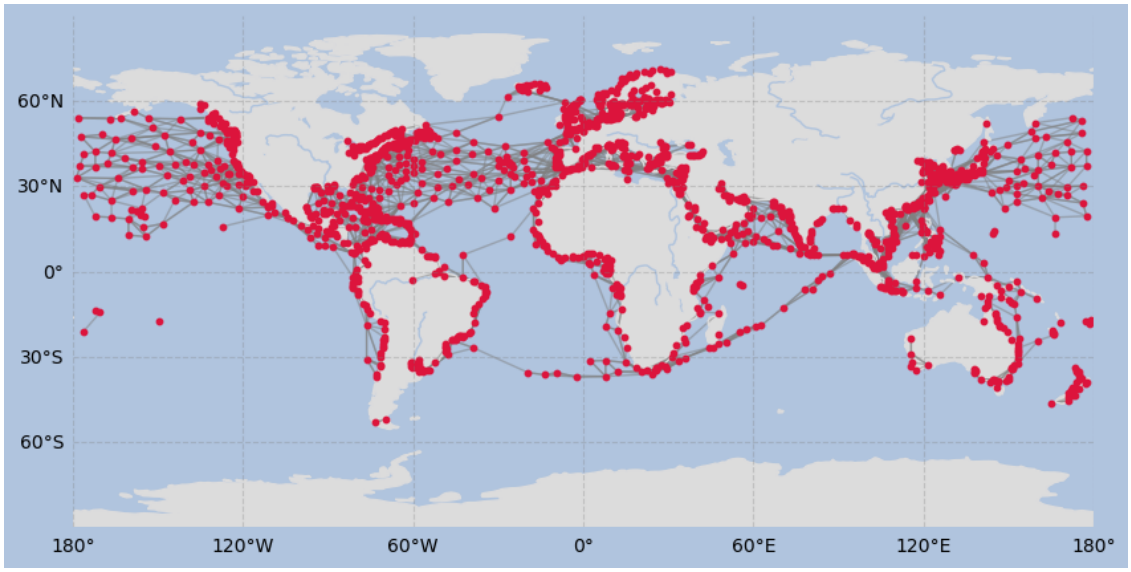
CHALMERS
UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG



Maritime Shipping Network Graph - a Model Derived from Vessel AIS Data

Creating and evaluating a graph representation of maritime vessel traffic using AIS data

Master's thesis in Computer science and engineering

HAMPUS LARSSON, WENDY PAU

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2024

Master's thesis 2024

Maritime Shipping Network Graph - a Model Derived from Vessel AIS Data

Creating and evaluating a graph representation of maritime vessel traffic using AIS data

HAMPUS LARSSON, WENDY PAU

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2024

Maritime Shipping Network Graph Representation
Creating and evaluating a graph representation of maritime vessel traffic using AIS data
HAMPUS LARSSON, WENDY PAU

Supervisor: Peter Damaschke, Data Science and AI.
Advisor: Kayed Mahra & Jesper Olsson
Examiner: Magnus Myreen, Computer Science and Engineering.

Master's Thesis 2024
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Constructed graph representation of the maritime shipping network.

Typeset in LaTeX
Gothenburg, Sweden 2024

Maritime Shipping Network Graph Representation
Creating and evaluating a graph representation of maritime vessel traffic using AIS data
HAMPUS LARSSON, WENDY PAU
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Maritime shipping shoulders more than 90% of global trade. Data science and ML, another immensely profitable industry, currently experiences an unprecedented evolution of techniques. This project proposes novel methods that apply data science techniques to the domain of maritime shipping. The main objective of this project is to construct a graph closely modelling the global maritime shipping structure, from which analytics can be derived. The node set is constructed using a pipeline of Change Point Detection to identify preliminary waypoints and reduce data quantity, KDE is utilised for geographical density estimation and partitioning the AIS data into different density areas, and lastly, the geospatial indexing framework S2 Geometry is used for final waypoint extraction to a node set. The edge set is constructed using a transition matrix that is used together with the final node set to construct the graph representation. Simulation results on the graph representation reveal the ability to construct routes with high resemblance to real-world routes. Further testing revealed high likeness between the most influential nodes in the graph representation and influential points-of-interest in the maritime shipping structure. In turn, the maritime shipping network graph representation is a tool for analysing the maritime shipping structure.

Keywords: AIS data, maritime, graph, network, kernel density estimation, S2 Google, traffic route, change point detection, path-finding, A*.

Acknowledgements

We would like to express our sincere gratitude to all those who have been a part of this project and contributed to its results. First and foremost, we sincerely thank our advisors, Jesper Olsson and Kayed Mahra for supervising this project. Their supportive natures, combined with heaps of domain knowledge, have been vital in shaping the project into what it is today.

We are also grateful for the support of professor Peter Damaschke, our supervisor at Chalmers University of Technology. It has been invaluable to have the guidance of someone as well-versed in the norms and customs of the scientific community as him.

Special thanks go out to professor Magnus Myreen at Chalmers University of Technology. His insightful feedback and correspondence has truly elevated this report.

Hampus Larsson & Wendy Pau, Gothenburg, 2024-06-25

Contents

List of Figures

List of Tables

List of Algorithms

1

Introduction

The maritime transportation sector stands as a cornerstone of our current economy. The demand for moving goods continues to surge, propelled by customer expectations for cost-effective, top-tier services and the expenses tied to providing these transport services. Maritime shipping dominates alternative modes of transport such as trucks, aeroplanes, or trains, due to its cost efficiency, rapid speeds, and dependable schedules [1]. Dating back through history, maritime transportation remains one of humanity's earliest means of connecting, playing an integral role in trade and cultural exchange. Presently, maritime transport shoulders more than 90% of global trade, particularly within global scale logistical networks [2]. In the last 30 years, containerised cargo has grown by more than 8% annually, with more than 5150 container ships globally operational in 2017 [3].

In recent years, data science and AI have exploded. The data science market is projected to increase from $96.3 billion (2022) to $378.7 billion (2030) [4], a potential largely based in the acquisition of data science and AI into new disciplines. As of 2023, 18.2% of firms in America report the usage of AI — a greater than three-fold increase since 2017 [5]. In the maritime shipping sector, data science is no foreign concept. As it stands, the query 'maritime shipping AI' yields approximately 80 900 results on Google Scholar [6]. However, due to the rapid development of data science and AI, many concepts are left unexplored in the context of maritime shipping.

The complex structure of global maritime shipping share a multitude of properties with that of a mathematical graph. Ports exhibit node-like properties, being crucial points of connectivity in the maritime shipping structure. Likewise, the movements of vessels between ports model the function of edges in a graph [7].

By leveraging emerging data science and AI/ML techniques, global maritime shipping and its network structure can be modelled as a graph. A graph representation of the maritime shipping structure could provide multifaceted benefits, primarily stemming from inherent properties of the graph structure. Firstly, a graph representation enables the visualisation of complex connections within the maritime structure, thus allowing stakeholders comprehensive insights into the flow of cargo, route optimisation, and bottleneck analysis. Secondly, by leveraging graph algorithms the analysis of key nodes, vital shipping lanes, and strategic ports is facilitated; decision-making processes are aided. Moreover, a graph representation would provide means for analysing dynamic changes in the shipping structure. In simulation and testing, the understanding of the global maritime shipping network could be furthered.

1.1 Problem Description

This thesis contributes to the maritime shipping sector by utilising data science techniques to construct a graph representation modelling the maritime shipping network, with naval points of interest as nodes and vessel routes as edges. A graph representation of the maritime shipping network is constructed using Automatic Identification System (AIS) data (Section 2.1), offering valuable insights into the operation and structure of maritime shipping. Through its construction via historical AIS data, the graph representation provides a unique platform for analysis where vital hubs and points of interest (POI) in the maritime shipping structure are deduced. Furthermore, shipping journeys can be planned using algorithmic approaches. As such, the construction of a "true-to-life" graph representation paves the way for savings analysis; both of economic and environmental nature.

One of the main challenges lies in producing a graph representation that is both meaningful and accurately reflects the intricacies of the actual maritime network. Determining what the nodes and edges shall represent poses a significant challenge. While the existing research commonly adopts an approach that treats ports as nodes and the connections between them as edges, this approach might overlook valuable insights [7]. One have to consider that there are additional elements that should be included within the node set, such as anchorages, sea lanes, and other geographical POI's.

Furthermore, the geographical density of data plays a crucial role in defining nodes. Certain locations experience heavy maritime traffic in contrast to others; the data density is uneven. Therefore, many clustering algorithms, such as the original DBSCAN [8], are not suited for the task of aggregating AIS data into nodes.

Another challenge, presented by the issue of uneven data coverage, is dealing with missing data. Since vessels often turn off their AIS, the AIS data typically have gaps and uneven geographical distribution. This missing data further complicates the problem since important milestones risk being omitted from the AIS data. Without comprehensive data coverage, there are difficulties in accurately discerning the actual routes undertaken by vessels, which in turn risk causing a disconnect between reality and the model graph representation.

1.2 Purpose

As alluded to, the primary aim of this project is to construct a close-to-reality graph representation of the maritime shipping network structure. With the purpose of allowing transfer of insights from the graph representation to real life scenarios, closeness to reality is a main concern.

To achieve a close to reality graph representation, focus lay on constructing both a comprehensive node set and a meaningful edge set. Thus, with the purpose of identifying nodes suitable for inclusion in the node set, maritime points of interest (waypoints) are extracted from AIS data. By applying ML techniques, the density

disparities and missing data mentioned in Section 1.1 can be combated. As the aim is to accurately represent maritime shipping in the form of a graph, edges are also a focal point of research. By observing historical vessel journeys, an edge set representative of actual ship journeys is constructed.

Finally, this project aims to demonstrate the congruence of the graph representation with real-world maritime traffic. With this objective in mind, path finding algorithms are explored to compare the navigation between nodes to that of the real life navigation of vessels between ports.

1.3 Limitations

One limitation in constructing the graph representation pertained to the time frame of AIS data integration. The graph is based on historical AIS data spanning from 2018 to the present day. Consequently, shipping patterns predating 2018 are not taken into account.

Another limitation imposed on the project concerns the "observation of actualised success". In Section 1.1 a key motivator of the project is explained to be the potential for environmental and economic savings. It is not within the scope of this project to see insights gathered from the graph representation actualised in the real world maritime sector. Instead, the scope is limited to simulation on historical data, as opposed to the identification and actualisation of future savings.

1.4 Ethical Considerations and Risks

The process of constructing a comprehensive maritime network graph representation featured a multitude of inherent risks and ethical considerations, that each required meticulous attention throughout the duration of the project work. Ethical considerations included, but are not limited to:

- **Unbiased analytics:** In the case of future real-world application of the graph representation for analytical purposes, fair and unbiased analytics will be of vital importance. Therefore, project work has to actively reject discriminatory practices, and consider the potential impact on diverse stakeholders.
- **Environmental obligations:** One could argue it is an ethical obligation to advocate for a sustainable tomorrow. Thus, the project always aims to highlight the environmental implications of optimised maritime traffic; the research has to align with the broader goals of society.

Apart from the necessity of the project aligning with the ethical considerations above, there are also risks in need of management to ensure successful completion of the study:

- **Geographical coverage and data sparsity:** The AIS data is uneven as seen to geographical coverage. There is a significant difference between AIS coverage in high density regions, as compared to low density regions. Therefore, the

generalisability of the maritime network graph would have been at risk if the unevenness could not be managed. One could even imagine some important trade route to be missing from the graph representation. However, areas with a high degree of AIS coverage also naturally happens to be areas of high importance for maritime shipping. Furthermore, given that AIS data extends back to 2018, it would be debatable to regard a maritime trade route as significant if it has been inactive ever since then. Even though the project did not fail due to this risk, in order to cover all bases of potential failure, the following mitigation is proposed at the start of work: In the case of insufficient global data coverage the project work will pivot to focus on an area more local in scope, but vast in its AIS coverage (e.g. the English Channel or the Mediterranean Sea).

- **Model alignment with reality:** In Section 5.5 the evaluation process is described. The model's usefulness can be validated through path-finding algorithms, and by comparing proposed routes with the historically travelled ones. Alas, this proved only partial usefulness; for the graph representation to truly consolidate its usefulness, one has to further make sure of its alignment with real maritime shipping structure via alternate avenues. Thus, if extra proof of alignment is required, further action can be taken. To diminish the severity of this 'risk' the following future mitigation is proposed: by engaging in dialogue with domain experts at our disposal, both in modelling and in maritime shipping, an extra level of certainty can be ensured.

2

Theory

In this chapter, theoretical concepts of importance to the purpose of this project are presented. Firstly, the concept of the Automatic Identification System is described, coupled with examples of its immense usability. Secondly, Googles S2 Geometry library and its usability for this project is explained. Subsequently, concepts pertaining to node set extraction from AIS data follow: Change Point Detection and Kernel Density Estimation.

2.1 Automatic Identification System

The Automatic Identification System (AIS) is a vital tool for monitoring vessels at sea and can be used for multiple purposes, as described in Section 3. It utilises VHF radio to broadcast information about the ship, such as its identification, location, speed, and direction. This real-time data is used to track vessels, avoid collisions, control traffic, and enforce maritime regulations. AIS data is also essential for studying traffic patterns, optimising routes, and aiding search and rescue operations [9].

2.2 Google S2 Geometry

In traditional cartography, so-called 'map projections' are employed and involve mapping functions from the set of points on the planet's surface to a set of points on a planar map. This is the cause of the infamous distortions in the Mercator projection. This distortion problem can be largely circumvented by using spherical geometry and spherical projections. While the earth is not perfectly spherical, it is more so than flat. A spherical projection, as the name implies, includes mapping points from the surface of the Earth to points on a mathematical sphere. This procedure achieves a distortion of only 0.56% [10]. To approach this problem, Google created the S2 library for spherical geometry.

The main idea of the framework is that the unit sphere can be decomposed into a hierarchy of cells, where the Earth is projected onto six top-level 'face cells' [11], [12]. Each top-level face cell can successively be subdivided into four smaller children cells, where the *level* of a cell represents the number of subdivisions starting with the face cell (see Figure 2.1). The levels range from 0 to 30, where a leaf cell is the smallest

cell at level 30, while the largest cell is a face cell at level 0. A cell is called '*S2Cell*' in the library, and each S2Cell is uniquely identified by a 64-bit '*S2CellId*'.

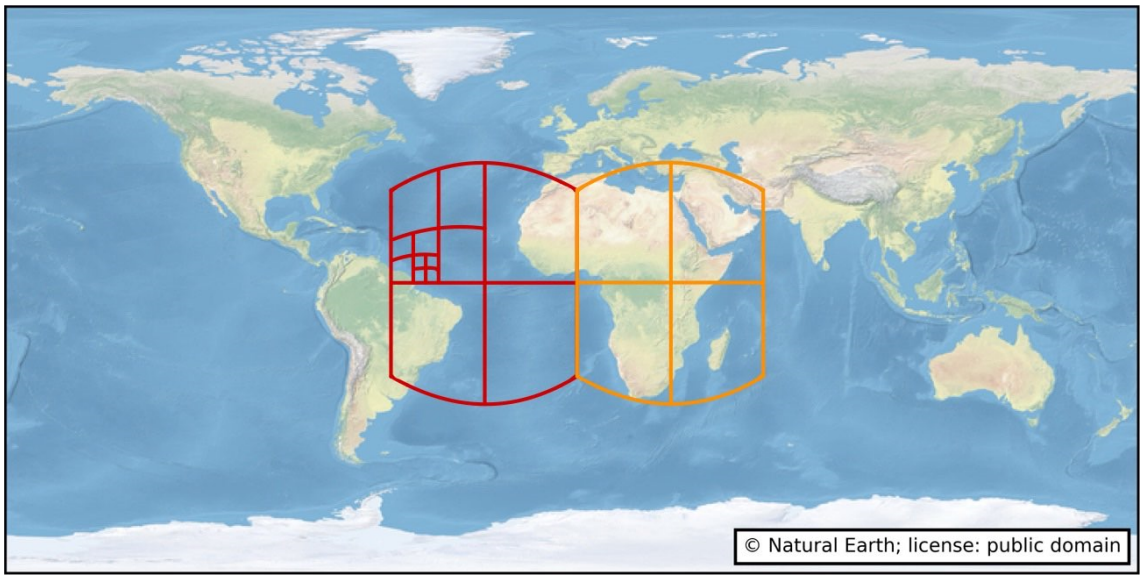


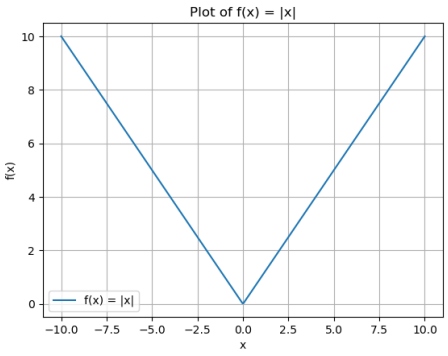
Figure 2.1: A world map showing two of the six face cells, where the red face cell has been subdivided into four smaller cells several times. The world map is plotted using Cartopy [13]. CAVEAT: This figure is a conceptual visualisation of cell divisions. The cell sizes are incorrect and do not represent the true sizes.

2.3 Change Point Detection

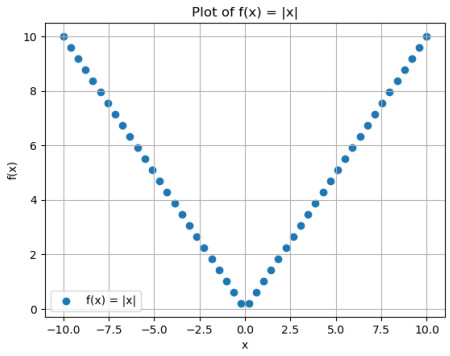
Change point detection is the task of finding changes in the underlying model of a signal. In Figure 2.2a the function $f(x) = |x|$ is displayed. This would be the underlying model of the signal. Say we observe this function and sample at a predetermined interval (Figure 2.2b). The task of change point detection is then to observe these discrete samples and determine points where the function $f(x)$ changes behaviour; such points are referred to as change points. If one were looking for change points in $f(x)$, then $x = 0$ would be a suitable candidate. For it is at this point where the derivative of the function abruptly changes from $f'(x) = -1$ to $f'(x) = 1$.

Change point detection can be performed both offline (as is the case in this report) and online (real-time). The offline case assumes a signal $y = \{y_1, ..., y_T\}$ that takes values in R^d space. Furthermore, y is assumed to be piece-wise stationary. This means that some characteristics of the signal change abruptly at some unknown times $t_1 < t_2 < t_K$, where K can be unknown. The task of change point detection lies in identifying these time instances $t_1, ..., t_K$.

This proves useful in detecting maritime waypoints. By observing changes in the 'speed' and 'course' signals of a vessel, change points can be found.



(a) Plot of the function $f(x) = |x|$.



(b) Discrete sampling of the function $f(x) = |x|$.

Figure 2.2: Plots of the function $f(x) = |x|$ in both a continuous manner and a discrete manner.

2.3.1 PELT

PELT — Pruned Exact Linear Time is an algorithm for detecting multiple change points in a sequence of data. It was proposed as a computationally efficient alternative to the Binary Segmentation and Segment Neighbourhood search methods [14]. The algorithm works as follows:

1. **Initialisation**: The algorithm starts with no change points and calculates the cost of this model. In this case, cost is defined as the cumulative cost of a segment of data points $x_0..x_i..x_n \in X$, where cost is the negative log likelihood of the data under a Gaussian distribution:

$$\text{cost} = -\sum_{i=1}^{n} \log\left(\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x_i - \mu)^2}{2\sigma^2}\right)\right)$$

   This cost is stored in a list of minimum costs, and the change point which led to this cost is stored in a separate list (initially empty).

2. **Iteration over data**: For each data point from 2 to n (where n is the total number of data points), the algorithm calculates the cost of models with each possible last change point and finds the one with the minimum cost. This cost is then added to the list of minimum costs, and the corresponding change point is added to the list of change points.

3. **Pruning**: To improve efficiency, the algorithm discards change points that could not possibly minimise the cost in future iterations. This is done by considering the cost and the rate of change of cost for each change point.

4. **Termination**: The algorithm stops when it has iterated over all data points. The detected change points are those stored in the list of change points.

PELT requires a penalty. This is most commonly chosen using a derivation of the Bayesian information criterion (BIC) as seen in Equation 2.1 [15].

$$\beta = p * \log n \tag{2.1}$$

Here p is a calibration parameter and n is the sample size.

2.4 Kernel Density Estimation

Kernel Density Estimation (KDE), also called the kernel estimator, is a non-parametric density estimator used for estimating the underlying probability density function (pdf) of a random variable [16]. Due to the non-parametric nature, requiring no assumption on the underlying parametric form, KDE allows more flexibility by automatically learning the density shape from the data. This is one of the things that makes KDE a suitable approach for data drawn from an unknown or complicated distribution.

Let $(X_1, X_2, ..., X_n)$ be independent random samples drawn from an unknown distribution with density function f. The kernel estimator can formally be expressed as

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right). \tag{2.2}$$

where $h > 0$ is the bandwidth (also called window width or smoothing parameter), and K is a *kernel* function that satisfies the condition

$$\int_{-\infty}^{\infty} K(x)dx = 1. \tag{2.3}$$

The basic idea of KDE involves applying the kernel function K on each sample, which returns individual smooth bumps that are then summed together to obtain a density estimator. Figure 2.3 shows an 1D example of KDE construction where the red bumps are the small bumps after applying the kernel function on each sample, and the blue line is the summed, resulting density estimator.

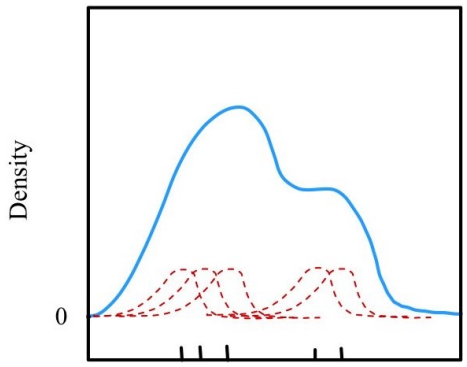


Figure 2.3: A conceptual visualisation of an 1D case of how KDE is constructed. The black lines indicate the locations of the five observations, the red bumps are the smoothed observations, and lastly, the blue curve is the resulting density estimate.

Bandwidth selection is the problem of selecting the smoothing bandwidth h and is a classical topic researched in non-parametric statistics. Figure 2.4 presents another KDE example that uses different values for h on a dataset randomly sampled from a normal distribution.

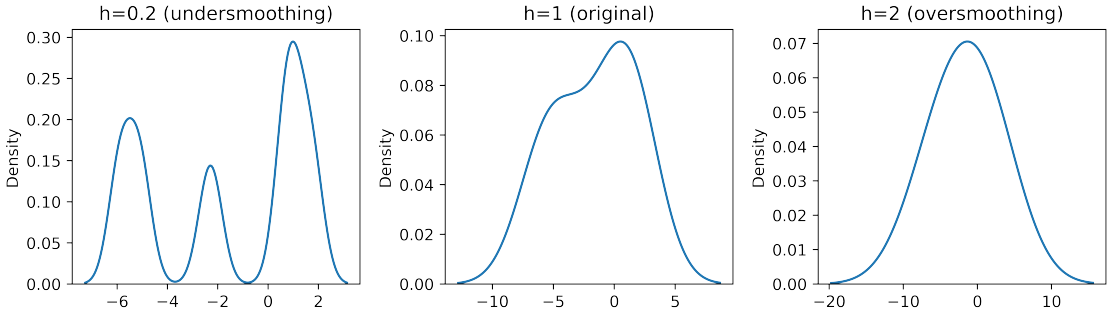


Figure 2.4: KDE with different smoothing bandwidths. The plots are created using *seaborn.kdeplot()* [17], with samples randomly drawn from a normal distribution using the Python package *NumPy* [18].

The left subfigure shows undersmoothing of the density estimate, where a lot of (wanted and unwanted) features are captured. Whereas, the right subfigure shows the effect of oversmoothing where all features are lost or obscured. There exists common approaches to bandwidth selection such as Silverman's Rule and Scott's Rule to name a few [19], [20]. For more details and discussions about KDE and bandwidth selection, see [16].

3

Related Work

As a consequence of the high monetary volume present in the maritime shipping sector, previous work has been carried out with the aim of optimising the industry. What is presented below is only a subset of the studies; albeit, a relevant subset. Because of the grand nature of the maritime sector, research has branched in a multitude of directions such as: route planning, shipping network creation, and vessel optimisation.

In Yang et al. [21], the authors successfully reconstructed the shipping service network between Europe and Asia while considering the *One Belt One Road* initiative. The proposal, as laid out by China, involved the improvement of the Budapest-Piraeus railway and the New Eurasian Land Bridge rail services. Considering the initiative, the paper explored the coping strategy for the Chinese liner shipping company. Their findings showed increased profit by $5 - 6\%$ using their shipping network, meeting more origin-destination demands, finding 'better' hub ports and lastly, discovering the changes and effects the initiative would come to bring. Here, they were able to see significant benefit of increased port calls to Piraeus. An immediate increase of cargo volume could be attributed to the increased capacity, and reduced freight rate of the East Europe rail. The increased capacity usage rate in the new shipping network alleviated pressure from the rest of the system. From these findings, the authors were able to gain managerial insights, showcasing the usefulness of shipping network design for controlling transportation costs and improving shipping efficiency. The findings were also useful for gaining understanding of the current shipping network and the effects of potential future changes.

A study similar to this project was conducted by Liu et al. [22], in which the authors proposed a framework of global maritime shipping network extraction using machine learning methods, including waypoint and berthing area identification, trajectory segmentation and separation, edge generation, and the construction of directed graphs modelling the global maritime shipping network using AIS data of bulk carriers in 2018. To prove and explore the application of the shipping network, a probability-based route planning method using the A* algorithm was proposed. The estimated shipping networks showed effectiveness for ship routing where generation of navigable routes given any departure and destination was one of the important applications, especially when there is no historical data on recorded paths between them. The paper highlighted some challenges such as the difficulty of recognising unique thresholds for changes in the navigation status for different ships in different

scenarios, and ineffective differentiation of waypoint areas among highly dense areas in large waters despite divided global regions and separately determined clustering parameters for each region. Moreover, the probability-based route planning method showed some unreasonable routes with high probabilities in large waters, where the authors suspect it to be due to the simplicity of the way of calculating the probability of node connections. For future work, the authors suggest either focusing on solving the mentioned problems, improving the methodology of the shipping network extraction, or conducting a deeper analysis of the estimated networks by following current research.

Filipiak et al. [23] proposed a novel approach for extracting maritime traffic networks from AIS data using the cumulative sum (CUSUM) method for change detection and a genetic algorithm (GA) for waypoint discovery. For edge discovery, the authors annotated the nearest waypoint for each AIS data-point by looking at every trajectory of the vessels passing an area of interest and tracking the visited waypoints based on historical AIS data. Then, by grouping the AIS data by *from_waypoint* and *to_waypoint*, a dataset containing edges is constructed. CUSUM was used for finding preliminary waypoints by detecting AIS data-points with a significant change in course or speed. The preliminary waypoints were partitioned using the spatial partitioning algorithm *k-d B-trees*, where each partition was treated separately by the GA to detect the final waypoints in a distributed manner. The spatial partitioning was a crucial step for using the GA, as the dense areas represented the fittest genes while sparse areas were ignored. The final network was evaluated by comparing four generated routes with the respective routes from a contemporary software within the Baltic Sea area with AIS data spanning over 8 weeks. The AIS data was further filtered, separating the data by passengers, tankers, and cargo ships. The resulting graphs showed a closer resemblance for the tanker and cargo vessels than the passenger vessels where the authors attribute it to data imbalance in the AIS messages caused by some passenger routes operating on shorter and repetitive patterns leading to numerous messages in small regions. Moreover, some irregularities in the results are attributed to the stochastic nature of the GA. Lastly, the authors highlighted that the proposed method could be improved due to its susceptibility to missing or incorrect AIS messages and bad performance in sparse areas. A suggestion was to use AIS trajectory reconstruction methods to alleviate the issue.

In a study conducted by Le Tixerant et al. [24], the authors investigated the uses of Automatic Identification System (AIS) data for maritime spatial planning (MSP). The results showed that AIS has many benefits in analysis and further studies where it can be used to describe sea-fishing activities, construct trajectory and position density maps, highlight hierarchical networks of maritime routes, asses risks of use conflicts in marine space by contributing to spatio-temporal interactions between activities for different time spans, and produce more qualitative spatial indicators by crossing AIS with other data. Another benefit is potentially having access to data on a worldwide scale, enabling analysis both at global and local levels. However, due to the nature of AIS data and being easily accessible while having an unclear legal framework, the authors discuss how AIS can infringe on public privacy where it is possible to extract leisure and fishing navigation activities and link it to a person and

how private organisations exploit, distribute and market it online raising concerns from shipping companies.

4

Data & Analysis

For this project, the dataset was provided by the company. The dataset contained AIS data from three different sources, together with a list of ports (see visualisation example of traffic data in Figure 4.1). The three sources were: *Gatehouse, AISHub* and *VesselFinder*, where the data ranged from year 2018 to the present day and consisted of a mix of terrestrial and satellite data. The vessel information that is common to all sources can be seen in Table 4.1 where it contains information about a vessel and its status at some time point.

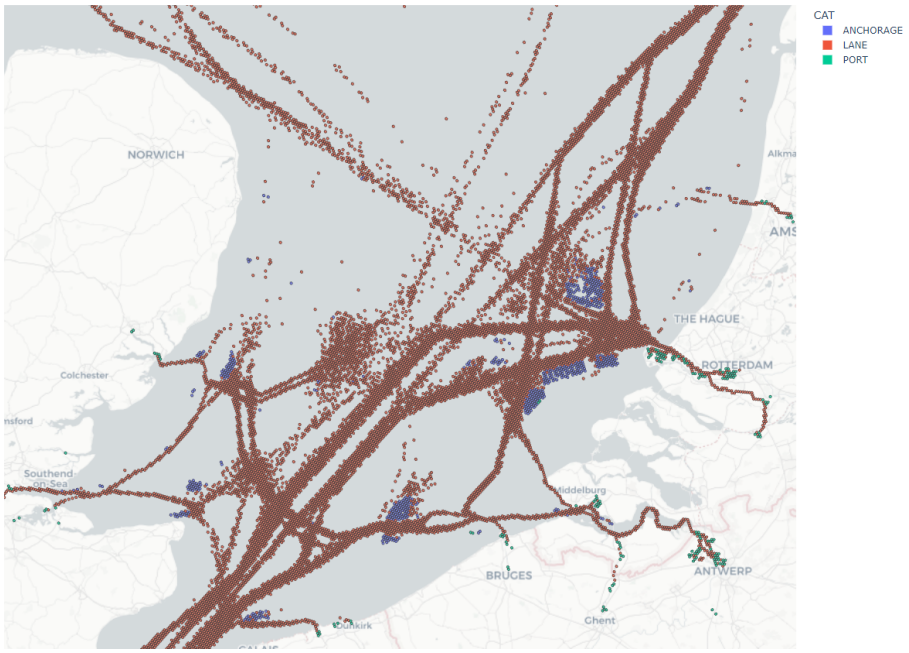


Figure 4.1: AIS data aggregated with H3 geographical index and coloured by the mode navigational status of samples in each hexagon.

Vessel Information	
Field name	**Description**
TIMESTAMP	Date and Time (in UTC) when position was recorded by AIS
IMO	Unique seven-digit vessel number
MMSI	Maritime mobile service identity (AIS identifier)
NAME	Name of the vessel
CALLSIGN	Callsign of the vessel
LATITUDE	Geographical latitude (WGS84)
LONGITUDE	Geographical longitude (WGS84)
COURSE	Course over ground (degrees)
SPEED	Speed over ground (knots)
HEADING	Heading of the vessel's hull (degrees)
NAVSTAT	Navigation status according to [25]
AISTYPE	Type of the vessel according to [26]
A	Dimension (meters) from AIS GPS antenna to the Bow of the vessel
B	Dimension (meters) from AIS GPS antenna to the Stern of the vessel (vessel length = A+B)
C	Dimension (meters) from AIS GPS antenna to the Port of the vessel
D	Dimension (meters) from AIS GPS antenna to the Starboard of the vessel (vessel width = C+D)
DRAUGHT	Current draught of the vessel (meters)
DESTINATION	Port of destination
ETA	Estimated time of arrival

Table 4.1: Available information about the vessel information. Some fields can be unavailable and is thus set as null. The descriptions were taken from [27], [28].

Tables 4.2, 4.3, 4.4, 4.5 show a single manufactured AIS data sample from the Gatehouse data, consisting of 21 additional data fields. The data sample was manufactured in the sense that it does not reflect any actual data transmitted by a vessel, but the general picture painted by the actual data content is structurally equivalent. This data consists more of port-to-port information such as *port of loading (pol), port of destination (pod)*, and *estimated time of arrival (ETA)*.

added_datetime	position_datetime	id	start_datetime	end_datetime	vessel_imo_number
2022-12-05 20:11:09.721494 UTC	2022-12-05 19:58:05.000000 UTC	999	2022-10-19 15:15:10.887625 UTC	null	9684689

Table 4.2: Example of AIS data sample structure, showing data fields 1-6.

input_pol_un_location_code	pol_un_location_code	pol_name	pol_country	pol_arrival_datetime
CNYSN	CNYSN	Yangshan	China	2022-07-20 00:00:00.000000 UTC

Table 4.3: Example of AIS data sample structure, showing data fields 7-11.

input_pod_un_location_code	pod_un_location_code	pod_name	pod_country	pod_arrival_datetime
LKCMB	LKCMB	Port of Colombo	Sri Lanka	2022-09-11 23:59:59.999999 UTC

Table 4.4: Example of AIS data sample structure, showing data fields 12-16.

sta	source_eta	eta	add_intermediate	matched_schedule
2022-09-11T18:00:00	2022-12-26 18:00:00.000000 UTC	2022-09-11T18:00:00	TRUE	null

Table 4.5: Example of AIS data sample structure, showing data fields 17-21.

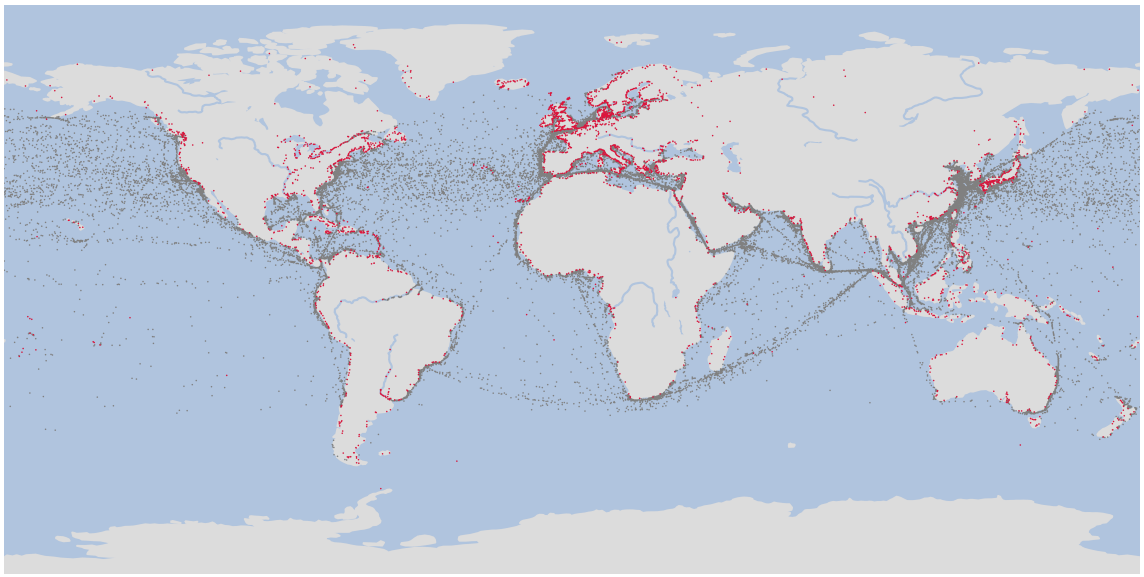


Figure 4.2: Global ports (red) from the port list excluding ports of type NaN.

The port data contains information about global ports such as location code, location name, latitude, and longitude (Figure 4.2, Table 4.6). The combination of the AIS data and the list of ports formed the foundation for constructing the graph representation.

Port Information	
Field name	**Description**
un_location_code	UN/LOCODE, the United Nations Code for Trade and Transport Locations
location_name	Name of port location
location_country	Country of port location
longitude	Geographical longitude
latitude	Geographical latitude
location_timezone	Timezone of port location
location_subdivision	Name of subdivision location of the port
location_port_type	Type of port e.g. harbour, seaport, river port
location_port_size	Size of port (VS, S, M, L, VL)

Table 4.6: Available information about the port data. Some fields can be unavailable and is thus set as null.

4.1 Data analysis

Data exploration and visualization were performed in order to gain understanding and insight of the data. This aided in discovering variable correlation or relationship that would be useful for the following steps in creation of the node and edge sets. Data visualization helps identify data imbalances, insufficient or missing data among the variables that could pose a future problem or to also help interpret the results. The analysis and visualization tasks consisted of the following:

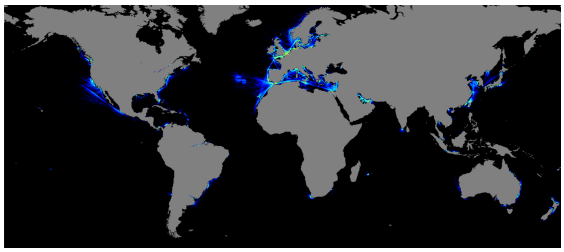
1. Data parsing, where relevant data such as vessel IMO, timestamp, coordinates, and port visits were extracted from CSV files.
2. Data exploration of each dataset from the providers and the port list.
3. Distribution map of ship points in different speed and course ranges to identify the quantity of stationary and moving states.
4. A histogram of the position time frequency for a vessel and journey.

4.1.1 AIS data sources

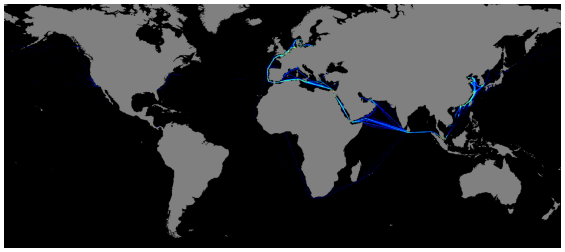
The AIS data from the three data sources consists of data from different time intervals and contains different amount of points, as can be seen in Table 4.7. Both the Gatehouse and AISHub data contains global data (Figure 4.3c, 4.3a), but the AISHub data is limited to land based stations. Moreover, the VesselFinder data covers only the Asia-Europe regions and contains both terrestrial and satellite data (Figure 4.3b).

Name	Start	End	#rows	Resolution
Gatehouse	2022-09-21	2022-12-31	1 973 916	Hourly
VesselFinder	2017-06-01	2018-06-30	1 002 733	Hourly
AISHub	2023-02-22	Ongoing	Many	Hourly

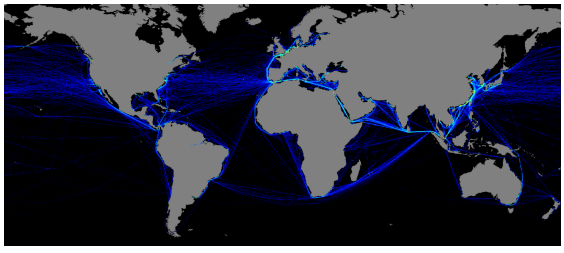
Table 4.7: Information about the data from the three AIS data sources.



(a) AISHub



(b) VesselFinder



(c) Gatehouse

Figure 4.3: Distribution plot of each dataset from the three AIS data sources.

Foremost, because of the computation time and data imbalance, it was decided to only use the AIS data from Gatehouse as it contains global data compared to the

other two sources that only covers specific areas. Moreover, as the Gatehouse data also contains additional vessel information involving port-to-port information, it helped in the extraction of the edge set. By knowing in which port a vessel starts and ends up in, it makes it possible to extract the journey, which was later used in the calculation of the transition matrix. More details in Section 5.3.

4.1.2 Ports

The list of ports contains ports of port types: NaN (missing port type), canal, ferry, harbour, marina, seaport, river port, port terminal, waterway region, deep-water seaport, offshore terminal, pier, and jetty or wharf. The ports of different types are shown in Figure 4.4 where the majority of the visible ports have no port type (red points).

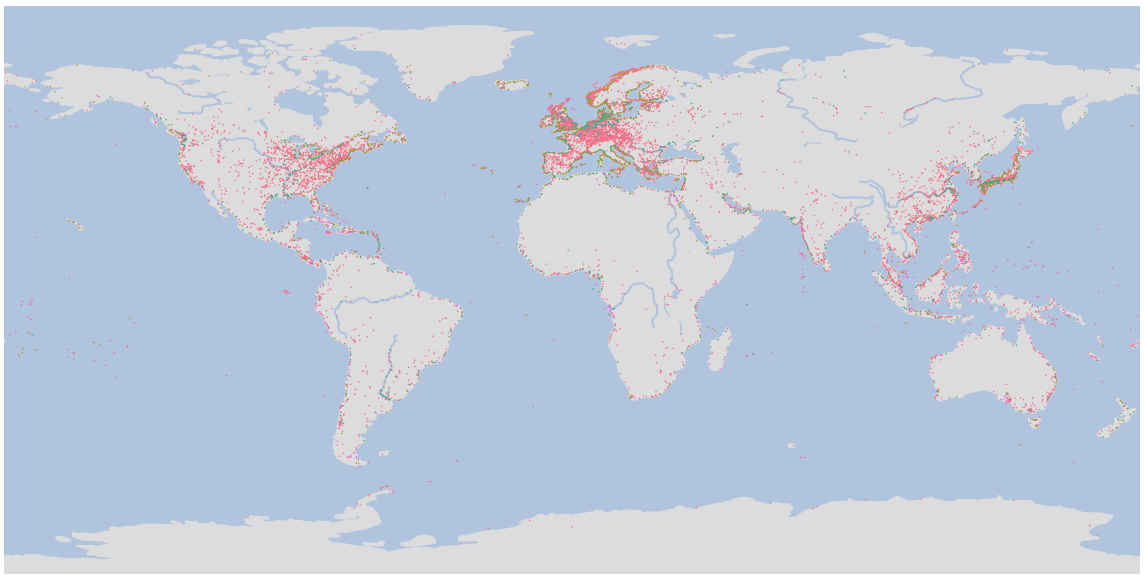


Figure 4.4: Categorical scatter plot of ports showing different port types. The red ports represent those with no port type (NaN).

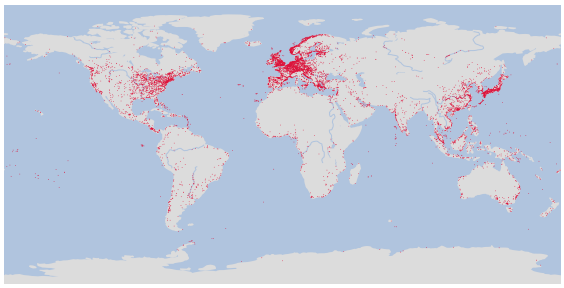


Figure 4.5: Ports with missing port type.

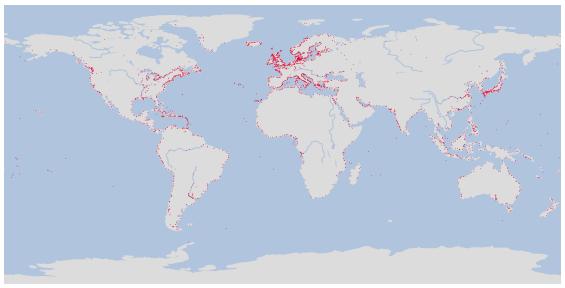


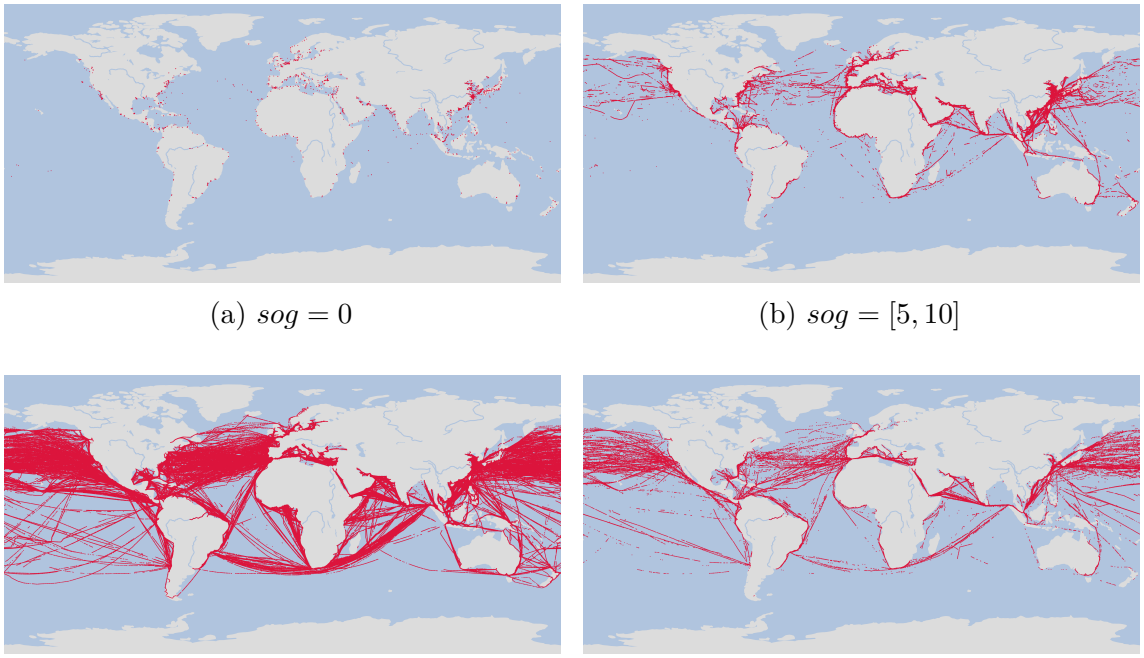
Figure 4.6: Ports excluding missing port types.

Further observations on the ports of missing port type shows that they are mostly located inland or at minuscule islands. An overview of the ports containing and

excluding ports with no port type can be seen in Figure 4.5, 4.6. Since the project focuses on the maritime transport, the inland ports are not of much interest. Moreover, since the port list excluding NaN port types contains most of the busiest container ports and covers the world, it was decided to exclude the NaN ports in the next steps as the final port list was deemed adequate for the project.

4.1.3 Speed and course over ground

The speed of a vessel relative to the Earth's surface is called *Speed over ground* (sog) and is usually measured by GPS. Four distribution plots of different sog intervals can be seen in Figure 4.7. A speed of 0 knots would imply a vessel not moving, which usually are when they are in a port or anchored. The most common speed globally seemed to be around 15–20 knots, while some vessels travel at up to 25 knots. A lower speed of around 5–10 knots are most common in dense areas such as Europe, East Asia and Central America, while a speed over 25 knots occurred in some few individual points in the world.

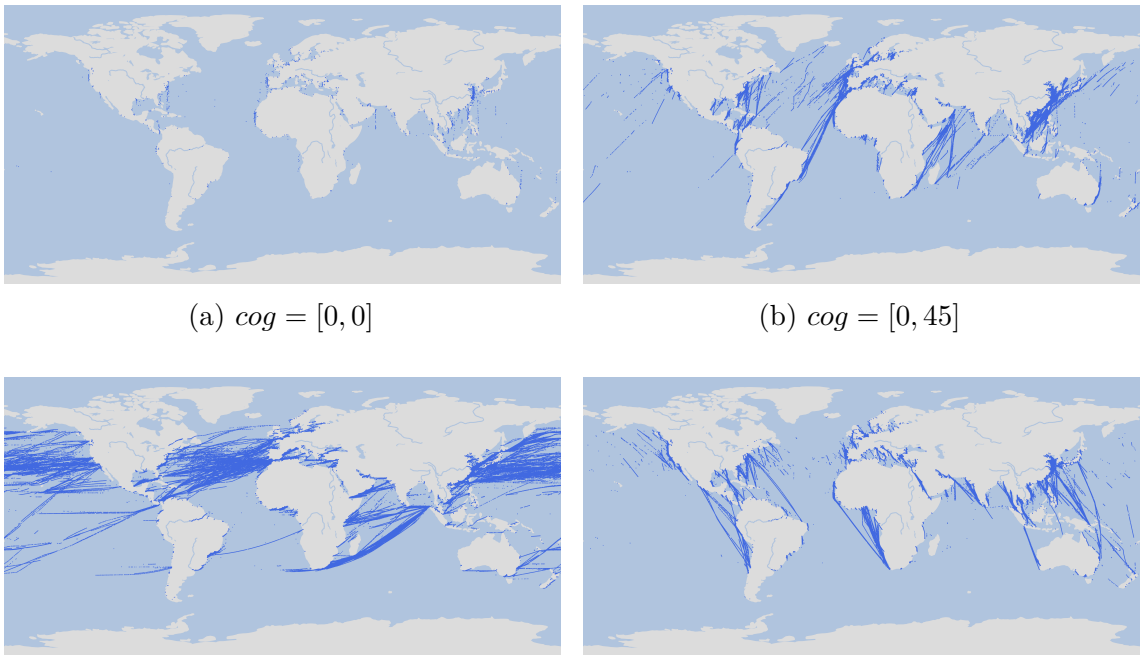


(a) $sog = 0$ (b) $sog = [5, 10]$

(c) $sog = [15, 20]$ (d) $sog = [20, 25]$

Figure 4.7: Distribution plots showing 4 different sog intervals.

The actual course of where a vessel is moving towards relative to the Earth's surface is called *Course over ground* (cog) and is also usually measured using GPS. Four distribution plots showing different cog intervals ranging from 0 to 360 degrees can be seen in Figure 4.8. Various complete routes between different regions can be identified in the plots, such as vessels travelling between America and Europe or Asia. Moreover, the course indicates which regions the vessel is travelling from and to, e.g., for vessels with $cog = [225, 270]$ it is more probable that the vessels are travelling from America to China than the other way.



(a) $cog = [0, 0]$ (b) $cog = [0, 45]$

(c) $cog = [225, 270]$ (d) $cog = [315, 360]$

Figure 4.8: Distribution plots showing 4 different cog intervals.

4.1.4 Position time frequency

Even though the AIS data resolution is hourly, it does not necessarily imply that the number of data for each vessel or journey is equal. The consequences of unequal amount of data points per journey leads to some vessel journeys being over- and underrepresented, which can later affect the construction of the edge set.

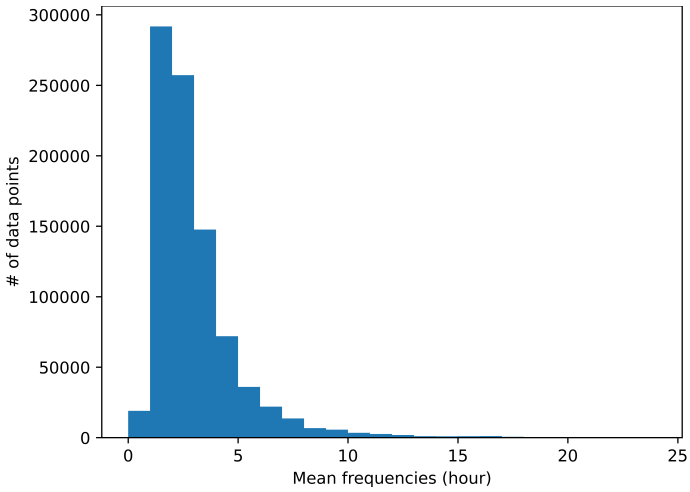


Figure 4.9: Count plot of the mean update frequencies for the journeys.

A journey was represented by grouping the dataset by a vessel's IMO, pol, pol and pod arrival time. This ensures that each group contains a journey for some vessel

starting at some port. By also grouping by the pol and pod arrival time, it ensures that the data points in the group are from the same journey as their start and end time are matching. Afterwards, the mean update frequency for a journey/group can be calculated, where the total occurrences are shown in Figure 4.9.

The figure indicates data imbalance for some journeys as a steep hill can be seen between mean frequency of 1–5 hours. In the way the edge set is constructed (Subsection 5.3) using a transition matrix, the journeys that have longer update time will have longer edges between the waypoint nodes, while the journeys with shorter update time will have many short edges between many nodes. This can pose a problem for the final graph construction, as longer edges would imply the possibility of travelling longer and illogical distances, such as a vessel starting in the United Kingdom and suddenly arrived at Yemen without passing any waypoints through the Suez Canal.

5

Methods

This chapter presents the methods used during this project to construct and evaluate the graph representation. The process employed consisted of three phases: Construction of the node set from historical AIS data (Section 5.2), construction of the edge set from historical AIS data (Section 5.3), and lastly, evaluation of the graph representation (Section 5.5).

5.1 Graph Representation

The purpose of this project was explained as 'the creation and evaluation of a comprehensive & global maritime shipping network graph representation'. For the sake of clarity in the methodologies below, a maritime shipping network graph representation is defined and explained as follows:

- A *graph representation* in the context of this project refers to an instance of the graph data structure. More precisely, the base of a graph representation is a graph $G = (V, E)$, where G is the graph consisting of nodes (vertices) V and edges E. This graph G is formed in accordance with some modelling rules in order to model some object O. To complete the graph representation of O, additional information I is embedded into the graph. Thus, the graph is extended to $G = (V, E, I)$, where I is of type dictionary and maps object-specific properties of O for each node.
- The *maritime shipping network*, in the context of this project, refers to the shipping by sea undertaken by cargo ships and other maritime shipping vessels. These vessels transport cargo between the world's seaports; often passing by intermediary goals such as canals, anchorages, or geographical points of interests.
- A maritime shipping network graph representation is a graph representation of the maritime shipping network. A model of the maritime shipping network is created by translating it into the structure of a graph G. Nodes and edges are created in a manner suitable to the maritime shipping structure.
- In the case of this project, the graph representation will model maritime shipping at a global scale. Nodes and edges are formed to preserve the historical accuracy of the AIS data, upon with the graph is built. Object specific information is embedded to convey position, and character of each node. Every edge is

weighted. The procedure and exact details of graph creation is explained in Sections 5.2–5.4.

5.2 Creation of Node Set

The data analysis seen in Chapter 4 served as a precursor to the node set creation. All methodologies seen in this section were derived from insights gathered during analysis of the AIS data. As such, by applying the methodologies described below, a node set representing the actual maritime shipping structure and its complexities could be formed.

Naval ports exhibit many node-like qualities. Thus, port are strong candidates for inclusion in the node set of the graph representation; the most obvious node-like behaviour being their role as origin and destination in the transport of cargo. Every cargo ship is routed between pairs of ports, and thus they are the focal points of maritime traffic. In addition, ports are naturally connected to each other via the cargo vessels. Connectivity between nodes is a crucial component in most graphs. On a high abstraction level, creation of port nodes include:

1. Obtaining a port list coupled with the respective *UN location code (UN/LOCODE)* for each port.
2. Gathering location coordinates for each port.
3. Embedding the information into nodes.

However, ports do not reveal the entire picture of maritime shipping. Two routes, both with origin O and destination D, can vary immensely when compared to each other. When comparing maritime journeys there are two other factors to consider apart from origin and destination: the geographical route undertaken and the time required to complete the journey. In the pursuit of capturing more journey context, the node set was extended past solely ports. Ship berthing areas will typically be located in the near vicinity of ports, although, one might also include other points of interests in the node set. To identify waypoints, common anchorages and other points of interest, the AIS data was utilised.

5.2.1 Change Point Detection

As previously mentioned, the complete dataset is far too vast to consider all data points when creating a node set. In addition, many data points do not contain valuable information for the cause of node set creation. The node set should contain both port nodes and waypoint nodes. To deduce meaningful waypoints, change point detection was applied.

Change point detection can be applied to all dimensions of the AIS data, however, interest lied mainly in the sog and cog data. A change in speed might indicate entrance into an area of interest, for example a canal, anchorage, or similar. Similarly, a change in course might indicate another area of interest; a vessel that drastically redirects its course is more in line with the expected behaviour at a node than at

an edge. Therefore, the change point set would come to include both speed change points and course change points.

To detect change points, the python package *ruptures* was used [29]. In Table 5.1 one can regard the configuration employed. *KernelCPD* is the C implementation of both the dynamic programming approach and the PELT approach for change point detection, where it was shown to be much faster in both cases [30]. At first BIC was used for calculating the penalty but showed unsatisfying results. Hence, it was opted to use the elbow method for choosing the penalty value instead.

Parameter	**Configuration**
Cost function	rbf
Search method	KernelCPD
Penalty	Elbow method
min_size	1

Table 5.1: Set-up of change point detection.

The change point detection was performed twice on the AIS dataset, first considering the speed of vessels as the signal, and second considering the course of vessels as the signal. Before applying the change point detection on the variables, the data was first sorted by their time of position and missing sog and cog values were removed. The algorithm was then applied on individual vessels grouped by their IMO to detect changes on a journey. The resulting change points were then compiled into the *change point set*.

5.2.2 Data Splitting Using Kernel Density Estimation

Having identified the change point set, further work needs to be done where the geographical density of the points needs to be considered. For example, consider sparse areas where the observations are scattered across large water and dense areas where the observations are densely packed. For sparse areas, a cluster containing more than 3 points could be rare, while clusters containing more than 10 points could be normal in dense areas.

Sparse	$-1 < est\_pdf \leq max\_low$
Mid	$max\_low < est\_pdf \leq min\_high$
Dense	$min\_high < est\_pdf \leq max\{est\_pdf\}$

Table 5.2: For an estimated pdf value of a data point *est_pdf*, the data point is categorized as one of the three areas: sparse, mid, and dense, according to the respective thresholding in the table. The variables *max_low* and *min_high* are manually chosen and represents the maximum pdf value for the sparse area and the minimum pdf for the dense area respectively.

Different constraints would thus be needed for respective areas because of their dissimilarities in density. By identifying different density areas and splitting the data

by density, the data becomes more homogeneous and thus, facilitates in waypoint extraction for the different areas.

For the kernel density estimation of the change points, *scipy.stats.gaussian_kde* from the SciPy package was used [31]. After obtaining the plot of the density estimation and density levels, the dataset was then split into three different areas: dense, mid, and sparse, by thresholding using the manually chosen thresholds: *max_low*, the maximum pdf value for sparse area and *min_high*, the minimum pdf value for dense area as seen in Table 5.2. The estimated pdf value, *est_pdf*, was evaluated for each data point and were then appended to one of the three areas according to the aforementioned thresholding.

5.2.3 Waypoint Extraction Using S2 Geometry

The Python package *s2sphere* was used for identifying the waypoints [32]. The *s2sphere* package is an implementation of a part of the C++ S2 Geometry Library mentioned in Section 2.2. In this section, the geographical coverage of the change point set needed to be considered. Here, there could be single points spread across wide waters i.e. sparse areas, and multiple packed points by a port i.e. dense areas.

Algorithm 1 Create Waypoints

```
df := AIS data
level := minimum cell size
min_level := maximum cell size
threshold := total number of points in a cell at level

tree ← root node
df_temp ← copy of df
for current_level = level...min_level, −1 do
    cell_counts ← group and count the rows by 'S2CellId'
    waypoints ← cell_counts ≥ threshold
    if waypoints is not empty then
        df_waypoints ← get rows from df_temp that has S2CellId in waypoints
        add waypoints to tree
        remove waypoints from df_temp
    end if
    cell_counts ← get parent cell ids for all cell ids in cell_counts
end for
waypoints ← collect all leaves in tree
return list of waypoints
```

To identify the waypoints, a waypoint was defined to be a cell with a *threshold* representing the maximum point count in the cell. The main idea was to start from the smallest cell *level*, and check if some cells contains n amount of points. Add the cells fulfilling the constraint in the waypoint set and repeat the process for the rest of

the set at $level - 1$ until no more cells are left, or it has reached min_level. At the end of the process, it resulted in a list of S2CellIds that represents the waypoints.

The first step in this process was to construct a list containing S2CellId of each point in the change point set at the smallest cell size, *level*. This was done by taking a point's longitude and latitude coordinates and converting it into a S2CellId using the constructor. In this step, the list could contain duplicates as the points could exist in the same cells at the same level. After obtaining the S2CellIds representing the change points, the waypoints were identified as described in Algorithm 1.

The waypoints are stored using a tree data structure to avoid duplicated waypoints. However, since ports are always a waypoint, an exception was needed to handle multiple ports in the same cell, i.e. they have the same S2CellId. To solve this issue, it was decided that a cell can contain multiple ports as the route up until the port of destination (pod) will be the same and that the difference between going to pod A versus pod B is negligible. Moreover, to enable evaluation of the network from port-to-port in the later stage, the port data was embedded in the respective waypoints.

The main idea for the tree structure was that for each non-existing waypoint in the tree: (1) a new branch is added to the root node, where the waypoint is the leaf node and all preceding nodes are the successive parent cells. Therefore, whenever a waypoint is extracted, it only requires checking if the parent cell/node already exists at the level subsequent to the root node. (2) If the parent cell already exists, then the algorithm ignores the new waypoint. (3) If the new waypoint is a port, then a new neighbouring node leaf is added by the current waypoint. (4) However, if the port already exists in the tree, then the new port data is embedded in the already existing tree leaf.

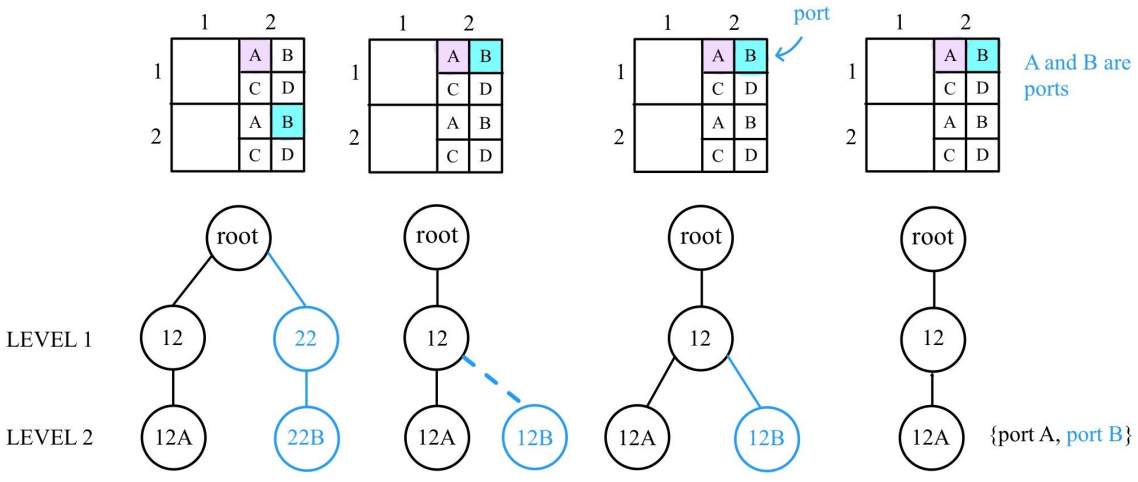


Figure 5.1: An example showing a newly extracted waypoint being added to the waypoint tree depending on 4 different cases.

A visual example of the tree construction is shown in Figure 5.1; an example showing a newly extracted waypoint being added to the waypoint tree depending on 4 different

cases. Each case is depicted with a S2Cell split into 2 levels together with its respective waypoint tree. Let the new waypoint cell be B, the cell marked in blue. Furthermore, let A, the pink cell, be an already existing waypoint cell in the tree. The 4 cases are then: (1) When there exists no parent cell subsequent to the root node that contains B. (2) There exists a parent cell at level 1 that contains B. (3) B is a port while A is not. (4) Both A and B are ports.

This subsection is an iterative process consisting of analysing the extracted waypoints and changing the argument values in Table 5.3. As mentioned in Subsection 5.2.2 the different density areas requires different constraints. The current method for deciding the values for the constraints was done by analysing the waypoints in different areas and adjust accordingly until satisfying waypoints were obtained.

Variable	Description
level	minimum cell size (higher level)
min_level	maximum cell size (smaller level)
threshold	total number of points in a cell at level

Table 5.3: Arguments for Algorithm 1.

5.3 Creation of Edge Set

This section outlines the process of creating an edge set suitable for representing connections between nodes in a maritime setting. The precursor to this section is the successful creation of a node-set according to Section 5.2. By considering the intended use-case of the graph representation (as per Section 5.4) the optimal structure of the graph can be inferred; the edges have to be directed in order to preserve information and be able to accurately model vessel journeys between nodes.

Furthermore, in the name of performance and since the evaluation method involves route planning, it would be nonsensical to consider a complete graph. Thus, edges were created in mirror of actual historical ship journeys — an edge models an actual plausible route undertaken by a ship between two nodes.

5.3.1 Transition Matrix

In graph theory, a transition matrix is typically a 2D $n \times n$ matrix, where n is the number of nodes in the graph. Each row in the matrix contains fractions representing the probability of transition from a node to every other node. As a result, each row has a sum of 1.

In the maritime case, considering the AIS data of vessel patterns (see Chapter 4), it is clear some origin-destination pairs are more likely than others. For example, if a vessel enters the Suez Canal from the Mediterranean Sea it is infinitely more likely that the vessel will exit the Suez Canal at Port Suez than that the vessel's next stop will be the port Shenzhen, China. While this might seem trivial, it was of essence that the transition matrix captured such real world relations. To ensure

that the transition matrix accurately models real world transitions between points of interest (nodes) all AIS data from Gatehouse was considered in the following transition matrix population algorithm (Algorithm 2):

Algorithm 2 Populate Transition Matrix

```
AIS := AIS data
N := Number of nodes in node set
Initialise matrix TM[N][N]
Initialise matrix Times[N][N]

Remove data rows that are not in any waypoint cells
sorted_ais ← sort by imo and datetime

freq ← datetime frequency to group by
grouped_ais ← sorted_ais grouped by imo, pol_un_location_code, datetime, freq
for group in grouped_ais do
    current_cell, prev_cell ← None
    current_time, prev_time ← None
    for row in group do
        if row[cell] == current then
            continue
        end if
        prev_cell ← current_cell
        current_cell ← row[cell]
        prev_time ← current_time
        current_time ← row[datetime]
        if prev_cell is not None then
            diff ← (current_time - prev_time).total_seconds()
            Times[prev_cell][current_cell] += diff
            TM[prev_cell][current_cell] += 1
        end if
    end for
end for

Normalise TM & Times by dividing each row element by TM's corresponding total row sum
```

The general operation of Algorithm 2 can be summarised as follows: the outermost loop iterates over groups of sorted AIS data. The data was sorted by time and IMO, port of loading and destination, and time period; group $G_i = g_{i0}, g_{i1}, ..., g_{in}$ then models a journey of the vessel i in chronological order. The logic of the algorithm was then to keep track of the previous cell visited and noticing when the vessel enters a new cell. Every time such a connection was discovered, it was added to the transition matrix TM. Then, every entry was normalised to ensure all entries in the

transition matrix are probabilities; using division by the total accesses to each row, normalisation was achieved.

TM then served as the basis upon which the graph was constructed. As described in Subsection 4.1.4, the update frequency for a journey had to be considered and was done by manually choosing a time period/frequency using the pandas groupby instruction, $pd.Grouper(position\_datetime, freq)$ in the *groupby* [33], [34]. This splits the journeys by a time interval, i.e. the chosen frequency. Let some journey take one month and the chosen frequency is one day. Then the journey will be split into groups with a days' interval. This ensured that a vessel can not travel from the start point to the end point in one step.

Also, one can note that a $Times$ matrix was updated every iteration alongside the transition matrix TM. $Times$ was also normalised in the same manner as TM. This mean estimation of travel time served as a precursor to the custom heuristic function later employed in A* evaluation (Section 5.5.2).

5.4 Graph Construction

This step entails the combination of the node set and the edge set to obtain a shipping network graph representation. Firstly, nodes were added to an empty graph data structure G. Nodes were in turn embedded with location data, signalling the position of each node. In addition, each node contains a *port* property, which is a boolean property indicating whether the node is a port.

Then, in order to suitably connect nodes to one another, the transition matrix (TM) was used; for every non-zero element in TM a directed edge is formed from the node corresponding to the row index to the node corresponding to the column index. Each edge was supplied with two properties: *weight* and *probability*. The probability from TM directly translates to the *probability* property. The *weight* property, on the other hand, required an additional step.

To ensure correct function, the A* algorithm requires that a weight is associated with each edge [35]. Furthermore, a lower weight should imply a lesser inclination to journey across the particular edge. Therefore, each edge was assigned a weight equal to the negative logarithm of the probability (Equation 5.1).

$$weight = -\log(probability) \tag{5.1}$$

Using more concrete notation, the weight assignment can be formulated as seen below (Equation 5.2):

$$weights[i][j] = -\log(TM[i][j]) \tag{5.2}$$

The above steps can be comprised and summarised in Algorithm 3, where $addNode(node)$ is a function of the graph data structure that allows for the addition of nodes into the node set. Likewise, $addEdge(start, end, probability, weight)$ is another utility

function of the graph. The function $addEdge(start, end, probability, weight)$ permits the addition of edges into the edge set.

Algorithm 3 Graph Construction Algorithm

```
G := Empty Graph
nodeDict := Dictionary with nodes as keys and coordinates as values
portDict := Dictionary with nodes as keys and ports as values
TM := Transition Matrix

for node in nodeDicts do
    G.addNode(node)
    G[node].coords = nodeDict[node]
    if node in portDict.keys() then
        G[node].port = portDict[node]
    end if
end for

for i in TM.rows do
    for j in TM.cols do
        if TM[i][j] > 0 then
            G.addEdge(i,j, probability=TM[i][j], weight=(-log(TM[i][j])))
        end if
    end for
end for
```

5.5 Graph Evaluation

The second phase of the project, following the construction of a graph representation, concerned evaluation. As per Section 1.2 the purpose of the project lie in creating a close-to-reality graph representation of the maritime shipping structure. For this project, two methods of evaluating closeness to real life were chosen. Firstly, by studying the properties of the graph representation it was possible to deduce characteristics that in turn could be utilised for comparison; The graph representation could be compared to its real life counterpart through the use of graph metrics.

Secondly, to ensure operational closeness to reality on top of the structural closeness, the graph representation was subjected to a set of test cases. The test cases were designed to test how well the graph representation lends itself to route planning. Real life major origin-destination pairs were tested on the graph representation and evaluated for closeness to their real life counterparts.

5.5.1 Graph Metrics

Three closely correlated metrics, often utilised in the pursuit of explaining and understanding graphs, are *Degree Centrality*, *Betweenness Centrality*, and *Closeness Centrality* [36]. All three measurements try to discern the most important nodes

in a graph: *Degree Centrality* measure the number of edges incident to each node, normalised by the maximum possible degree in the graph. It is often the case that nodes with a high degree centrality represent vital hubs in the graph structure.

In Equation 5.3 the formula is presented as it was implemented to measure degree centrality in the graph representation. $C_D(j)$ denotes the degree centrality of node j, and $TM[i][j]$ denotes the transition matrix entry corresponding to the connection between nodes i and j. *sgn* is a sign function returning 1 if $x > 0$ and 0 otherwise. Finally, the summation was normalised with respect to the maximum possible degree in the graph.

$$C_D(j) = \frac{\sum_{j=i}^{n} sgn(TM[i][j])}{n-1} \tag{5.3}$$

Betweenness Centrality, on the other hand, will quantify the extent to which nodes lie on the shortest path between other pairs of nodes. As such, the metric is useful for detecting bottlenecks in the graph structure.

The implemented version of betweenness centrality is found below in Equation 5.4, where $C_B(v)$ denotes betweenness centrality. In addition, $\sigma_{ij}(v)$ denotes the number of shortest paths from node i to j passing through node v, and $\sigma_{ij}(v)$ denotes the total number of shortest paths from node i to j, with which the measure is normalised.

$$C_B(v) = \frac{\sum_{i \neq j \neq v} \sigma_{ij}(v)}{\sigma_{ij}} \tag{5.4}$$

Finally, *Closeness Centrality* measure the closeness to other nodes for nodes in the graph. As a result, one can detect nodes with high potential for quick interaction with other nodes. The implementation of closeness centrality was precisely derived from Equation 5.5.

In Equation 5.5, $d(u, v)$ denotes the distance between nodes u and v, and $n - 1$ denotes the total number of reachable nodes from node v.

$$C_C(v) = \frac{n-1}{\sum_{u=1}^{n-1} d(u, v)} \tag{5.5}$$

These above metrics were examined in the network graph representation formed from the procedures in Section 5.4. Apart from the three metrics above, *density*, *average clustering coefficient*, *assortativity*, and *reciprocity* were measured. Subsequently, these metrics were plotted together with the centrality measures in order to visualise the closeness to reality of the network graph representation.

5.5.2 A* Algorithm

A* ('A-star') is a path-finding algorithm in the family of weighted graph-traversal algorithms. The algorithm was employed to find the shortest path between two nodes in a graph, considering both the distance travelled and the estimated distance to

the goal node. Like Dijkstra's algorithm, A* operates by maintaining a lowest-cost path tree from the starting node to the goal node [35]. However, unlike Dijkstra's algorithm, A* utilises a heuristic function; A* expands potential paths using the following formula (equation 5.6):

$$f(n) = g(n) + h(n) \tag{5.6}$$

in which $f(n)$ denotes the total estimated cost of reaching the goal through node n, $g(n)$ denotes the cost of reaching node n from the starting node, and $h(n)$ is the heuristic function that estimates the cost of reaching the goal node from node n. One should note that in the case of $h(n) = 0$, A* is computationally equivalent to Dijkstra's algorithm.

Algorithm 4 A*

```
function ASTAR(start, goal)
    openSet ← {start}                          ▷ Set of nodes to be evaluated
    cameFrom ← empty map  ▷ Map to store the previous node in optimal path
    gScore[start] ← 0                          ▷ Cost from start along best known path
    fScore[start] ← heuristic(start, goal)     ▷ Estimated total cost from start to
goal
    while openSet is not empty do
        current ← node in openSet with the lowest fScore   ▷ Get the node with
lowest fScore
        if current = goal then
            return reconstructPath(cameFrom, current)
        end if
        Remove current from openSet
        for all neighbor of current do
            tentativeGScore ← gScore[current] + distance(current, neighbor)
            if tentativeGScore < gScore[neighbor] then
                cameFrom[neighbor] ← current
                gScore[neighbor] ← tentativeGScore
                fScore[neighbor] ← gScore[neighbor] + heuristic(neighbor, goal)
                if neighbor not in openSet then
                    Add neighbor to openSet
                end if
            end if
        end for
    end while
    return failure                                 ▷ No path exists
end function
```

In Algorithm 4 the implementation of the A* algorithm is presented where it makes a couple of simplifications and assumptions. Firstly, *heuristic*(*node*, *goal*) is assumed to be an appropriate heuristic function. As stated, this could be the

Euclidean distance, or even the Manhattan distance if the application domain calls for it. Moreover, working implementations of $distance(node1, node2)$ and $reconstructPath(cameFrom, current)$ are assumed.

For the sake of robustness, it was necessary to be able to verify expected behaviour of the graph representation. A*, as presented in Algorithm 4, was one such tool for evaluation. For implementation the Python package *NetworkX* and its method *astar_path(G, source, target, heuristic=None, weight='weight')* was employed [37]. Furthermore, to be able to effectively path find in the maritime setting, a custom heuristic function had to be supplied.

Euclidean distance is a common heuristic function when handling distances on a 2-dimensional plane (Equation 5.7). However, in real life, shipping vessels traverse the sea on a globe. Additionally, the positions of nodes in the graph are supplied in coordinate format (see Section 5.4). Haversine distance exists to fill this gap and provide the tools for distance calculation on a globe. More specifically, the Haversine formula is used to find the distance d in-between two points lying on the surface of a sphere (globe) utilising their latitudes (ϕ_1, ϕ_2) and longitudes (λ_1, λ_2) (Equation 5.8).

$$dist(x_1, y_1, y_2, y_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{5.7}$$

$$d = 2r \cdot \arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos(\phi_1) \cdot \cos(\phi_2) \cdot \sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right) \tag{5.8}$$

Equation 5.8 consists of the following elements:

- d: the distance between the two points.
- r: the radius of the sphere (the earth has a radius of approximately $6,371$ km).
- ϕ_1 and ϕ_2: the latitudes of the two points in radians.
- λ_1 and λ_2: the longitudes of the two points in radians.

Even though Haversine distance more closely models the distance between two nodes in the graph, context is left out. Firstly, the Earth is not a perfect sphere. The fact is that the Earth's shape more closely resembles that of an oblate spheroid. Vincenty's formulae [38], remedies this fallacy of the Haversine formula. Still, innate properties of distance based heuristic functions render them complicated for global maritime applications. When traversing the sea, the shortest distance route might not always be possible. These impossibilities are often found where waters are separated by a landmass. In such a case, the shortest distance would be to traverse the landmass in a straight line between the waters. However, in reality, the vessel has to avoid the landmass and navigate around it.

Travel Time Heuristic

To combat the above problems, a time based heuristic function was used. By considering the actual time required to travel between geographical POI's as the heuristic cost, one can circumvent the issues related to a distance based heuristic function. The key component in constructing such a heuristic function lies in study of the historical travel patterns of vessels. The first component of the time based heuristic function is seen in Algorithm 2.

In tandem with calculating the transition matrix entries, the mean travel time between nodes is calculated and stored in the matrix *Times*. The matrix is of size nxn, where n is the number of nodes in the node-set. Each element in the *Times* matrix is calculated as per Equation 5.9. *current_time* and *prev_time* are defined as in Algorithm 2, i and j refers to node i and node j in the node-set; $i, j \in nodeset$ and $i \neq j$.

$$\begin{aligned} \Delta time &= current\_time - prev\_time \\ Times[i][j] &\leftarrow Times[i][j] + \Delta time \end{aligned} \tag{5.9}$$

Following the calculation of each matrix element in *Times*, all elements were normalised by dividing by the total number of accesses to the matrix (full procedure seen in Algorithm 2). The resulting *Times* matrix was then stored in memory, upon which the heuristic function could be constructed (Algorithm 5). Algorithm 5 and its operation can be summarised as follows: first the expected travel time between the two input nodes is read from *Times*, a penalty value is then returned if the time read equals 0, otherwise the time read is returned.

Algorithm 5 Travel Time Heuristic

```
function TIME_HEURISTIC(node1, node2)
    Times := Matrix of size #nodes × #nodes containing estimated travel times
    time = Times[node1][node2]
    if time == 0 then
        time = 1e9
    end if
    return time
end function
```

5.5.3 A* - Test Cases

To verify the expected behaviour using A* one need to intelligently design suitable test cases. Thus, for a test case to be suitable, it has to test conformity of the graph representation to reality. That is, for any pair of ports, the A* algorithm should be able to suggest a feasible route. Ideally, every origin-destination port pair in existence would be scrutinised. Although, such extensive testing is not feasible. Instead, intelligent design has to be employed in order to form a suitable subset of origin-destination pairs; a subset of pairs covering all areas of interest during testing.

Research has uncovered an imbalance in the importance within the set of all maritime shipping routes. While there is an infinite number of possible routes, as alluded to above, only a subset are truly viable routes for global scale maritime shipping [39]. The central axis of the maritime structure connects North America, Europe, and Pacific Asia through the Suez Canal, the Strait of Malacca, and the Panama Canal. Transatlantic and transpacific shipping involves a large spread of ports. As a consequence, routes are not as set in stone as in the central axis structure. Trans-Indian ocean traffic serves predominantly as intermediary traffic from Pacific Asia to Europe, which implies a series of clearly defined routes between the Strait of Malacca and Bab el-Mandeb. Deviations from these standard routes are only viable in the case of coastal and local shipping. Figure 5.2 displays theses standard routes. A core route is defined as a route catering towards the major commercial shipping demands, and a secondary route is defined as a route connecting minor markets.

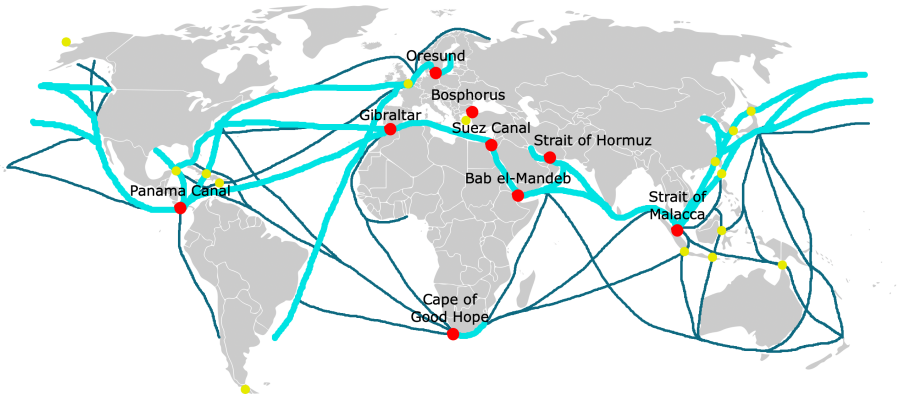


Figure 5.2: Main maritime routes divided up into core and secondary routes [40]. Red circles denote primary choke points, whereas yellow circles denote secondary choke points.

Furthermore, Figure 5.2 also visualises so-called *choke points*. Choke points are geographical locations that serve as motivators for the standard shipping routes; they are strategic locations through which vessels are routed to optimise the shipping structure. Thus, main routes are formed by routing via choke points. For a choke point to be classified as primary, it has to meet a certain criterion: There should be a limited set of cost-effective alternative routes, not routed via the choke point. The importance of well functioning major choke points is clearly illustrated by the Suez Canal blockage in 2021. Estimations reveal that by freezing the Suez Canal for six days, a loss of $10 billion each day is recorded [41]. Table 5.4 lists all primary choke points.

Choke point	Route
1: Bab el-Mandeb	Red Sea
2: Bosphorus	Black Sea
3: Cape of Good Hope	Atlantic — Indian Ocean
4: Gibraltar	Atlantic — Mediterranean Sea
5: Suez Canal	Mediterranean Sea — Red Sea
6: Strait of Hormuz	Persian Gulf
7: Strait of Malacca	Indian Ocean — South China Sea
8: Panama Canal	Atlantic-Pacific
9: Oresund	Baltic Sea — Nordic Sea

Table 5.4: Primary choke points and the naval route they occupy.

In order to subject the graph representation to meaningful test-cases, this knowledge of the maritime shipping structure was leveraged. The test set includes origin-destination pairs where the historic journey would have followed a standard naval route. Furthermore, origin-destination pairs were selected to test the willingness of routing via choke points. A well engineered graph representation will naturally lend itself to routing via choke points, as is the case in the real world scenario. In Table 5.5 the chosen subset of test-cases can be regarded. The first two columns display port codes; codes used to identify naval ports. Thus, each row is a test case, where the choke points tested are displayed in the third column.

Port 1	Port 2	Test
VNSGN (Ho Chi Minh City)	NLRTM (Port of Rotterdam)	1,4,5,7
USLGB (Long Beach)	CNTXG (Port Of Tianjin)	-
CNTXG (Port Of Tianjin)	DEHAM (Port of Hamburg)	1,4,5,7
HKHKG (Port of Hong Kong)	IRMRX (Port of Mahshahr)	6,7
USNYC (Port of New York)	KRPUS (Port of Busan)	8
SGSIN (Port of Singapore)	NLRTM (Port of Rotterdam)	1,4,5,7
BEANR (Port of Antwerp)	SAJED (Port of Jeddah)	4,5
RUPKC (Port of Petropavlovsk)	NLRTM (Port of Rotterdam)	1,4,5,7,8
CNLYG (Port of Lianyungang)	RUNVS (Port of Novorossiysk)	1,2,5,7
CNTXG (Port of Tianjin)	SENYN (Port of Nynäshamn)	1,4,5,7,9
SENYN (Port of Nynäshamn)	USHNL (Port of Honolulu)	8,9
IRMRX (Port of Mahshahr)	RUNVS (Port of Novorossiysk)	2,6
SAJED (Port of Jeddah)	CLVAP (Port of Valparaiso)	4,5,8
ZADUR (Port Of Durban)	ZACPT (Cape Town)	3

Table 5.5: Test cases used for evaluating the graph representation using the A* algorithm. Routes are tested both ways. Choke points are represented by their row number in Table 5.4.

Tests were conducted using the configuration of A* described above in Section 5.5. The resulting paths were saved and visualized.

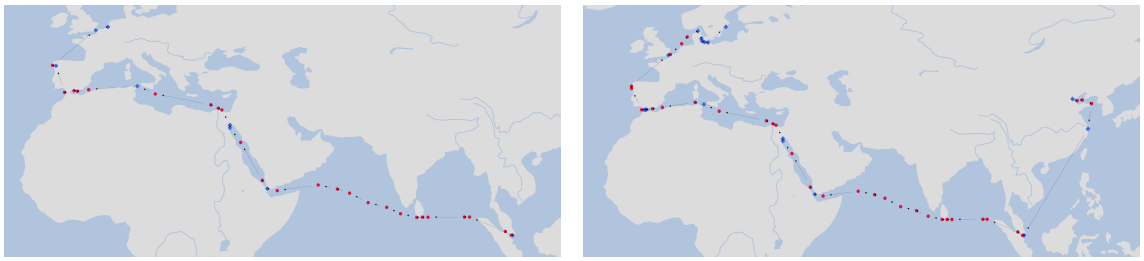
6

Results

This chapter presents the results obtained following the methodologies presented in the previous chapter (Chapter 5). Firstly, the results of evaluating the graph representation via A* are relayed. Thereafter, graph metrics pertaining to structural properties are presented. Results covering the inner workings of the graph representation follow last; detected change points, change point splitting results, waypoints extracted, and the transition matrix.

6.1 Graph Evaluation Using A*

The graph representation was subjected to evaluation through A* on the test cases stated in Section 5.5.3. The graph representation proved successful path-finding on all the test cases, without failing a single test case. In Figure 6.1a the A* path from the port of Singapore (SGSIN) to the port of Rotterdam (NLRTM) is visualised. The proposed path crosses multiple primary choke points: the Strait of Malacca, Bab el-Mandeb, the Suez Canal, and Gibraltar. A secondary choke point is also crossed in the form of the English canal. No expected choke points were omitted in the path.



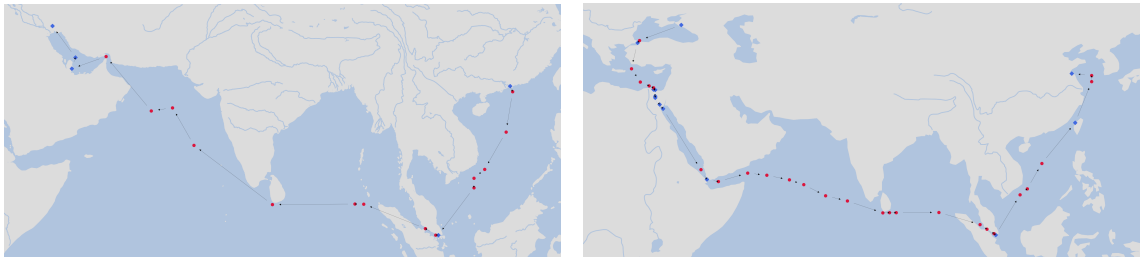
(a) SGSIN → NLRTM.

(b) CNTXG → SENYN.

Figure 6.1: A* paths: Port of Singapore to Port of Rotterdam, Port of Tianjin to Port of Nynäshamn. Ports visited on the path are marked by a polygon. Non-port nodes visited are marked by a circle.

In Figure 6.1b the A* path between the port of Tianjin (CNTXG) and the port of Nynäshamn (SENYN) is visualised. The port of Tianjin is the largest seaport in southern China, with a large volume of vessels journeying from here on export endeavours. The port of Nynäshamn is a smaller seaport, located in Sweden. The discovered A* path connects the two, omitting no expected choke point; The path is

routed via the secondary choke point in the China sea, through the Strait of Malacca, Bab el-Mandeb, the Suez Canal, Gibraltar, the secondary choke point the English canal, and finally Oresund. Note that the passage through the secondary choke point in the China sea is only implied, as the route visits a coastal port in China.



(a) HKHKG → IRMRX. (b) RUNVS → CNLYG.

Figure 6.2: A* paths: Port of Hong Kong to Port of Mahshahr, Port of Novorossiysk to Port of Lianyungang. Ports visited on the path are marked by a polygon. Non-port nodes visited are marked by a circle.

In Figure 6.2a the A* path from the port of Hong Kong (HKHKG) to the Iranian port of Mahshahr (IRMRX) is visualised. The discovered best path by A* resemble the core route in Figure 5.2. Also, the expected choke points of Strait of Hormuz and Strait of Malacca are visited.

In Figure 6.2b the suggested A* path from the Russian port of Novorossiysk to the Chinese port of Lianyungang is visualised. In this case, all expected choke points are traversed: Bosphorus, the Suez Canal, Bab el-Mandeb, the strait of Malacca. Furthermore, two secondary choke points are found on the suggested path.

(a) KRPUS → USNYC. (b) ZACPT → ZADUR.

Figure 6.3: A* paths: Port of Busan to Port of New York, Cape Town to Port of Durban. Ports visited on the path are marked by a polygon. Non-port nodes visited are marked by a circle.

In Figure 6.3a the A* path from the port of Busan (KRPUS) to the port of New York (USNYC) is visualised. The A* path closely mimics the core route in Figure 5.2. Furthermore, three secondary choke points are passed through together with traversal through the primary choke point, the Panama Canal. Lastly, in Figure 6.3b the A* path between two South African ports is visualised: Cape Town to Durban. The choke point Cape of Good Hope is included in the path.

Table 6.1 shows the estimated travel time for each test case; based on mean historic travel time as per Section 5.5.2.

Port 1	Port 2	Time 1 → 2 (hours)	Time 2 → 1 (hours)
VNSGN	NLRTM	103.17	74.55
USLGB	CNTXG	117.68	98.75
CNTXG	DEHAM	114.46	145.46
HKHKG	IRMRX	65.23	49.95
USNYC	KRPUS	203.44	174.48
SGSIN	NLRTM	91.97	92.52
BEANR	SAJED	64.82	63.97
RUPKC	NLRTM	142.73	131.45
CNLYG	RUNVS	76.16	155.40
CNTXG	SENYN	137.94	134.32
SENYN	USHNL	132.66	204.84
IRMRX	RUNVS	99.56	96.75
SAJED	CLVAP	96.42	128.83
ZADUR	ZACPT	10.77	27.40

Table 6.1: Estimated travel times (in hours) for the test cases.

6.2 Graph Metrics

The graph was constructed in accordance with Subsection 5.4. The resulting graph contained a total of 1748 nodes. Graph edges totalled in at 33 462 after construction via the transition matrix. The resulting graph is visualised in Figure 6.4. A red dot indicates the existence of a node at the specified location.

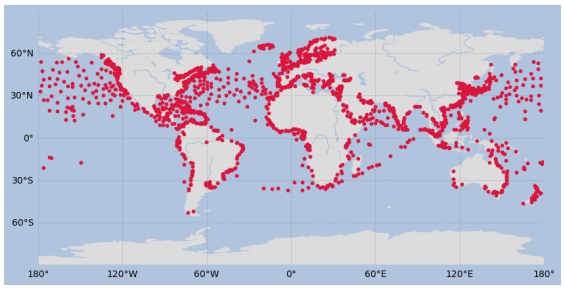


Figure 6.4: A plot of all nodes in the graph representation of the maritime structure. Edges are omitted for ease of viewing.

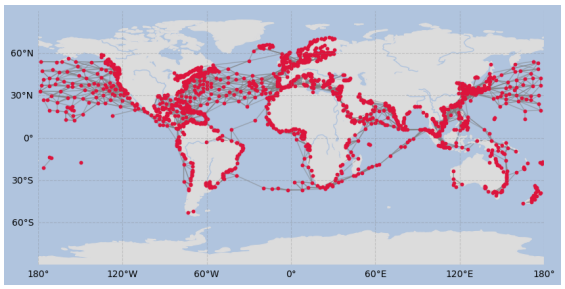


Figure 6.5: A plot of all nodes and edges in the graph representation of the maritime structure. Edges are directed in the actual graph; here, direction is omitted.

As stated above, there were a total of 7288 edges. Almost all of these are visible in Figure 6.5. Edges where nodes have a difference in longitude greater than 350 are omitted in the plot. These are edges that cross the International Date Line (180th meridian), and would have spanned the entire plot. In Section 6.6 it is mentioned

that the transition matrix has a *sparsity ratio* of 0.00101. In graph theory, *density* is a measurement of how dense connections are within a graph. To calculate density, the below equation is utilised (Equation 6.1):

$$Density = \frac{\#Edges}{\#PossibleEdges} \tag{6.1}$$

Calculating the density of the graph representation yields 0.00239, which is an increase when compared to the sparsity ratio of the transition matrix; a natural consequence of multiple waypoints having the same coordinates.

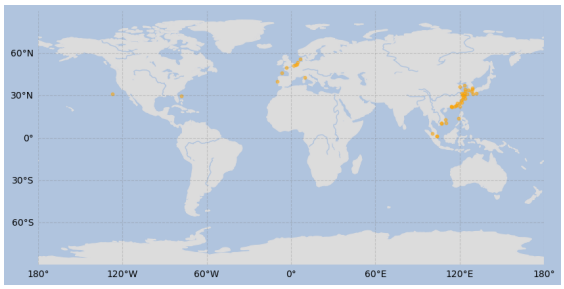
Another metric is *Average Clustering Coefficient* (*ACC*). Ranging from 0 to 1 the *ACC* tells the average tendency of nodes to cluster. In the case of the graph representation, an *ACC* of 0.333 could be recorded. This *ACC* indicates that the nodes tend to form local clusters to a degree. In other words, nodes can form clusters, but connections outside immediate neighbours also occur.

Metric	**Value**
Density	0.00239
ACC	0.333
Assortativity	0.480
Reciprocity	0.718
Degree Centrality (Mean)	0.005
Betweenness Centrality (Mean)	0.004
Closeness Centrality (Mean)	0.032

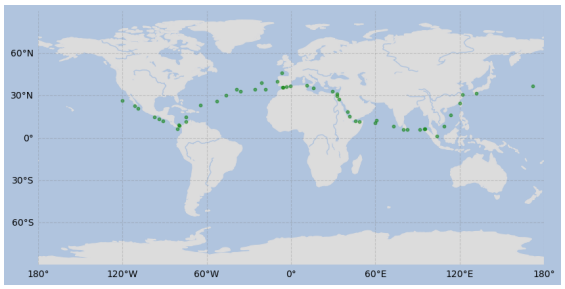
Table 6.2: Metrics concerning the graph representation.

In Table 6.2 the average degree-, betweenness-, and closeness centrality as per Section 5.5.1, are displayed together with the ACC, density, assortativity, and reciprocity. As can be noted, the assortativity coefficient of the graph representation was calculated as 0.480 and the reciprocity coefficient was calculated as 0.718.

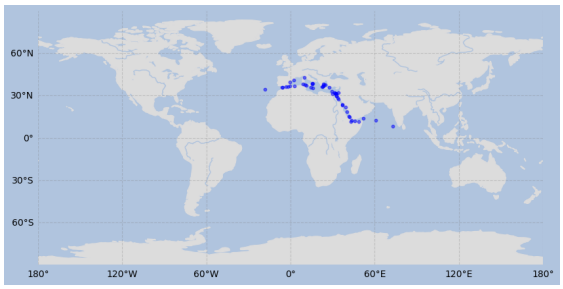
In Figure 6.6, the top 50 nodes with respect to degree centrality, betweenness centrality, and closeness centrality, are visualized.



(a) Degree Centrality.



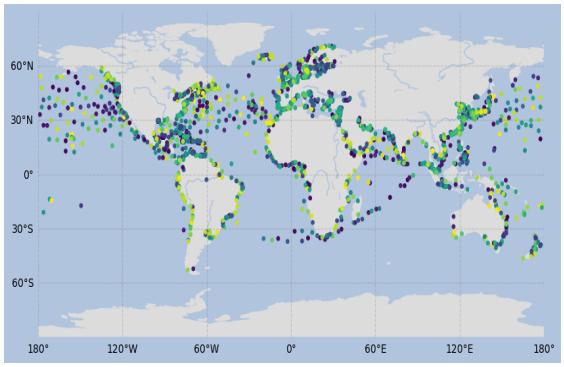
(b) Betweenness Centrality.



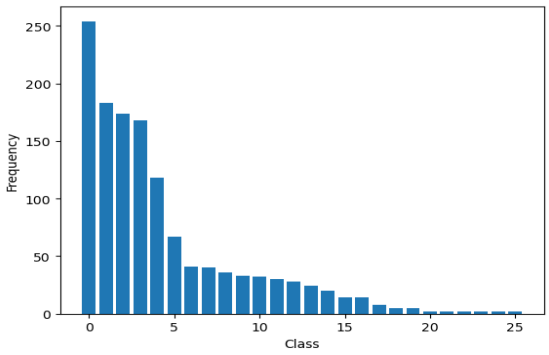
(c) Closeness Centrality.

Figure 6.6: Scatter plots of the 50 most important nodes with respect to degree, betweenness, and closeness centrality.

Since the ACC alluded to the tendency for some clustering, community detection was performed in the pursuit of uncovering any communities/modules in the graph. The uncovered communities can be regarded in Figure 6.7a, coupled with a bar plot over class frequency seen in Figure 6.7b.



(a) The results of community detection in the graph representation. Each colour represents belonging to a different class.



(b) The frequency of which community classes occur. A few classes dominate.

Figure 6.7: Visuals of the community detection on the graph representation.

6.3 Change Point Detection

The first step of construction of the node set for the graph representation is to apply change point detection as presented in Subsection 5.2.1. The task of change point detection aimed to single out a subset of the total dataset; only a select few data points display significant changes in speed or course behaviour. Figure 6.8 shows the final data points identified as change points.

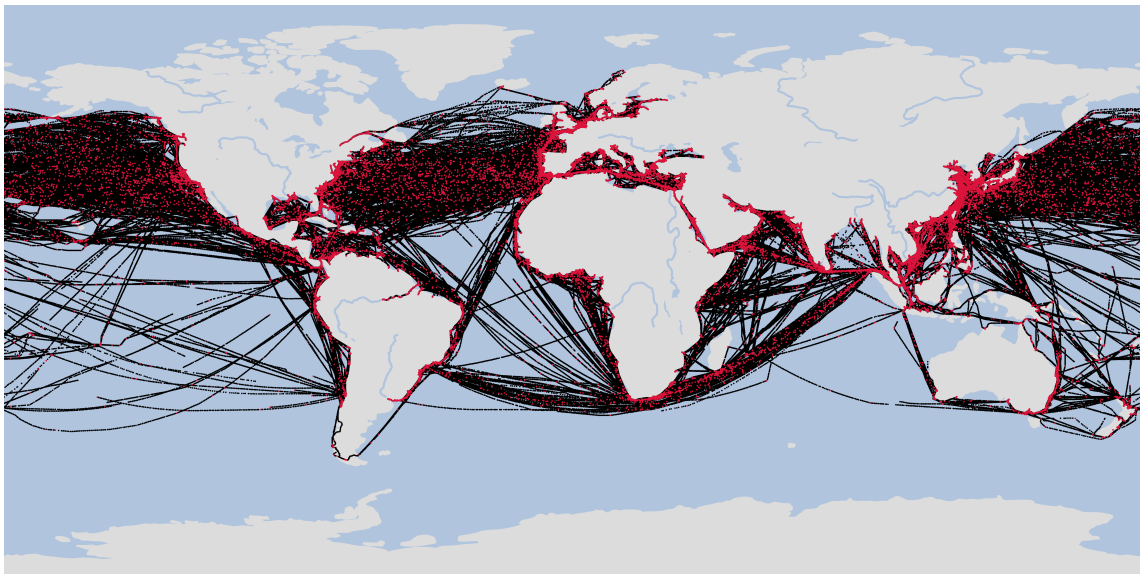


Figure 6.8: Results of change point detection on AIS dataset. The red dots represent identified change points, and the black markings portray all maritime traffic extracted from the AIS data.

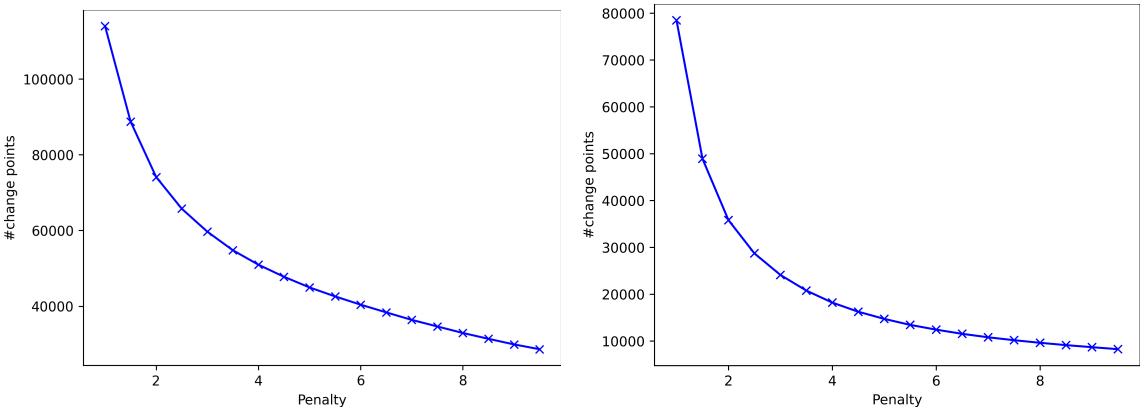
Compared to the original AIS dataset, which consisted of 1 973 916 data points, the change point set was reduced to a total of 130 470 data points. Thus, dimensionality could be reduced whilst keeping all important data necessary for waypoint identification. Of the total change points detected, 61 715 were solely course change points, 49 679 were speed change points, and 19 076 were both course and speed change points (Table 6.3).

Data	#points
AIS data	1 973 916
Change Points	130 470
Course Change Points	61 715
Speed Change Points	49 679
Course & Speed Change Points	19 076

Table 6.3: Number of change points compared to total number of AIS data points. Row 3-4 is the number of points, where a point is a change point based on either the course or speed value, while the last row is based on both values.

Figure 6.9 shows the resulting plots of the elbow method that was used to decide

the penalty value for cog and sog. The penalties range from 0 to 10 with a step of 0.5. From the figures, it seems like the elbow of the curve is at value 2.5 for sog and 3 for cog. But by testing other values around those, a penalty value of 2.3 for sog and 2 for cog showed the best results.



(a) Chosen penalty value for sog: 2.3 (b) Chosen penalty value for cog: 2

Figure 6.9: Elbow method for choosing the penalty parameter for change point detection.

The Figure 6.10, 6.11 displays the signals (sog, cog) from four different vessels. The start of each segment of alternating colour represents at which points the algorithm detects the change points. In the top figure in of the sog example, the algorithm performs well where the 'blocks of signals' are properly segmented. However, in the bottom figure with less occurring 'blocks' the algorithm performs inadequately where it misses to segment the two rightmost blocks and instead segments a seemingly flat part.

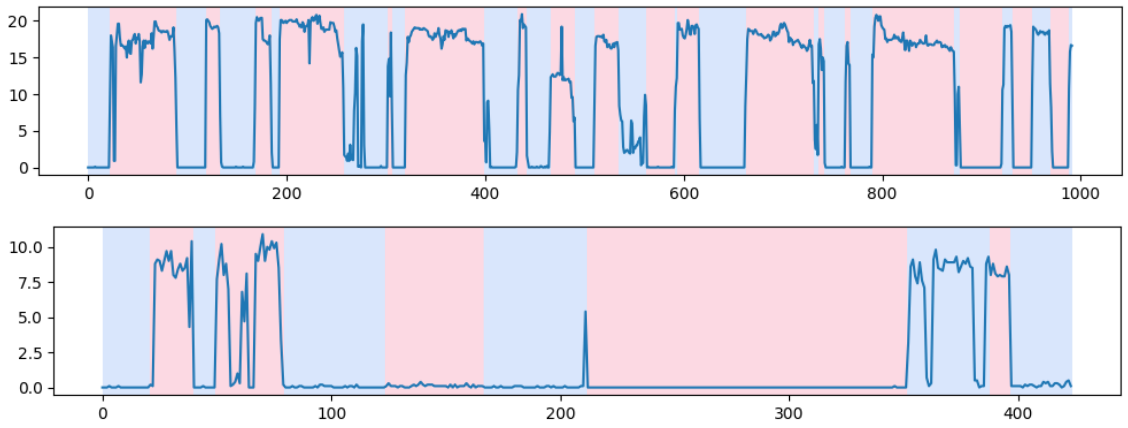


Figure 6.10: Change point detection on sog (blue lines). The detected change points are the start of each segmentation block.

In the change point detection on cog, the top figure shows again good performance of the algorithm where it was able to segment tightly and noisy signals. However, the

bottom figure shows an example of drastically varying signals where the algorithm performed worse.

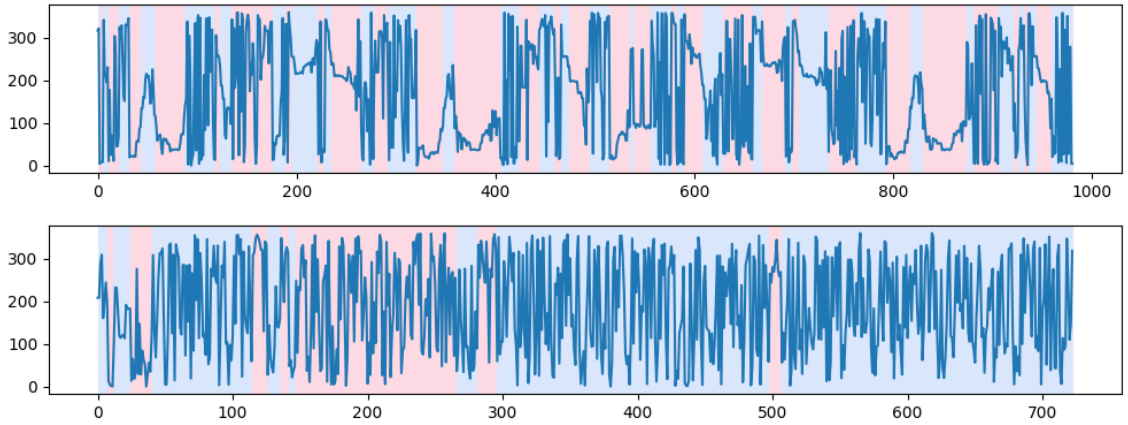


Figure 6.11: Change point detection on cog (blue lines). The detected change points are the start of each segmentation block.

6.4 Data Splitting of Change Points

After running the KDE on the change point set, the results are shown in Figure 6.12. The yellow to green spots shows where the density of points is the highest and can, thus, be classified as dense areas. Around the dense areas, some faint light blue spots can be seen where it becomes fainter across large waters. These areas can further be split into mid dense and sparse areas.

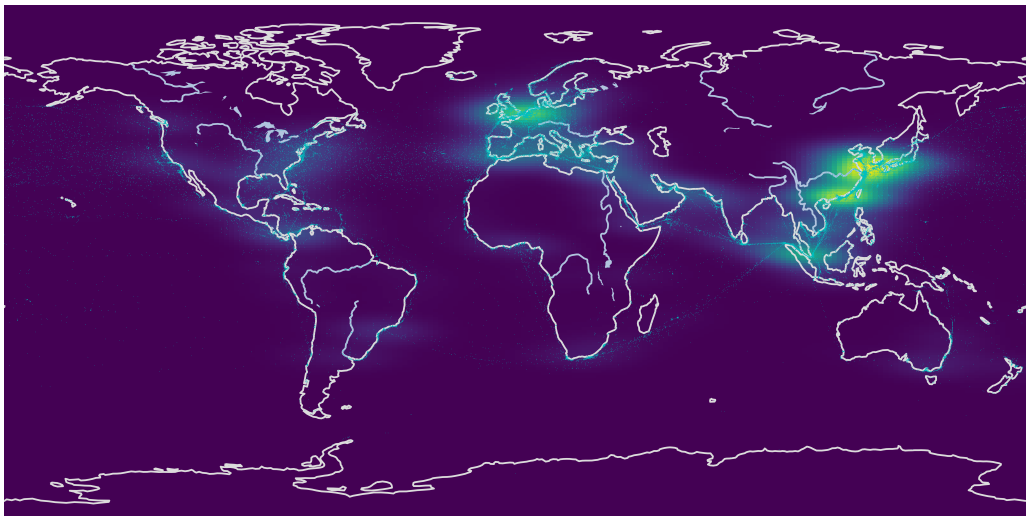


Figure 6.12: Heat map of the result from KDE of change points. The change points are also plotted as scatter points in the figure.

From observing the density plot above, the change points were split as described in Subsection 5.2.2, into three datasets by the following areas with the respective

thresholds seen in Table 6.4. The resulting split of change points can be seen in the Figure 6.13 where the points belonging to the sparse areas were coloured green, the mid-points were coloured pink, and the dense points were coloured blue. Observing further, it can be seen that the dense areas covers the majority of the world's vital waterways for global trade: the English Channel, Malacca strait, Hormuz strait, Suez Canal, and the Panama Canal.

	Min	**Max**
Sparse	0	0.09
Mid	0.09	0.28
Dense	0.28	~ 1.62

Table 6.4: The *min* and *max* threshold pdf values for a point to be categorised as one of the following regions: sparse, mid, and dense.

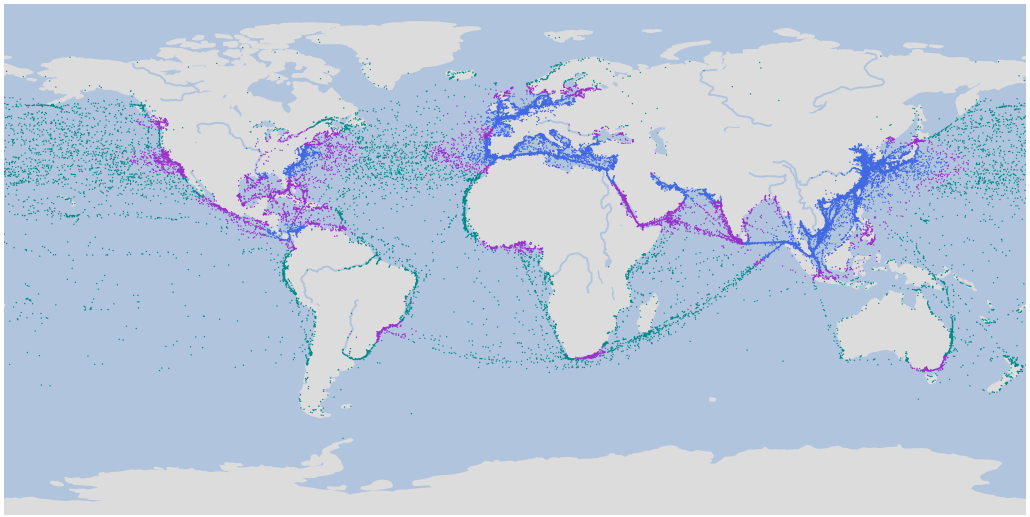


Figure 6.13: Combined map plot of change points filtered by KDE. Sparse (green), mid (purple), and dense (blue).

6.5 Extraction of Waypoints

The waypoints were extracted using the methods described in Subsection 5.2.3. For each data set, the maximum and minimum cell levels and threshold for point count were set as in Table 6.5. The resulting waypoints after combining the waypoints from each split of change point set can be seen in Figure 6.14. A total of 2686 waypoints, which is approximately, $\sim 2\%$ were extracted from the change point set where the cell levels ranged from 4 to 8. The larger cells/polygons (blue) are positioned in sparse and mid dense areas, while the smallest cells (green) are positioned in the denser areas.

	level	min_level	threshold
Sparse	7	4	8
Mid	8	5	6
Dense	8	6	30

Table 6.5: Final chosen values for the variables in Table 5.3 used for Algorithm 1.

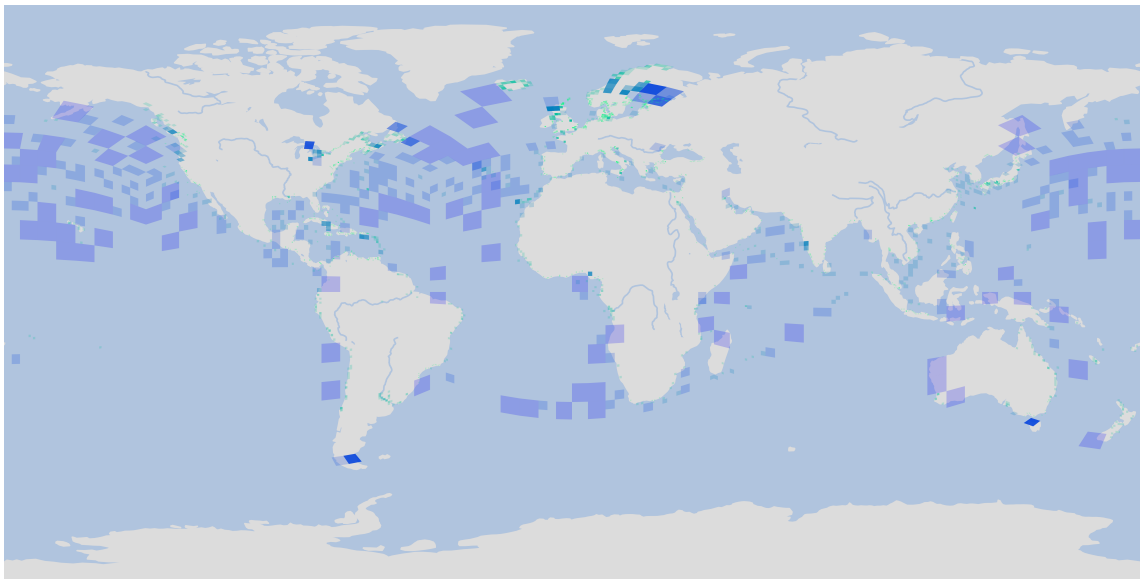
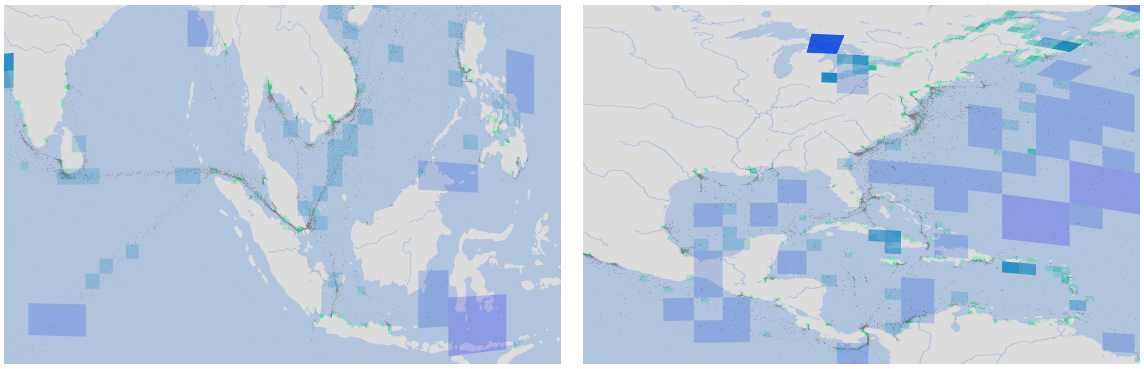


Figure 6.14: Map plot showing waypoints as polygons of different levels ranging from level 4-8.



(a) The Malacca Strait.

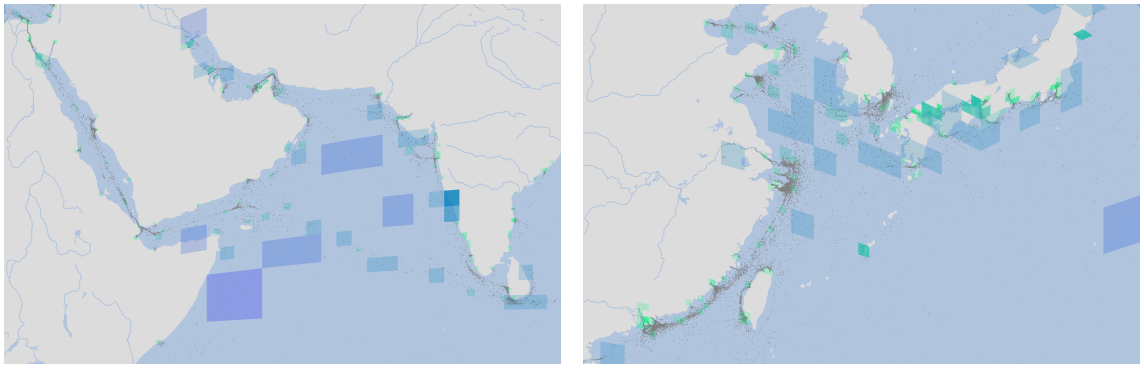
(b) The Panama Canal.

Figure 6.15: Map plot showing two of the vital waterway regions with the extracted waypoint in Figure 6.14 with scatter points of the AIS data.

Four examples focused on the East Asia region, Malacca Strait, Panama Canal, Suez Canal and Hormuz Strait can be seen in Figure 6.15, 6.16. The figures show the location of the waypoints on a map with the respective AIS data in those regions. The cells of varying sizes are located in a way that encompasses some data points.

Some cells are positioned in a way where the points are gathered in the middle, meanwhile some cells are positioned on the edge, barely encompassing any or no points at all.

While the larger cells covers more points, some of them also covers multiple routes, as seen below the Panama Canal in the second figure of Figure 6.15. Moreover, even though some smaller cells are perfectly located there are some that do not cover all the points in a cluster as seen in the left figure below to the right and below the Suez Canal. All of this demonstrates the difficulty of choosing the values for the cell levels and point count thresholds in such a way that perfectly encompass all the points of interest in each cell and do not cover too much area.



(a) The Suez Canal and Hormuz Strait. (b) East Asia region.

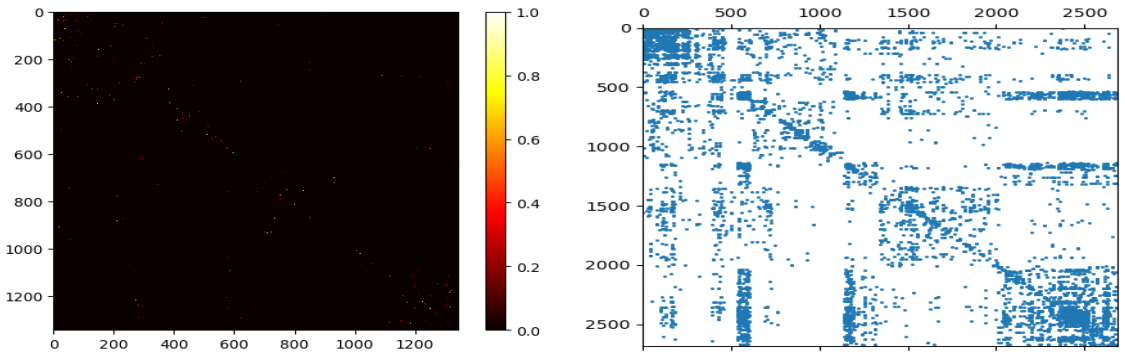
Figure 6.16: Map plot of the waypoints in the East Asia region and around the Suez Canal and Hormuz Strait.

6.6 Transition Matrix

Following the proceedings detailed in Subsection 5.3.1 a transition matrix could be formed from which edges will be extracted in the graph representation. There were a total of 2686 waypoints resulting from the previous steps in the process. As such, the transition matrix took the shape of a 2686×2686 matrix.

In Figure 6.17a one can regard a heat map representation of the transition matrix. The transition matrix is sparse, as can be deduced from the figure. The level of sparsity is further clarified in Figure 6.17b. Displaying the sparsity pattern of the transition matrix, Figure 6.17b allows for an understanding of any patterns present in the transition matrix.

Summing up non-zero transition probabilities yields the following sparsity ratio: $7288/7\,214\,596 \simeq 0.00101 = 0.101\%$ - 0.101% of transition probabilities are non-zero. Furthermore, it should be noted that even though the sparsity pattern might appear symmetrical at first glance, it is not. Despite the high prevalence of zero values, it can be noted that there are white dots present in the heat map. Therefore, there exist elements with a value of 1; the probability of transition on this edge is 100% when starting at the row-index node. All in all, there are a total of 140 such node pairs where the probability of transition is 100%.



(a) Max-pooled heat-map of values in transition matrix. Dimensions are reduced two-fold using a pooling window of size 2.

(b) The sparsity pattern of the transition matrix. Blue signifies a non-zero value. Note that markers are enlarged for ease of viewing.

Figure 6.17: Transition matrix plots detailing the sparseness of it.

As can be seen in Figure 6.17a, there are transition probabilities in the range $0 < p < 1$. This is further illustrated in Figure 6.18 where it shows the distribution of the transition probabilities. Because of the sparsity, most transition probabilities are 0. If there are to be no self loops in the graph representation, all diagonal transition probabilities should be 0. Calculation of the diagonal elements in the transition matrix reveal them all to be 0.

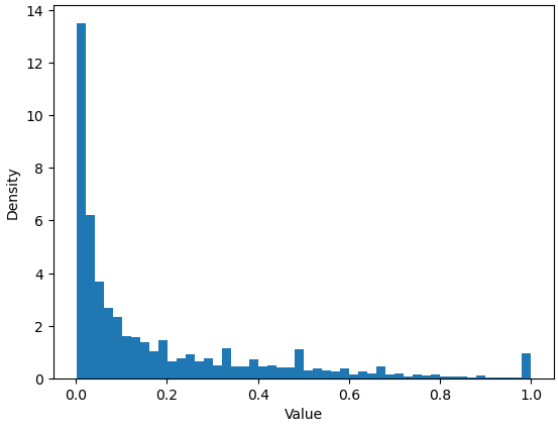
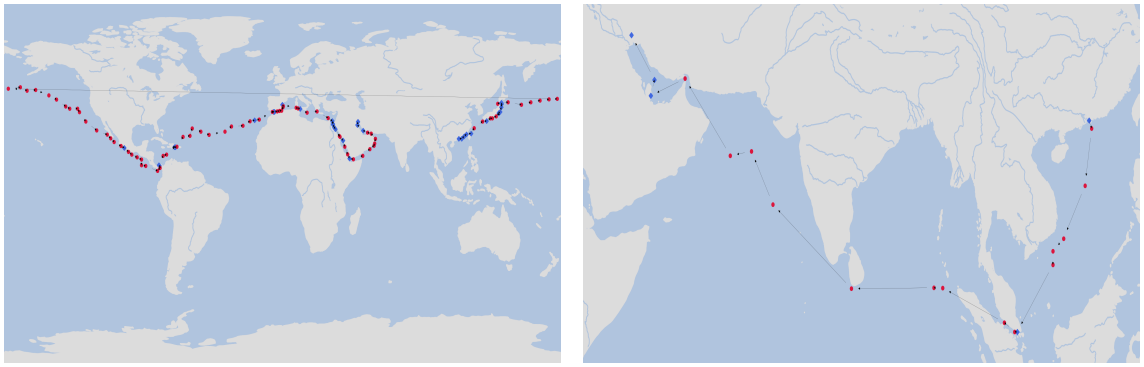


Figure 6.18: Density plot of the transition matrix values.

Position Time Frequency

The results of using different frequencies when constructing the edge set is seen in the following two examples of routes from Port of Hong Kong to Port of Mahshahr (Figure 6.19), and Port of Busan to Port Of Tianjin (Figure 6.20) with frequency of 1 to 2 days. In the first example, a shorter frequency of one day led to the path finding algorithm choosing a longer path by crossing the Pacific Ocean instead of directly passing through the Malacca Strait as in the same example but with frequency of 2 days. Comparing the two routes, it can be seen that the path in the left figure contains more waypoints and shorter edges, whereas the right figure takes longer

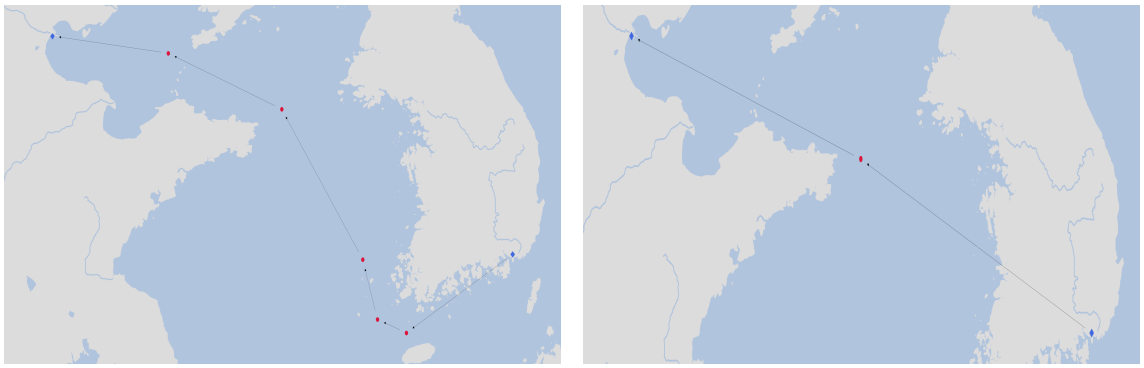
edges between the waypoints leading to fewer nodes. The first can lead to a vessel stopping by more, unnecessary waypoints before moving on, while the second can lead to a vessel passing through fewer waypoints, making the path illogical and missing important stops.



(a) Frequency of 1 day. (b) Frequency of 2 days.

Figure 6.19: Port of Hong Kong (HKHKG) to Port of Mahshahr (IRMRX).

This is further showcased in the second example with a route starting in South Korea to China, where it can be seen that a shorter frequency can also lead to better fitting paths. In the left figure, the algorithm chose to travel around the country and seemingly follows the historical data paths well, compared to the right figure where the algorithms instead chose to cross the country and only stop in a middle waypoint before reaching the port of destination. These differences however are not specifically illogical compared to the first example where the results are greatly different.



(a) Frequency of 1 day. (b) Frequency of 2 days.

Figure 6.20: Port of Busan (KRPUS) to Port Of Tianjin (CNTXG).

7

Discussion

In Section 1.2 the purpose of this project was described as 'to construct a close-to-reality graph representation of the maritime shipping network structure'. In this chapter, the results (Section 6) are discussed through the lens of the stated purpose, beginning with the results pertaining to evaluation of the graph representation using the A* algorithm.

7.1 Graph Evaluation Using A*

In Figures 6.1a–6.3b and Appendix A.1 the resulting A* paths are visualised. These paths were formed from subjecting the graph representation to a set of test cases (Section 5.5.3), all designed to test choke points and core routes. The first test (Figure A.1), testing the path between the port of Rotterdam and the Vietnamese port of Ho Chi Minh City, showed no sign of deviation from the expected core route. However, note that the node distribution is not equal both ways. The nature of the two paths are identical, where the same core routes are being traversed through the same choke points. The selected nodes, on the other hand, differ. Since both subsets of utilised nodes represent the historical route without any deviations, one representation is not to prefer over the other.

Although, nodes serving the same purpose might benefit from being combined. This argument is further supported by Figure A.3a, which displays the suggested A* path from Tianjin port to port Hamburg; after leaving the port of origin, no node is visited until arrival at the Strait of Malacca. Other test-cases (e.g., Figure A.4) reveal the existence of nodes in this area. Furthermore, seeking the shortest path, no drastic manipulation of trajectory should occur in this part of sea. As such, it is safe to assume theses existing nodes could be utilised in cases like this one (See also Figure A.10a for another similar situation involving Tianjin port). Another mitigation might involve considering shortening edges in the china sea.

In Figure A.5 the paths between Long Beach and port of Busan are visualised. Here the strong preference for the Suez Canal, seen in previous test cases, is replaced with preference for the Panama Canal. Still, core routes are being traversed. This suggested A* path might be of preference since only one canal has to be travelled, as compared to the one canal and three straits in the Suez path.

The test case of the journey between the port of Rotterdam and the Russian port of

Petropavlovsk (Figure A.8) sets itself apart; the two paths do not follow the same core route. Such a result might be caused by differing intermediary stops in the historic AIS data. If certain ports are typically visited together with the destination when initiating the journey from the origin, then the suggested A* path will be biased to one containing these ports.

All in all, core routes are pursued without fail and the expected choke points for each test are all included on their respective paths. The graph representation accurately models global maritime shipping as far as the routes are concerned.

In Table 6.1 the estimated travel times for each test case is listed. The listed travel times are optimistic estimations. Estimations provided by Maersk reveal maritime shipping usually require a span of 20 to 45 days [42]. The reason behind the optimistic estimates in Table 6.1 can be attributed to wait-times. When the graph representation is scrutinised via A*, no overhead time is added to account for time spent in ports waiting. To more accurately model the travel times between ports, further work is required. Since the travel times are directly derived from A* and the time based heuristic function, a change in the heuristic will in turn change the estimated travel times. If the heuristic function was to be optimised to return more accurate times, it is possible some A* path suggestions would change. There might be some geographical locations featuring heavy wait-times; these locations would then be favoured to a lesser degree. There is also the possibility that the degree of which intermediary ports are included in the A* path would decrease — In 2021, container vessels spent an average of 0.8 days (19.2 hours) in a port during a port call [43].

7.2 Network Metrics

In Table 6.2 various different metrics obtained from the graph representation are presented. Firstly, *Density* is recorded as 0.00239; the graph representation can be concluded to be very sparse.

Secondly, *ACC* was measured to be 0.333. ACC is a coefficient ranging from 0 to 1. 0.333 means that, on average, for any given node in the graph representation, $n \in G$, 0.333 of its neighbours are also neighbouring each other. The ACC not being closer to 1 is correlated to the density: there are not enough edges to form densely connected clusters. In a maritime shipping network graph representation, an extreme ACC would probably not be desirable since it would create redundancy in routes.

As for the *reciprocity*, a value of 0.718 means that 71.8% of edges go both ways. This value would tend towards the maximum 1 if nodes are combined as discussed in Section 7.1.

The mean *Degree Centrality* of 0.005 indicates that the average node does not exhibit hub-like structure. A result caused by the average node being an intermediary step on a core route. In Figure 6.6a the nodes with most hub-like structure are visualised. It can be noted that there is a disproportionate amount of hub-like nodes in China/Asia, which is synonymous with real world maritime shipping.

Figure 6.6b displays the top 50 nodes as seen to betweenness centrality. These match up exactly with the central axis of maritime traffic (Section 5.5.3). This result indicates that the graph representation has been able to capture the core routes seen in Figure 5.2.

7.3 Extraction of Waypoints

The results from extracting waypoint cells using S2 Geometry (Section 6.5), shows the difficulty of choosing the optimal thresholds and cell levels for each density dataset. Considering how the edge set is later constructed, an optimal cell is one such that it covers the majority of the data points at the area of interest. However, some resulting cells were wrongly positioned where it barely misses the cluster of points and some cells covered too many points. This further shows the difficulty of choosing thresholds but could also be due to the algorithm for adding waypoints. As the algorithm skips adding waypoints whose parent cells already exists in the waypoint tree, this leads to that only the first smallest child cell of the four children to represent the area of interest, while perhaps the second-smallest child was a better representation. A possible solution would be to consider all children and calculate the mean of the cells to choose the optimal cell.

Other reasons directly affecting the result of the waypoint extraction is the density data splitting (Section 6.4), and change point detection (Section 6.3). In this study, three different density levels were chosen where the split could be directly evaluated when choosing the threshold and cell levels in the subsequent step of waypoint extraction. Choosing thresholds to obtain optimal partitions showed to be difficult where each partition contained varying density level of points, making it later difficult to threshold the point count for each partition. This could be alleviated by using some method to choose the optimal number of partitions. Another possible solution is to make the data splits more homogeneous by removing data points in denser areas.

The change point detection could also further be improved. The change point detection algorithm performed well while using the elbow method to choose the penalty value for each of the features cog and sog. Examples of the detected change points (Figure 6.10, 6.11), shows the algorithm being able to seemingly identify the changes, but the algorithm also shows difficulty in some other cases. Compared to sog where a vessel's speed are mostly constant or have obvious changes depending on if the vessel is travelling in large water or inside a highly trafficked area with a speed limit, the course is adjusted more often depending on various factors such as the weather, wind, or avoiding obstacles. This is shown by the oscillating lines in Figure 6.11 indicating constant adjustment of vessel course. However, the fluctuations could also be attributed to the course value ranging from 0 to 360°, implying that the vessel is experiencing a small fluctuation in course angled around 0 and 360°. This could be due to the wrapping of the angle between time steps or unrepresentative journeys, as the data is only grouped by vessel IMO (Section 5.2.1). This issue was not tended to because of timing constraints imposed on the project, thus no further experiments on the grouping was be performed. A possible solution to alleviate the

problem would be to apply cos and sin transformations to the cog before Change Point Detection, or also grouping by *port of loading* (*pol*) and *port of destination* (*pod*) in the groupby step. Improving the change point detection, further processing the change points or the original dataset, would lead to more defined and qualitative change points, which would facilitate waypoint extraction in the later steps.

7.4 Transition Matrix

In Figure 6.17 the sparseness of the transition matrix is visualised. There is a strong possibility that the sparseness is a natural product of modelling maritime traffic. Since the transition matrix is derived from historical vessel journeys (Algorithm 2), it can be hypothesised maritime traffic exhibits sparse behaviour. That is, for every possible vessel position, there should only be a couple of possible alternatives for the next position. This theory finds reason in the fact that a vessel cannot travel from position p_0 to position p_1 without first covering any intermediary positions between the two; a vessel can only select its next position as one directly next to its current position.

The results showcase the effect of using different position time frequency when grouping the journeys for each vessel (Figures 6.19 - 6.20). A smaller frequency leads to vessels visiting more waypoints, following more logical routes and not visually crossing land. Comparatively, a larger frequency allows the A* algorithm to find more paths, although at the price of less detailed routes in highly trafficked areas. However, the paths found by the larger frequency are also occasionally more viable as compared to the smaller frequencies. A smaller frequency leads to a longer path crossing the whole world in the case of routing from Hong Kong to Iran, instead of the optimal, shorter path (Figure 6.19). Another problem with a smaller frequency is the case of not finding paths in some cases. This could be due to the frequency not being large enough, and as such not enough 'subsequent' waypoints are found in the transition matrix algorithm (Algorithm 2). It then follows that edges are not formed to a high enough degree in the construction of the graph. This could also be a contributing factor in decreasing performance in too small frequencies; promoting the case with shorter frequency to pick an illogical path, as there are no edges forming the optimal/historical path. There seem to be a trade-off between obtaining detailed paths and obtaining more and optimal paths. A possible solution would entail the implementation of a dynamic frequency when constructing the transition matrix, such that the frequency varies depending on the data points. Moreover, it could be worth looking into preprocessing the dataset depending on the number of data points for each vessel journey and frequency, or further processing of the edge set by manually removing illogical edges but instead get to use a common frequency that performs approximately well.

8

Conclusion

In this project, a graph model representing the global maritime shipping network has been constructed and evaluated for likeness to real maritime traffic. Specifically, the graph representation accurately models the core aspects of maritime traffic. Evaluating the graph representation using the path-finding algorithm A*, it was found that the suggested paths all included expected strategic geographical locations.

The graph representation was constructed utilising historical AIS data gathered from naval vessels. Thus, the structure of the graph representation reveal insights about the actual maritime shipping structure; insights that could be utilised in the pursuit of a more environmentally sustainable maritime sector.

Although the project provides important intel concerning the maritime shipping industry, the project has limitations in its applicability. Since the focal point lie on replicating the global maritime structure from AIS data spread across the globe, the graph representation does not suit itself for usage in local waters; path-finding and structural intel will lack accuracy.

In conclusion, the project demonstrates a maritime shipping network graph representation well suited for analysis and path-finding in global waters. The graph representation has usability both for shipping companies in need of routing on global waters and for analysis of the maritime shipping structure. Further work is required to increase accuracy in local waters.

Bibliography

[1] Q. Meng, S. Wang, H. Andersson, and K. Thun, "Containership routing and scheduling in liner shipping: Overview and future research directions," *Transportation Science*, vol. 48, no. 2, pp. 265–280, 2014.

[2] C. Ducruet, "The geography of maritime networks: A critical review," *Journal of Transport Geography*, vol. 88, p. 102 824, 2020.

[3] M. Christiansen, E. Hellsten, D. Pisinger, D. Sacramento, and C. Vilhelmsen, "Liner shipping network design," *European Journal of Operational Research*, vol. 286, no. 1, pp. 1–20, 2020.

[4] DataVersity. "The growing impact of ai on data science in 2023." (2023), [Online]. Available: `https://www.dataversity.net/the-growing-impact-of-ai-on-data-science-in-2023/` (visited on 05/07/2024).

[5] K. McElheran, J. F. Li, E. Brynjolfsson, *et al.*, "Ai adoption in america: Who, what, and where," *Journal of Economics & Management Strategy*, 2024.

[6] Google Scholar. "Google scholar search query: Maritime shipping ai." (), [Online]. Available: `https://scholar.google.com/scholar?hl=sv&as_sdt=0%2C5&q=maritime+shipping+AI&oq=maritime+shipping` (visited on 05/16/2024).

[7] C. Ducruet and T. Notteboom, "The worldwide maritime network of container shipping: Spatial structure and regional dynamics," *Global networks*, vol. 12, no. 3, pp. 395–423, 2012.

[8] X. Li, P. Zhang, and G. Zhu, "Dbscan clustering algorithms for non-uniform density data and its application in urban rail passenger aggregation distribution," *Energies*, vol. 12, no. 19, p. 3722, 2019.

[9] S. Global. "Maritime case studies." (2023), [Online]. Available: `https://spire.com/case-studies/?filterbycategory=maritime&utm_term=ais&utm_campaign=Search+AIS+BM&utm_source=adwords&utm_medium=ppc&hsa_acc=4934961383&hsa_cam=18028798704&hsa_grp=141497870113&hsa_ad=616244029423&hsa_src=g&hsa_tgt=kwd-12477886&hsa_kw=ais&hsa_mt=b&hsa_net=adwords&hsa_ver=3&gad_source=1&gclid=Cj0KCQiA67CrBhC1ARIsACKAa8TjCfjZ_HRbjEJnte_r0YEkJMU5Ubnw8d4ly8W9uG5XhkMOwoxT1-caAvTREALw_wcB` (visited on 12/03/2023).

[10] Google. "S2 geometry library." (2017), [Online]. Available: `https://github.com/google/s2geometry` (visited on 04/04/2024).

[11] Google. "S2 cells." (2017), [Online]. Available: `http://s2geometry.io/devguide/s2cell_hierarchy` (visited on 04/04/2024).

[12] Google. "Earth cube." (2017), [Online]. Available: http://s2geometry.io/resources/earthcube.html (visited on 04/04/2024).

[13] Met Office, *Cartopy: A cartographic python library with a matplotlib interface*, Exeter, Devon, 2010 - 2015. [Online]. Available: https://scitools.org.uk/cartopy.

[14] R. Killick, P. Fearnhead, and I. A. Eckley, "Optimal detection of changepoints with a linear computational cost," *Journal of the American Statistical Association*, vol. 107, no. 500, pp. 1590–1598, 2012.

[15] G. Schwarz, "Estimating the dimension of a model," *The annals of statistics*, pp. 461–464, 1978.

[16] Y.-C. Chen, "A tutorial on kernel density estimation and recent advances," *Biostatistics & Epidemiology*, vol. 1, no. 1, pp. 161–187, 2017.

[17] M. L. Waskom, "Seaborn: Statistical data visualization," *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, 2021. DOI: 10.21105/joss.03021. [Online]. Available: https://doi.org/10.21105/joss.03021.

[18] C. R. Harris, K. J. Millman, S. J. van der Walt, *et al.*, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. DOI: 10.1038/s41586-020-2649-2. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2.

[19] B. W. Silverman, *Density estimation for statistics and data analysis*. Routledge, 2018.

[20] D. W. Scott, *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons, 2015.

[21] D. Yang, K. Pan, and S. Wang, "On service network improvement for shipping lines under the one belt one road initiative of china," *Transportation Research Part E: Logistics and Transportation Review*, vol. 117, pp. 82–95, 2018.

[22] L. Liu, R. Shibasaki, Y. Zhang, N. Kosuge, M. Zhang, and Y. Hu, "Data-driven framework for extracting global maritime shipping networks by machine learning," *Ocean Engineering*, vol. 269, p. 113 494, 2023.

[23] D. Filipiak, K. Węcel, M. Stróżyna, M. Michalak, and W. Abramowicz, "Extracting maritime traffic networks from ais data using evolutionary algorithm," *Business & Information Systems Engineering*, vol. 62, no. 5, pp. 435–450, 2020.

[24] M. Le Tixerant, D. Le Guyader, F. Gourmelon, and B. Queffelec, "How can automatic identification system (ais) data be used for maritime spatial planning?" *Ocean & Coastal Management*, vol. 166, pp. 18–30, 2018.

[25] VesselFinder. "Ais navigation status." (2024), [Online]. Available: https://api.vesselfinder.com/docs/ref-navstat.html (visited on 04/16/2024).

[26] VesselFinder. "Ais ship types." (2024), [Online]. Available: https://api.vesselfinder.com/docs/ref-aistypes.html (visited on 04/16/2024).

[27] VesselFinder. "Ais dataset." (), [Online]. Available: https://api.vesselfinder.com/docs/response-ais.html (visited on 05/21/2024).

[28] AISHub. "Ais data api." (), [Online]. Available: https://www.aishub.net/api (visited on 05/21/2024).

[29] C. Truong, L. Oudre, and N. Vayatis, "Ruptures: Change point detection in python," *arXiv preprint arXiv:1801.00826*, 2018.

[30] ruptures. "Kernel change point detection: A performance comparison." (), [Online]. Available: https://centre-borelli.github.io/ruptures-docs/examples/kernel-cpd-performance-comparison/ (visited on 05/09/2024).

[31] P. Virtanen, R. Gommers, T. E. Oliphant, *et al.*, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020. DOI: 10.1038/s41592-019-0686-2.

[32] J. Gillham *et al.* "S2sphere." (2017), [Online]. Available: https://pypi.org/project/s2sphere/ (visited on 04/04/2024).

[33] T. pandas development team, *Pandas-dev/pandas: Pandas*, version v2.2.2, Apr. 2024. DOI: 10.5281/zenodo.10957263. [Online]. Available: https://doi.org/10.5281/zenodo.10957263.

[34] W. McKinney, "Data Structures for Statistical Computing in Python," in *Proceedings of the 9th Python in Science Conference*, S. van der Walt and J. Millman, Eds., 2010, pp. 56–61. DOI: 10.25080/Majora-92bf1922-00a.

[35] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," vol. 4, no. 2, pp. 100–107, 1968. DOI: 10.1109/tssc.1968.300136. [Online]. Available: https://doi.org/10.1109/tssc.1968.300136.

[36] D. J. Klein, "Centrality measure in graphs," *Journal of mathematical chemistry*, vol. 47, no. 4, pp. 1209–1223, 2010.

[37] A. Hagberg and D. Conway, "Networkx: Network analysis with python," *URL: https://networkx. github. io*, 2020.

[38] T. Vincenty, "Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations," *Survey review*, vol. 23, no. 176, pp. 88–93, 1975.

[39] T. Notteboom, A. Pallis, and J.-P. Rodrigue, *Port economics, management and policy*. Routledge, 2022.

[40] T. Notteboom, A. Pallis, and J.-P. Rodrigue, *Port Economics, Management and Policy*. New York: Routledge, 2022. DOI: 10.4324/9780429318184. [Online]. Available: https://doi.org/10.4324/9780429318184.

[41] J. Guo, S. Wang, D. Wang, and T. Liu, "Spatial structural pattern and vulnerability of china-japan-korea shipping network," *Chinese Geographical Science*, vol. 27, pp. 697–708, 2017.

[42] Maersk. "A short guide on ocean freight transit times." (2023), [Online]. Available: https://www.maersk.com/logistics-explained/transportation-and-freight/2023/09/27/sea-freight-guide (visited on 05/19/2024).

[43] Statista Research Department, *Median time spent in port by container ships worldwide by segment 2021*, Published by Statista Research Department, Mar 20, 2024, Mar. 2024. [Online]. Available: https://www.statista.com/statistics/1101596/port-turnaround-times-by-country/.

A

Appendix 1

A.1 A* paths

(a) Resulting A* path for the direction VNSGN → NLRTM

(b) Resulting A* path for the direction NLRTM → VNSGN

Figure A.1: A* paths for test case VNSGN ↔ NLRTM

(a) Resulting A* path for the direction USLGB → CNTXG.

(b) Resulting A* path for the direction CNTXG → USLGB.

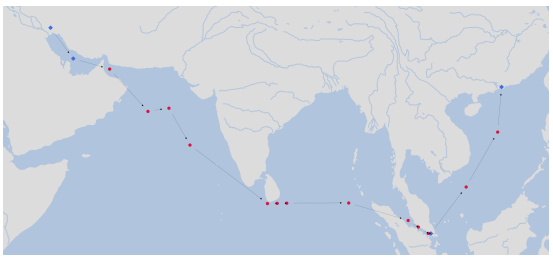
Figure A.2: A* paths for test case USLGB ↔ CNTXG.

(a) Resulting A* path for the direction CNTXG → DEHAM.

(b) Resulting A* path for the direction DEHAM → CNTXG.

Figure A.3: A* paths for test case CNTXG ↔ DEHAM.

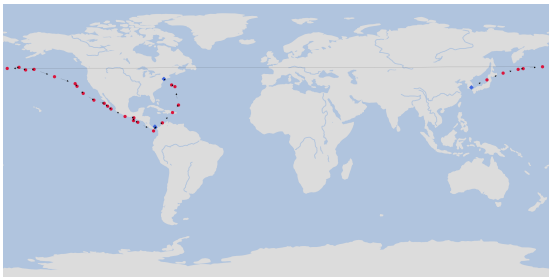
(a) Resulting A* path for the direction HKHKG → IRMRX.



(b) Resulting A* path for the direction IRMRX → HKHKG.

Figure A.4: A* paths for test case HKHKG ↔ IRMRX.

(a) Resulting A* path for the direction USNYC → KRPUS.



(b) Resulting A* path for the direction KRPUS → USNYC.

Figure A.5: A* paths for test case USNYC ↔ KRPUS.

(a) Resulting A* path for the direction SGSIN → NLRTM.

(b) Resulting A* path for the direction NLRTM → SGSIN.

Figure A.6: A* paths for test case SGSIN ↔ NLRTM.

(a) Resulting A* path for the direction BEANR → SAJED.

(b) Resulting A* path for the direction SAJED → BEANR.

Figure A.7: A* paths for test case BEANR ↔ SAJED.

(a) Resulting A* path for the direction RUPKC → NLRTM.

(b) Resulting A* path for the direction NLRTM → RUPKC.

Figure A.8: A* paths for test case RUPKC ↔ NLRTM.

(a) Resulting A* path for the direction CNLYG → RUNVS.

(b) Resulting A* path for the direction RUNVS → CNLYG.

Figure A.9: A* paths for test case CNLYG ↔ RUNVS.

(a) Resulting A* path for the direction CNTXG → SENYN.

(b) Resulting A* path for the direction SENYN → CNTXG.

Figure A.10: A* paths for test case CNTXG ↔ SENYN.

(a) Resulting A* path for the direction SENYN → USHNL.

(b) Resulting A* path for the direction USHNL → SENYN.

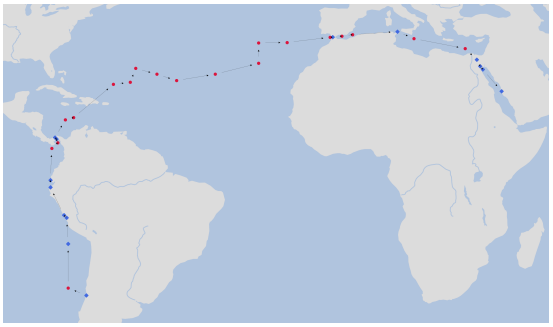
Figure A.11: A* paths for test case SENYN ↔ USHNL.

(a) Resulting A* path for the direction IRMRX → RUNVS.

(b) Resulting A* path for the direction RUNVS → IRMRX.

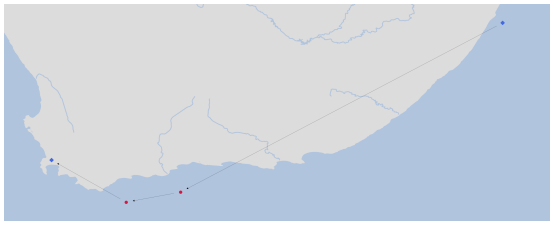
Figure A.12: A* paths for test case IRMRX ↔ RUNVS.

(a) Resulting A* path for the direction SAJED → CLVAP.

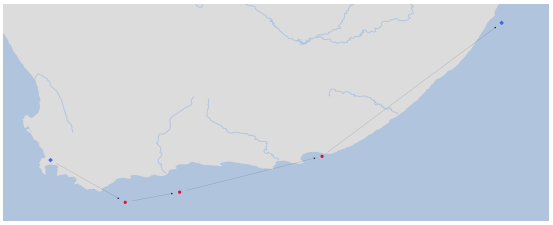


(b) Resulting A* path for the direction CLVAP → SAJED.

Figure A.13: A* paths for test case SAJED ↔ CLVAP.



(a) Resulting A* path for the direction ZADUR → ZACPT.



(b) Resulting A* path for the direction ZACPT → ZADUR.

Figure A.14: A* paths for test case ZADUR ↔ ZACPT.