International Journal of Data Science and Analytics (2018) 5:111–136
https://doi.org/10.1007/s41060-017-0092-8

REGULAR PAPER

Maritime pattern extraction and route reconstruction from incomplete AIS data

Andrej Dobrkovic[1] · **Maria-Eugenia Iacob**[1] · **Jos van Hillegersberg**[1]

Received: 1 February 2017 / Accepted: 15 December 2017 / Published online: 6 January 2018

Abstract
Effective barge scheduling in the logistic domain requires advanced information on the availability of the port terminals and the maritime traffic in their vicinity. To enable a long-term prediction of vessel arrival times, we investigate how to use the publicly available automatic identification system (AIS) data to identify maritime patterns and transform them into a directed graph that can be used to estimate the potential trajectories and destination points. To tackle this problem, we use a genetic algorithm (GA) to cluster vessel position data. Then, we show how to enhance the process to allow fast computation of incremental data coming from the sensors, including the importance of adding a quad tree structure for data preprocessing. Focusing on a real case implementation, characterized by partially incomplete and noisy AIS data, we show how the algorithm can handle routes intersecting the regions with missing data and the repercussions this has on the route graph. Finally, postprocessing is explained that handles graph pruning and filtering. We validate the results produced by the GA by comparing resulting patterns with known inland water routes for two Dutch provinces followed by the simulation using synthetic data to highlight the strengths and weaknesses of this approach.

Keywords Genetic algorithms · Pattern recognition · Data mining · Automatic identification system · Synchromodal logistics

1 Introduction

The availability of reliable information is a prerequisite of any planning. With the ongoing process of continuous cost minimization in the logistics sector, the ability to adequately and timely provide solid information about the present and future state is essential for any type of planning. For Dutch logistics service providers (LSPs), it is essential to maximize the utilization of inland water transportation resources. The most important such resource of LSPs are barges that in an ideal scenario are fully utilized with no or minimal waiting times.

LSPs define terminal disturbances as events when a deep sea vessel makes an unscheduled arrival at the port terminal [2]. Since deep sea vessels have priority over barges, the occurrence of such an event forces barges into a waiting state, until the terminal becomes available to service them again. Minimization of expenses through maximization of resource utilization is only possible if there is an effective way to estimate deep sea vessel destinations, and arrival times, thereby estimating terminal disturbances, and creating optimal scheduling policies taking those into account.

The problem of estimating destination and arrival time for more than one hour ahead cannot be solved by relying solely on position and heading parameters of one entity. As indicated in [3], long-term predictions require identification of sailing patterns for all vessels in the region of interest. Following the successful completion of that step, it is possible to estimate the final position of an entity by associating its current position, and heading to the known pattern for the area, and extrapolating all possible destinations from the graph.

This paper focuses on the problem of maritime pattern extraction. To obtain current positions of the vessels we

This paper is the extended version of the DSAA'2016 special session paper "Maritime Pattern Extraction from AIS data Using a Genetic Algorithm"[1].

✉ Andrej Dobrkovic
a.dobrkovic@utwente.nl

Maria-Eugenia Iacob
m.e.iacob@utwente.nl

Jos van Hillegersberg
j.vanhillegersberg@utwente.nl

1 Industrial Engineering and Business Information Systems, University of Twente, Enschede, The Netherlands

use publicly available Automatic Identification System (AIS) data and zoom in on two specific regions in the Netherlands. The research done by [3] and [4] shows that the basis for pattern identification is waypoint extraction, and, although standard clustering algorithms, such as DBSCAN, can be used for this task, the problem of varying traffic density requires a different approach. In [3], we have shown that a genetic algorithm (GA) can be a viable alternative to other machine learning approaches, and although it is resource intensive, it can deliver accurate results once good criteria for the GA fitness function have been found.

Our goal is to show that by sequential waypoint (WP) discovery using a genetic algorithm, we can extract sailing patterns and transform them into a directed graph. We address the main drawback of evolutionary algorithms, characterized by high computational time, by using quad tree structures to preprocess AIS data and isolate areas of high concentration for the genetic algorithm to discover. This is followed by the discovery of one pair of waypoints at a time. Once those waypoints are found and saved, all vessel positions from the waypoint's neighborhood are removed and the discovery process is repeated. This approach allows for fewer checks of the fitness function, which improves the method's overall performance. The process is repeated until there are no more points that have not been assigned to a cluster. By using this approach, we show that evolutionary algorithms can be used to solve real life data mining and prediction problems. Also, due to its' ability to reduce the resource requirements, this approach becomes viable for future usage in real-time pattern extraction from large volume of streaming data.

Additionally, in the real case business environment, the AIS data relayed to LSPs contains gaps and noise. Depending on the resources used to collect data, the range and the position of AIS receivers, it is expected to have regions with missing AIS information. To have an industry acceptable solution, the algorithm is expected to handle routes with missing data, and give the best possible estimate from the available input.

Thus, the main contribution of this paper is a novel approach on how to adapt a genetic algorithm to handle the real life data mining problem of pattern extraction. While we use maritime data to prove the approach, the same method could be applied to other modes of transport without restrictions. Finally, as maritime routes have the tendency to change over time due to influence of other external factors such as weather and tides [5], the ability of the genetic algorithm to gradually evolve with the new data, will make it the best candidate for providing real-time naval pattern awareness for typical logistics enterprises, utilizing standard PC hardware and accessing imperfect publicly available AIS data. Considering data science domain in general, we show the robustness of the evolutionary algorithms to extract knowledge from noisy and incomplete sequential data. Data mining algorithms in general deal with complete data sets. However, shifting to the big data domain and processing streaming data, introduces new challenges, as the typical algorithms provide the answer based on the snapshot of the actual state. Due to the fact that the evolutionary algorithms contain memory and can adapt to the changes in the input data streams, they are more resilient to the inconsistencies in comparison with the other data mining approaches. Our novel contribution aims at expending the data science body of knowledge by showing that periodic inconsistencies in large data sets can be effectively tackled by memory-based algorithm, such as the modified genetic algorithm, that we present in this paper.

The overall research methodology followed in this study is the design science research methodology [6].

The remainder of the paper is organized as follows. In Sect. 2 we present the problem background and some related research on this topic. Section 3 is dedicated to the solution design. Here we present the genetic algorithm and the required improvement steps. The validation of the extracted patterns is given in Sect. 4. Section 5 is used to compare this approach with the current state of the art, and discuss the future work. Concluding remarks and a summary of the work done are given in Sect. 6.

2 Problem background

2.1 Synchromodal IT platform

The concept of synchromodality was introduced in the Netherlands in 2011 with the goal of allowing the usage of the most suitable mode of transport at all times during the fulfillment of an order, through cooperation within supply chains, transport chains and infrastructures [7]. According to [8] and [9], the core idea of the synchromodal concept is the real-time switching between the transport modes.

In 2013, the Synchromodal IT project was started, in an effort to provide a unified platform to integrate various stakeholders in the logistics domain and manage the process. The added value the Synchromodal IT platform brings is the ability to provide essential information for process optimization that LSPs either could not acquire on their own, or the expense of doing so would not justify the potential benefits [2].

2.2 Long-term arrival time prediction

As mentioned earlier, deep sea vessels are given priority over barges at port terminals, as they represent larger clients bringing higher volumes of cargo, and revenue. Consequently, the arrival of a deep sea vessel, makes the terminal unavailable for barges, which is referred to as terminal disturbance. If a barge is forced into idle mode due to this disturbance, or the cargo containers onboard cannot be loaded/unloaded, expenses for LSPs will increase. Considering the fact that most barges have to be booked in advance, these disruption events are posing constant challenges for the planers.

To solve the problem, the platform needs to predict the occurrence of these disruptions, and this is possible if we can predict the arrival time of deep sea vessels. The publicly available AIS data can be used to give current information about the position of any vessel, its heading, and speed. The literature review in [2] on using AIS data for predicting arrival times, shows that there are two distinctive approaches for estimating future position of a vessel depending on time frame.

The short-term prediction is used to identify anomalies in vessel behavior, indicating suspicious activities like smuggling, piracy, or illegal dumping. These kinds of predictions analyze motion patterns of individual vessels and are accurate for time frames of up to one hour in the future. On the contrary, the long-term prediction is not feasible through the analysis of individual patterns only. Therefore, it is required to extract route patterns, and use them for predicting potential vessel destinations—end points.

2.3 Clustering and route pattern extraction

To extract maritime patterns from AIS data, it is required to have a good algorithm capable of identifying route waypoints. Pallotta et al. [4] define the TREAD methodology that uses incremental DBSCAN to isolate turning points, and connects them afterward to get shipping lanes. Lei et al. [10] developed the TMP algorithm that also uses DBSCAN to discover "hot regions" using them into a modified probabilistic suffix tree to obtain the probability distribution for discrete events occurring in a sequence, thus discovering moving behavior.

Giannotti et al. [11] introduce the T-pattern algorithm to extract frequent temporally annotated sequences of dense spatial regions from trajectories. Rinzivillo et al. [12] use progressive clustering to aggregate trajectories in their entirety based on their similarity in geographic, temporal or attribute space. However, the last two approaches are more suited for non-naval patterns, where initial trajectories do not have the tendency to considerably deviate from one to another due to external factors.

Handl and Knowles [13] show that unsupervised learning problems, such as data clustering, can be solved using an evolutionary approach. The paper also points out that the correct clustering solution often corresponds to the trade-off between two or more clustering objectives, such as minimization of overall deviation, and connectivity. Taking into account that the problem of discovering route patterns is based on clustering of positional data, we conclude that the evolutionary concepts can be used for waypoint identification.

Soares Junior et al. [14] use an unsupervised iterative procedure to segment trajectories. They identify representative points in the data, and use Minimum Description Length to compute similarities, and measure homogeneity needed by the GRASP-UTS algorithm to build new route segments through the modification of the representative points. Research done by Lee et al. [15] and Li et al. [16] use what authors refer to as the "partition-and-group framework" to cluster trajectories. In [15] a trajectory clustering algorithm TRACULUS is developed, which divides trajectories into line segments, and clusters each subset in order to discover common subtrajectories. This process generates a set of representative trajectories, that can be used to analyze special areas of interest. Since the trajectory data are received from the sensors incrementally, the approach is adopted in [16], introducing micro cluster maintenance, and macro cluster creation. This enables the processing of large amounts of data as it is being received, and allows for incremental trajectory clustering. We find this approach to be a good fit for our problem of route extraction as vessel positions coming from the AIS data are recorded sequentially, preventing the solution to be applied to an entire data set, and requiring a form of incremental pattern discovery. However, due to the fact that already extracted routes may not be valid if weather conditions change, and to the possibility of having incomplete and partially noisy data from the real case environment, we chose to base our solution on a genetic algorithm as it can adapt to trajectory changes, and it is showing robustness when handling imperfect input.

2.4 Varying density

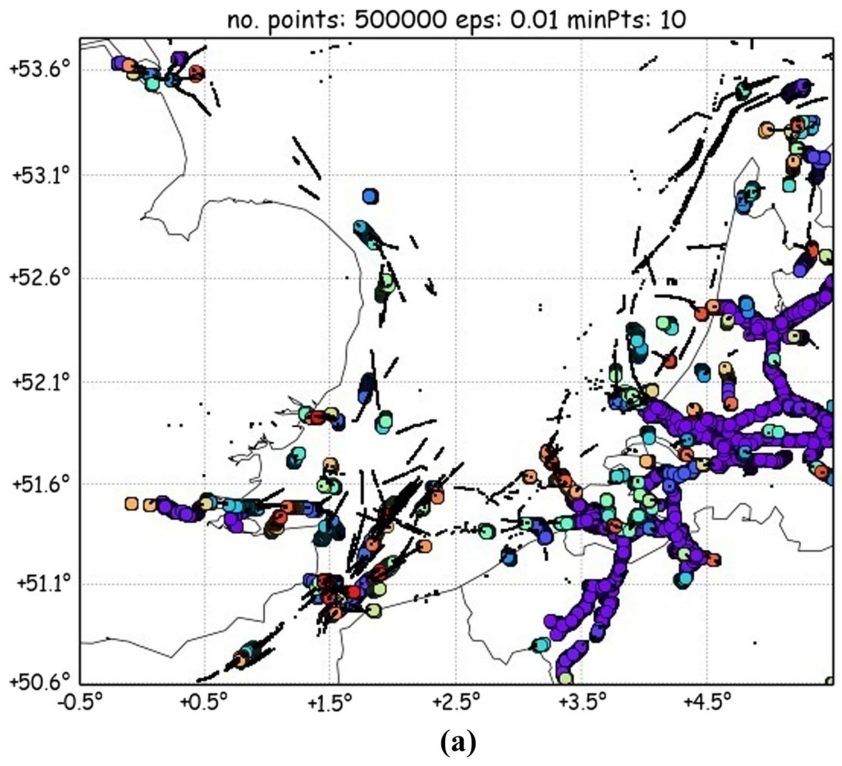
Dobrkovic et al. [3] investigate various algorithms for waypoint discovery in the area of the North sea, including DBSCAN, modified ant colony optimization, and a genetic algorithm. They explain that in a real case application algorithms have to cope with clustering maritime data points where density varies. This makes finding optimal parameters for all clustering algorithms challenging, as shown in Fig. 1 where DBSCAN either misclassifies an entire area of the Netherlands as one cluster, or the area in UK as noise.

3 Solution design

3.1 Genetic algorithm

3.1.1 Conceptual idea

The problem calls for finding a way to identify waypoints in maritime trajectories, i.e., special regions of interest where lanes intersect, as well as connecting them in such a way that it allows to depict all possible routes between the points.



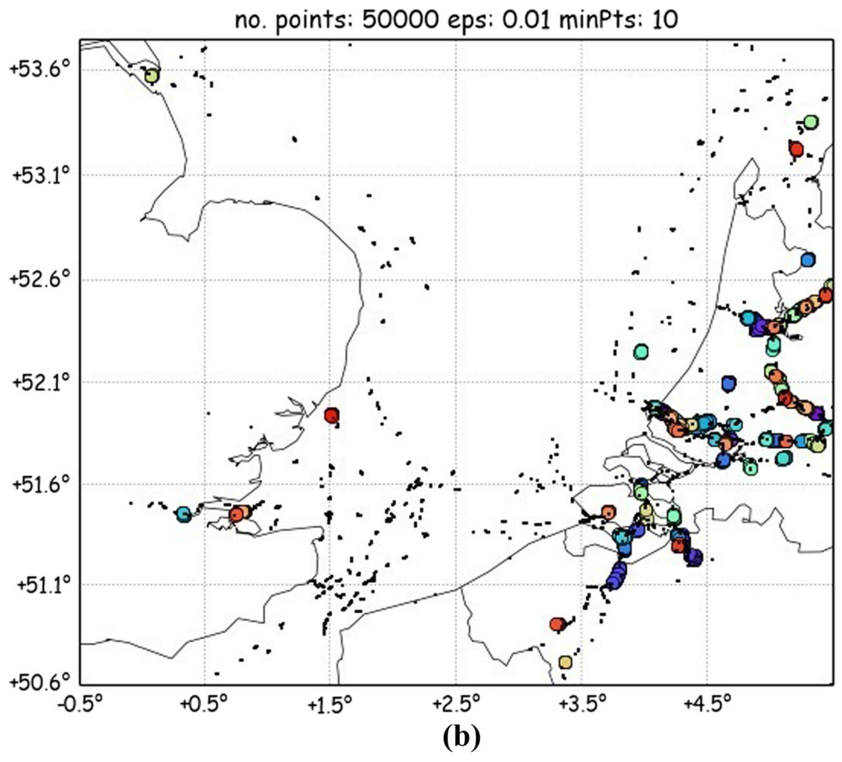


Fig. 1 Using DBSCAN to cluster waypoints from AIS data. Vessel points identified as the noise are presented as black dots. Points contained in a cluster are depicted as circles, where those that belong to the same cluster share the same unique color. This figure shows the difficulty DBSCAN has when dealing with varying data size and density (color figure online)

Let R be the set of vessel points falling inside the area of interest. We define a waypoint F as the set of vessel points falling inside a circle:

$$F = \left\{ (x, y) \in R \,\middle|\, (x - x_c)^2 + (y - y_c)^2 \le r^2 \right\}, \tag{1}$$

where (x_c, y_c) is the center of the circle and r is its radius. Let P denote the cardinality of F, that is:

$$P = |F| \tag{2}$$

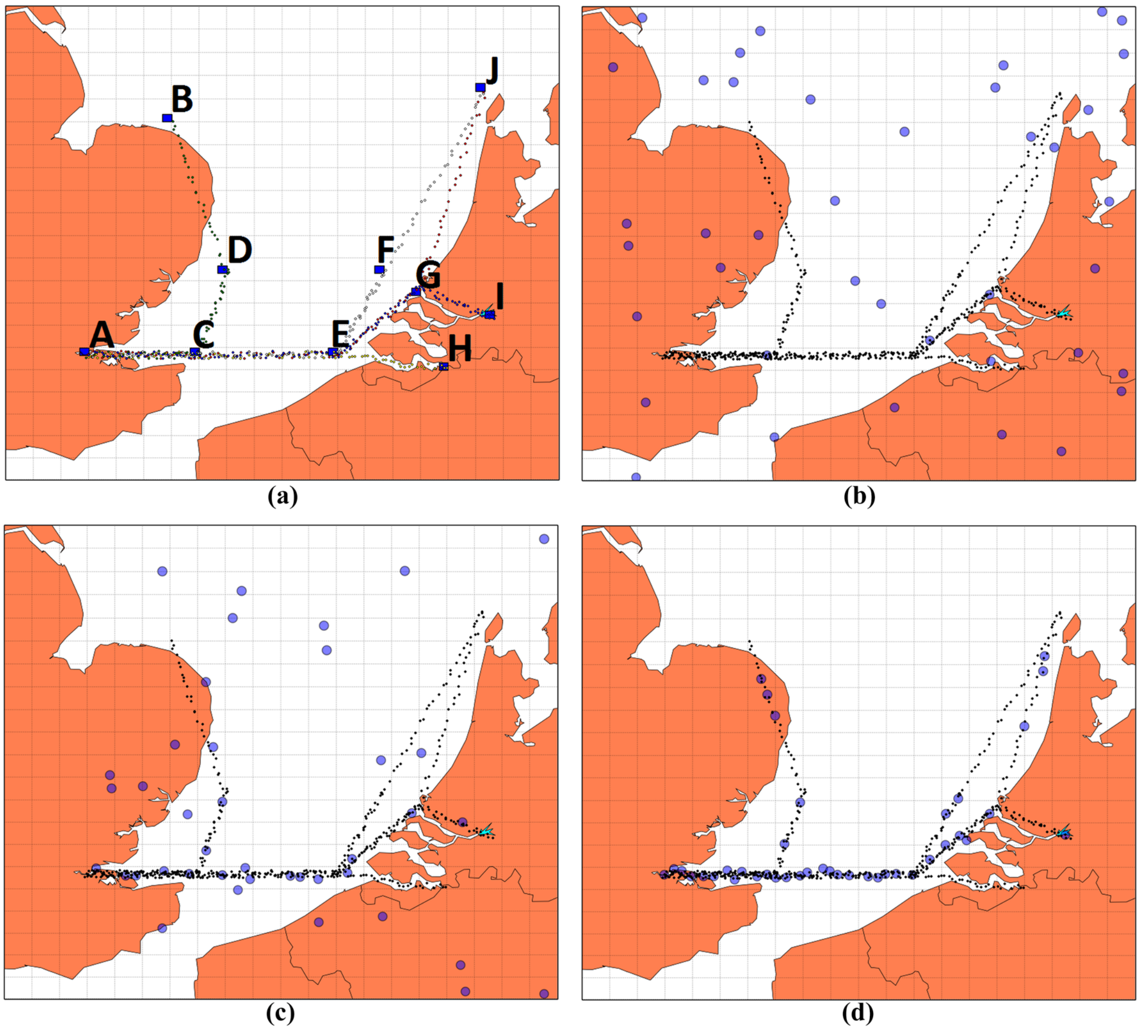


Fig. 2 An illustration of the learning process of a genetic algorithm used to discover predefined routes by converging arbitrary waypoints toward the areas containing vessel positions

Assuming that every lane is comprised of a finite number of evenly distanced points, we can build a set of waypoints that contain all points of that lane. These definitions allow us to define the fitness function of a genetic algorithm that seeks to maximize P such that F contains as many points as possible, by moving the center of F, that is,

$$\max_{(x_c, y_c)\in R} P\left(x_c, y_c\right) \tag{3}$$

For n waypoints we can define the maximization function:

$$\max_{(x_c, y_c)\in R} \sum_{i=1}^{n} P\left(x_{c_i}, y_{c_i}\right) \tag{4}$$

An illustration on how these formulas impact the convergence of waypoints toward routes, and their intersections is shown in Fig. 2. Starting with Fig. 2a, we make an arbitrary set of routes and simulate vessel movements with varying density. Then for a fixed number of waypoints, we apply the genetic algorithm with the fitness function (4). The remaining figures show the result of the learning process after 1 (see Fig. 2b), 50 (see Fig. 2c), and 500 (see Fig. 2d) generations, respectively. As expected, GA converges first toward the area of higher density such as lane C-E. However, if a sufficient number of waypoints is provided, all points from all lanes can be covered.

3.1.2 Encoding

We use direct value encoding to represent one waypoint in one gene. Every gene contains three floating values: latitude, longitude, and radius. The solution for our problem is contained within the chromosome of the genetic algorithm that contains N number of genes (see Fig. 3). It is important to note that, while we intend to use waypoints with varying radius in

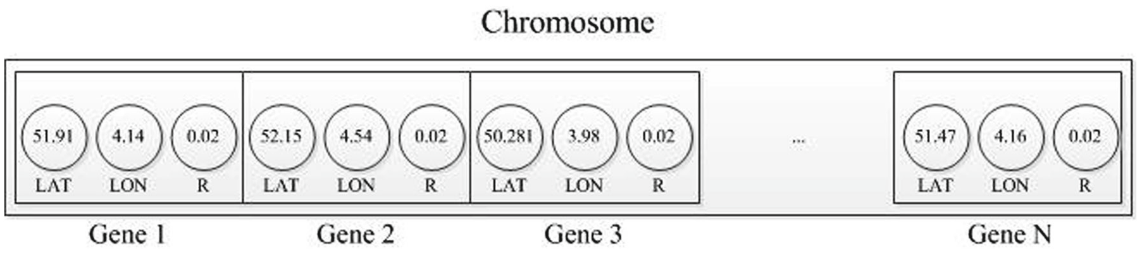


Fig. 3 Chromosome encoding

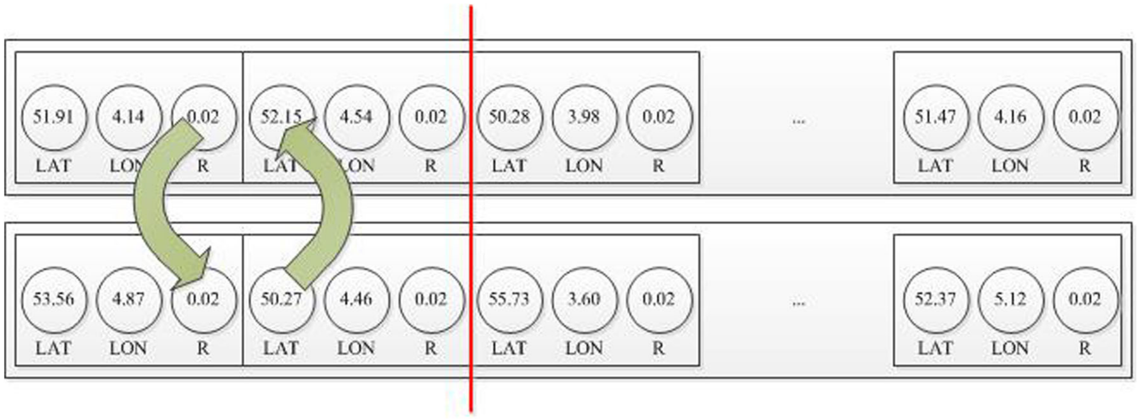


Fig. 4 One-point crossover

the future, for this problem, that radius value is locked, and is equal for all genes, and, as such, it can be omitted.

3.1.3 Fitness function

The fitness function of the genetic algorithm is designed to force waypoints to converge to the area with a high concentration of points. As defined in (1), using values contained in genes we create circles with the center in (LON, LAT), and radius r. Then, we count the number of vessel points contained in this circle. The overall fitness value of each chromosome is the sum of all points contained within all waypoints, divided by total number of points (NP) in the area:

$$\text{FIT} = \frac{\sum_{i=1}^{n} P\left(x_{c_i}, y_{c_i}\right)}{\text{NP}} \tag{5}$$

The penalty function is also added to prevent the occurrence of the same waypoint more than once within the chromosome. Also, we want to force the genetic algorithm to avoid placing waypoints close to each other so they start overlapping. Therefore, we check for the intersection between all circles from a solution, and if found, we penalize the entity with the minimum fitness score.

3.1.4 Initialization

We generate initial values for each gene in the chromosome randomly from the possible values within the problem area. Radius is fixed for all genes, and is area-specific. The number of genes is fixed before the execution of the algorithm, and depends on the complexity of the maritime traffic within the area. Since the number of genes directly influences the computational intensity, we start the experiments with an arbitrary small number, and increase it until a good coverage of the region with waypoints is achieved.

3.1.5 Crossover and mutation

This implementation of the genetic algorithm uses a one-point crossover, due to the fact that the order of waypoints is not important, while we want to keep their number fixed. In GA terms, the chromosome length must remain the same, while genes are exchanged between two entities. Two parents are selected by the roulette wheel selection. Then, a random value in the range from 0 to the number of genes is generated, and both parents exchange their genetic material before that random point (see Fig. 4). After two new offspring are created, a mutation check is performed. If an entity fails, one of its latitude or longitude values gets replaced with a new, randomly generated one.

3.2 Improving the algorithm

3.2.1 The bottleneck of the classic approach

The algorithm presented in the previous section is a typical genetic algorithm, applied to the problem of discovering waypoints. Given a large enough chromosome length, and processing power, all points from the area will be clustered into a set of neighboring circles. However, industry applications require the processing of a large number of points in a limited time frame, utilizing standard PCs only.

At the core of the fitness function is the radius distance check (based on the Pythagorean theorem), which tests if the point (x_p, y_p) is inside a circle with center (x_c, y_c), and radius r.

$$\sqrt{\left|x_p - x_c\right|^2 + \left|y_p - y_c\right|^2} < r \tag{6}$$

or

$$\left|x_p - x_c\right|^2 + \left|y_p - y_c\right|^2 < r^2 \tag{7}$$

This function will be executed NC times:

$$\mathrm{NC} = \mathrm{POP} * \mathrm{EP} * \mathrm{N} * \mathrm{PT}, \tag{8}$$

where parameters POP, EP, N and PT, represent the GA population size, number of epochs, chromosome length, and number of points in the area, respectively. The multiplication of N, and PT is the bottleneck of the process, requiring exponentially more computational power with the increase in traffic density.

The choice of distance function to be used significantly impacts the speed and the execution time of the algorithm. The Euclidian distance function (7) favors fast execution on standard PC hardware, and is acceptable for general route extraction within small regions. In the specific maritime case we calculate the distance using the Haversine formula (9).

$$d = 2r{*}\arcsin\left(\sqrt{\sin^2\left(\frac{y_p - y_c}{2}\right) + \cos\left(y_p\right)\cos\left(y_c\right)\sin^2\left(\frac{x_p - x_c}{2}\right)}\right) \tag{9}$$

Inserting the generally accepted value for the Earth's mean radius of 6371_{km} into $2r$ in (9) the distance check can be expressed as the following formula:

$$r > 6371{*}\arcsin\left(\sqrt{\sin^2\left(\frac{y_p - y_c}{2}\right) + \cos\left(y_p\right)\cos\left(y_c\right)\sin^2\left(\frac{x_p - x_c}{2}\right)}\right)_{\mathrm{km}} \tag{10}$$

Since Earth is not a perfect circle, the distance function (10) will give minor inaccuracies, and the usage of trigonometric calculations will make this function slower in comparison with (7), yet for our specific problem we find these constraints acceptable. If higher precision is required, or greater speed, then alternative distance checks have to be considered.

3.2.2 Improving the genetic algorithm

Improving the solution requires adding several steps to reduce the number of point-to-circle distance checks. First we use a quad tree structure [17] to quickly subdivide the area into smaller regions, and query only those with points adjacent to the testing circle. We also use the same structure to discover areas of high concentration, and direct the GA to converge toward them. Second, we change the process of discovering solutions, and reduce the size of the chromosome to two waypoints only. After solving their position, all points contained inside waypoints, as well as extended neighboring ones are removed from data structure. This leads to fewer checks at every consecutive call. Third, we use the center of mass (COM) to fine tune the final position of waypoints to the data. The fourth step is the creation of a directed graph, by checking the trajectories of every individual vessel and connecting the WPs, which intersected the trajectory of that vessel. Finally, we prune the graph to remove faulty nodes, and edges, caused by noise in AIS data.

In Fig. 5, all steps required to perform maritime route extraction are given. Those that are part of the new incremental genetic algorithm are also highlighted.

3.2.3 Data preprocesing with quad tree

While most clustering algorithms struggle with finding good sets of clusters with varying data density, a quad tree structure has the ability to effectively identify such areas, and also do it in only one pass. Using a quad tree we partition the region by recursively subdividing it into four smaller regions, until a threshold, designating the maximum number of points per region, is reached. We also introduce one adaptation, to prevent further partitions after the specified depth level. This prevents high concentration of very small regions in the area with the highest density. In Fig. 6a, we focus on the example with data in the vicinity of the Rotterdam port. We use a quad tree to subdivide the area, limiting it to the maximum depth level seven. To reduce the number of calls to the GA's fitness function, we start by sending only the regions with the highest depth level in the quad tree as input, and omitting all the others. When completed, we add another layer of data, and repeat the process. Since the GA will ignore all points in the neighborhood of already discovered waypoints, we can assure that the evolution will be done on the number of points within the acceptable range.

We show the result of preprocessing the data of 17.099, and 1.370.346 points in Fig. 6b, d, and c, e, respectively.

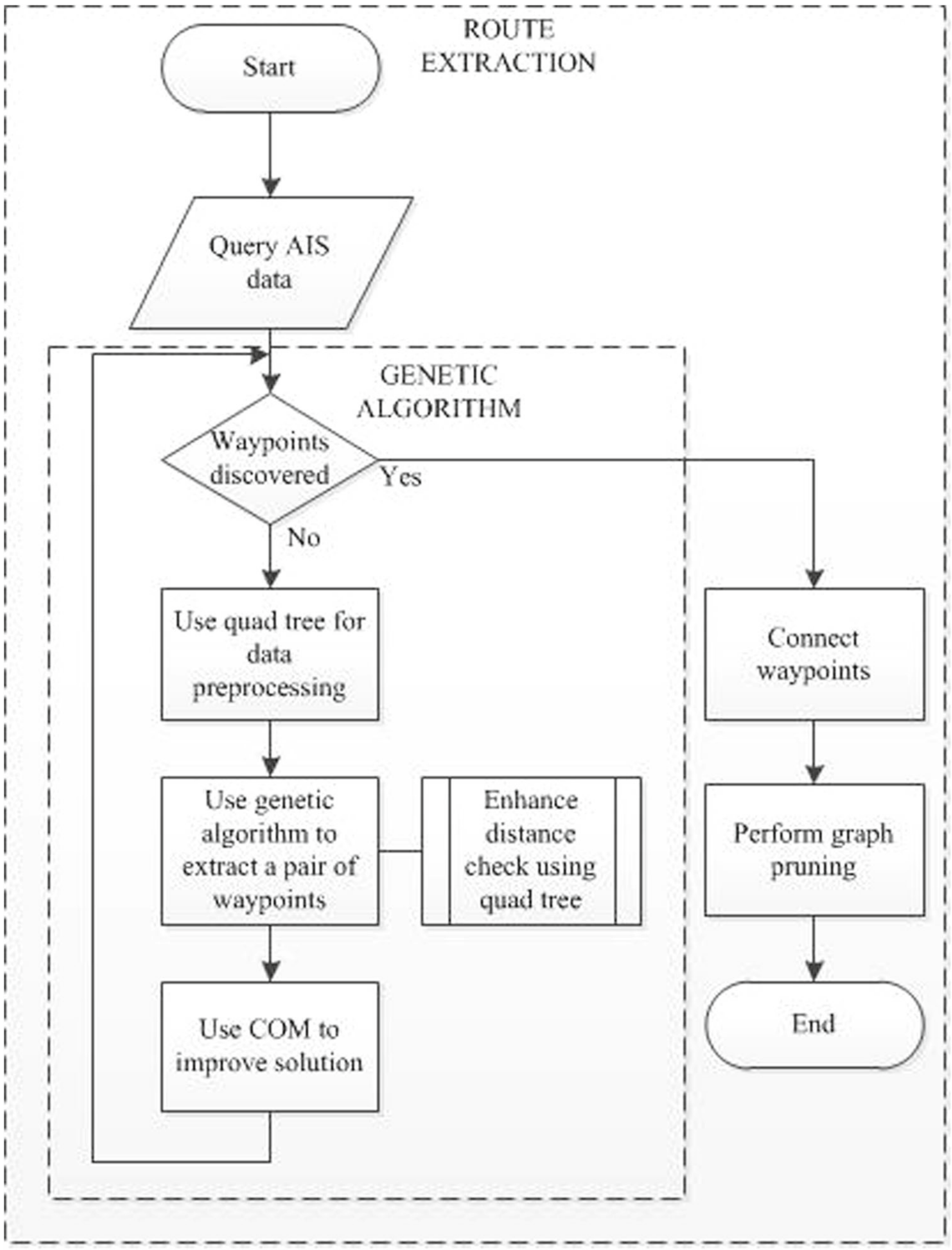

ROUTE EXTRACTION
Start
Query AIS data
GENETIC ALGORITHM
Waypoints discovered
Yes
No
Use quad tree for data preprocessing
Use genetic algorithm to extract a pair of waypoints
Enhance distance check using quad tree
Use COM to improve solution
Connect waypoints
Perform graph pruning
End

Fig. 5 Steps in improving pattern extraction

For the smaller one, the quad tree returns four subregions containing 2.150 points. In the other example, we significantly reduce the area, while still containing 94.57% points (1.296.065 out of 1.370.346).

3.2.4 Incremental waypoint discovery

Complementary with the previous process is the incremental waypoint discovery. In this step, we use GA to discover a pair of waypoints, and then remove all points within bounding circles and near neighborhood from further consideration. The process is divided into phases. Per each phase a pair of waypoints will be added to the solution. Every phase contains the following steps:

- load data, and solution waypoints,
- remove data in vicinity of previously discovered (solution) waypoints,
- initialize genetic algorithm, and use it to find the best fit for two new waypoints,
- add new waypoints to the solution and return it.

To give a visual representation of waypoint discovery we select six phases that illustrate the learning process the best. The results after completing phases 0, 1, 2, 5, 9 and 17 are

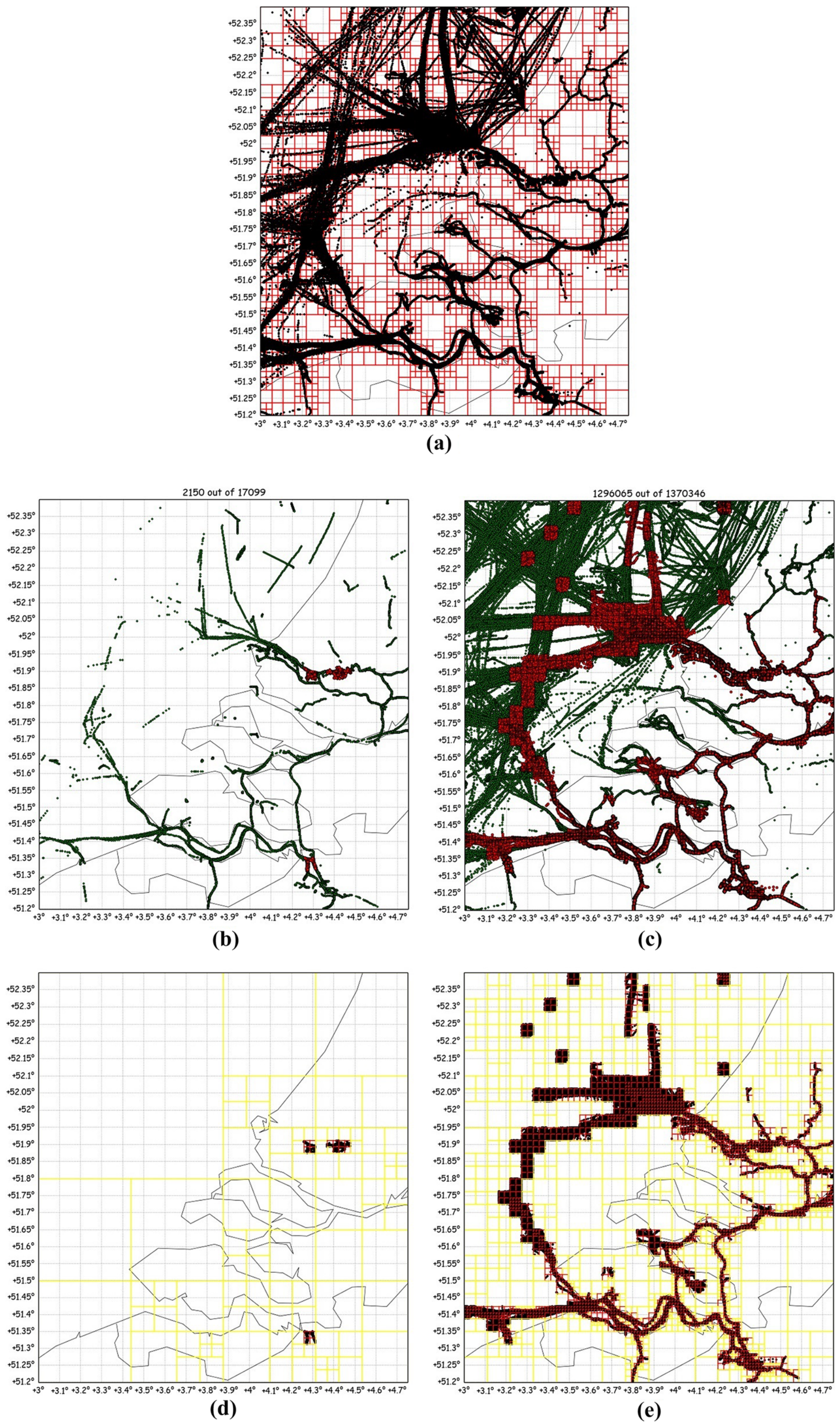


Fig. 6 Subdividing the region using a quad tree, and identifying high-density areas. Section **a** shows all subregions returned by the quad tree. In the bottom section **b**, **c** we show how this helps to reduce the number of points that will be processed later. Green and red points are AIS vessel positions. Green points will be ignored at the present density level, while the red ones are contained within high-density regions, and will be used. The images at the bottom **d**, **e** show only the points that the GA will see (color figure online)

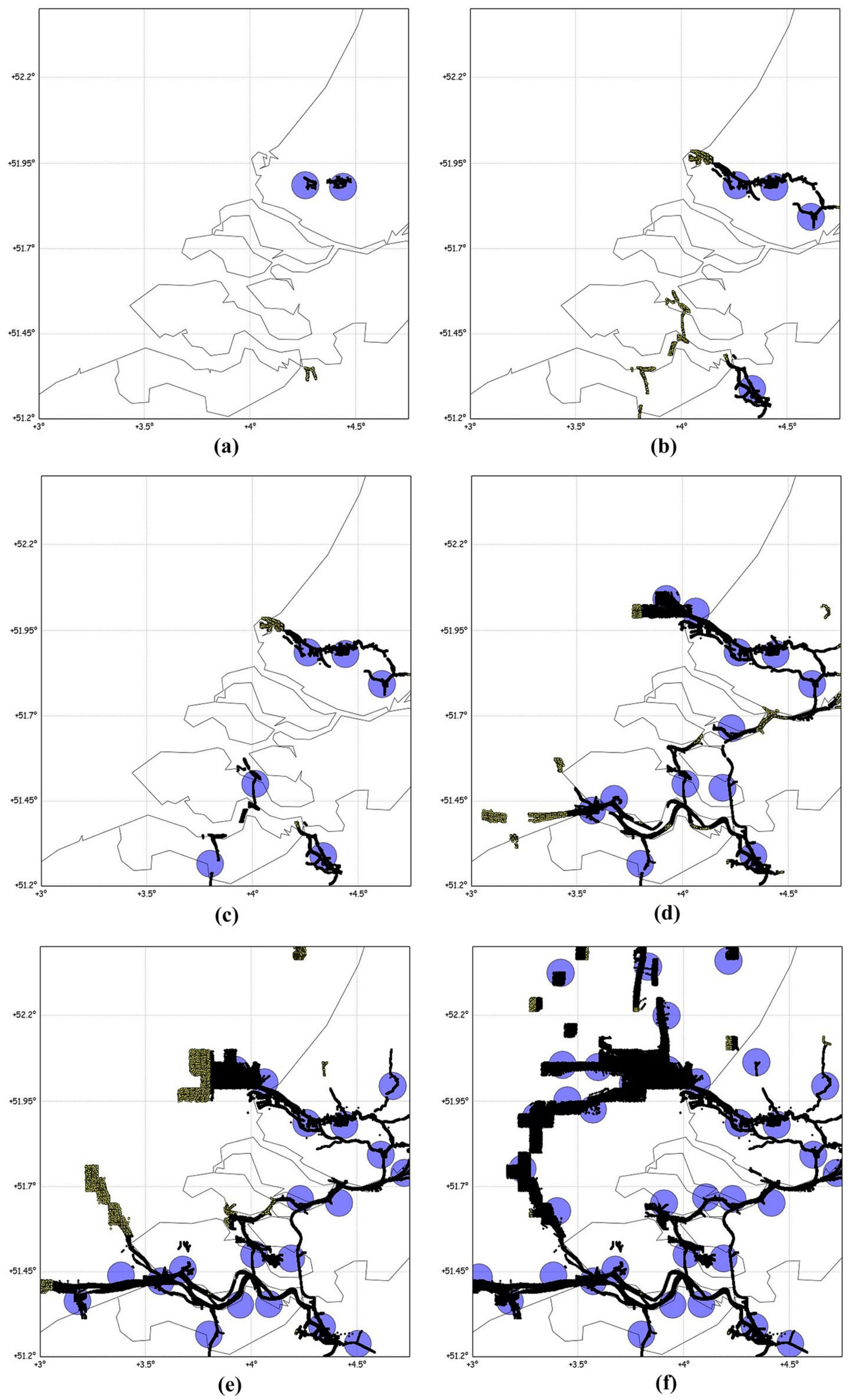


Fig. 7 Incremental waypoint discovery. Here we have six phases of the learning process. Discovered waypoints are shown as blue circles. Vessel points classified as part of a waypoint are depicted as black and they will be ignored from further GA epochs. The remaining points are shown as yellow dots on the map (color figure online)

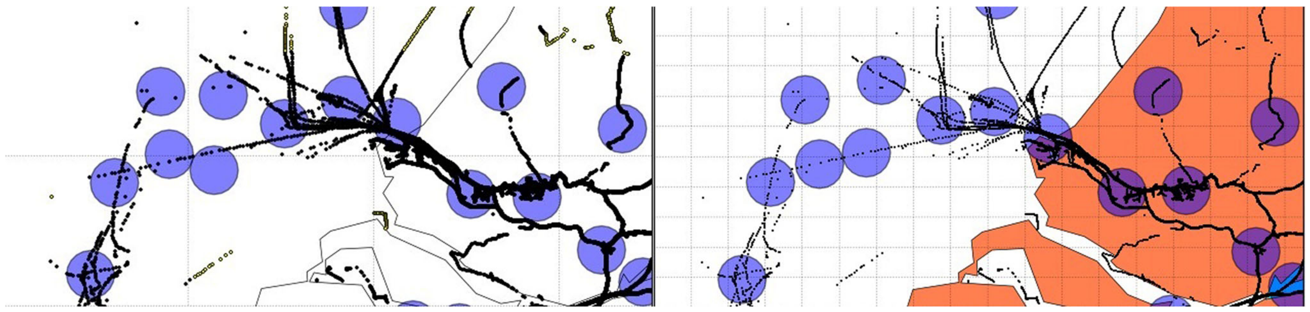


Fig. 8 Waypoints before and after fine tuning

given in Fig. 7. We start by loading 2.000.000 AIS points worldwide, and extract only those contained in the desired region, leaving us with 17.099. Using a quad tree this number is further reduced to 2.150, and sent as input for the genetic algorithm. The GA finds centers for two waypoints, and adds them to the solution, thus ending phase 0. For phase 1, 5.000.000 AIS points are being loaded, and the same steps are repeated, providing two more waypoints to the solution. Since there are still more points to be clustered within new pair, no additional data are loaded in phase 2. Instead, using the same data, another WP pair is extracted and saved. The output of phase 1 and phase 2 also shows how algorithm discards points in the WP vicinity, improving execution speed for all successive epochs of the GA. On the map two types of points are displayed. Black color is used for the points that are within the cluster or in its neighborhood, and as such, will be removed from all further considerations. Yellow ones are those not clustered, and will be the input for the next phase of the GA. These processes are being repeated until all AIS points are loaded, and waypoints are extracted.

Improving distance check function

The quad tree structure used to subdivide the area can be applied to reduce the number of distance check points in the fitness function of the GA. Instead of checking distance between the circle encoded in a gene, and all available points, a bounding box can be created around that circle, and then can be tested for intersection with regions in the quad tree. This returns only areas that intersected with the circle, and then the tree structure can be used to quickly retrieve the points from that area only. The benefit of this approach is that it reduces the search area for all distance checks, thereby reducing processing power used to calculate fitness.

Fine tuning waypoints using the center of mass

The second, trivial yet effective improvement, to find positions of the waypoints is to use their center of mass. When the genetic algorithm reaches its objective, we can use genes of the best entity to obtain waypoint centers. However, the fitness function in the present form can only find the center in the close proximity of the ideal one. Adding an additional constraint to make this possible, would only further decrease the execution speed. Therefore, this step is postponed until the genetic algorithm completes the evolution, and is used only on the best entity. We take all the points contained in the circle defined by a gene's parameters, separate them along x, and y axis and calculate the new center of a circle by dividing the sum with the number of points.

$$x_c = \frac{1}{n} * \sum\nolimits_{k=1}^{n} x_{p_k}, \text{ and } y_c = \frac{1}{n} * \sum\nolimits_{k=1}^{n} y_{p_k} \tag{11}$$

Figure 8 shows the readjustments of WP positions after applying the formula for the center of mass.

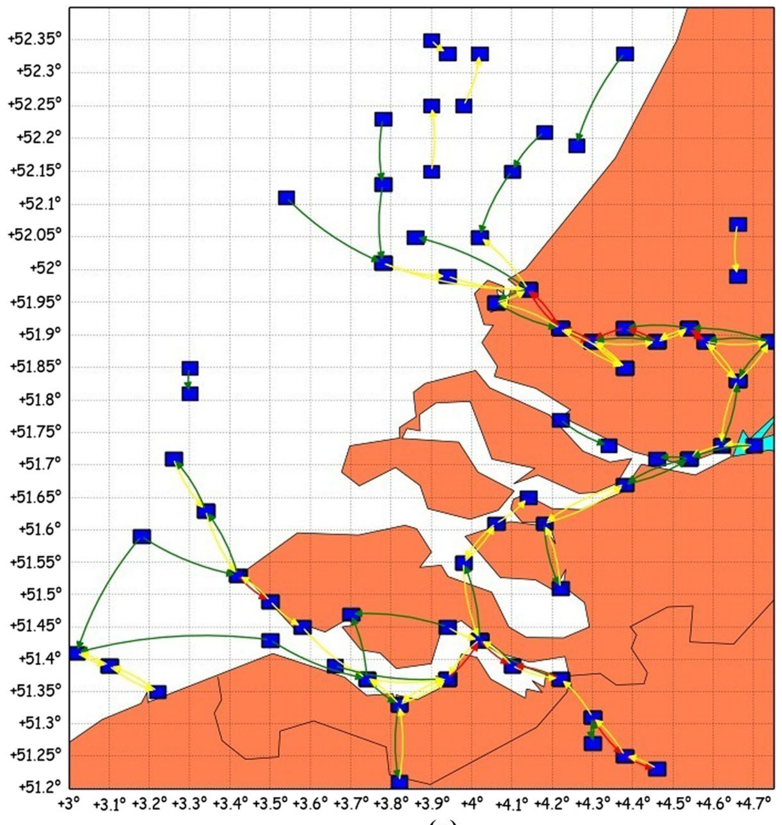
3.3 Route extraction

3.3.1 Creating the graph

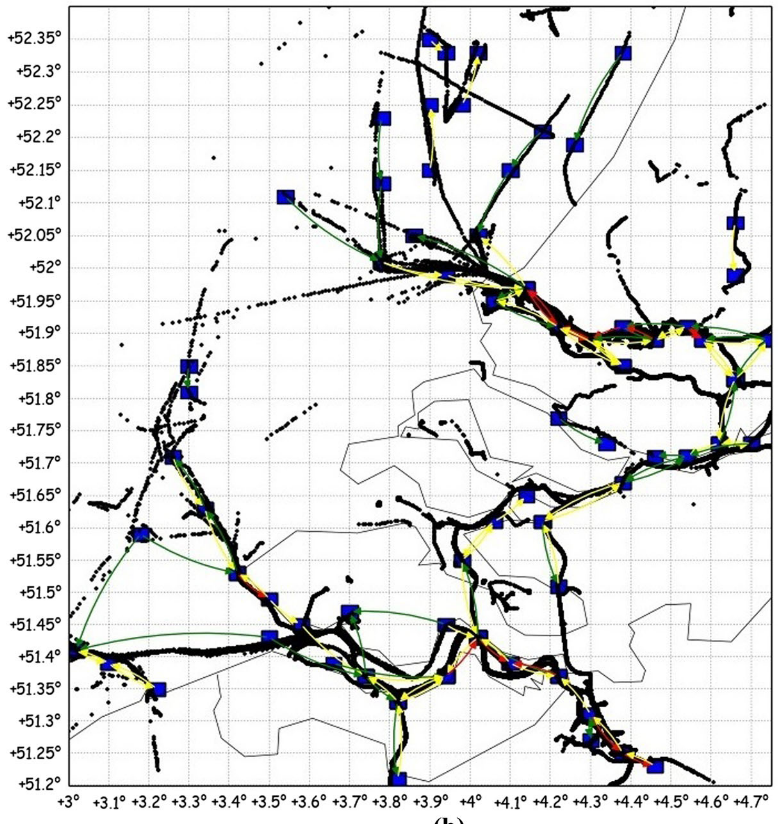
The process of route extraction begins when the GA completes the waypoint discovery. The objective is to create a directed graph. Nodes of this graph are waypoints provided by the GA, while edges still need to be discovered. This is a straightforward task of separating AIS data according to a unique identifier (MMSI) of every vessel. For each MMSI we check the sequence of waypoints the vessel travels through, and create edges between them. Every time a unique MMSI passes through two nodes, we increase the weight of the edge by one. An example of such a directed graph is given in Fig. 9a, while in Fig. 9b the same graph is drawn on top of AIS positional data used for the calculation.

Although waypoint centers are ideal to be converted to nodes, future enhancements require an alternative approach. Maritime routes are not as stable as the road, and railroad networks, and have the tendency to change, due to external factors such as weather conditions and tides. Our aim is to use this solution for future real-time route extraction from streaming data. This will cause some routes to appear, and disappear depending on external influences, and with them their nodes, and edges will also change status from active to inactive. The GA cannot guarantee that the same waypoint will be discovered more than once with the exact same center. To avoid losing history when nodes and their corresponding

Fig. 9 Extracted routes represented by the graph



(a)



(b)

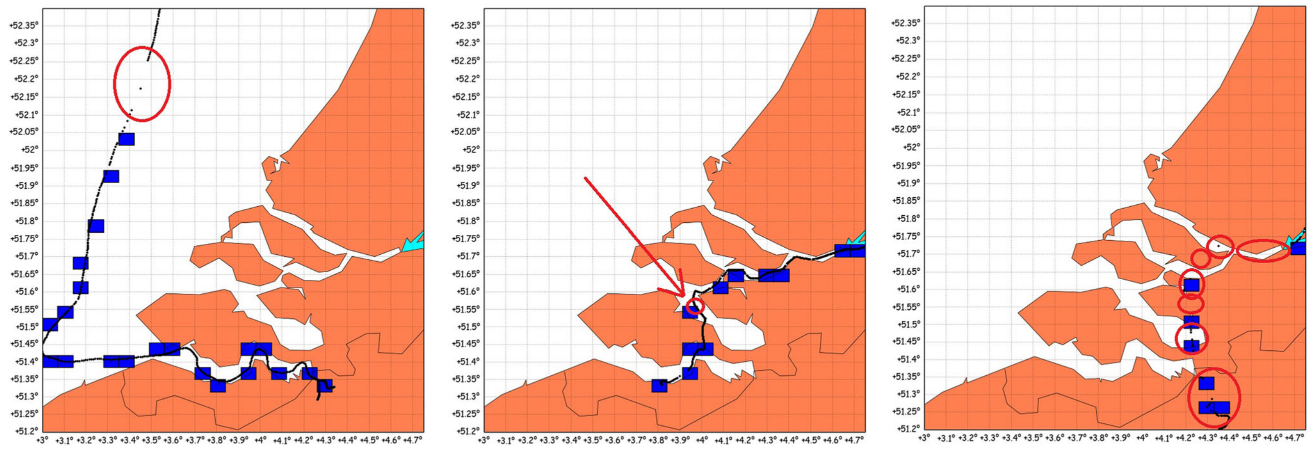


Fig. 10 Inaccuracies in AIS data

edges become inactive, we split the region into cells. The nodes of our graph will become cells that contain centers of waypoint circles. Thus, by partially sacrificing some precision, we are able to store all historical data for each cell. If a node has to be reactivated in the future, it will immediately connect with all active nodes it was connected with edges before.

3.3.2 Pruning and noise filtering

The graph in Fig. 9 was created on a reduced data set with negligible noise. The difficulty when working with AIS data, especially that obtained from free public sources, is the occurrence of noise, and inconsistencies. The genetic algorithm assumes that all inputs are correct. If the data are not clean, there will be a negative impact on the result, as false nodes, and edges are created.

When dealing with AIS data, it is also important to be aware that some vessels may turn off their AIS transponders, or radio stations occasionally fail to pick AIS signals sent over radio waves. This causes gaps in ship trip data. Figure 10 shows three unique vessel trips as received by the algorithm, together with nodes that get connected with this data. The trip on the left has one negligible data loss (marked by the red ellipse), but still allows the algorithm to create valid edges between the nodes. The second one, in the middle is also acceptable in terms of quality, although it is important to point out that this trip narrowly misses one node, due to the waypoint to cell conversion. The last one, on the right, is problematic, with large segments of trajectory missing. From those points that were available, the algorithm ran intersection tests with nodes, and added edges where the collision was true. Due to large segments missing, distant nodes will be incorrectly connected.

Real-time applications require to use data in its impure form. If we assume that data contains irregularities, but that their occurrence is low, we have to adjust the algorithm to handle them. In Fig. 11 we show the result of the same algorithm using all available data. In Fig. 11a, we show the waypoints on top of positional data, as discovered by the GA. Next to it (see Fig. 11b), we see the route graph, which contains an unacceptable level of wrong connections due to inconsistencies in the AIS data.

Solving this problem requires postprocessing. First, we store average weights of all edges. If a node is connected by more than one edge, and if one of the edges has a weight with very low value, then that edge is treated to be the result of noisy data, and is removed. In the image, the color shows the weight range for every edge. Blue: 50 or more, Red: 10–50. Yellow: 2–10. Green: 1. Another way to remove faulty routes is to reduce the weight of all edges by a given small value (1–3). This leads to the deletion of some of the wrong connections. Figure 11c, d shows the same graph after noise filtering. On the left we show the new graph with nodes and edges, while on the right, the same graph is drawn with the input AIS positions on top. We can conclude from the picture that all major routes are still contained in the graph after filtering.

The last step in the postprocessing, and route extraction is graph pruning. In Fig. 12a, we zoom on one segment of the previously extracted graph. In the path from A to E and vice versa, nodes labeled B, C and D are only intermediaries, and, as such, can be removed without losing route information. In Fig. 12b, we can also find a one way route from A to G via F. Since F is only a transit node, it can also be safely removed. The result is shown in Fig. 12c. Utilizing these steps, we make the graph easier to read.

3.3.3 Handling missing region data

In the real case implementation, it is not possible to assume perfect coverage of the entire region. While AIS receivers

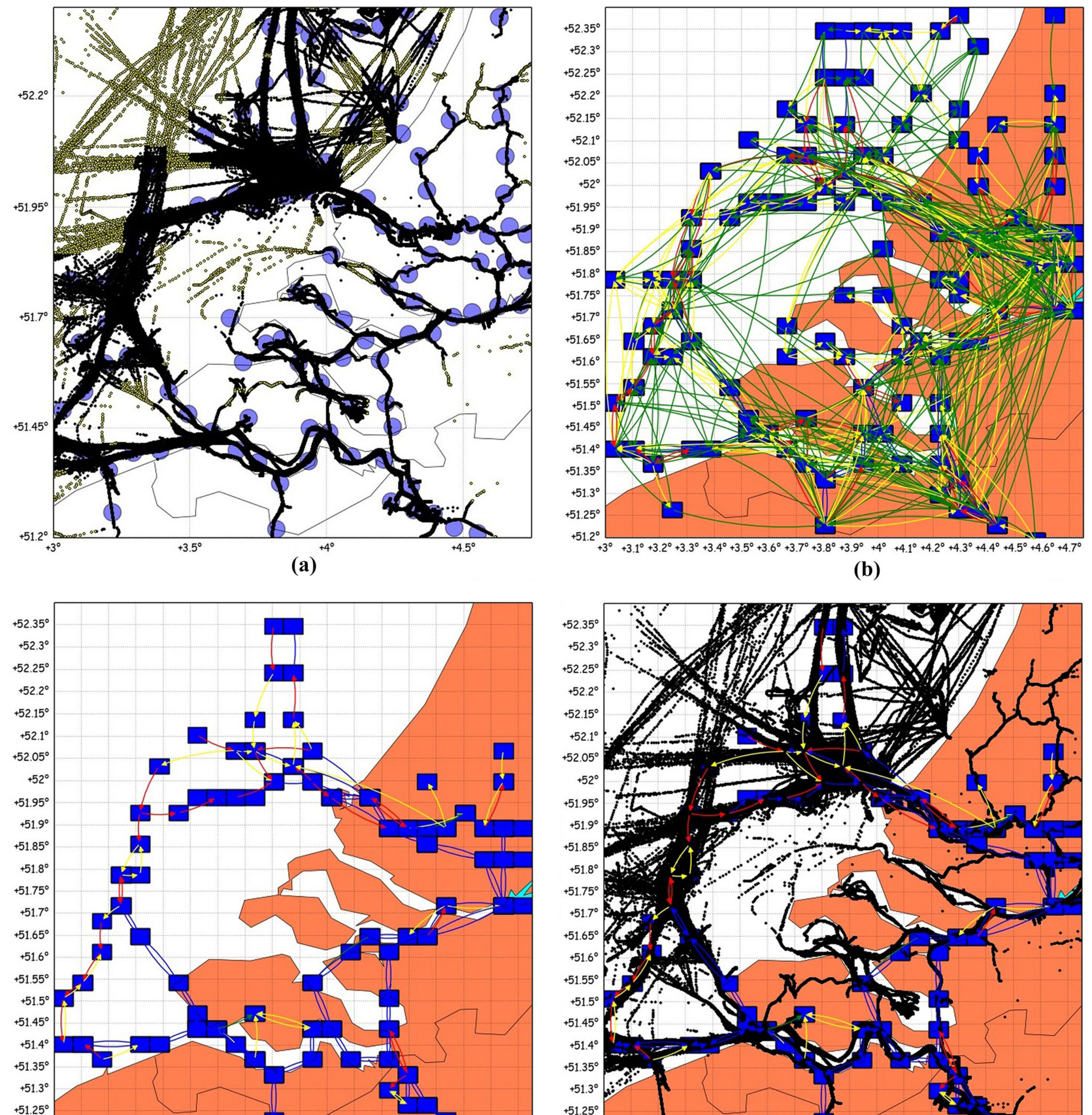


Fig. 11 Postprocessing—noise filtering

typically cover coastal regions well, the high seas areas can be beyond the reception range. Consequently, AIS data broadcasted by the vessels will not be recorded.

Depending on the location of the missing data, it is possible to minimize the impact of the missing data. The question this solution has to answer is what routes a vessel may take based on the current position, and the corresponding end points of the voyage. Hence the resulting graph must do the same. Let us consider an example where a ship can sail from waypoint M to waypoint N either visiting waypoints A and B, or taking a route through waypoint C (see Fig. 13a). The shape of the routes between the waypoints is not important (e.g., it can be either a straight line, a curved path avoiding an island, or any other shape), as long as the main charac-

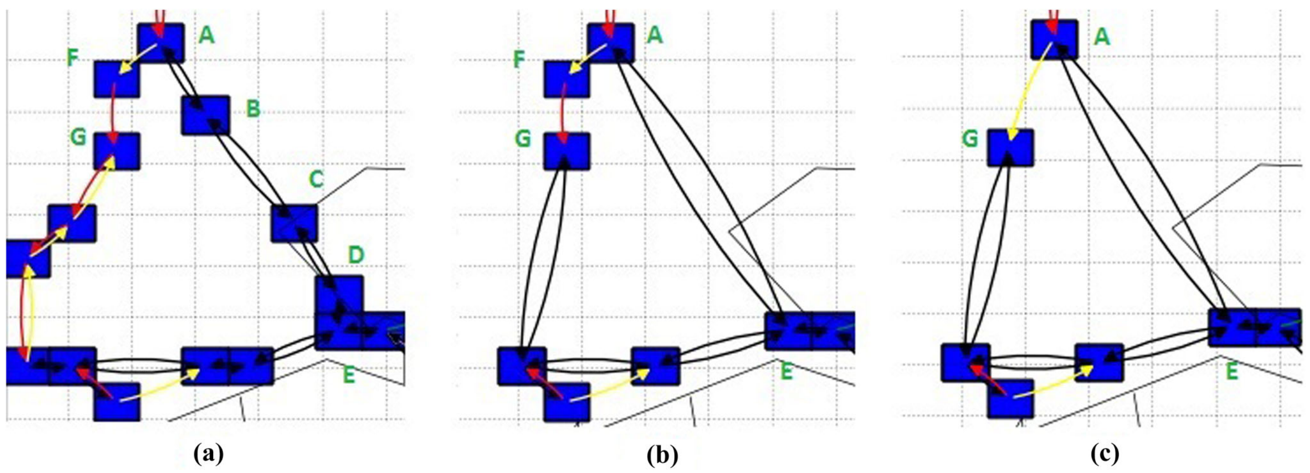


Fig. 12 Graph pruning

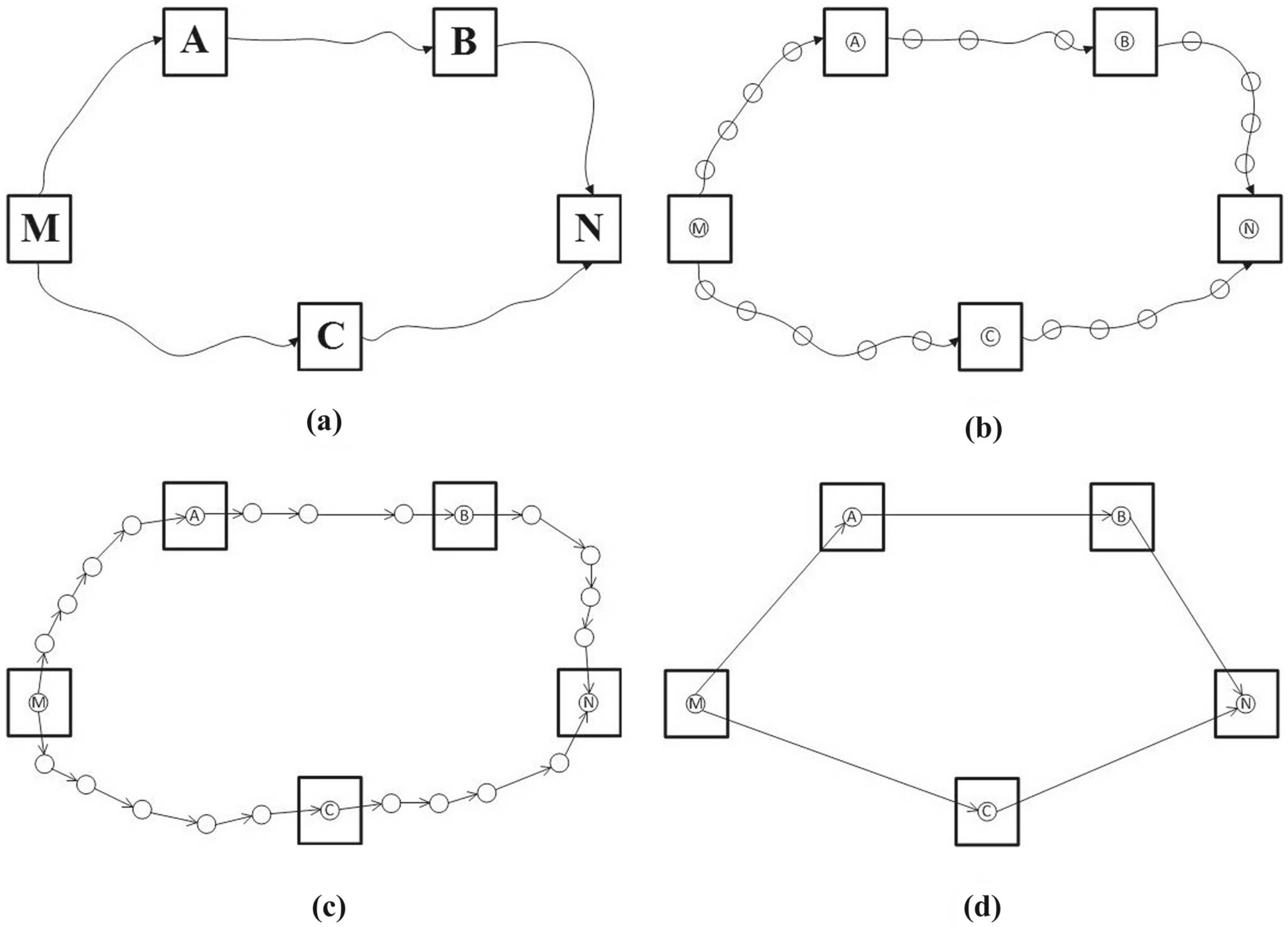


Fig. 13 Graph extraction from complete data set. Section **a** shows an arbitrary route connecting two points M, and N, via points A, and B, or point C. Section **b** gives the illustration of GA fitting waypoints for routes. In **c** waypoints are connected with edges to form a directed graph. Section **d** shows the same graph after filtering, and pruning

teristics related to average speed, and sailing time can be maintained. Therefore, if there are data segments missing from the high seas, the number of waypoints extracted will be reduced. Yet, as long as the starting, and ending points are contained, the algorithm is capable of extracting the possible routes. The first example will assume that the algorithm

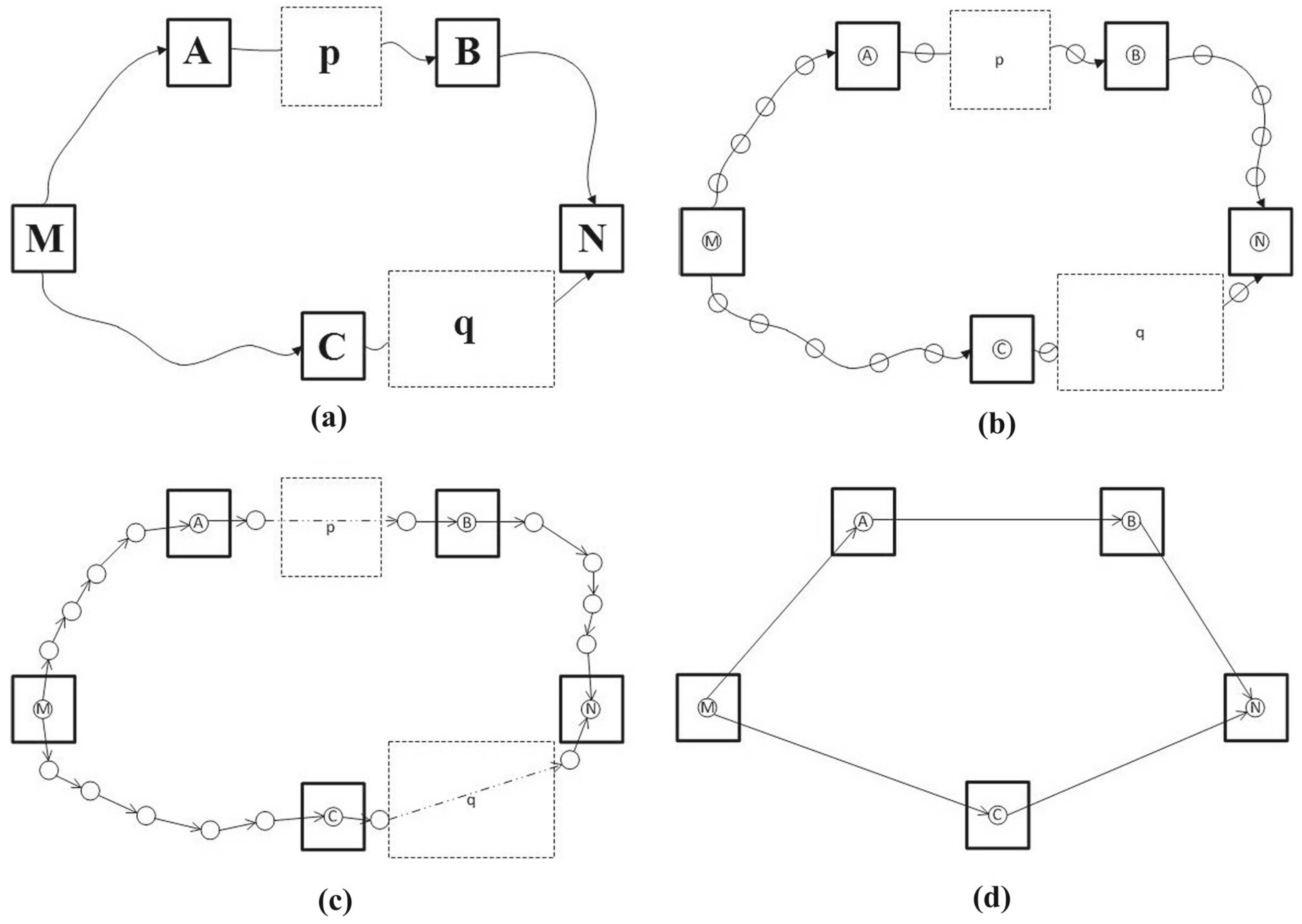


Fig. 14 Graph extraction with missing data points. Section **a** shows an arbitrary route between points M, and N with areas p, and q from which no AIS data are received. In **b** waypoint discovery is illustrated for this example. Section **c** shows how the algorithm connects nodes with edges. Section **d** shows the result after filtering, and pruning

uses the complete data set. In Fig. 13a, we can identify two arbitrary routes where a vessel can sail from M to N via the route M–A–B–N, or using the route M–C–N. The next one, Fig. 13b shows waypoints discovered by GA, and in Fig. 13c those waypoints are connected, and form a directed graph. After removing transition nodes the graph will look like the depiction shown in Fig. 13d.

The next picture (Fig. 14) uses the same example with addition of two blank areas p, and q. The data that the algorithm will receive, including the missing regions is shown in Fig. 14a. The discovery process is the same as described above, with Fig. 14b showing waypoint discovery, Fig. 14c the process of connecting the waypoints to form the graph, and Fig. 14d the final result after removing the transition nodes. Comparing Figs. 13b, and 14b we can see that the second one will not be able to discover waypoints in the missing regions p, and q. This will have some impact on the graph creation, as the algorithm will use vessel data to connect the waypoint where the vessels were last seen with the first one that follows, hence the shape of the two graphs will be different. However, since the blank regions did not contain the areas A, B or C, the graphs will look the same after the pruning process, which removes transition nodes.

The last example (Fig. 15) illustrates the most difficult case. Here blank regions p, and q overlap areas A, and C. The waypoint discovery Fig. 15b, and the graph creation Fig. 15c will be handled as in the previous examples. Due to the fact that no waypoints could be discovered in A and C, the algorithm will connect the nodes for which there are data, and that will impact the shape of the graph result after the pruning is complete, which is shown in Fig. 15d.

Using these illustrations we demonstrate the potential of this solution to reconstruct the route in relation to the available data. As long as the blank regions do not occlude key waypoints, we can assume that it is possible to preserve key route information. Since these data gaps typically occur in high seas, we can use this approach to discover main route characteristics even without an ideal input set. Also, it is

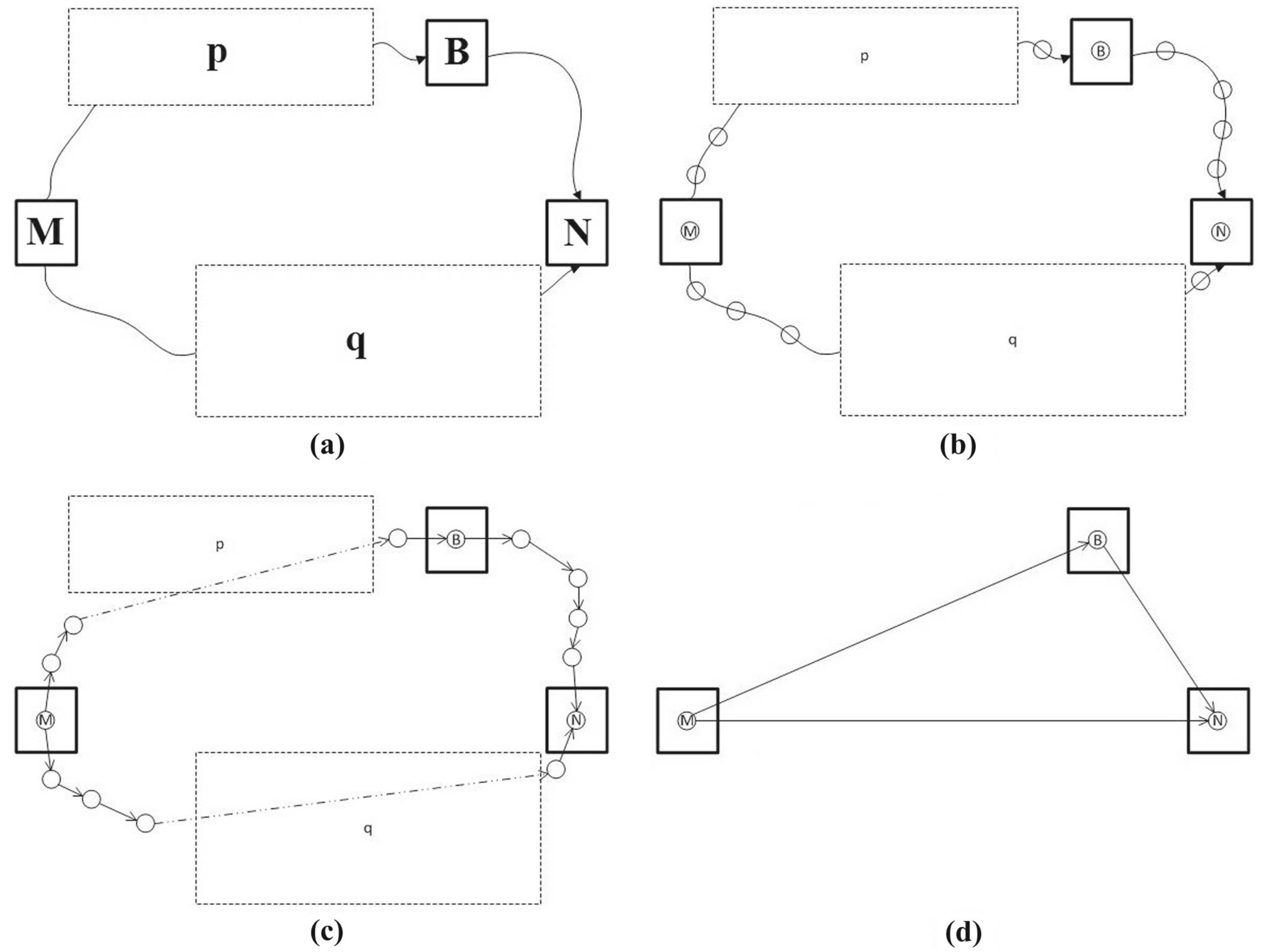


Fig. 15 Graph extraction with blank regions overlapping key areas. Routes are given in (**a**), waypoint discovery in (**b**), the graph creation in (**c**), and the final result after filtering and pruning in (**d**)

important to note that these examples assume that there may be other routes leading to, or from the areas A, B and C. If that was not the case, then these areas would also be removed by the algorithm as transition nodes, resulting in a straight line from M to N.

4 Testing and validation

4.1 Execution time

The GA enhancement steps described in the solution design are made to enable the genetic algorithm to efficiently process large data volumes, which is typical for real life industrial applications. Therefore, we measure the time it takes for the improved algorithm to discover maritime waypoints and compare it with the execution time of the standard approach.

Since these two algorithms process data in a different manner, we design the test and set up algorithm parameters such that we can measure execution time of every epoch in similar conditions. For the time test, we set the area to the coordinates of South Holland province. The population size is set to 50, mutation is 1%, and both algorithms are capped at 100 epochs. The standard algorithm is tested with a chromosome length of 20. In the case of the improved algorithm, it is set to run 20 phases.

We run the standard genetic algorithm three times, using data sets containing 2150, 165,200, and 591,300 points. With each epoch, as the algorithm converges toward the solution, the number of intersection tests between WP, and points increases, slowing down the execution time. In direct relation to the formula (8), adding more points has a considerable negative effect on the performance as shown in Fig. 16. This reveals the limitation of the standard approach, as with the increase in points increases the requirement for additional WPs. In terms of the GA that means that the number of chromosomes also has to increase, resulting in exponential surge of the required time per epoch.

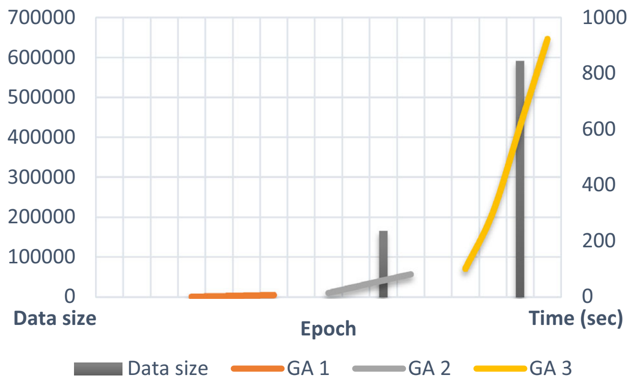


Fig. 16 Execution time per epoch in relation to data size. Time in seconds per each of 100 epoch is shown as the line on top of the bar depicting the number of points the GA is processing

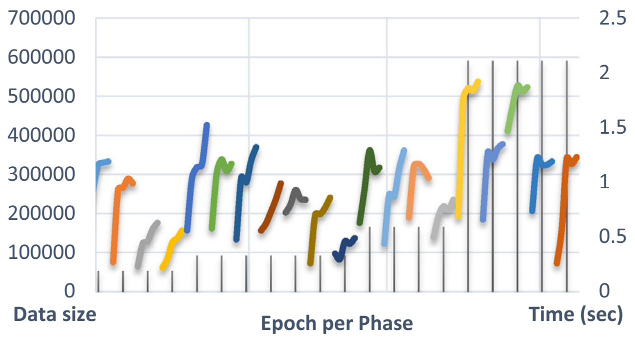


Fig. 17 Epoch execution time of the enhanced genetic algorithm. The algorithm is running 20 phases. The number of points used per phase are given as gray bars. Above, every phase also contains the curve depicting the execution time for each of the 100 epochs ran by the GA in that phase

Next, we focus on the performance of the enhanced solution. In Fig. 17, we show by means of a scatter plot the execution time in seconds for all 20 phases, each running 100 epochs. Associated with each phase is the bar plotted value showing the number of points loaded. From the chart we can see that the time for each epoch stays within the interval 0.5–2.0 s. While the execution slows down throughout the epochs (which is typical for the GA), we notice that switching to the next phase enables the algorithm to omit the points used in the fitness check, and maintain a steady execution time. The chart highlights the main positive characteristic of the improved genetic algorithm, namely the ability to handle the increased amount of data without a significant impact on the overall performance.

4.2 Extraction quality

The second essential test is the extraction quality of the algorithm, i.e., the extent to which it accurately represents maritime routes. To make that assessment in a real case environment, we can only rely on the visual comparison. Therefore, we choose to test the algorithm on the area corresponding to two Dutch provinces, while focusing on inland waterways. A good algorithm will generate the graph corresponding to the map of rivers and canals in these two provinces. If there is an edge across nonexistent waterway, that would be a clear indication of the poor extraction quality.

In Figs. 18, and 19, we show: the AIS positional data used for the experiment with the discovered waypoints, extracted graph after pruning, and the noise reduction, the same graph on top of positional data, and the actual waterways map for the region. The South Holland was chosen as the province with the heaviest inland traffic, and many intersections, while Overijssel was chosen as its opposite. In both cases we do not identify any inconsistencies such as routes going through the land areas, and the main routes from the map are also present in the graph. Therefore, we conclude that the algorithm produces an acceptable/correct route representation. However, we note the presence of noise in closely positioned waypoints, mostly attributed to the inaccuracies in input data, as shown in Fig. 10. In Test 1—Overijssel, there are some routes that the algorithm does not identify, but this not a defect of the algorithm: it is due to the fact that there are narrow canals, on which commercial vessels that are required to broadcast AIS signals, cannot sail (Fig. 18, blue dotted lines).

For the actual maps we used the online source from the *"Waterrecreatie Nederland"*[18].

4.3 Simulation with complete and incomplete AIS data

The extraction quality test from the previous section gives an indication that the algorithm performs as expected, but it does not allow us to quantify its quality, and compare it with other approaches. Due to the fact that different algorithms use different AIS data sets that are not publicly available, we are not able to compare the performance of this solution with others under the same conditions. To lessen this drawback, and allow future comparisons we propose to use a set of simulated AIS data with enumerated routes.

We use the same test used in [3] containing four overlapping routes, two horizontal and two vertical (see Fig. 20). This configuration allows for the clear comparison of the extracted routes with the simulated ones. The routes are labeled a, b, c and d. For every route, the simulation generates new vessels at both ends. The simulation runs for a given number of intervals, and during each one, the vessel position is updated, and new vessels are generated at route ends. The vessel trajectory is given as a vector with direction toward the opposite end, and the magnitude of the vector is set according to the predefined lane specific parameter indicating the average speed. To prevent all vessels having the exact same path, another vector is added representing the deviation from the path. The movement delta per every interval is calculated as the sum

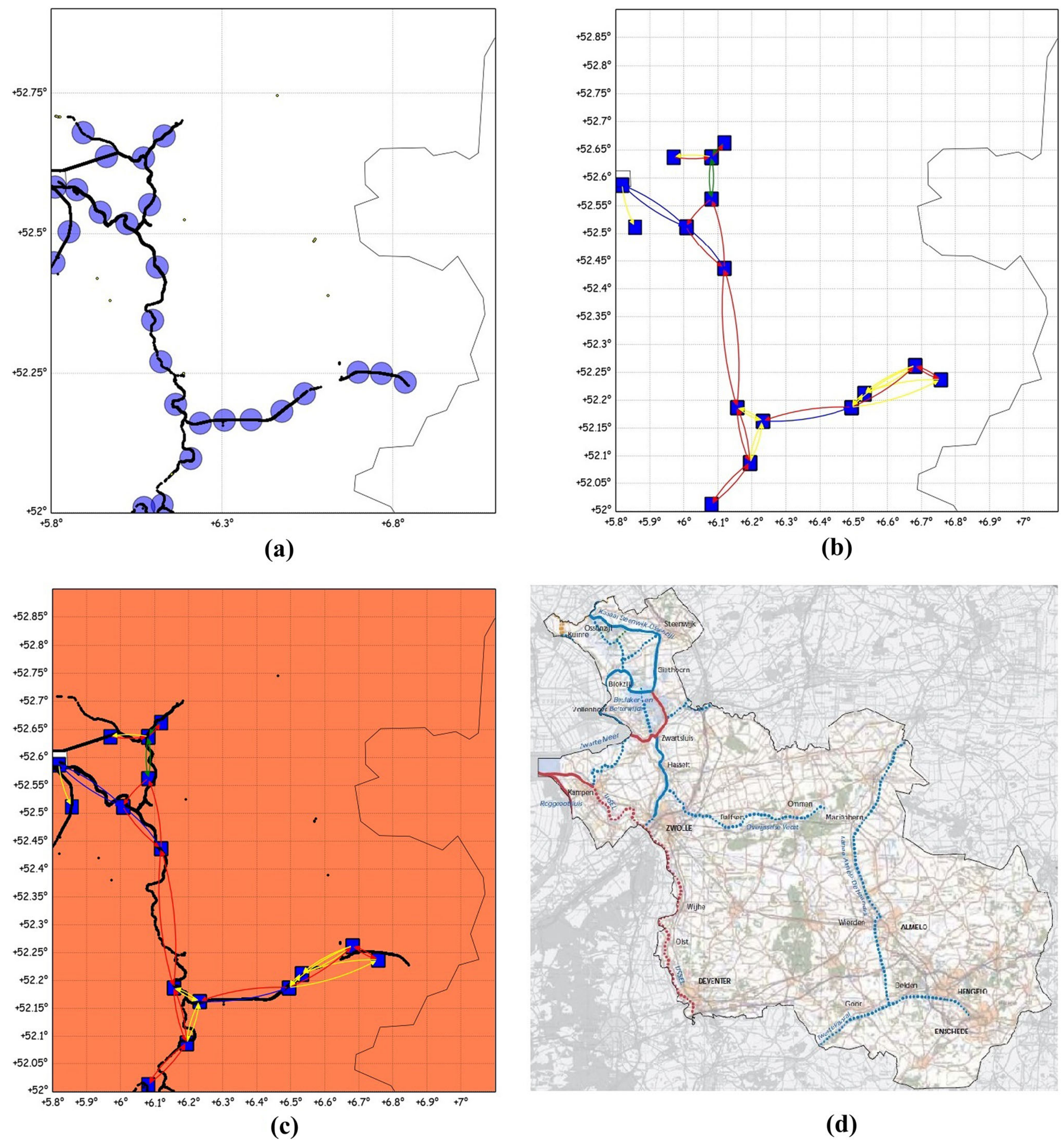


Fig. 18 Test 1—Overijssel. Starting from **a** AIS data and waypoints discovered from it, **b** shows the directed graph representing extracted routes, **c** shows the same graph plotted on top of the loaded AIS points, and **d** the actual waterways map for the province

of these two vectors. Since we want to test our algorithm on lanes that have different traffic density, the lane b is set to generate five times more vessels per interval than the lanes a, and d. The lane c represents a less travelled route generating one third of the traffic from the lanes a, and d. The average speed of the vessels is the same for lanes a, b, and c, while ships in lane d are set to move twice as fast. The intersections between the routes are labeled as capital red letters A, B, C, and D.

We run the simulation, setting it to generate 8000 ships over the interval of 200 time units. The position of every vessel is stored and used by the following experiments. Since the

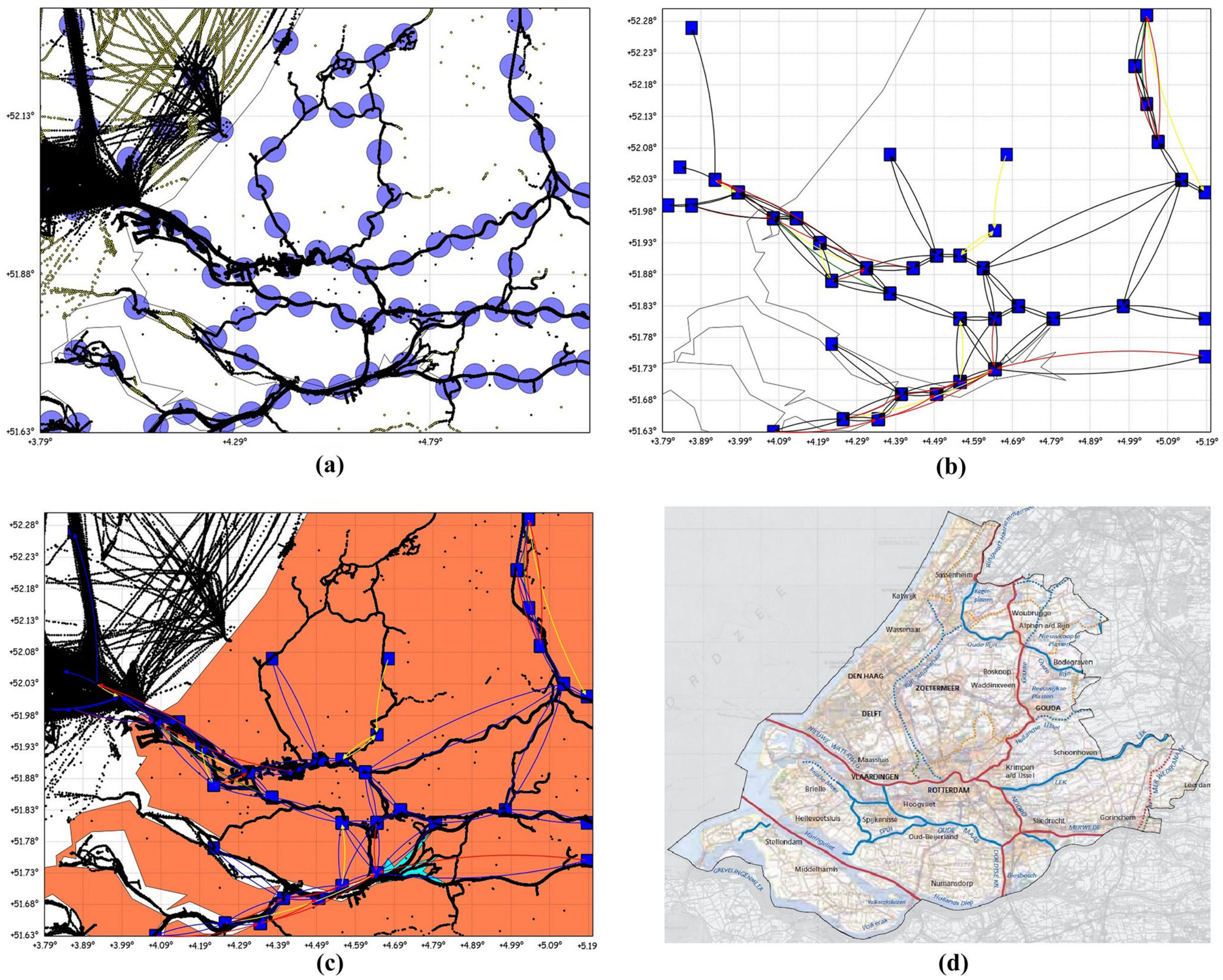


Fig. 19 Test 2—South Holland. Starting from **a** AIS data and waypoints discovered from it, **b** shows the directed graph representing extracted routes, **c** shows the same graph plotted on top of the loaded AIS points and **d** the actual waterways map for the province

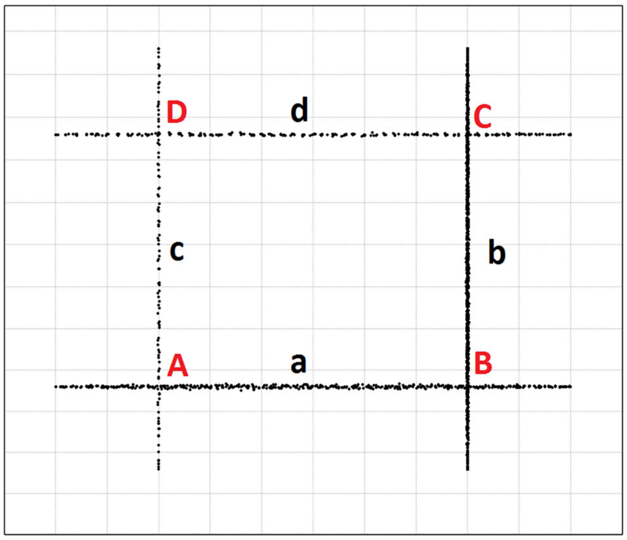


Fig. 20 Simulated AIS data for the benchmark test

test presented in the Execution time section showed that the enhanced GA solution offers much higher execution speed with the same or better extraction quality, the standard GA was ignored, and all tests were performed on the enhanced GA. The algorithm was run ten times, using 100, 800 and 2000 epochs. For the same number of epochs the test was repeated three times. The first, and the second run had population, and elite size set to 50 and 2, respectively. The third one has those values doubled: 100 and 4. After these nine tests, we run one more test with significant increase on mutation size to 20%. To evaluate the algorithm we calculate the number of correctly extracted lanes in comparison with the total number of lanes, and do the same for the intersection points. The final score is the average of these two values. The results are given in Table 1

First we set our attention to the route accuracy. Based on the scores, we conclude that with sufficient number of

Table 1 Simulation results

No.	Input data				Parameters				Results				
	Simulation	Routes	Intersections	Algorithm	Population size	Elite size	Mutation (%)	Max. epochs	Routes	Correct (%)	Intersections	Correct (%)	Total (%)
1	SIM 1—complete data	4	4	Enhanced GA	50	2	8	100	3	75.00	1	25.00	50.00
2	SIM 1—complete data	4	4	Enhanced GA	50	2	8	100	2	50.00	0	0.00	25.00
3	SIM 1—complete data	4	4	Enhanced GA	100	4	8	100	3	75.00	0	0.00	37.50
4	SIM 1—complete data	4	4	Enhanced GA	50	2	8	800	4	100.00	3	75.00	87.50
5	SIM 1—complete data	4	4	Enhanced GA	50	2	8	800	4	100.00	1	25.00	62.50
6	SIM 1—complete data	4	4	Enhanced GA	100	4	8	800	4	100.00	3	75.00	87.50
7	SIM 1—complete data	4	4	Enhanced GA	50	2	8	2000	4	100.00	3	75.00	87.50
8	SIM 1—complete data	4	4	Enhanced GA	50	2	8	2000	4	100.00	3	75.00	87.50
9	SIM 1—complete data	4	4	Enhanced GA	100	4	8	2000	4	100.00	3	75.00	87.50
10	SIM 1—complete data	4	4	Enhanced GA	100	4	20	2000	3	75.00	0	0.00	37.50
11	SIM 2—blank regions	4	3	Enhanced GA	50	2	8	100	3	75.00	0	0.00	37.50
12	SIM 2—blank regions	4	3	Enhanced GA	50	2	8	100	3	75.00	1	33.33	54.17
13	SIM 2—blank regions	4	3	Enhanced GA	100	4	8	100	3	75.00	0	0.00	37.50
14	SIM 2—blank regions	4	3	Enhanced GA	50	2	8	800	4	100.00	2	66.67	83.33
15	SIM 2—blank regions	4	3	Enhanced GA	50	2	8	800	4	100.00	3	100.00	100.00
16	SIM 2—blank regions	4	3	Enhanced GA	100	4	8	800	4	100.00	1	33.33	66.67
17	SIM 2—blank regions	4	3	Enhanced GA	50	2	8	2000	4	100.00	3	100.00	100.00
18	SIM 2—blank regions	4	3	Enhanced GA	50	2	8	2000	4	100.00	2	66.67	83.33
19	SIM 2—blank regions	4	3	Enhanced GA	100	4	8	2000	4	100.00	2	66.67	83.33
20	SIM 2—blank regions	4	4	Enhanced GA	100	4	20	2000	3	75.00	0	0.00	37.50

epochs, the enhanced GA handles route extraction very well. The tests 1–3 were run with setting the limit to only 100 epochs. As this number of epochs is typically insufficient for the GA to converge to a good solution, we use it to compare how the accuracy increases with the increase in the epoch limit. Decreasing the number of epochs allows for fast execution, yet lowers the quality. However, due to preprocessing provided by the quad tree structure, the algorithm was able to quickly converge waypoints to the lanes, and correctly identify 2 or 3 out of 4, scoring between 50–75% for this part. Upon close inspection of the missed routes, we found the spread of waypoints to be correct in the large segment of the lane, yet the low number of epochs prevented the algorithm to accurately extract waypoints near the starts, and the ends, hence impacting the score. All tests (1–10) show that increasing the population size has negligible effect on the overall score, and this parameter is removed from further considerations. With tests 4–9, increasing the number of epochs to 800, and finally to 2000, allowed the algorithm to fit all the waypoints, and correctly identify all the lanes. From this we conclude that the further increase in the number of epochs would only decrease the execution speed without providing improved precision.

The intersection points proved to be more challenging for the GA. Although tests 4–9 had 75% score, not a single configuration managed to achieve 100% precision. This is due to the fact that the AIS point density within the intersection was not significantly greater than the density of the points in the neighborhood, hence in all of those tests the GA tended to be stuck in the local maximum, close to the optimal. Yet without a high mutation value, could not reach it. Test no. 5 illustrates this well, with the number of accurate intersection detections dropping to only one, due to the fact that the close neighboring waypoints were discovered first. The final test (no. 10) was performed to check the algorithm's behavior with a large mutation value, and see if that would improve, or decrease the ability to detect the intersection points. As shown in Table 1, where the final score for the test 10 dropped to 37.5%, a large mutation value in the long run negatively impacts the solution contained within the elite chromosomes, eventually preventing the genes of the best entity to be preserved for future epochs. This causes accuracy to drop in both lane, and intersection detection. Balancing between the speed, and the extraction quality we find that the algorithm performs the best by using the parameters given in test 4. The graphical representation of the input data, waypoint extraction, graph creation, and the final result after filtering, and pruning for test 4 is given in Fig. 21, which scored 87.5% on our test. To explain the missing 12.5%, we zoom in on the intersection point B (the lover right one) in Fig. 21b, c. We can see how GA narrowly missed the intersection, and the impact it had on the final result Fig. 21d, even though the routes were accurately extracted.

The tests are repeated again with the same parameters, but instead of the complete data, we use the same simulation, and occlude two regions. We pick the regions in such a way that one occludes part of the lane a, and the other the intersection point C. Table 1 contains the parameters, and results for the tests 11–20. The simulation input data, after introducing the blank regions, are shown in Fig. 22a. As with the tests 1–3, small epoch limit of 100 in tests 11–13 prevents the algorithm to find a good solution, yet the preprocessing of quad tree still allows for a 75% accuracy of the route extraction. From 800 to 2000 epochs (tests 14–19) the solution performed with an incomplete data set in a similar fashion as in the case with the complete AIS data. There is a good extraction result for lanes, and intersections are either correctly identified, or there is a narrow miss attributed to the algorithm being stuck in the local maximum in the vicinity of those points. This time the genetic algorithm was more successful in identifying the intersection points, but this is attributed to the random choice of picking initial waypoints, and cannot be guaranteed. Test 20 with a high mutation again showed that increasing this parameter has considerable negative effect on the score.

The main point of tests 11–20 is to see how the GA handles missing data. Comparing Figs. 21 and 22 and the scores in Table 1, we can see that there is no performance drop caused by the blank region overlapping a part of the lane a. The variations in score are solely due to the stochastic nature of the GA. The waypoint C could not be detected as there was no input data in the vicinity, yet in this example it didn't have a significant influence on the results. Figure 22 shows the input data set, waypoint discovery, and graph creation for the test 15. From the presented experiments, we may conclude that the enhanced GA provides a robust solution to extract maritime routes in a real case scenario using standard PC hardware.

5 Related and future work

5.1 Future requirements of the synchromodal platform

The solution presented in this paper shows how to perform maritime route reconstruction using a genetic algorithm. Although related research such as [4,12,19] also deal with route extraction, and [20] can also provide up to 60 minute prediction in milliseconds, our decision to use an evolutionary approach is based on future needs of the Synchromodal IT platform.

The ideal Synchromodal IT platform should be able to handle data from large waterways areas, such as entire Atlantic ocean, and be able to estimate vessel arrival times 48 hours in the future. The current process of collecting, and storing AIS data, the available computational power, and the

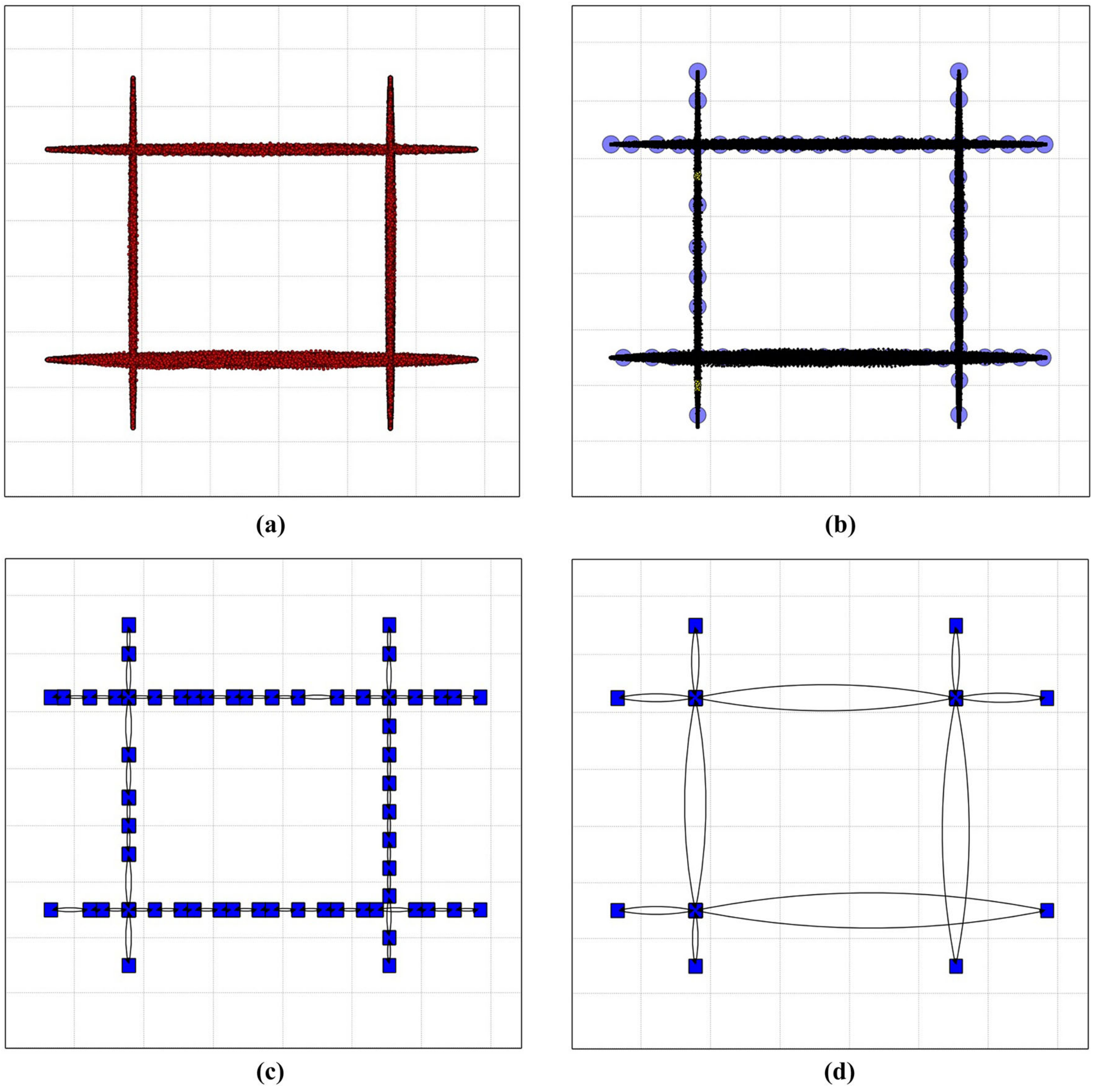


Fig. 21 Simulation with complete data set. Section **a** shows AIS points used as input. Section **b** shows the final result of GA fitting the waypoints. In **c** waypoints are connected to form the directed graph. Section **d** shows the same graph after filtering and pruning

extant algorithms' efficiency are limiting factors to achieve such an objective. However, working toward this scenario imposes that the following requirements and limitations must be taken into account:

- The solution must handle large volumes of streaming AIS data in real time,
- Preferred collection method for AIS data is through a range of independent costal receivers, limiting visibility, and causing periodic blank areas,
- The AIS data comes from different sources, and is typically fragmented,
- As sailing patterns away from harbor areas change due to the influence of external factors such as the weather, the result from the algorithm with the corresponding predictions must be capable to adapt to these changes in real time.

The current state of the art approaches treats input AIS data on an "as is" basis. The area under consideration is fully covered by AIS receivers, and the collected vessel data is deemed complete. There are no missing regions. On top of that, the patterns are extracted from the coastal areas where the route changes due to weather factors are not observed. In such conditions, the genetic algorithm can be outperformed both in terms of precision and speed. To the best of our knowledge, there is no solution that is addressing the problem of maritime route extraction between points including partially occluded

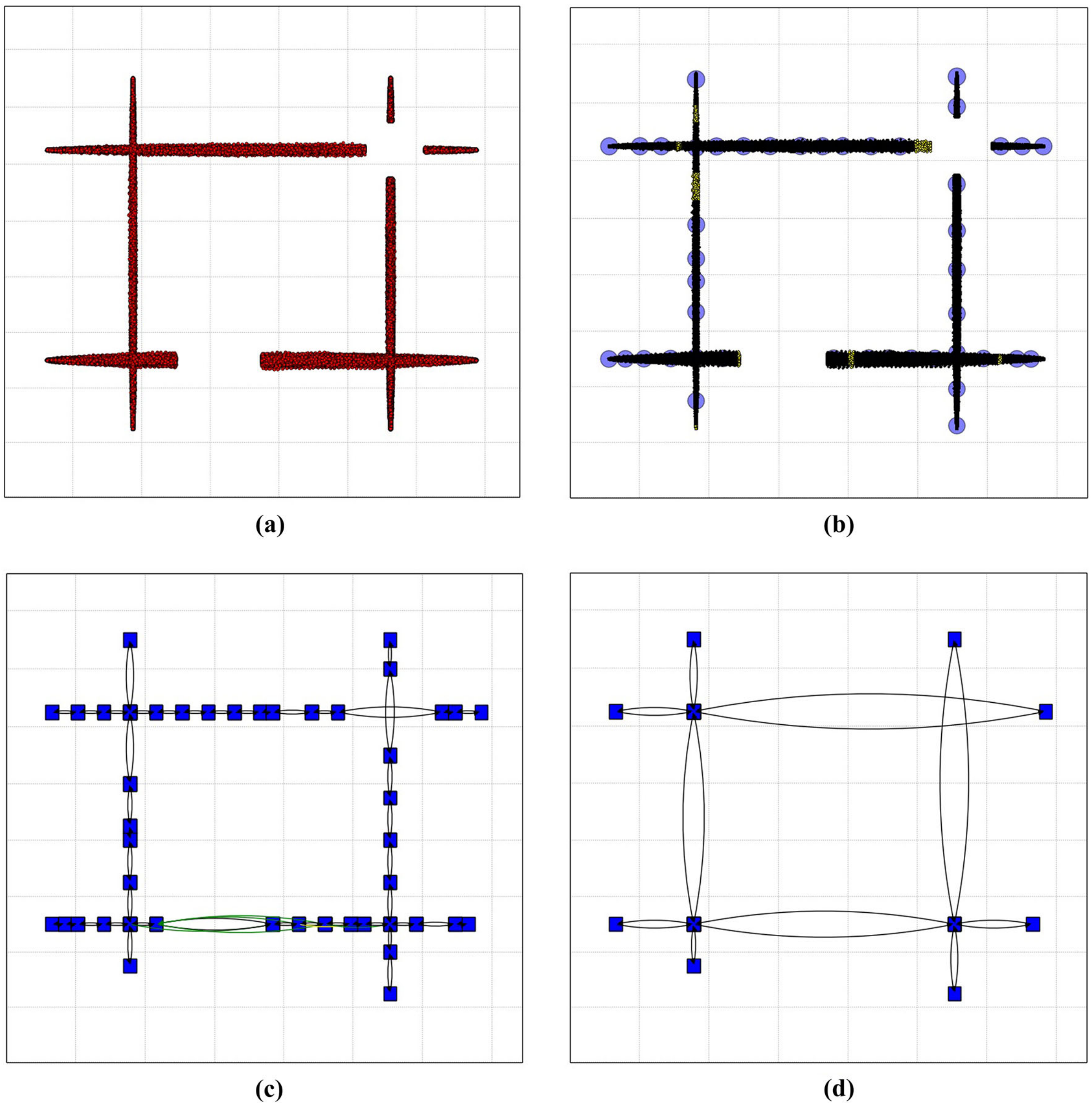


Fig. 22 Simulation with two blank areas occluding the data set. Section **a** shows AIS points used as input. Section **b** shows the final result of GA fitting the waypoints. In **c** waypoints are connected to form the directed graph. Section **d** shows the same graph after filtering. and pruning

areas. As demonstrated on waypoint extraction in [3], clustering solutions such as DBSCAN that contain no memory, are highly vulnerable, and intolerant to missing regions. The novelty of our approach lies in providing the means to discover maritime patterns when dealing with imperfect data, typical for a real case business environment.

Additionally, the requirements of the future Synchromodal IT platform suggest that the improved solution must not only be able to remember and adapt, but also to forget. To clarify, we consider the scenario when one sailing pattern changes due to external factors (weather, tide) [5]. If we consider a large waterway area, such as the Atlantic ocean, it is unfeasible to recalculate all patterns every time the change occurs. Storing the AIS data for such a large region, in contrast to handling real-time data streams would pose an additional challenge. Therefore, the solution must adapt in real time relying on the fragmented data currently available. Our GA approach has shown it is capable of evolving with the changes. It is our intention to explore what is the most suitable option to fade, and forget waypoints over time, if the traffic density decreases, thus endowing the solution with real-time adaption capabilities. To that end, the evolutionary approach seems to be a good candidate.

5.2 Discussion on related work

Ideally, the strong and weak points of this solution can best be illustrated by comparing them with other route extraction algorithms. Due to unavailability of AIS data used by the other approaches, we are unable to measure the differences in precision, and execution speed. As different environments can favor one solution over the other, in this section we present the current state of the art, and discuss the similarities with, and differences from our approach.

By short-term prediction, we assume estimating vessel points up to 60 minutes in the future [2]. For such a time interval the routes are not considered, instead the prediction is solely based on the data of the individual vessel, and its history. Perera and Soares [21] use Extended Kalman Filter to estimate the position, velocity, and acceleration of an vessel, and use these values to predict its position. Ristic et al. [22] focus on anomaly detection. With the assumption that the sailing patterns are extracted, they use Kernel density estimator to estimate the future position of a vessel, and compare it with known patterns to identify anomalous behavior.

Long-term predictions, and analysis of maritime patterns calls for a different approach. Rather than learning from each individual vessel, the data are grouped, and analyzed to discover route waypoints, lanes, and their characteristics. The common line for all approaches is the clustering algorithm, and typically it is based on DBSCAN or an improved version of it. Pallotta et al. [4] use incremental DBSCAN to identify waypoints as part of the THREAD algorithm to identify maritime routes. The challenge we faced when testing DBSCAN was that the fine tuning of parameters for good waypoint identification is not possible when dealing with areas with varying density. Due to the very high traffic density in the vicinity of Rotterdam, DBSCAN would give one of two options: a good waypoint spread within the Rotterdam harbor, and the rest treated as noise, or a good waypoint spread around the Rotterdam area, while the entire Rotterdam harbor area was classified as one huge cluster. Therefore, we conclude that this approach is possible as long as the region under consideration maintains a similar level of traffic density, but does not suffice for our needs.

The improvements of clustering have been addressed in [12] and [20]. Rinzivillo et al. [12] use OPTICS algorithm to achieve the progressive clustering and aggregate trajectories. Xiao et al. [20] use lattice-based DBSCAN, and with that approach significantly reduce the problem scale, allowing for fast route extraction, that the authors claim to be in milliseconds on standard PC computer. Containing DBSCAN within the lattice also solves the density problem. The genetic algorithm we use also successfully handles areas with varying density, although we have to report a slower execution time, which increases with the number of epochs. Rinzivillo et al. [12] and Xiao et al. [20] both use the complete data for the area of their consideration, and do not report any missing regions, or incorrect data. If the entire data set is available, both OPTICS, and lattice-based DBSCAN can effectively identify the waypoints. However, if AIS data come in streams which contain fragmented routes, DBSCAN (and its enhanced versions) is unable to cope, as it contains no memory. On the other hand, our genetic algorithm is proving to be more robust, clustering first the available points, and then converging to the higher density area, as additional AIS data become available. In section 2.3.3, we have showed that unless starting waypoints have been occluded, our solution is able to cope with missing regions.

Big data represent an additional dimension to the complexity of clustering moving entities, and trajectories, due to the fact that clustering algorithms used to group the objects according to their properties, do not scale effectively to the increasing volume. Extending the work of Rinzivillo et al. [12], Andrienko et al. [19] use a human analyst to combine clustering, and classification for large data sets. That approach implies that a human analyst will first select a subset of the objects, and experiment with various parameters, in order to build a suitable classifier. That classifier is then used to cluster data points, with the option that the analyst can further supervise the process, and alter the clusters when needed. The solution we propose in this paper is also able to scale with an increasing number of AIS data through incremental waypoint discovery, illustrated in section 0. Due to the quad tree preprocessing, and the removal of data in the vicinity of the extracted waypoints, the genetic algorithm is able to maintain the execution speed even with a continuous increase in input data (see Fig. 17). Although the requirements of the Synchromodal IT platform favor the option of fully autonomous solution, the potential benefits to improve the results through human–machine symbiosis are considerable, and will be investigated in the future.

6 Conclusion

In this paper, we have shown how to apply the genetic algorithm to cluster waypoints, and used them to create a directed graph to represent maritime lanes. We have discussed the limitations of the standard approach, and proposed a set of steps to improve the speed of the algorithm, enabling it to be used in real life business situations, where large data volumes have to be processed quickly. Additionally, we have shown how to connect waypoints to form nodes and edges in a route graph, and how to perform pruning and noise filtering on it. Adding more focus on the problem of dealing with missing AIS data, usually from the high seas, we have shown that our approach is robust, and can identify the lanes without significant drop in accuracy, even when it is not possible to have complete vessel data. Finally, we have tested the proposed solution on real time, and simulated data to assess the extraction quality/accuracy and the execution speed.

Based on the work, and test results presented in this paper, we conclude that evolutionary machine learning is suitable for data mining purposes of synchromodal logistics, and that this improved genetic algorithm is a good foundation for industrial applications. For the future work we are interested in exploring how to handle periodic changes in maritime traffic due to external factors, such as tides, or changing weather conditions. This will also require a mechanism for the algorithm to "forget" a portion of the previous data, and adapt in real time with the new AIS info. Further improvements in terms of data processing are required to enable the handling of big data. Narrow misses of close waypoints also have to be addressed through the introduction of niching schemes, making it possible to jump out of the local maximum and correctly converge to the optimal waypoints.

Acknowledgements This work is part of the SynchromodalIT project (http://www.dinalog.nl/nl/projects/r_d_projects/synchromodalit/) which is partly funded by the Dutch Institute of Advanced Logistics (DINALOG).

Compliance with ethical standards

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Dobrkovic, A., Iacob, M.E., Van Hillegersberg, J.: Maritime pattern extraction from AIS data using a genetic algorithm. In: 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA), pp. 642–651. IEEE (2016)
2. Dobrkovic, A., Iacob, M.E., van Hillegersberg, J., Mes, M., Glandrup, M.: Towards an approach for long term AIS-based prediction of vessel arrival times. In: Logistics and Supply Chain Innovation, pp. 281–294. Springer (2016)
3. Dobrkovic, A., Iacob, M.E., van Hillegersberg, J.: Using machine learning for unsupervised maritime waypoint discovery from streaming AIS data. In: Proceedings of the 15th International Conference on Knowledge Technologies and Data-driven Business, p. 16. ACM (2015)
4. Pallotta, G., Vespe, M., Bryan, K.: Vessel pattern knowledge discovery from AIS data: a framework for anomaly detection and route prediction. Entropy **15**(6), 2218–2245 (2013)
5. Lampe, O.D., Kehrer, J., Hauser, H.: Visual analysis of multivariate movement data using interactive difference views. In: VMV, pp. 315-322 (2010)
6. Peffers, K., Tuunanen, T., Rothenberger, M.A., Chatterjee, S.: A design science research methodology for information systems research. J. Manag. Inf.Syst. **24**(3), 45–77 (2007)
7. Oonk, M.: Smart logistics corridors and the benefits of intelligent transport systems. In: Transport Research Arena (TRA) 5th Conference: Transport Solutions from Research to Deployment (2014)
8. van Riessen, B., Negenborn, R.R., Dekker, R.: Synchromodal container transportation: An overview of current topics and research opportunities. In: Computational Logistics, pp. 386-397. Springer (2015)
9. Lu, M., Borbon-Galvez, Y.: Advanced logistics and supply chain management for intelligent and sustainable transport. In: 19th ITS World Congress (2012)
10. Lei, P.-R., Su, J., Peng, W.-C., Han, W.-Y., Chang, C.-P.: A framework of moving behavior modeling in the maritime surveillance. J. Chung Cheng Inst. Technol. **40**(2), 33–42 (2011)
11. Giannotti, F., Nanni, M., Pinelli, F., Pedreschi, D.: Trajectory pattern mining. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 330–339. ACM (2007)
12. Rinzivillo, S., Pedreschi, D., Nanni, M., Giannotti, F., Andrienko, N., Andrienko, G.: Visually driven analysis of movement data by progressive clustering. Inf. V. **7**(3–4), 225–239 (2008)
13. Handl, J., Knowles, J.: An evolutionary approach to multiobjective clustering. IEEE Trans. Evolut. Comput. **11**(1), 56–76 (2007)
14. Soares Júnior, A., Moreno, B.N., Times, V.C., Matwin, S., Cabral, L.D.A.F.: GRASP-UTS: an algorithm for unsupervised trajectory segmentation. Int. J. Geogr. Inf. Sci. **29**(1), 46–68 (2015)
15. Lee, J.G., Han, J., Whang, K.Y.: Trajectory clustering: a partition-and-group framework. In: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, pp. 593–604. ACM (2007)
16. Li, Z., Lee, J.-G., Li, X., Han, J.: Incremental clustering for trajectories. In: Kitagawa, H., Ishikawa, Y., Li, Q., Watanabe, C. (eds.) Database Systems for Advanced Applications: 15th International Conference, DASFAA 2010, Tsukuba, Japan, April 1–4, 2010, Proceedings, Part II, pp. 32–46. Springer, Berlin Heidelberg (2010)
17. Finkel, R.A., Bentley, J.L.: Quad trees a data structure for retrieval on composite keys. Acta Inf. **4**(1), 1–9 (1974)
18. Waterrecreatie Nederland: BRTN 2008–2013. http://waterrecreatienederland.nl/themas-projecten/landelijk-routenetwerk/brtn-2008-2013/ (2016). Accessed 29 Jan 2016
19. Andrienko, G., Andrienko, N., Rinzivillo, S., Nanni, M., Pedreschi, D., Giannotti, F.: Interactive visual clustering of large collections of trajectories. In: IEEE Symposium on Visual Analytics Science and Technology, 2009, VAST 2009, pp. 3-10. IEEE (2009)
20. Xiao, Z., Ponnambalam, L., Fu, X., Zhang, W.: Maritime traffic probabilistic forecasting based on vessels' waterway patterns and motion behaviors. IEEE Trans. Intell. Transp. Syst. **18**(11), 3122–3134 (2017)
21. Perera, L.P., Soares, C.G.: Ocean vessel trajectory estimation and prediction based on Extended Kalman filter. In: Proceedings of 2nd International Conference on Adaptive and Self-adaptive Systems and Applications, pp. 14–20 (2010)
22. Ristic, B., La Scala, B., Morelande, M., Gordon, N.: Statistical analysis of motion patterns in AIS data: anomaly detection and motion prediction. In: 2008 11th International Conference on Information Fusion, pp. 1–7. IEEE (2008)