to the shortest path found. Deng et al. (2017) proposed an online version based on Long et al. (2013).

Under the states of stopping or low speed navigation, the trajectory direction varies greatly over a short time due to GPS error or the swing of the ship's heading, and the use of a direction-preserving algorithm will lead to under compression. Figure 3 shows an anchoring ship's trajectory simplification result using a position-preserving algorithm and a direction-preserving algorithm, respectively. It can be seen that just ten points are enough to reflect the movement of an anchoring ship under the influence of tidal current. The simplified trajectory based on the direction-preserving algorithm saves most of the original trajectory points. To solve this problem, the direction-preserving algorithm is used only when the speed is greater than a pre-set speed boundary.

2.3. *Speed-preserving algorithm.* In some existing position-preserving algorithms or direction-preserving algorithms, speed preservation is included. For example, Zhu et al. (2014) and Meratnia and By (2004) determined whether or not to save points using the difference in the speed values between the previous and next trajectory segment, and if the difference exceeded a certain speed threshold, they saved the point. Potamias et al. (2006) determined whether to save points using the difference in speed values between the previous two practical points and the previous two sample points. Vries and Someren (2012) determined whether to save points using the difference between the actual speed and the linearly interpolated speed, and according to their experiments, this method could provide better stop retention at the same compression rate. However, this algorithm is an off-line algorithm.

These studies indicate that a single simplification algorithm used on ship trajectory data will not get satisfactory results. Therefore, it is necessary to establish an algorithm considering position, direction and velocity preservation.

https://doi.org/10.1017/S037346331900064X Published online by Cambridge University Press

3. ALGORITHM DESCRIPTION: IMPROVED DOTS ALGORITHM. DOTS minimises the ISSED to preserve the location of the original trajectory as much as possible. The effect is better than that of existing algorithms. However, DOTS lacks direction preservation and speed preservation functions. Therefore, the main purpose of this paper is to improve the DOTS such that it retains direction and speed information.

3.1. *Definitions.* Before establishing the Vessel Trajectory Simplification algorithm, the following definitions should be given.

Definition 1 (Location point): A location point, which is denoted as $p_i = (x_i, y_i, t_i)$, is a location sample with a position and timestamp. x_i is the latitude, y_i is the longitude, t_i is the timestamp and $i \in [1, n]$.

Definition 2 (Trajectory stream): The trajectory stream is defined as a series of location points arranged in chronological order, which is denoted as $T = \{p_1, p_2, \ldots, p_k, \ldots\}$. The compressed trajectory stream of T is denoted as $T' = \{p'_1, p'_2, \ldots, p'_k, \ldots\}$.

Definition 3 (The direction of a trajectory segment line): The line segment between two adjoining points in T is denoted as $\overrightarrow{l_j} = \overrightarrow{p_j p_{j+1}}$, $1 \le j \le n-1$. True north is 0°, and the angle of the clockwise rotation to the $\overrightarrow{l_j}$ is the direction of trajectory segment line, which is denoted as $\theta(\overrightarrow{l_j})$.

Definition 4 (The angular difference between two directions): For two directions θ_1 and θ_2, the angular difference between θ_1 and θ_2, which is denoted as $\Delta(\theta_1, \theta_2)$, is defined as the minimum of the angle of the clockwise rotation from θ_1 to θ_2 and that from θ_2 to θ_1. That is, $\Delta(\theta_1, \theta_2) = \min\{|\theta_1 - \theta_2|, 2\pi - |\theta_1 - \theta_2|\}$.

Definition 5 (The compression angular error of a segment line): $\overrightarrow{l_{sj}} = \overrightarrow{p_{sj} p_{sj+1}}$ is a compressed segment line in T', and its compression angular error, which is denoted by $\varepsilon_\theta(\overrightarrow{l_{sj}})$, is defined as the greatest angular difference between $\theta(\overrightarrow{l_{sj}})$ and $\theta(\overrightarrow{l_k})$, where $\overrightarrow{l_k} = \overrightarrow{p_k p_{k+1}}$ is one segment line in T approximated by $\overrightarrow{l_{sj}}$. This means that $\varepsilon_\theta(\overrightarrow{l_{sj}}) = \max(\Delta(\theta(\overrightarrow{l_{sj}})\theta(\overrightarrow{l_k})))$, and $t_{sj} \le t_k < t_{k+1} \le t_{sj+1}$.

Definition 6 (The difference between two speeds): For two speeds v_1 and v_2, the difference between v_1 and v_2, which is denoted by $\Delta(v_1 v_2)$, is defined as the absolute value of $v_1 - v_2$, which means that $\Delta(v_1 v_2) = |v_1 - v_2|$. The speed on this segment is replaced by the average speed $v(\overrightarrow{l_j})$, and the unit is knots (kn).

Definition 7 (The compression speed error of a segment line): $\overrightarrow{l_{sj}} = \overrightarrow{p_{sj} p_{sj+1}}$ is a compressed segment line in T', and its compression speed error, which is denoted by $\varepsilon_v(\overrightarrow{l_{sj}})$, is defined as the greatest speed difference between $v(\overrightarrow{l_{sj}})$ and $v(\overrightarrow{l_k})$, where $\overrightarrow{l_k} = \overrightarrow{p_k p_{k+1}}$ is one segment line in T approximated by $\overrightarrow{l_{sj}}$. This means that $\varepsilon_v(\overrightarrow{l_{sj}}) = \max(\Delta(v(\overrightarrow{l_{sj}})v(\overrightarrow{l_k})))$, and $t_{sj} \le t_k < t_{k+1} \le t_{sj+1}$.

Definition 8 (The compression rate): The size of T is defined as the number of location points, which is denoted as $|T|$. The size of T' is denoted as $|T'|$. With the compression rate $r = |T'| / |T|$, the smaller r is, the more points that need to be deleted.

https://doi.org/10.1017/S037346331900064X Published online by Cambridge University Press

3.2. *DOTS algorithm.* In this paper, the online compression of an AIS trajectory is realised by improving the DOTS algorithm. Therefore, the DOTS algorithm needs to be described first. The DOTS algorithm uses incremental Directed Acyclic Graph (DAG) construction to realise online trajectory compression, and its process is summarised in Algorithm 1. The function of DAG is to choose the minimum ISSED point in the V_k layer as the parent of p_j. It can create a $k+1$ layer compression path tree as shown in Figure 4. The 1-alive-element layers will decode and output a single vertex to the simplified trajectory T' step by step. The meaning of 1-alive-element layers is represented in Figure 5, in which the alive vertices are marked in light shading, and the dead ones are marked in dark shading. An "alive" vertex is the point that is connected to the V_{k+1} layer, and a "dead" vertex is the point that is not connected to the V_{k+1} layer.

Algorithm 1 Incremental DAG construction.

```
0   Taking V_1, V_2, ..., V_k
1   V_{k+1} = ∅;   Term_k = ∅
2     WHILE p_j ≪ TRAJECTORY_DATA_STREAM
3     FOR p_i in V_k
4       IF p_i ∈ Term_k
5         CONTINUE
6       ENDIF
7       IF LISSED_i^j < ε
8       V_{k+1} = V_{k+1} ∪ {p_j}
9         Parent(p_j) = p_i
10        BREAKFOR
11      ELSEIF LISSED_i^j > ηε
12      Term_k = Term_k ∪ {p_i}
13      ENDIF
14    ENDFOR
15    IF Term_k == V_k
16      BREAK WHILE
17    ENDIF
18  ENDWHILE
19  Minimise ISSED locally according to Equation (4)
20  Try to decode and output according to the decoding and output procedure
```

SED can be described by equation:

$$SED_k = \sqrt{\left(x_k - x'_k\right)^2 + \left(y_k - y'_k\right)^2} \tag{1}$$

where:

$$x'_k = x_i + \frac{t_k - t_i}{t_j - t_i}(x_j - x_i) \quad y'_k = y_i + \frac{t_k - t_i}{t_j - t_i}(y_j - y_i)$$

where i and j are the start and end point numbers of the approximated trajectory and k is the number of any point between p_i and p_j, and $1 \le i < k < j \le |T|$.

https://doi.org/10.1017/S037346331900064X Published online by Cambridge University Press

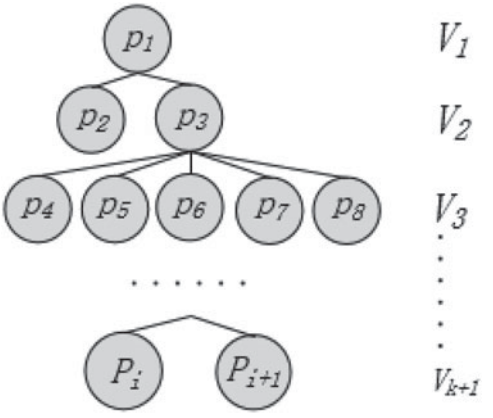


Figure 4. Compression path tree.

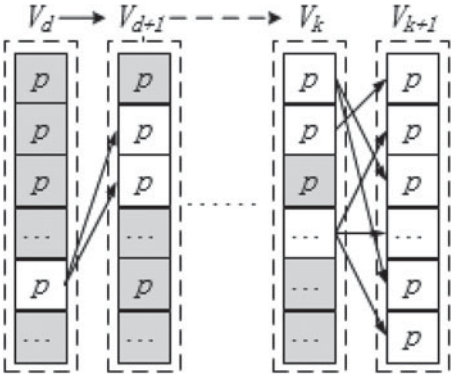


Figure 5. Decoding procedure of DOTS algorithm.

$LISSED_i^j$ can be described by equation:

$$\begin{aligned} LISSED_i^j = \sum_{i<k<j} SED^2\left(p_k, p_k'\right) &= \left(c_1^2 + c_3^2\right)(j - i - 1) + \left(c_2^2 + c_4^2\right)\left(S_{t^2}^{j-1} - S_{t^2}^{i}\right) \\ &+ 2\left(c_1 c_2 + c_3 c_4\right)\left(S_t^{j-1} - S_t^i\right) + \left(S_{x^2}^{j-1} + S_{X^2}^i\right) + \left(S_{y^2}^{j-1} + S_{y^2}^i\right) \\ &- 2c_1\left(S_x^{j-1} + S_x^i\right) + 2c_3\left(S_y^{j-1} + S_y^i\right) - 2c_2\left(S_{xt}^{j-1} + S_{xt}^i\right) + 2c_4\left(S_{yt}^{j-1} + S_{yt}^i\right) \end{aligned} \tag{2}$$

where:

$$c_1 = \frac{x_i t_j - x_j t_i}{t_j - t_i} \quad c_2 = \frac{x_j - x_i}{t_j - t_i} \quad c_3 = \frac{y_i t_j - y_j t_i}{t_j - t_i} \quad c_4 = \frac{y_j - y_i}{t_j - t_i}$$

$$S_x^i = \sum_{j=1}^{i} x_j \quad S_y^i = \sum_{j=1}^{i} y_j \quad S_t^i = \sum_{j=1}^{i} t_j \quad S_{x^2}^i = \sum_{j=1}^{i} x_j^2$$

https://doi.org/10.1017/S037346331900064X Published online by Cambridge University Press

$$S^i_{y^2} = \sum_{j=1}^{i} y_j^2 \quad S^i_{t^2} = \sum_{j=1}^{i} t_j^2 \quad S^i_{xt} = \sum_{j=1}^{i} t_j x_j \quad S^i_{yt} = \sum_{j=1}^{i} t_j y_j$$

The ISSED of the simplified trajectory T' can be described by equation:

$$ISSED = \sum_{p_{k_i} \in T'}^{i} LISSED_{k_i}^{k_{i+1}} \tag{3}$$

If there are multiple points in V_k that satisfy $LISSED_i^j < \varepsilon$, the optimal point is selected using equation:

$$Parent^*(p_j) = \alpha_{p_i} + LISSED_i^j \tag{4}$$

α_{p_i} is the ISSED from p_1 to p_i.

This section briefly describes the structure of the compressed path tree and the construction process of the layers in the DOTS algorithm and gives the calculation formula of the distance scale in the DOTS algorithm. The DOTS algorithm can be used to create a ship trajectory compression path tree online and output sampling points. The distance scale used by DOTS can preserve the spatial and temporal information of the ship trajectory.

There are two key points in DAG that can be optimised to limit the compression angle error and the compression speed error. One is that when p_j selects the parent node in V_k (Step 7 to 10), it needs to add the judgment of the compression angle error and compression speed error so that the two errors are within an allowable range. The other is that in the judgment of the termination condition of DAG (Step 11 to 12), the judgment of the termination condition of the angle error and speed error should be added. How to implement these two key points will be explained in Sections 3.3 and 3.4, and the implementation process will be explained in Section 3.5.

3.3. *Optimising DOTS algorithm using direction-preserving.* To make DOTS have the function of direction preservation, DOTS is modified. According to a pre-set speed boundary v_b, if $v \leq v_b$, the position-preserving algorithm is used, and otherwise the compression angular error judgement is added to the parent vertex selection process, which is denoted as $\varepsilon_\theta(\overrightarrow{p_i p_j}) \leq \varepsilon_\theta$. The time complexity of DOTS is $O(N^2/M)$, $N = |T|$, and M is the number of layers. If the program judges each $\varepsilon_\theta(\overrightarrow{p_i p_j})$, the time complexity is $O((N/M)^2)$, and the time complexity of the whole algorithm is $O((N/M)^2 N)$. To make the optimisation algorithm not increase the time complexity of the original algorithm, the concept of *fdr* can be used. $fdr(\overrightarrow{p_i p_{i+1}})$ represents a directional range of the segment line $\overrightarrow{p_i p_{i+1}}$, the centre direction is $\theta(\overrightarrow{p_i p_{i+1}})$ and the width is $2\varepsilon_\theta$. It is denoted as:

$$fdr\left(\overrightarrow{p_i p_{i+1}}\right) = [\theta\left(\overrightarrow{p_i p_{i+1}}\right) - \varepsilon_\theta \theta\left(\overrightarrow{p_i p_{i+1}}\right) + \varepsilon_\theta] \tag{5}$$

Given $fdr(\overrightarrow{p_i p_j})$, if $\theta(\overrightarrow{p_i p_j}) \in fdr(\overrightarrow{p_i p_j})$, then $\varepsilon_\theta(\overrightarrow{p_i p_j}) < \varepsilon_\theta$.

Here:

$$fdr\left(\overrightarrow{p_i p_j}\right) = \bigcap_{i \leq h \leq j} fdr(\overrightarrow{p_h p_{h+1}}) \tag{6}$$

and $fdr(\overrightarrow{p_i p_j})$ can be obtained by:

$$fdr\left(\overrightarrow{p_i p_j}\right) = fdr(\overrightarrow{p_i p_{j-1}}) \cap fdr(\overrightarrow{p_{j-1} p_j}) \tag{7}$$

https://doi.org/10.1017/S037346331900064X Published online by Cambridge University Press

Given $fdr(\overrightarrow{p_i p_{j-1}})$, the time complexity of calculating $fdr\left(\overrightarrow{p_i p_j}\right)$ is $O(1)$, which is the same as that of calculating $LISSED_i^j$.

In DOTS, the search termination condition for p_i is $LISSED_i^j > \eta\varepsilon$. To minimise the compression rate, the termination condition based on the directional compression is defined as:

$$\text{If } fdr\left(\overrightarrow{p_i p_j}\right) = \emptyset, \text{ then } \theta(\overrightarrow{p_i p_{j+k}}) \notin fdr(\overrightarrow{p_i p_{j+k}}) \tag{8}$$

If the condition Equation (8) is satisfied, it is impossible to find a point after p_j that satisfies $\theta(\overrightarrow{p_i p_{j+k}}) \in fdr(\overrightarrow{p_i p_{j+k}})$.

Proof of Equation (8). According to Equation (7), there is:

$$fdr\left(\overrightarrow{p_i p_{j+1}}\right) = fdr\left(\overrightarrow{p_i p_j}\right) \cap fdr\left(\overrightarrow{p_j p_{j+1}}\right)$$
$$fdr\left(\overrightarrow{p_i p_{j+2}}\right) = fdr\left(\overrightarrow{p_i p_{j+1}}\right) \cap fdr\left(\overrightarrow{p_{j+1} p_{j+2}}\right)$$
$$\cdots\cdots$$
$$fdr\left(\overrightarrow{p_i p_{j+k}}\right) = fdr\left(\overrightarrow{p_i p_{j+k-1}}\right) \cap fdr\left(\overrightarrow{p_{j+k-1} p_{j+k}}\right)$$

If $fdr\left(\overrightarrow{p_i p_j}\right) = \emptyset$, then $fdr\left(\overrightarrow{p_i p_{j+k}}\right) = \emptyset$, then $\theta(\overrightarrow{p_i p_{j+k}}) \notin fdr(\overrightarrow{p_i p_{j+k}})$.

This section solves two problems related to direction preservation. One solution is to convert the judgment of $\varepsilon_\theta(\overrightarrow{p_i p_j}) < \varepsilon_\theta$ to that of $\theta(\overrightarrow{p_i p_j}) \in fdr(\overrightarrow{p_i p_j})$, since $fdr(\overrightarrow{p_i p_j})$ can be obtained quickly with time complexity $O(1)$ when $fdr(\overrightarrow{p_i p_{j-1}})$ is known. The other is that the direction termination condition can dynamically determine the range of the direction compression check to ensure the minimum direction compression rate.

3.4. *Optimising DOTS algorithm using speed-preserving.* Similar to the direction-preserving algorithm, by defining a feasible speed range, as long as the velocity of the compressed trajectory is within the feasible speed range, the original trajectory can be considered to be replaced by the compressed trajectory.

Definition 9 (Feasible speed range): $\overrightarrow{p_k p_{k+1}}(1 \le \mathrm{k} < \mathrm{n})$ is one segment of T. The feasible speed range of $\overrightarrow{p_k p_{k+1}}$, which is denoted by $fvr(\overrightarrow{p_k p_{k+1}}|\varepsilon_v)$, is defined as a speed range $[v_l v_h]$ under the condition of the speed threshold ε_v. The centre speed is $v(\overrightarrow{p_k p_{k+1}})$, the width is $2\varepsilon_v$, $v_l = \max(0, v\left(\overrightarrow{p_k p_{k+1}}\right) - \varepsilon_v)$ and $v_h = v\left(\overrightarrow{p_k p_{k+1}}\right) + \varepsilon_v$. Then, $fvr\left(\overrightarrow{p_k p_{k+1}} \mid \varepsilon_v\right) = [v\left(\overrightarrow{p_k p_{k+1}}\right) - \varepsilon_v\left(\overrightarrow{p_k p_{k+1}}\right) + \varepsilon_v]$.

Given that $\overrightarrow{p_i p_j}$ $(1 \le i < j \le n)$ is a subset of T, according to the definition of fvr, is obtained $fvr(\overrightarrow{p_i p_j}|\varepsilon_v)$ as:

$$fvr\left(\overrightarrow{p_i p_j} \mid \varepsilon_v\right) = \bigcap_{i \le k < j} fvr(\overrightarrow{p_k p_{k+1}}|\varepsilon_v) \tag{9}$$

From here in this paper, $fvr(\overrightarrow{p_i p_j})$ is used to represent $fvr\left(\overrightarrow{p_i p_j} \mid \varepsilon_v\right)$.

THEOREM 1: *For the trajectory stream* $T = \{p_1, p_2, \ldots, p_k, \ldots\}$, $v(\overrightarrow{p_i p_j}) \in fvr(\overrightarrow{p_i p_j})$ *is a sufficient and necessary condition for* $\varepsilon_v(\overrightarrow{p_i p_j}) \le \varepsilon_v$.

https://doi.org/10.1017/S037346331900064X Published online by Cambridge University Press

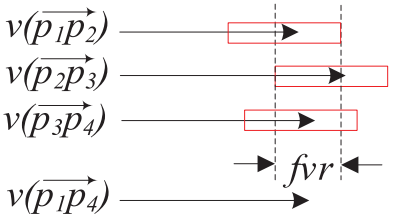


Figure 6. Illustration of Theorem 1 and the speed termination condition.

Proof of the sufficient condition. If $v(\overrightarrow{p_ip_j}) \in fvr(\overrightarrow{p_ip_j})$, according to Equation (9), when $i \le k < j$, then $v(\overrightarrow{p_ip_j}) \in fvr(\overrightarrow{p_kp_{k+1}})$ and $\Delta(v(\overrightarrow{p_ip_j}), v(\overrightarrow{p_kp_{k+1}})) \le \varepsilon_v$. As $\varepsilon_v(\overrightarrow{p_ip_j}) = \max(\Delta(v(\overrightarrow{p_ip_j}), v(\overrightarrow{p_kp_{k+1}}))) (1 \le k < k+1 \le j)$, $\varepsilon_v(\overrightarrow{p_ip_j}) \le \varepsilon_v$.

Proof of the necessary condition. If $\varepsilon_v(\overrightarrow{p_ip_j}) \le \varepsilon_v$, then $\varepsilon_v(\overrightarrow{p_ip_j}) = \max(\Delta(v(\overrightarrow{p_kp_{k+1}}) v(\overrightarrow{p_ip_j}))) \le \varepsilon_v$ $(i \le k < j)$, and $v(\overrightarrow{p_ip_j}) \in fvr(\overrightarrow{p_kp_{k+1}})$. Therefore, there must be $v(\overrightarrow{p_ip_j}) \in fvr(\overrightarrow{p_ip_j})$.

According to Theorem 1, check $\varepsilon_v(\overrightarrow{p_ip_j}) \le \varepsilon_v$ is equal to check $v(\overrightarrow{p_ip_j}) \in fvr(\overrightarrow{p_ip_j})$. If the value of $fvr(\overrightarrow{p_ip_j})$ is known, then:

$$fvr(\overrightarrow{p_ip_{j+1}}) = fvr(\overrightarrow{p_ip_j}) \cap fvr(\overrightarrow{p_jp_{j+1}}) \tag{10}$$

According to Equation (10), after checking $v(\overrightarrow{p_ip_j}) \in fvr(\overrightarrow{p_ip_j})$, the time complexity of obtaining $fvr(\overrightarrow{p_ip_{j+1}})$ is reduced from $O(N)$ to $O(1)$; thus, the time complexity of checking $v(\overrightarrow{p_ip_{j+1}}) \in fvr(\overrightarrow{p_ip_{j+1}})$ is $O(1)$.

This is the same as the direction-preserving algorithm's termination condition. The speed termination condition is as follows:

$$\text{If } fvr(\overrightarrow{p_ip_j}) = \emptyset, \text{ then } v(\overrightarrow{p_ip_{j+k}}) \notin fvr(\overrightarrow{p_ip_{j+k}}) \tag{11}$$

Figure 6 is used to better explain the meaning of Theorem 1 and the speed termination condition. $v(\overrightarrow{p_1p_2})$, $v(\overrightarrow{p_2p_3})$ and $v(\overrightarrow{p_3p_4})$ are three continuous route speeds. If the compression speed error is ε_v, the error allowable range of three speeds can be obtained (represented by rectangles). If speed $v(\overrightarrow{p_1p_4})$ of the simplified route $\overrightarrow{p_1p_4}$ is in three rectangles, it means that $v(\overrightarrow{p_1p_4})$ can replace $v(\overrightarrow{p_1p_2})$, $v(\overrightarrow{p_2p_3})$ and $v(\overrightarrow{p_3p_4})$. To improve calculation speed, the intersection fvr of $v(\overrightarrow{p_1p_2})$, $v(\overrightarrow{p_2p_3})$ and $v(\overrightarrow{p_3p_4})$ can be obtained from Equation (10). If $v(\overrightarrow{p_1p_4})$ is in the range of fvr, the errors between $v(\overrightarrow{p_1p_4})$ and any of the three speeds will not exceed ε_v. If fvr is $\emptyset$, it means that the three speeds do not intersect, so $v(\overrightarrow{p_1p_4})$ must not belong to fvr, which is the speed termination condition.

This section solves two problems related to speed preservation. One solution is to convert the judgment of $\varepsilon_v(\overrightarrow{p_ip_j}) \le \varepsilon_v$ to that of $v(\overrightarrow{p_ip_j}) \in fvr(\overrightarrow{p_ip_j})$ since $fvr(\overrightarrow{p_ip_j})$ can be obtained as quickly as $fdr(\overrightarrow{p_ip_j})$ with time complexity $O(1)$ when $fvr(\overrightarrow{p_ip_{j-1}})$ is known. The other is that the speed termination condition can dynamically determine the range of the speed compression check to ensure the minimum speed compression rate.

3.5. *Algorithm implementation.* The modified algorithm is called the DOTS* algorithm in this paper, and the layer construction process is shown in Algorithm 2. The process needs to have two variables, V_{k+1} and $Term_k$. V_{k+1} stores the points of the current

https://doi.org/10.1017/S037346331900064X Published online by Cambridge University Press

Algorithm 2 Layer construction process of DOTS*.

Taking $V_1, V_2, \ldots, V_k$

$V_{k+1} = \emptyset$; $Term_k = \emptyset$ //Initialise variables V_{k+1} and $Term_k$.
while $p_j \ll TRAJECTORY_DATA_STREAM$ // Get p_j from ship trajectory data stream in chronological order.
  Parent$(p_j) = \emptyset$; $p_j.ISSED = \infty$ // Initialise variables of p_j.
  for p_i in V_k in reverse order //Get p_i in reverse order in V_k.
    Obtain $fdr(ij)$ and $fvr(ij)$
    if $p_i \in Term_k$ // If $Term_k$ contains p_i.
      continue // Return to step 4
    end if
    if Parent$(p_i) == \emptyset$ // If the parent of p_i is null.
      $Term_k = Term_k \cup \{p_i\}$; continue // Merge p_i into $Term_k$, then return to step 4.
    end if
    HasParent = false; IsTerm = false // Initialise variables
    If $LISSED(i,j) < \varepsilon$ **and** $v(ij) \in fvr(ij)$ // If the compression position error and speed error meet the error threshold.
      HasParent = true
    else if $LISSED(i,j) > \eta\varepsilon$ **or** $fvr(i,j) == \emptyset$ // If position or speed termination condition is satisfied.
      IsTerm=true
    end if
    if $v(j, j+1) > v_b$ // If direction compression is available.
      if $\theta(ij) \notin fdr(ij)$ // If the compression direction error does not meet the error threshold.
        HasParent = false
      else if $fdr(i,j) == \emptyset$ // If direction termination condition is satisfied.
        IsTerm=true
      end if
    end if
    // If p_i can be used as the parent of p_j and the compression position error is lower than before.
    if *HasParent==true* **and** $p_j.ISSED > p_i.ISSED + LISSED(ij)$
      Parent$(p_j) = p_i$; //Update p_j.
      $p_j.ISSED = p_i.ISSED + LISSED(ij)$
      $V_{k+1} = V_{k+1} \cup \{p_j$ and these points are not in V_{k+1} before $p_j\}$ //Update V_{k+1}
    else if *IsTerm == true* // If p_i satisfied the termination condition.
      $Term_k = Term_k \cup \{p_i\}$ // Merge p_i into $Term_k$
    end if
  end for
  if $Term_k == V_k$ // If $Term_k$ equals V_k.
    break while // End V_{k+1} layer insertion.
  end if
end while
Try to decode and output according to the decoding and output procedure

https://doi.org/10.1017/S037346331900064X Published online by Cambridge University Press

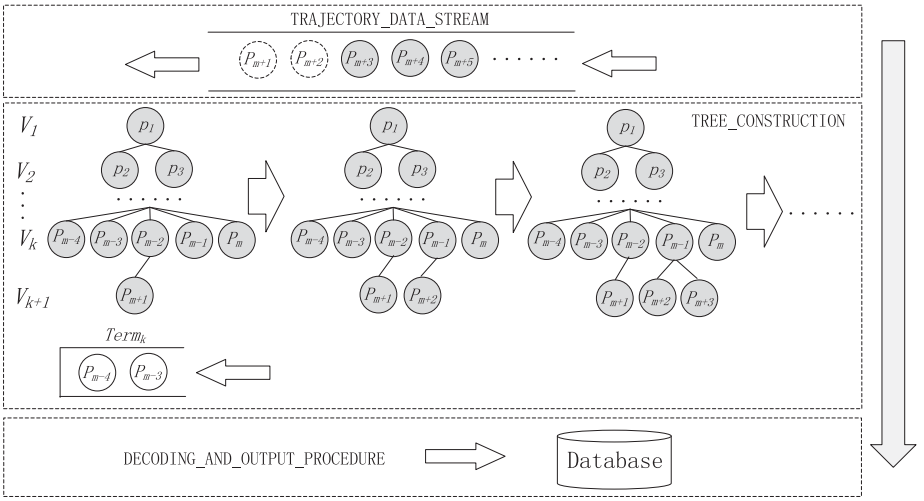


Figure 7. DOTS* algorithm structure. The top part is used to manage unprocessed points. The middle part is used to build the compression path tree. The bottom part is used to output the final simplified trajectory points.

layer, and $Term_k$ stores the points in V_k that satisfy the termination condition. To reduce the time complexity of calculating $\varepsilon_\theta(\overrightarrow{p_i p_j}) \leq \varepsilon_\theta$ and $\varepsilon_v(\overrightarrow{p_i p_j}) \leq \varepsilon_v$, in step 4, search for the parent node of p_j from V_k in reverse order. *HasParent* indicates whether p_j has a parent node, and *IsTerm* indicates whether p_i satisfies the termination condition. If *HasParent* == *true*, steps 26 to 28 select the minimum ISSED point in V_k as the parent of p_j and update p_j and V_{k+1}. "Merge" means that if the point is not in V_{k+1}, it will be inserted, and otherwise the point is updated. The decoding and output process of DOTS* is the same as that of DOTS.

Figure 7 used to better explain Algorithm 2. The figure is composed of three main parts, which are represented by dotted rectangles. TRAJECTORY_DATA_STREAM manages received but unprocessed points and outputs them to the compression path tree in sequence. TREE_CONSTRUCTION is responsible for the construction of the compression path tree. $Term_k$ manages the points in the V_k layer that satisfy the termination conditions. DECODING_AND_OUTPUT_PROCEDURE is responsible for outputting simplified trajectory points to the database. TREE_CONSTRUCTION is the main part of the algorithm, the V_{k+1} layer is created in order from left to right in Figure 7. p_{m+1}, p_{m+2} and p_{m+3} are newly added points in V_{k+1}. For the points satisfying three conditions of $LISSED(i,j) < \varepsilon$, $v(ij) \in fvr(ij)$ and $\theta(ij) \notin fdr(ij)$ at the same time (the last condition is judged only when the speed is greater than v_b), the point whose $p_i ISSED + LISSED(ij)$ is the minimum as the parent of p_j. In addition, p_i satisfying conditions Parent(p_i) == $\emptyset$, $LISSED(i,j) > \eta\varepsilon$, $fvr(i,j) == \emptyset$ and $fdr(i,j) == \emptyset$ is put into $Term_k$. When $Term_k$ contains all points in V_k, the construction of V_{k+1} is completed, and then the decoding and output procedure is executed.

The layer construction process of DOTS* does not increase the time complexity of DOTS. The main reason is that in step 5 in Algorithm 2, the calculation process can be explained using Figure 8. There are 10 points, V_1 to V_3 are the layers that have been processed, and V_4 is the layer being processed. In V_4, after processing p_8, $fdr(7,8)$ and $fvr(7,8)$ need to be temporarily preserved. When dealing with p_9, $fdr(7,9)$ and $fvr(7,9)$ are calculated by $fdr(7,8)$ and $fvr(7,8)$ first and then the parent of p_9 is searched in reverse order

https://doi.org/10.1017/S037346331900064X Published online by Cambridge University Press

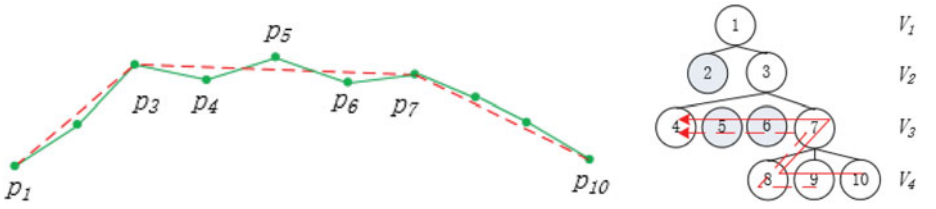


Figure 8. The left diagram is the original (green solid line) and simplified (red dashed line) trajectories. The right diagram is the corresponding compression path tree. The dashed arrow is the computation direction of *fdr* and *fvr* of p_9, and the solid arrow is the computation direction of *fdr* and *fvr* of p_{10}.

in layer V_3. $fdr(6,9)$ and $fvr(6,9)$ can be calculated by $fdr(7,9)$ and $fvr(7,9)$, until V_3 is fully searched. Similarly, when dealing with p_{10}, $fdr(7,10)$ and $fvr(7,10)$ can be calculated by $fdr(7,9)$ and $fvr(7,9)$. Therefore, the time complexity of obtaining $fdr(ij)$ and $fvr(ij)$ is $O(1)$. It should be noted that there is a list maintained in DOTS* that stores the points that have been received but have not yet been inserted into the tree. If the current point finds a parent in the tree, to ensure the correct calculation of *fdr* and *fvr*, DOTS* will insert the current point and these points are not in V_{k+1} before the current point into the tree (Step 28 in Algorithm 2). For example, in Figure 8, p_5 and p_6 did not find the parent in V_2, but p_7 found the parent p_3, DOTS would ignore p_5 and p_6, and DOTS* would need to insert p_5, p_6 and p_7 into V_3. For points without parents, they need to be inserted into $Term_k$ and do not perform the error check, but they must be used to calculate *fdr* and *fvr*. Therefore, steps 9 to 11 are located behind step 5.

Experimental results indicate that when $v(j,j+1) > v_b$, the position-preserving algorithm is performed before executing the direction-preserving algorithm, and the simplification result is better than the direction-preserving algorithm only. Because only the direction-preserving algorithm is performed, in order to make the position compression error not too large, the direction error threshold must be set to a smaller value, which is not conducive to obtaining a low compression rate. When the position-preserving algorithm is added, a higher direction error threshold can be set. In the case of a low compression rate, it can not only get a small position error but also retain the steering information.

The time complexity of computing *fdr* and *fvr* are $O(1)$, which is the same as that of LISSED. Therefore, the time complexity of DOTS* is $O(N^2/M)$, the same as that of DOTS.

4. EXPERIMENTAL RESULTS.

To validate the efficiency and effectiveness of the simplification, the DOTS* algorithm is compared with typical position-preserving and direction-preserving algorithms using the same sample trajectories of 976 ships under the same compression rate. The total number of original AIS position reports is 1,621,319. The data were collected through a network of AIS base stations set up by VTS along the Ningbo coast. The Ningbo Maritime Bureau built 13 base stations and a command centre in 2013, which can achieve full coverage of AIS signals in major port areas. The AIS location data is stored in the form of latitudes and longitudes. Before compression, the geographic coordinates can be converted to planar coordinates using the Mercator coordinate conversion equation (Zhang et al., 2015).

4.1. *Noise filtering.*

AIS data is never perfectly accurate due to GPS noise and other factors. If the noise is not filtered, the experimental results will be affected. Due to the large

https://doi.org/10.1017/S037346331900064X Published online by Cambridge University Press

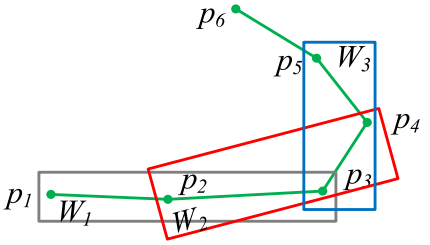


Figure 9. Multiple moving interpolation smoothing.

inertia of a ship, the ship's trajectory should be continuously changing, and an abnormal jump point can be considered as noise. Based on this idea, this paper uses multiple moving interpolation smoothing to filter the AIS noise in trajectories. The specific steps are explained in Figure 9. There are six points that can be used to calculate the course and speed from p_1 to p_5. p_3 is the checked point. There are three moving windows: W_1, W_2 and W_3 and each window contains three points. In the three windows, the position, course and speed of the first point and the third point, and the time of the three points to interpolate the position, course and speed of the intermediate point are used, respectively. If the difference between the interpolations of a window and the received values is greater than the pre-set threshold value, 1 is added to the noise count of p_3. When the noise count of p_3 is greater than 2, it can be deleted as noise. For the first two points and the last three points of the whole trajectory, if they increase the noise count in the calculation, they are deleted; otherwise, they are retained. The use of multiple interpolation filtering rather than one interpolation filtering is to prevent normal points from being deleted as noise when encountering points with large changes in course or speed, such as p_3. In this paper, the position threshold is set to 0·5 NM, the course threshold is set to 20° and the speed threshold is set to 2 kn. As the course is unstable at low speed, the algorithm only filters the course at the points whose speed is greater than v_b. In addition, for distances between two adjacent points larger than 0·5 NM, they are cut into two trajectories.

4.2. *Error definition.* To fully measure the effectiveness of the DOTS* algorithm, the algorithm was tested with seven error criteria, including the average SED error $\overline{\varepsilon(T,T')}$, maximum SED error max $\varepsilon(T,T')$, average direction error $\overline{\varepsilon_\theta(T,T')}$, maximum direction error max $\varepsilon_\theta(T,T')$, average speed error $\overline{\varepsilon_v(T,T')}$, maximum speed error max $\varepsilon_v(T,T')$ and stop retention rate p_{stop}. The stop retention rate is equal to the number of the compression trajectory stops divided by the number of the original trajectory stops. The error equations are as follows:

$$\overline{\varepsilon(T,T')} = \frac{1}{n}\sum_{i=1}^{n} SED(p_i, p_i') \tag{12}$$

$$\overline{\varepsilon_\theta(T,T')} = \frac{1}{n-1}\sum_{i=1}^{n-1} \Delta(\theta(\overrightarrow{p_i p_{i+1}}), \theta(\overrightarrow{pred(p_i)succ(p_{i+1})})) \tag{13}$$

$$\overline{\varepsilon_v(T,T')} = \frac{1}{n-1}\sum_{i=1}^{n-1} \Delta(v(\overrightarrow{p_i p_{i+1}}), v(\overrightarrow{pred(p_i)succ(p_{i+1})})) \tag{14}$$

https://doi.org/10.1017/S037346331900064X Published online by Cambridge University Press

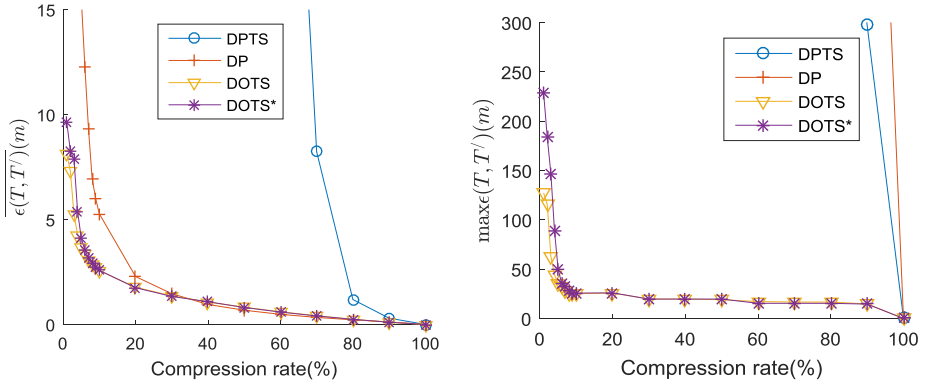


Figure 10. Average SED error (left) and Maximum SED error (right) comparison.

$$\max \varepsilon(T, T') = \max_{1 \le i \le n} SED(p_i, p') \tag{15}$$

$$\max \varepsilon_\theta(T, T') = \max_{1 \le i \le n-1} \Delta(\theta(\overrightarrow{p_i p_{i+1}}), \theta(\overrightarrow{pred(p_i)succ(p_{i+1})})) \tag{16}$$

$$\max \varepsilon_v(T, T') = \max_{1 \le i \le n-1} \Delta(v(\overrightarrow{p_i p_{i+1}}), v(\overrightarrow{pred(p_i)succ(p_{i+1})})) \tag{17}$$

$$p_{stop} = T'_{stop} / T_{stop} \tag{18}$$

4.3. *Error comparison.* In this paper, the stop regions are the regions where the speed is less than 1 kn, and the same stop region does not count repeatedly. We set $v_b = 4$ kn. Through the experiment, it was found that different ε_θs and ε_vs will obtain different upper compression rates. Therefore, to obtain different compression rates, ε_θ and ε_v are set to the three combinations of 0·1 kn and 5°, 1 kn and 5° and 4 kn and 20°, respectively.

Figure 10 gives the $\overline{\varepsilon(T, T')}$ and max $\varepsilon(T, T')$ comparisons of the four algorithms. It can be seen that the $\overline{\varepsilon(T, T')}$ and max $\varepsilon(T, T')$ of DOTS* are larger than those of DOTS under a very small compression rate. The main reason is that DOTS* considers the direction and speed changes and, to achieve a small compression ratio, DOTS* must set a larger ε than DOTS under a very small compression rate. However, when the compression ratio is not very small, DOTS* uses LISSED as the compression scale. This only changes the position of the sampling point, and the influence on the compression ratio is not obvious. The reason why $\overline{\varepsilon(T, T')}$ and max $\varepsilon(T, T')$ of DPTS and DP are significantly larger than those of DOTS* is that DPTS and DP do not preserve the time characteristics of the trajectory.

Figure 11 shows the $\overline{\varepsilon_\theta(T, T')}$ and max $\varepsilon_\theta(T, T')$ comparisons of the four algorithms. As a ship's course is unstable when stopping or sailing at a low speed, the experiment only compares the direction error of the speed greater than 4 kn. The AIS data volume is large and contains a large number of stop positions. Therefore, the $\overline{\varepsilon_\theta(T, T')}$ of DOTS* is not much different from other algorithms except for DPTS. However, DOTS* preserves the points with large directional changes, thus making its max $\varepsilon_\theta(T, T')$ significantly better than those of other algorithms.

Figure 12 gives the $\overline{\varepsilon_v(T, T')}$ and max $\varepsilon_v(T, T')$ comparisons of the four algorithms. The $\overline{\varepsilon_v(T, T')}$ of DOTS* is close to that of DOTS but is significantly smaller than those of DP and DPTS, mainly because both DOTS* and DOTS preserve the time information of

https://doi.org/10.1017/S037346331900064X Published online by Cambridge University Press

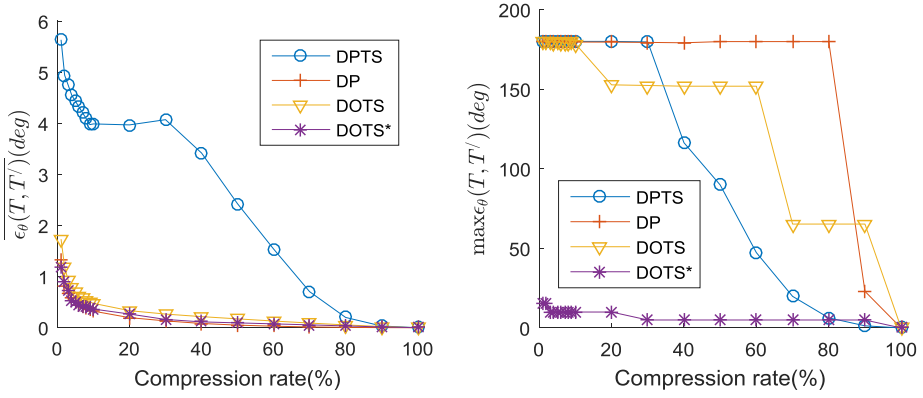


Figure 11. Average direction error (left) and Maximum direction error (right) comparison.

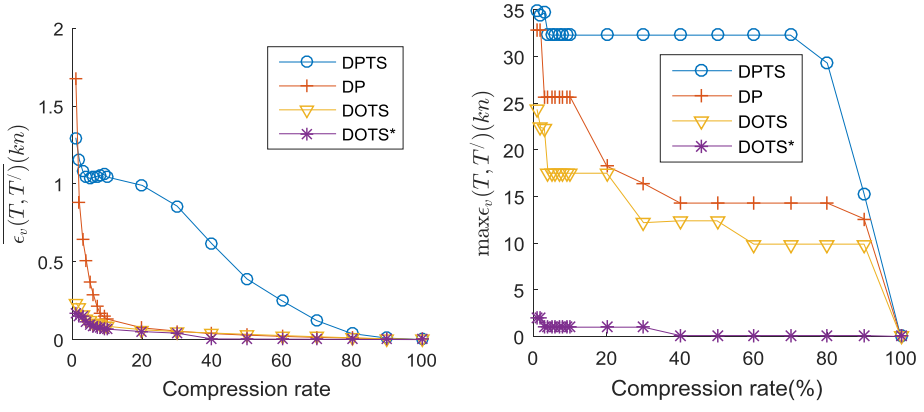


Figure 12. Average speed error (left) and Maximum speed error (right) comparison.

the trajectory. However, DOTS* limits the magnitude of the speed change and thus, max $\varepsilon_v(T, T')$ is significantly better than for the other algorithms.

Figure 13 shows the p_{stop} comparisons of the four algorithms. As DOTS* considers speed preservation, it can preserve more stops than the other algorithms at the same compression rate.

4.4. *The validation of visual observation.* To evaluate the visual effectiveness of the DOTS* algorithm and to analyse the macroscopic traffic flow situation, the original and simplified AIS trajectories were imported into an Electronic Chart Display and Information System (ECDIS). The total number of simplified AIS positions is 166,007, which is approximately 10% of the original trajectories. It can be seen from Figure 14 that the simplified trajectories preserve the shape of the original trajectory very well, including the stand on vessel (right side diagram), the course changing vessel (upper left corner diagram) and the anchored vessel (lower left corner diagram). Therefore, the simplified trajectories can describe the macroscopic traffic flow situation almost as accurately as the original trajectories.

https://doi.org/10.1017/S037346331900064X Published online by Cambridge University Press

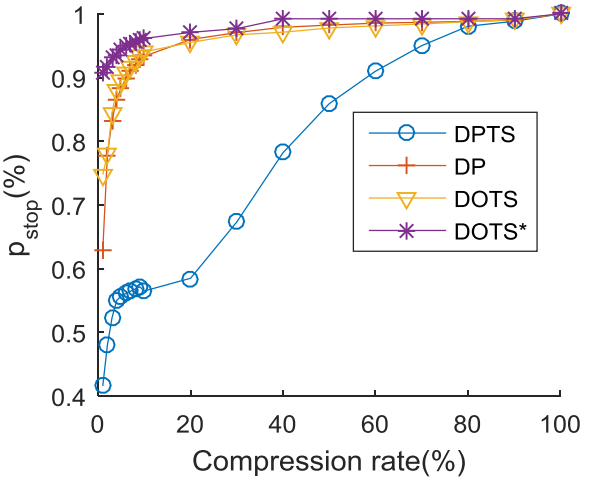


Figure 13. Stop retention rate comparison.

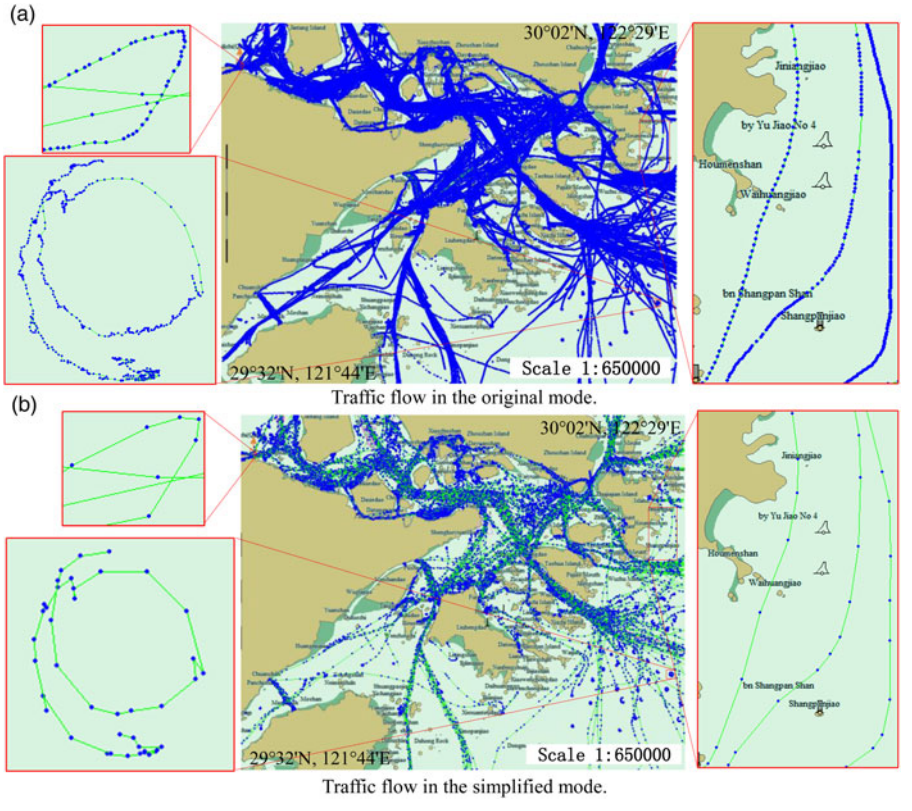


Figure 14. Macroscopic traffic flow comparison before and after compression.

https://doi.org/10.1017/S037346331900064X Published online by Cambridge University Press

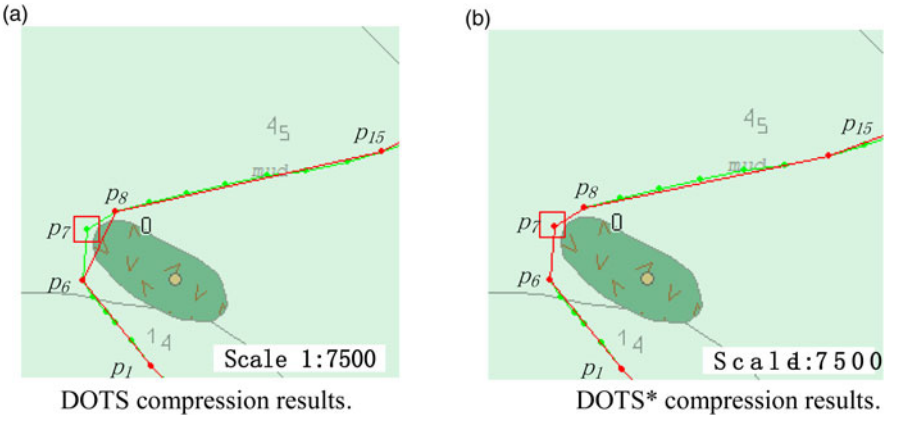


Figure 15. Preserving important direction changing information.

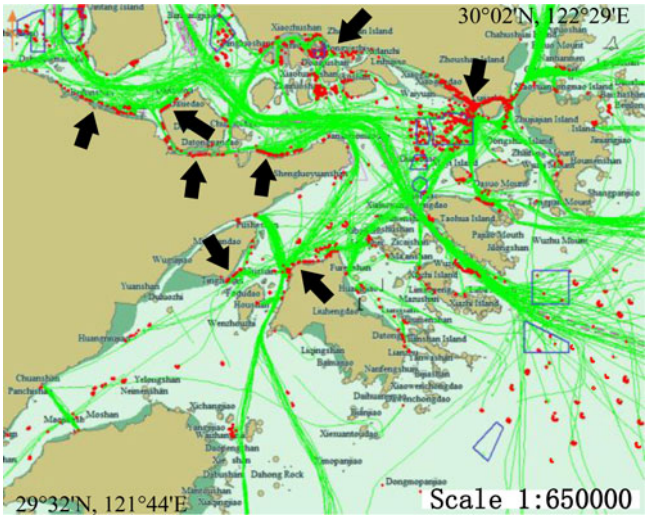


Figure 16. Stop area distribution. The red points are the stop locations, and the areas surrounded by blue solid lines are anchorages.

4.5. *Direction-preserving.* Figure 15 shows the effect of compressing a vessel trajectory around a shoal by using DOTS and DOTS* at the same LISSED threshold. The starting turning point is p_6, the ending turning point is p_8, $\theta(\overrightarrow{p_6p_7}) = 5$, $\theta(\overrightarrow{p_7p_8}) = 59$ and $\theta(\overrightarrow{p_6p_8}) = 26$. As LISSED$(6, 8) < \varepsilon$, in the simplified trajectory compressed by DOTS, $\overrightarrow{p_6p_7}$ and $\overrightarrow{p_7p_8}$ are replaced by $\overrightarrow{p_6p_8}$. DOTS made the simplified trajectory cross the shoals, but the original trajectory was around the shoals. In the DOTS* algorithm, $fdr(6, 7) = [-5, 15]$, $fdr(7, 8) = [49, 69]$, and $fdr(6, 8) = fdr(6, 7) \cap fdr(7, 8) = \emptyset$; thus, $\theta(\overrightarrow{p_1p_3}) \notin fdr(6, 8)$, that is, $\varepsilon_\theta(\overrightarrow{p_6p_8}) > \varepsilon_\theta$. Point p_7 was preserved, and important course change information was not lost.

4.6. *Stop retention.* Figure 16 shows the distribution of stop locations in the study area, including berthing, anchoring and drifting ships. The stop locations within or near

https://doi.org/10.1017/S037346331900064X Published online by Cambridge University Press

the anchorage are anchored ships. The stop locations near the land are berthing ships. The arrows indicate some of the main wharf areas of Ningbo Zhoushan port.

5. CONCLUSIONS. The wide adoption of AIS has led to a huge volume of spatial trajectories. To reduce memory requirements and delete redundant data, an AIS trajectory online simplification algorithm is proposed. First, this paper describes previous work on AIS trajectory simplification, and analyses the problems of these algorithms. Subsequently, the paper describes research on trajectory simplification from three aspects: position-preserving algorithms, direction-preserving algorithms and speed-preserving algorithms. Second, an improved DOTS online compression algorithm is proposed, which is called DOTS*. In order not to increase the time complexity of DOTS, the concepts of FDR and FVR are used to solve the problem of direction preservation and speed preservation. Finally, to validate the efficiency and effectiveness of the proposed algorithm, DOTS* is compared with typical position-preserving and direction-preserving algorithms. Comparisons include average SED error, maximum SED error, average direction error, maximum direction error, average speed error, maximum speed error and stop retention rate. The experimental results show that, at the same compression rate, DOTS* has obvious improvement in information preservation. To evaluate the visual effectiveness of the DOTS*, the results of macro traffic flow compression, direction-preserving and stop retention were visualised. Simplified trajectories can describe the macroscopic traffic flow situation almost as accurately as the original trajectories. Direction preservation can avoid the change of original trajectory semantics meaning. Distribution of stop locations conforms to anchorage and wharf planning. The above statistical analysis and visualisation show DOTS* can preserve the position, direction and speed information well using online compression.

The limitation of this paper is that the error threshold is set uniformly. For different types and sizes of ships, stability and manoeuvrability are different; intuitively, large ships and small ships, container ships and oil tankers should be considered separately. In addition, hydrological conditions should be taken into consideration in compression. This requires the combination of electronic chart data. Future research will take these influencing factors into consideration to obtain a more scientific and more satisfactory simplified trajectory. A future extension study is to use the DOTS* algorithm for incremental learning of ship behaviour to improve the ability of automatic perception of changes in the marine traffic environment.

FINANCIAL SUPPORT

This work was partly supported by “the National Natural Science Foundation of China” (grant number 51709165) and the Fundamental Research Funds for the Central Universities (grant number 3132015009). The author gratefully acknowledges support from the Liu Kong Aiju Education Fund in Ningbo University.

REFERENCES

Aarsæther, K.G. and Moan, T. (2009). Estimating Navigation Patterns from AIS. *The Journal of Navigation*, **62**, 587–607.

Altan, Y.C. and Otay, E.N. (2017). Maritime Traffic Analysis of the Strait of Istanbul based on AIS data. *The Journal of Navigation*, **70**, 1367–1382.

https://doi.org/10.1017/S037346331900064X Published online by Cambridge University Press

Bertolotto, M. and Zhou, M. (2007). Efficient and consistent line simplification for web mapping. *International Journal of Web Engineering and Technology*, **3**, 139–156.

Cao, W. and Li, Y. (2017). DOTS: An online and near-optimal trajectory simplification algorithm. *Journal of Systems and Software*, **126**, 34–44.

Chen, C.J., Lee, T.Y., Huang, Y.M. and Lai, F.J. (2009). Extraction of characteristic points and its fractal Reconstruction for terrain profile data. *Chaos Solutions & Fractals*, **39**, 1732–1743.

Chen, M., Xu, M. and Fränti, P. (2012). A fast $O(N)$ Multiresolution Polygonal Approximation Algorithm for GPS Trajectory Simplification. *IEEE Transactions on Image Processing*, **21**, 2770–2785.

Deng, Z., Han, W., Wang, L., Ranjan, R., Zomaya, A.Y. and Jie, W. (2017). An efficient online direction-preserving compression approach for trajectory streaming data. *Future Generation Computer Systems*, **68**, 150–162.

Douglas, D. and Peucker, T. (1973). Algorithm for the reduction of the number of points required to represent a digital line or its caricature. *Journal of the Canadian Cartographer*, **10**, 112–22.

Gudmundsson, J., Katajainen, J., Merrick, D., Ong, C. and Wolle, T. (2009). Compressing spatio-temporal trajectories. *Computational Geometry: Theory and Applications*, **42**, 825–841.

International Telecommunication Union (ITU). (2014). Technical Characteristics for an Automatic Identification System Using Time-Division Multiple Access in the VHF Maritime Mobile Band. http://www.itu.int/rec/R-REC-M.1371/en. Accessed February 2014.

International Maritime Organization (IMO). (2014). International Convention for the Safety of Life at Sea (SOLAS). China Communications Press Co., Ltd.

Keogh, E., Chu, S., Hart, D. and Pazzani, M. (2001). An Online Algorithm for Segmenting Time Series. *IEEE International Conference on Data Mining*, **289**, 289–296.

Ke, B., Shao, J. and Zhang, D. (2017). An Efficient Online Approach for Direction-Preserving Trajectory Simplification with Interval Bounds. *IEEE International Conference on Mobile Data Management*, 50–55.

Liu, G., Iwai, M. and Sezaki, K. (2013). An Online Method for Trajectory Simplification Under Uncertainty of GPS. *Information and Media Technologies*, **6**, 665–674.

Long, C., Wong, R.C.-W. and Jagadish, H.V. (2013). Direction-preserving trajectory simplification. *VLDB Endowment*, **6**, 949–960.

Long, C., Wong, R.C.-W. and Jagadish, H.V. (2015). Trajectory Simplification: On Minimizing the Direction-based Error. *VLDB Endowment*, **8**, 49–60.

Mazaheri, A., Montewka, J., Kotilainen, P., Sormunen, O.-V.E. and Kujala, P. (2015). Assessing Grounding Frequency using Ship Traffic and Waterway Complexity. *The Journal of Navigation*, **68**, 89–106.

Meratnia, N. and By, R.A.D. (2004). Spatiotemporal Compression Techniques for Moving Point Objects. *International Conference on Advances in Database Technology-EDBT*, **2992**, 765–782.

Meng, Q., Yu, X., Yao, C. and Li, X. (2017). Improvement of OPW-TR Algorithm for Compressing GPS Trajectory Data. *Journal of Information Processing Systems*, **13**, 533–545.

Muckell, J., Olsen, P.W. Jr, Hwang, J.-H., Lawson, C.T. and Ravi, S.S. (2013). Compression of trajectory data: a comprehensive evaluation and new approach. *Geoinformatica*, **18**, 435–460.

Pallero, J.L.G. (2013). Robust line simplification on the plane. *Computers & Geosciences*, **61**, 152–159.

Pallotta, G., Vespe, M. and Bryan, K. (2013). Vessel Pattern Knowledge Discovery from AIS Data: A Framework for Anomaly Detection and Route Prediction. *Entropy*, **15**, 2218–2245.

Potamias, M., Patroumpas, K. and Sellis, T. (2006). Sampling Trajectory Streams with Spatiotemporal Criteria. *International Conference on Scientific & Statistical Database Management*, **52**, 275–284.

Ristic, B., Scala, B.L., Morelande, M. and Gordon, N. (2008). Statistical analysis of motion patterns in AIS Data: Anomaly detection and motion prediction. *International Conference on Information Fusion*, **29**, 1–7.

Shi, S. and Charlton, M. (2013). A new approach and procedure for generalising vector-based maps of realworld features. *GIScience & Remote Sensing*, **50**, 473–482.

Shu, Y.Q., Daamen, D., Ligteringen, H. and Hoogendoorn, S. (2013). Vessel Speed, Course, and Path Analysis in the Botlek Area of the Port of Rotterdam, Netherlands. *Transportation Research Record Journal of the Transportation Research Board*, **2330**, 63–72.

Sidibé, A. and Shu, G. (2017). Study of Automatic Anomalous Behaviour Detection Techniques for Maritime Vessels. *The Journal of Navigation*, **70**, 847–858.

Silveira, P.A.M., Teixeira, A.P. and Soares, G.C. (2013). Use of AIS Data to Characterise Marine Traffic Patterns and Ship Collision Risk off the Coast of Portugal. *The Journal of Navigation*, **66**, 879–898.

Vries, G.K,D.D. and Someren, M.V. (2012). Machine learning for vessel trajectories using compression, alignments and domain knowledge. *Expert Systems with Applications*, **39**, 13426–13439.

https://doi.org/10.1017/S037346331900064X Published online by Cambridge University Press

Wu, F., Fu, K., Wang, Y. and Xiao, Z. (2017a). A Graph-Based Min-# and Error-Optimal Trajectory Simplification Algorithm and Its Extension towards Online Services. *International Journal of Geo-Information*, **6**, 1–21.

Wu, L., Xu, Y.J., Wang, Q., Wang, F. and Xu, Z.W. (2017b). Mapping Global Shipping Density from AIS Data. *The Journal of Navigation*, **70**, 67–81.

Wang, J., Zhu, C., Zhou, Y. and Zhang, W. (2017). Vessel Spatio-temporal Knowledge Discovery with AIS Trajectories Using Co-clustering. *The Journal of Navigation*, **70**, 1383–1400.

Zhang, S.K., Shi, G.Y., Liu, Z.J., Zhao, Z.W. and Wu, Z.L. (2018). Data-driven based automatic maritime routing from massive AIS trajectories in the face of disparity. *Ocean Engineering*, **155**, 240–250.

Zhang, S.K., Liu, Z.J., Cai, Y., Wu, Z.L. and Shi, G.Y. (2016). AIS Trajectories Simplification and Threshold Determination. *The Journal of Navigation*, **69**, 729–744.

Zhang, S.K., Liu, Z.J., Zhang, X.H., Shi, G.Y. and Cai, Y. (2015). A method for AIS track data compression based on Douglas-Peucker algorithm. *Journal of Harbin Engineering University*, **36**, 595–599. (in Chinese)

Zheng, Y. (2015). Trajectory Data Mining: An Overview. *ACM*, **6**, 1–41.

Zhu, F.X., Miao, L.M. and Liu, W. (2014). Research on Vessel Trajectory Multi-Dimensional Compression Algorithm Based on Douglas-Peucker Theory. *Applied Mechanics and Materials*, **694**, 59–62.

https://doi.org/10.1017/S037346331900064X Published online by Cambridge University Press