

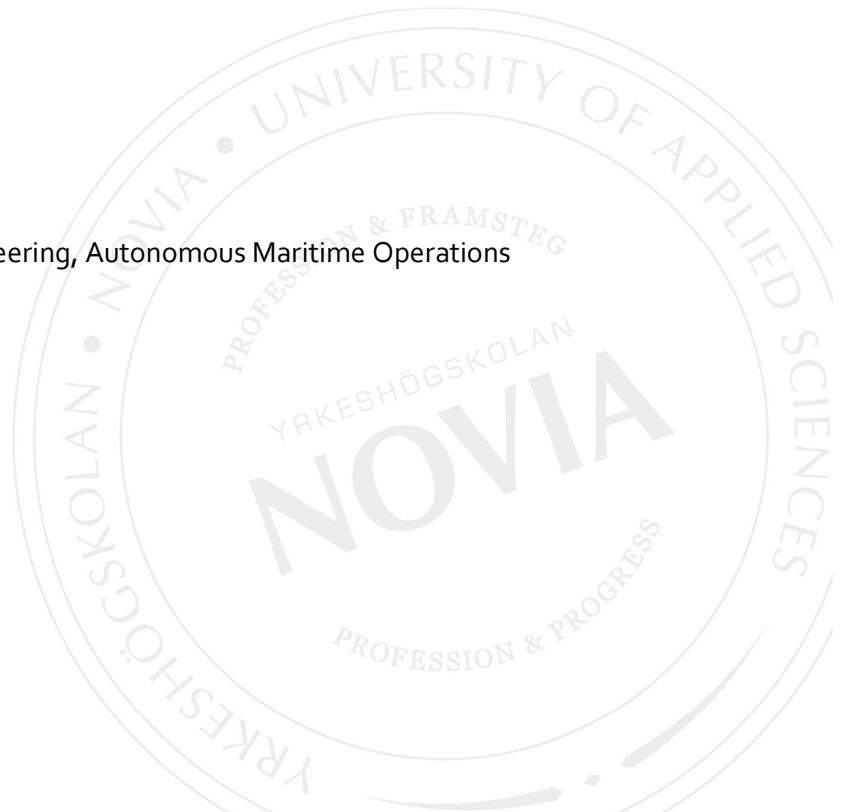
Automatic Route Planning and Calculation based on S-100 for Autonomous Vessels Navigation

Georgios Karamanoglou

Master thesis

Degree Programme Master of Engineering, Autonomous Maritime Operations

Place and year Turku, 2022



DEGREE THESIS

Author: George Karamanoglou

Degree Programme and place of study: Master of Engineering, Autonomous Maritime Operations, Turku, Finland

Specialization: MSc Degree Thesis

Supervisor(s): Katarina Sandström

Title: Automatic Route Planning Calculation based on S-100 for Autonomous Vessels Navigation

Date May, 2022 Number of pages 115 Appendices: 5

Abstract

Automatic Route Planning and Distance Calculation can be further developed by using the same factors as used in Route Planning by a Navigator while using the new Hydrographic S-100 Standards. These can be used to make this process automatically while considering the weather, depth, navigational dangers, and other related information like a regular Navigator does in reality. The purpose of this Thesis is to explain how Route Planning and Distance Calculation works, show the implementation of the new S-100 IHO standard and most commonly used pathfinding algorithms in automatic route calculation. Finally, explain why accurate and adaptable route calculation is essential to Autonomous Ships and Shipping in general in the future.

For collecting the information for this Thesis, the chosen research methods were partly theoretical and partly constructive research, in the form of ascertaining and defining the problem to be solved, after presented the challenges that need to be tackled, literature review of researchers in the field of Pathfinding Algorithms applications, comparison of the literature collected, design of the process to solve the issue at hand, and theoretical evaluation of the various methods available depending on their use suitability and efficiency.

In conclusion, the technology is here for using the new S-100 IHO ENC Standard for Automatic Route Calculation while using, as the research has revealed out of a diverse set of pathfinding algorithms, ranging from Dijkstra to A*, genetic, including ant colony algorithms, which have been proven by empirical studies that they can be potentially employed in route navigation for ships. The adoption of Automatic Route Planning and Calculation in Autonomous Ship Navigation is advocated by using deep learning algorithms; further cybersecurity concerns are also present, including but not limited to signal jamming and malicious attacks on the ship's communication equipment. Nevertheless, diverse solutions have been also identified to mitigate the threats and ensure that the ships can successfully attain their goals in reaching targeted destinations. In particular, the use of neural networks and deep reinforcement learning algorithms is a prevalent solution.

Language: English

Keywords: navigation, route planning, electronic charts, Routeing, route calculation, machine learning, geography, algorithms, data structures, pathfinding problems, wayfinding problems, safety check, maritime spatial data infrastructure, S-100, MSDI, IHO, IMO, ECDIS, GIS, ENC.

Acknowledgement

This Thesis is a materialization of my experience all these years as a Navigation Officer and research in Pathfinding Algorithms which in the Maritime Domain remains a well guarded secret. Therefore I would like to thank my Supervisor Katarina Sandström for her support during this process and her candid comments which helped me grow academically and professionally, secondly Mr. Micael Vuorio, Head of Maritime Academy and Training Center Aboa Mare, who welcomed us in the Program and helped us as best as he could, third Leif-Christian Östergård, the Aboa Mare and Novia Teams for his great job at making the program a reality in collaboration with Novia and last but not least the amazing Team of peers I had the privilege to work, collaborate, learn and grow as a person, professional and academic.

Abbreviations

ACA: ant-colony algorithms,
ACO: ant colony optimization,
ACPF: adversarial cooperative pathfinding,
ADSS: Active Dynamic Signage System
AIS: Automatic Information System,
ANN: artificial neural networks
ARA*: Anytime Algorithms,
BAN: body area networks,
BFS: breadth-first algorithm,
CATZOC: Categories of Zones of Confidence,
CO2: Carbon Dioxide,
COLREGS: Collision Regulations,
COTP: Captain of the Port,
CPF: cooperative pathfinding,
CPU: central processing unit
DA*: Dynamic A* algorithm,
DDPG: Deep Deterministic Policy Gradient,
DDQN: double Q-network,
DFS: Depth First Search,
ECDIS: Electronic Chart Display Information System,
EGC: Enhanced General Calling,
ENC: Electronic Navigational Charts,
ETA: Estimated Time of Arrival,
FIFO: first-in-first-out,
GA: genetic algorithms,
GC: Great Circle,
GIS: Geographic Information Science,

GNSS, Global Navigation Satellite Systems,
GoM: Corrected Metacentric Height,
GPS: Global Positioning System,
GPU: Graphical Processing Unit,
GT: Gross Tonnage,
GUI: Graphic User Interface,
IC: Interoperability Catalogue,
ICT: Information and communications technology,
IDA Iterative Deepening Algorithm,
IDDFS: Iterative Deepening Depth First Search,
IEC: International Electrotechnical Committee,
IHO: International Hydrographic Organisation,
IMO: International Maritime Organisation,
INM-C: Inmarsat-C,
ISO: International Standards Organization,
KHOA: Korean Hydrographic and Oceanographic Office,
LPA*: Lifelong Planning A* algorithm,
LPA*: Lifelong Planning A*,
MAPF: multiple agent pathfinding
MAPF: multiple agent pathfinding,
MAV: micro aerial vehicles,
MOSP: minimization of open stacks,
MPA: Marine Protected Areas,
MSDI: Marine Spatial Data Infrastructure,
MSDIWG: Marine Spatial Data Infrastructures Working Group,
NGIA: National Geospatial Intelligence Agency,
NOAA: National Oceanic and Atmospheric Administration
NOX: Nitrogen Oxides,
NWPS: Nearshore Wave Prediction System,

OCIMF: Oil Companies International Maritime Forum,
OSPF: open shortest path first
PMN: Precision Marine Navigation
PORTS: Physical Oceanographic Real-Time System,
PROTIDE: Probabilistic Tidal window Determination,
PSSA: Particularly Sensitive Sea Areas,
RAM: Random Access Memory,
RL: Rhumb Line,
RoRo: Roll On Roll Off Carriers,
SCCOOS: Southern California Coastal Ocean Observing System,
SCCOOS: Southern California Coastal Ocean Observing System,
SDI: Spatial Data Infrastructure,
SIRE: Ship Inspection Report,
SOX: Sulphur Oxides,
SVM: Support Vector Machines
T&P: Temporary and Preliminary,
UAV: unmanned aerial vehicles,
UCS: Uniform Cost Search,
UKC: Under Keel Clearance,
UKHO: United Kingdom Hydrographic Organization,
UML: Unified Modelling Language,
UPS: Uninterruptible Power Supplies,
USGS: United States Geological Survey,
USV: unmanned surface vehicles
XTE: Cross Track Error,
ZOC: Zones of Confidence,

Table of contents

1	Introduction	1
1.1	History of Route Planning and Calculation.....	2
1.2	Statement of Purpose, Research Questions and Research Approach.....	6
1.3	Structure of the Thesis.....	7
2	Research Methodology.....	7
3	Methods currently used for Automatic Route Planning.....	8
4	Types of Sailing Methods.....	17
4.1	Depiction of Great Circle and Rhumb Line Navigation on Charts.....	18
4.2	Great Circle Navigation.....	19
4.3	Factors to consider during Great Circle Navigation	19
4.4	Rhumb Line Navigation.....	21
5	Factors affecting Route Calculation.....	21
6	History, Present and Future of Electronic Navigational Charts	23
7	S-100 ISO Standards.....	28
7.1	Types of S-100 Layers.....	31
7.2	Data structures and attributes used in S-100.....	33
7.3	Display and data integrity of S-100 ENC.....	36
8	Pathfinding / Wayfinding Algorithms.....	37
8.1	Pathfinding problem variations	38
8.1.1	Pathfinding algorithms adoption in video games	40
8.1.2	Pathfinding algorithms adoption in robotics	42
8.1.3	Pathfinding algorithms adoption in ship routing	44
8.2	Attributes of Pathfinding / Wayfinding Algorithms important to the process	44
8.2.1	Dijkstra algorithm.....	45
8.2.2	Breadth-First Search (BFS)	49
8.2.3	Depth-First Search (DFS).....	53
8.2.4	Iterative Deepening Depth First Search (IDDFS)	54
8.2.5	Bi-directional Search	56
8.2.6	Uniform Cost Search	57
8.2.7	Breadth-First Search with Heuristic Function.....	58
8.2.8	A* algorithm.....	59
8.2.9	Genetic algorithms.....	62
8.2.10	Ant Colony algorithms	65
8.3	Comparison of Pathfinding / Wayfinding Algorithms.....	68
9	Automatic Route Calculation	71

9.1	Automatic Route Planning.....	71
9.2	Automatic Route Safety Check.....	73
9.3	Automatic Route Calculation Process.....	77
9.4	Potential Applications.....	78
10	Artificial Intelligence.....	83
10.1	Artificial Intelligence and application of new technological innovations in Route Finding	83
10.2	Machine Learning	84
10.3	Deep Learning.....	86
10.4	Comparison between Machine and Deep Learning.....	89
11	Cybersecurity concerns	91
12	Conclusions.....	92
13	Bibliography.....	96
	List of Tables.....	9
	List of Figures.....	0

"The real voyage of discovery consists not in seeking new landscapes, but in having new eyes."

Marcel Proust, French Novelist known for Remembrance of Things Past or In Search of Lost Time

1 Introduction

Currently, route planning and distance calculation are mainly being done manually by a navigator, for passage planning for actual navigation at sea or automatically by dedicated software, either web-based or app-based, which is mainly used by office personnel who don't have access to ECDIS (Electronic Chart Display Information System) or charts, such as shipbrokers, charterers, operators, and other parties interested in such information. I've had the personal experience of inaccurate route measurements made by routing software by several hundred nautical miles, which in turn caused questions as that could result in significant delay and commercial impact and from the investigation the result was that the routing software didn't take into account several small islands and plotted the route over them. The advances in GIS (Geographic Information System) and digital hydrography, such as the new hydrographic S-100 Standards, can be used to this process automatically while taking into account the weather, depth, navigational dangers, and other related information like a regular navigator does in reality.

Route planning has evolved from an approximate science before the advent of real time communication and the Internet, where the vessel's ETA (Estimated Time of Arrival) was a time window of several days, meaning the vessel could arrive at any time during these weeks, down to the ETA matching almost closely the ATA (Actual Time of Arrival) by only a few hours after an intercontinental voyage of several weeks, something that by its own right it's a marvel of logistics.

That of course wouldn't be possible without the discovery of a series of technological inventions, such as long-range radio navigation such as Decca, Loran-C, the advent of satellite navigation with systems such as the GPS (Global Positioning System), GLONASS, Beidou, the internet, real time information feeds, API (Application Programming Interface) serving as a communication interface between different software applications and the evolution in the field of graph theory and pathfinding algorithms. Logistics and process innovations such as Just In Time (JIT) arrivals, advanced port management to ensure prompt ships turnover, analysis of ships energy efficiency, standardisation of commodity paperwork and introduction of incoterms and electronic bills of lading,

which help vessels process their cargo and customs related paperwork faster and help for a quicker vessel turnover and a multitude of other innovations.

Even after these innovations, there is room to further develop route planning to become not only more accurate in terms of accuracy but to be able to do constant route calculations and improvements on the current voyage, as a standalone system if it will be placed onboard an autonomous vessel.

1.1 History of Route Planning and Calculation

Route calculation is the result of a rigorous planning procedure called voyage planning. In the past voyage planning had to be made painstakingly on paper charts by plotting it first to general charts, then to other smaller charts. To get a “rough” distance measurement for an intercontinental voyage, it could take from 30minutes to even 1.5 hours, depending on the navigators’ experience. For a more accurate reading, it could take from 1.5 up to 4 hours. From personal experience, the most challenging part was looking at the chart catalogue (see below Figure 1) and realizing that it would take the navigator quite a while to find the charts in the drawers and even worst when the chart he/she was looking for so bad was not there. Before routeing software, there were distance tables as either standalone books (such as reeds distance tables or np350 admiralty distance tables, Lloyd’s maritime atlas) or as part of other books/publications (such as brown’s almanac or routeing charts). Some distance tables used low and high-powered ship routes, where depending on the vessel’s engine horsepower, they would indicate the respective route to take. The assumptions used back then on the currents/weather are outdated due to climate change and the increase of vessel sizes and engine horsepower. These distance tables were used not only from the ships but also from the people ashore, who depended on the accuracy of this information by various stakeholders such as shipbrokers, charterers, operations managers. Accuracy was quite an issue due to the lack of sharing of information and experience at the time, and only people who were part of a specific group or company benefitted from this information exchange. Specialized software developed in-house by companies with resources such as AtoBviaC by BP, as it was quite a feat at the time, to contain all of the known tanker and bunkering ports in custom in-house software. Manual plotting of courses on ECDIS is the prevalent method now as ECDIS is mandatory equipment by IMO

now, and paper charts are only one of the alternatives and currently only being used for training and emergency in case of ECDIS failure. Consulting routing charts and sailing directions / pilot books for the prevailing weather conditions in the area. Every route after it is drawn must be evaluated through the automatic safety check function, by the ECDIS software, in order to highlight potential issues in the plotted Route, which must then be, in turn, manually checked by the navigator and revise route or passage planning, as needed.

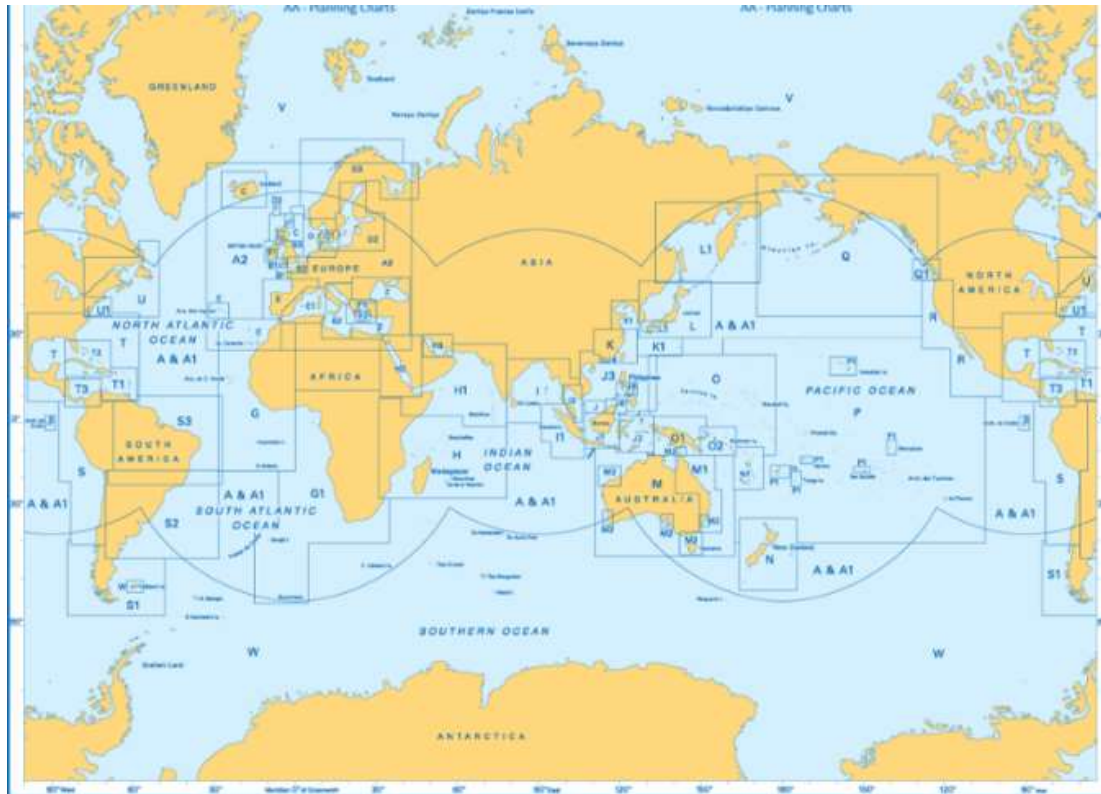


Figure 1 Routing Publications Covers and Admiralty Chart Catalogue (UKHO, 2020)

Currently, a navigation officer on board a modern ship above 500 GT (Gross Tonnage) should have at the minimum an ECDIS available at his disposal for route planning, per mandatory IMO (International Maritime Organisation) requirements. Besides a standard ECDIS, a modern navigation officer can use touchscreen back of bridge navigation software for planning routes such as the one shown in Figure 2 and depending on the company's budget. They may have a weather routing software such as the one shown in Figure 3, where the planned Route can be optimized by the software to avoid heavy weather while minimizing fuel consumption and arriving at the shortest possible time. Also selecting the charts for the voyage so they can be ordered if needed, is done

automatically by uploading the route and the ECDIS will generate a list with all charts required for the voyage, or if the navigator chooses to manually select them from the ENC cell catalogue using a GUI (Graphic User Interface) like the one in Figure 4.



Figure 2 Using a touchscreen Back of Bridge Navigation software for planning routes (Navtor, 2015)

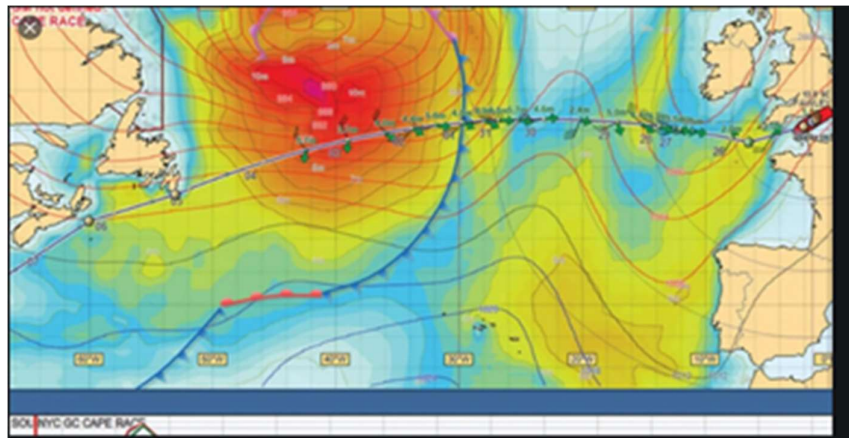


Figure 3 Modern Weather Routing Software showing the weather affecting a specific route (SPOS, 2016)

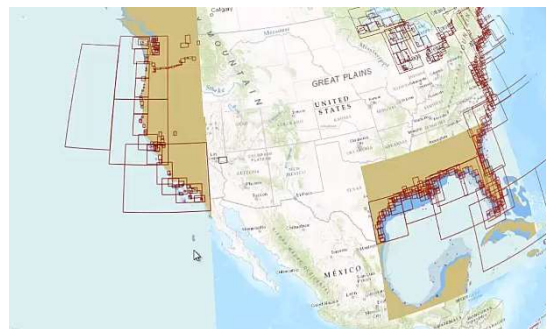


Figure 4 A graphic ENC Cell Catalogue for a specific area as shown in all modern ECDIS (ESRI, 2015)

S-52 is the IHO (International Hydrographic Organisation) standard for displaying navigational info on ECDIS, and S-57 is the IHO Standard for transfer of navigational data from the hydrographic office to the ENC producer and from there to the individual ECDIS. Each item has its own individual and group code and is displayed in layers as shown in the Figure 5 below. The bathymetric layer is adjustable according to the safety depth and safety contour, which is considering the vessel's individual squat and UKC (Under Keel Clearance) calculations and the sailing area Zone of Confidence (ZOC) which depicts the hydrographic data accuracy of the area and in accordance with the company navigational policy.

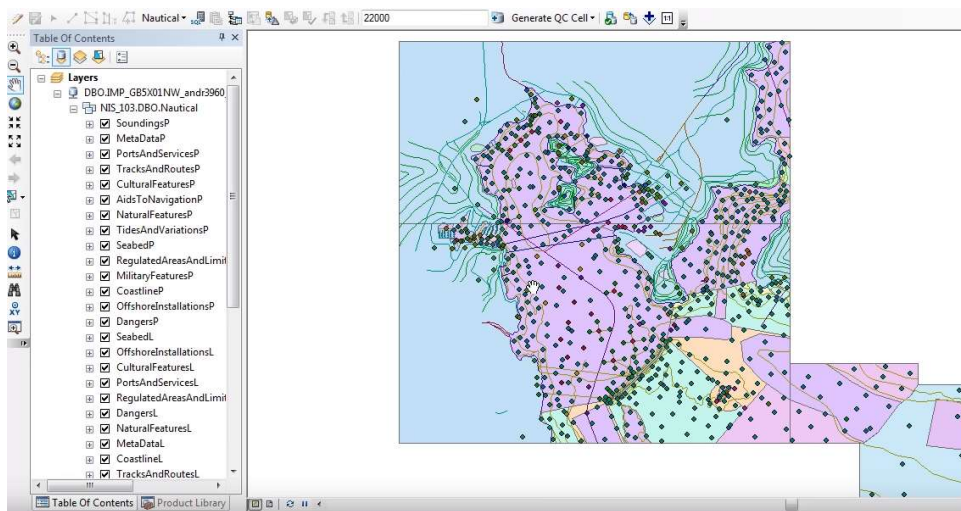


Figure 5 S-57 ENC Raw Data display in Arc Map Maritime Module during construction of an ENC. (ESRI, 2015)

Currently, there is an abundance of companies working in the field of maritime routeing, but few are the ones who are distinguished as go-to providers and even referenced in agreements and charter parties. Each company is using a different model for calculation, with most of them having manually plotted generic routes and then established main waypoints so that they can combine many destinations in a single route. The vast majority aren't trade-specific like Worldscale, where the global Tanker Trade depends on making single voyage fixtures, and AtoBviaC, which was developed initially for the tanker industry. A few companies, such as MarineTraffic and Searoutes, use historical AIS (Automatic Information System) routes, which have been "cleaned" from clutter such as waiting, deviations, and similar occurrences which affect the actual voyage route. They remain generic, and even if they apply a seasonal filter for weather for

oceanic crossings, it still refers to past data and not the actual ones at present. A significant issue is that many companies ignore the individual vessel's draft and dimensions, and thus many shorter routes are being lost.

1.2 Statement of Purpose, Research Questions and Research Approach

The challenges I faced during my tenure as a navigation officer, while striving to discover improved ways to overcome these issues and the realisation that the vast majority of Routeing Software available at the time offers a generic produced distance and only a few might have a selection for the ships draft and air draft, resulted to motivating me to research the topic further and compose this Thesis. These experiences made me realise that successful route / passage planning is an essential part of route calculation and without it, accurate and safe results couldn't be possible.

Despite the strides in advancement in the field of autonomous navigation, the focus on the actual passage / route planning by autonomous ships has been seriously underestimated because the companies that are researching for autonomous solutions, are equalizing marine navigation to autonomous car navigation driving on a road or a freeway, where the car has to follow the lanes.

In this thesis, I will attempt to answer the following research questions:

- 1) Why route calculation is more than a tabletop exercise by analysing the factors behind route / passage planning both using traditional and contemporary means of navigation, that includes the challenges faced during the planning of a voyage plan both from route planning software and the navigator's perspectives?
- 2) How to implement the use of the new S-100 IHO standard and pathfinding algorithms in automatic route calculation?
- 3) Why accurate and adaptable route calculation is important to autonomous ships and shipping in general in the future?

1.3 Structure of the Thesis

This thesis explains in the first chapter to the reader in brief, the history and evolution of route planning through the ages and the research statement, questions and approach to the thesis subject. Chapter two covers the methods and factors influencing automatic route planning and calculation. Chapter three explains the types of sailing methods and an operational point of view and that of the ENC (Electronic Navigational Charts) usage. In the fourth chapter the factors influencing route calculation are described. The history and evolution of electronic navigational charts are covered in the fifth chapter and in the sixth chapter the S-100 ENC standard is explained in detail technically and operationally. In the seventh chapter, pathfinding algorithms are presented and demonstrate their usefulness and application in video games, robotics and ship routing. Then the attributes of most well-known pathfinding algorithms are detailed and then compared between each other. The automatic route calculation for ships is covered in the eighth chapter by explaining how the process will work and what are its potential applications. The ninth chapter is about artificial intelligence and makes a distinction about the difference between machine learning and deep learning and its applications. Any potential cybersecurity issues are covered in the tenth chapter. The research methodology is explained in the eleventh chapter and the twelfth chapter has the conclusions of the thesis.

2 Research Methodology

For collecting the information for this thesis, the chosen research methods are partly theoretical and partly constructive research, in the form of ascertaining and defining the problem to be solved, after presenting the challenges that need to be tackled, literature review of researchers in the field of pathfinding algorithms applications, comparison of the literature collected, design of the process to solve the issue at hand, and theoretical evaluation of the various methods available depending on their use suitability and efficiency. Unfortunately, qualitative analysis methods such as interviews were not possible due to the demanding schedule of my current work and because several of the persons of interest, I approached with an important role in the industry were hesitant to disclose any information due to the commercial sensitivity of the subject. Sending questionnaires was not considered as a method, because the concept of the topic is

more about exploring the possible issues and possibilities of automatic route calculation would be restricting the exploration of the topic. Unstructured Interviews with general agenda of discussion, which encourage open discussion seem to be the most appropriate for this thesis, because the informant is allowed to speak freely about the topics covered and the interviewer would control the informant to only a small extent as long as the informant stays on topic (Zhang, Y., & Wildemuth, B. M., 2009).

The literature review was made by following a simple 7-step structure according to Allison Kirsop,2021: First consider your audience - who are you writing for? This is important for the scope of the literature to be researched. Second, what type of review are you being asked to write - systematic or narrative? Do you know the difference? Third, who are the main players? Identify the research teams of your topic. Fourth, discuss timeframes correctly - be specific, and don't generalize. Fifth step, back up your comments with convincing evidence. Sixth, be aware of accurately presenting conflicting data/correct citation and referencing. Seventh, know how to structure your review.

3 Methods currently used for Automatic Route Planning

As of now, automatic route planning processes are carefully guarded, and therefore these are not publicly known, except a few, but in my opinion, they seem to be divided into the following categories based on the method they use:

- 1) Distances being calculated manually by experienced navigators for each class of ship.
- 2) Distances being sourced by AIS data of the corresponding size/class of ship. Depending on the size, draft, type, and trade of the vessel, there is variance in its routing.
- 3) Or by being indicated in the way the route is being chosen/toggled in their interface, such as via nodes/waypoints.
- 4) Wayfinding formulas applied on Open-Source maps, not considering the depth of the sea, but only the layer attributed as the sea. Information is not available on the topic due to the fact companies in the field, do not disclose these.

When calculating distances by AIS data and the size of the vessel, more specifically the companies collect the AIS positions of all vessels of a specific class, over a specific period of time e.g., 2 years, 1 year or as needful, and create traffic density maps. these density maps show at their thickest areas, the routes most widely used and the main points of connecting these lines are the waypoints of the voyage. By analyzing all traffic for a specific area, you can then filter by each size of vessel, as shown below in Figures 6 -15, and get a clearer picture of the routes each vessel takes depending on its size. If we want to further analyze it, we will have to identify when the vessel is laden or ballast according to his AIS status and by filtering the data per month of year over a period of years. This way the final result is more custom made for the specific size and type of vessel rather than a general routeing software. Results are fully custom made, but still are the better than what you can have with a generic wayfinding formula that disregards all trade specific and environmental factors such as depth, currents, winds, swell, seasonal storms.

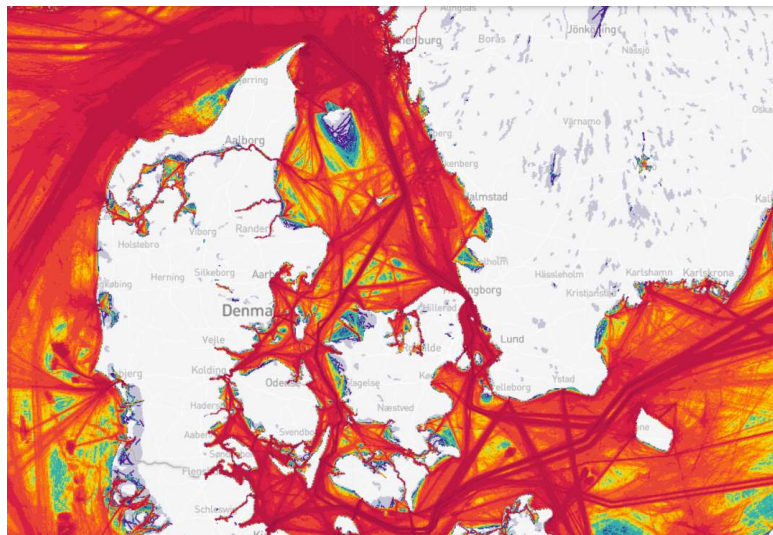


Figure 6 All Year 2019 Traffic Density Map (MarineTraffic, 2021)

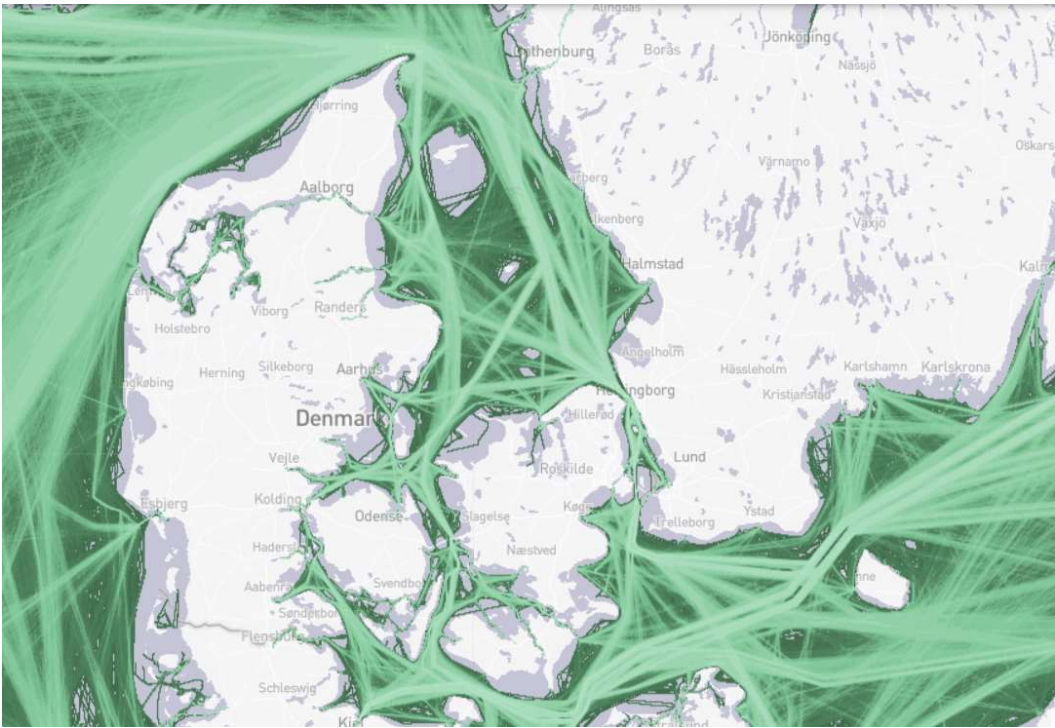


Figure 7 Handysize Bulk Carrier Year 2019 Traffic Density Map (MarineTraffic, 2021)

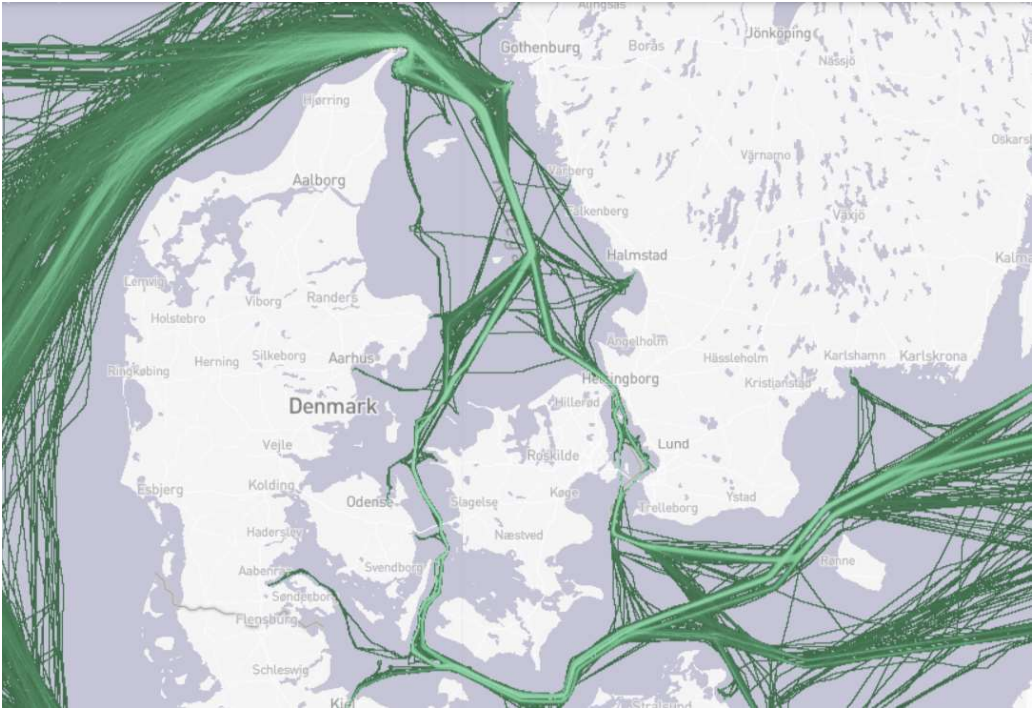


Figure 8 Handymax Bulk Carrier Year 2019 Traffic Density Map (MarineTraffic, 2021)

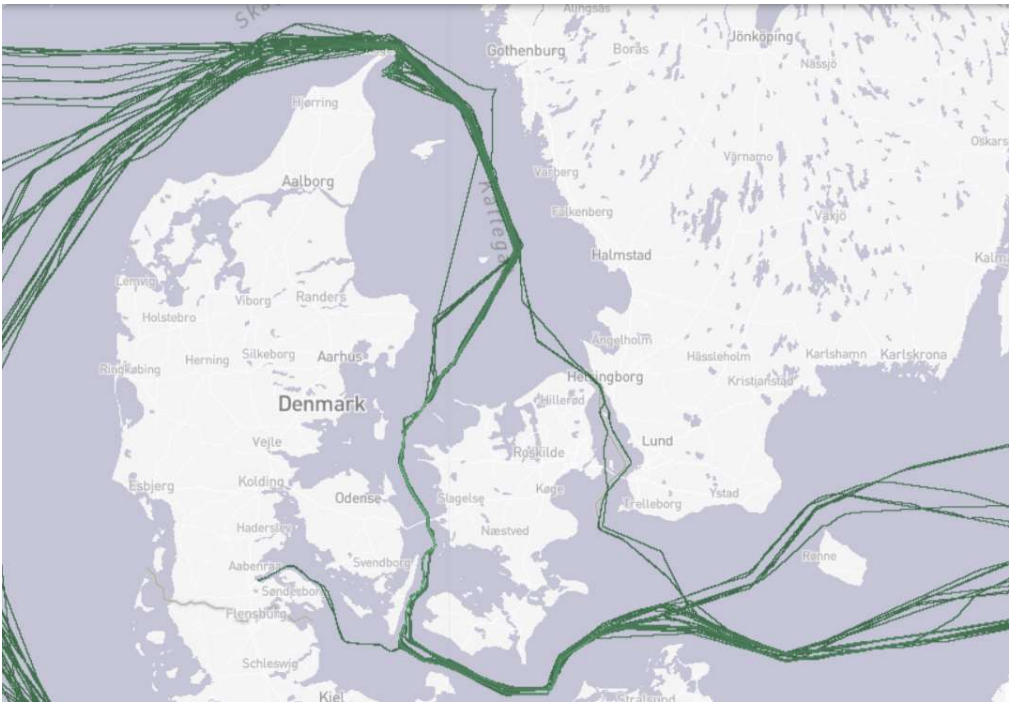


Figure 9 Capesize+ Bulk Carrier Year 2019 Traffic Density Map (MarineTraffic, 2021)

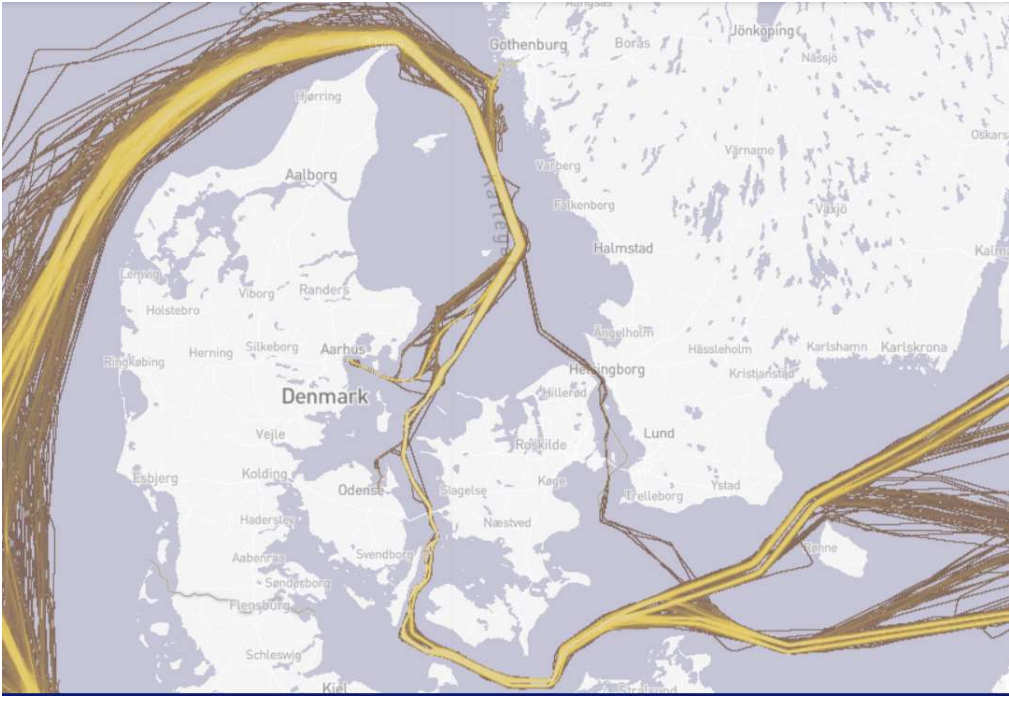


Figure 10 Large Container Ships >3,000 GT Year 2019 Traffic Density Map (MarineTraffic, 2021)

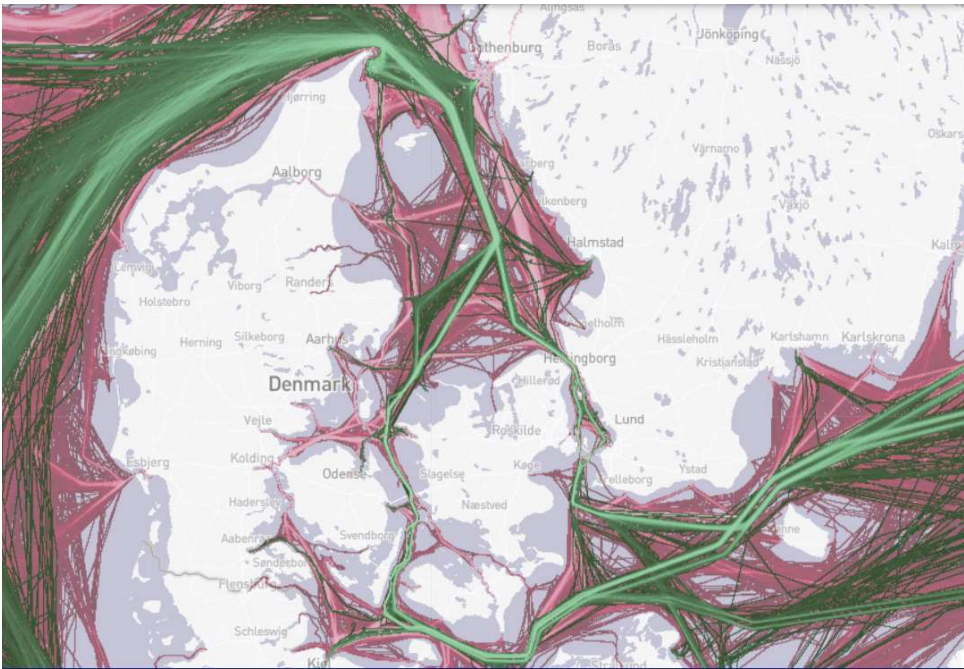


Figure 11 Handymax Bulk Carrier Traffic (green) overlaid on top of Handysize Tankers (pink) Year 2019 Traffic Density Map (MarineTraffic, 2021)

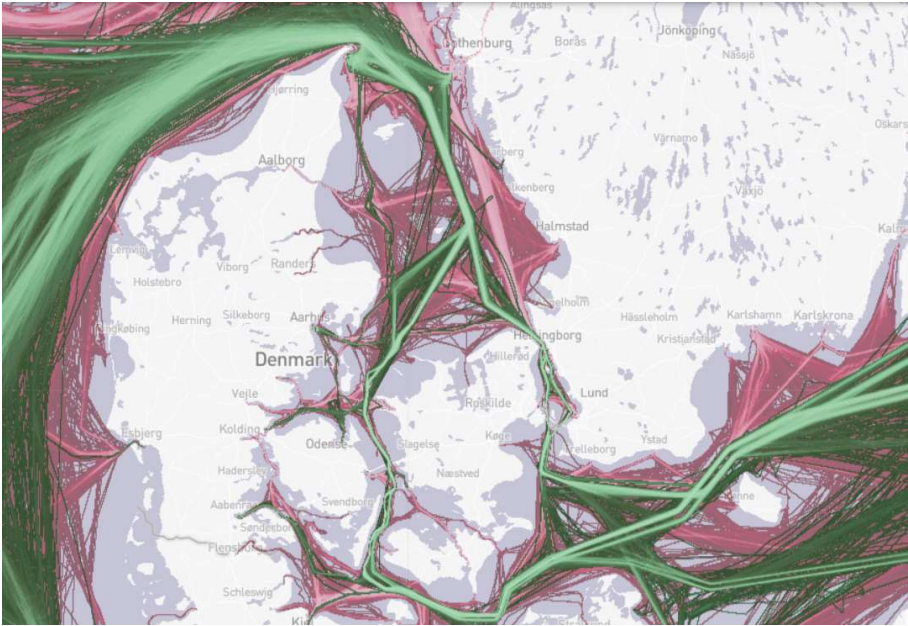


Figure 12 Panamax Bulk Carrier Traffic (green) overlaid on top of Handysize Tankers (pink) Year 2019 Traffic Density Map (MarineTraffic, 2021)

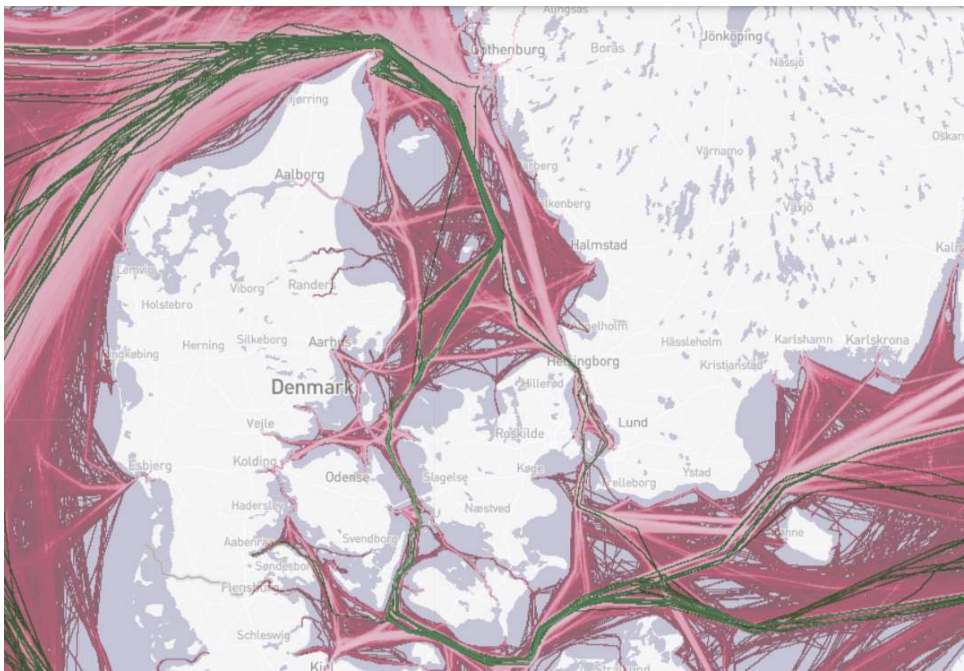


Figure 13 Capesize+ Bulk Carrier Traffic (green) overlaid on top of Handysize Tankers (pink) Year 2019 Traffic Density Map (MarineTraffic, 2021)

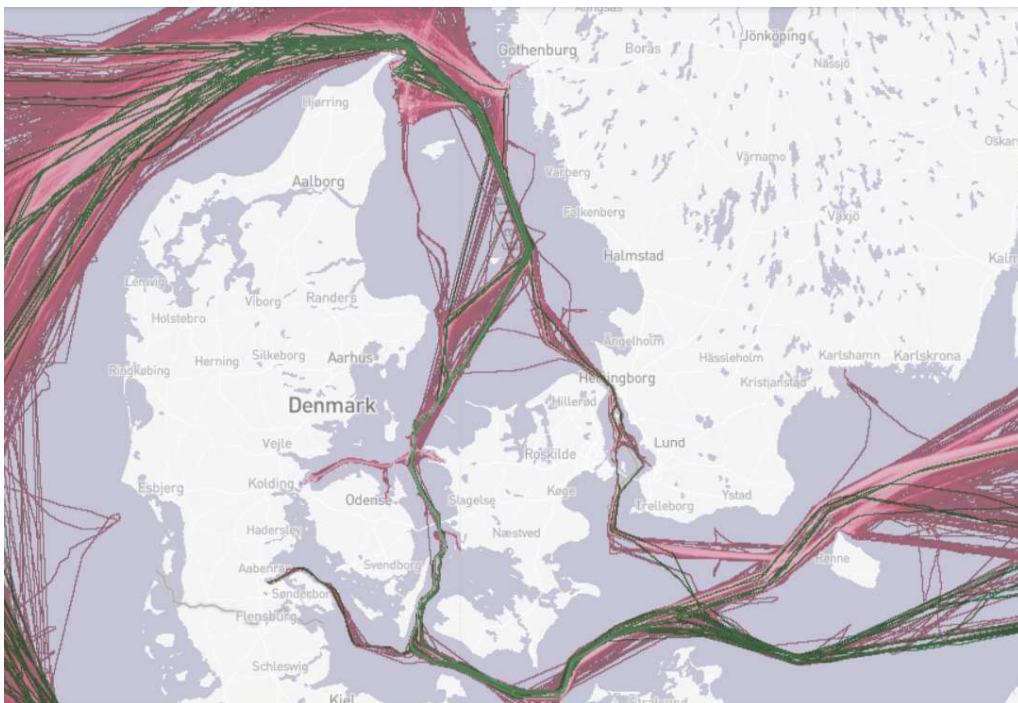


Figure 14 Capesize + Bulk Carrier Traffic (green) overlaid on top of Aframax /LR2 + Tankers (pink) Year 2019 Traffic Density Map (MarineTraffic, 2021)

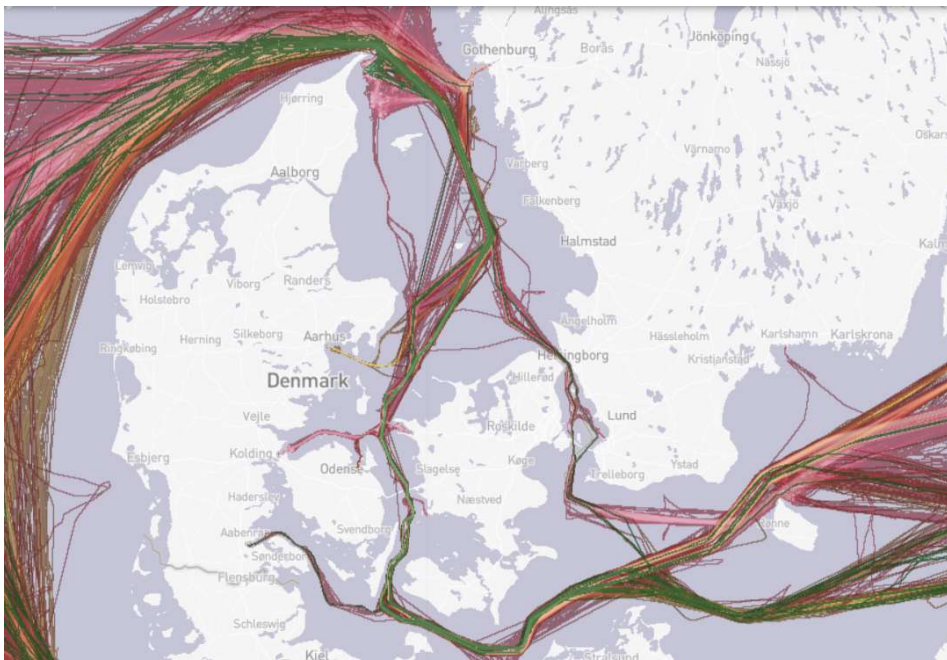


Figure 15 Capesize + Bulk Carrier Traffic (green) overlaid on top of Aframax /LR2 + Tankers (pink) vs. Large Container Ships >3,000 GT (yellow) Year 2019 Traffic Density Map (MarineTraffic, 2021)

Another method is when the calculated route is being chosen/toggled in the software interface, such as via nodes/waypoints. The results are variable depending on company to company due to their algorithm, a number of nodes, and other factors not visible, as they are done in the backend of the development process. Some “bottleneck” coastal areas, such as the One Fathom Bank in the Malacca Strait, straits in the Indonesian Archipelago, or in the Caribbean Sea, might have draft restrictions attributed so any vessels exceeding this draft will not be navigated through these points. This draft is not considering any squat effect, waves, swell, tides, and any other applicable factors.

The route will be calculated predominantly through an algorithm by using ENCs (Electronic Navigational Charts) as the means to recognize navigational dangers and the depth of the sea. The type of ENCs that will be used can be the S-100, which contains meteorological, hydrographic, and nautical layers of navigational information in layers, as illustrated in Figure 16 below.

An alternative method will be by using historic AIS routes, which are filtered by the type of ship, size class (handymax, panamax, etc.), and draft and then used as a measure to compare against the algorithm calculated route, as a means of weighted average

verification. The weight average value to be indicated by the number of vessels with similar dimensions. As a safety precaution, it's always strongly recommended; the route always is validated by a certified professional if used for navigation.

Route optimization has the potential to significantly contribute to helping reducing fuel consumption, thus emissions of CO₂ and SO_x (Sulphur Oxides) / NO_x (Nitrogen Oxides). Due to the continuous evolution and revision of ship energy efficiency calculation standards by IMO, in this thesis, we won't be expanding further on the subject.

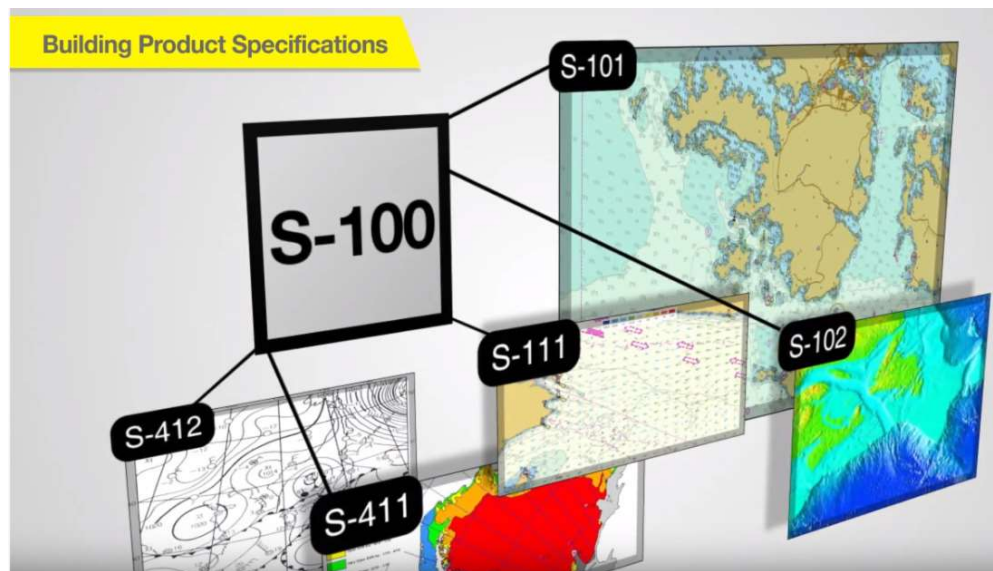


Figure 16 S-100 Main ENC layers (KHOA, 2017)

Currently, automatic route calculations, except when done manually, are only considering the shortest navigable route but without detailed draft checking. If they are carried out on official ENCs, there should be specific criteria implemented that will ensure the calculation of the safest possible route.

The route planning criteria will be divided by order of most important to least, as shown in the Appendix I "Route Planning Criteria" and from Figures 17 to 19.

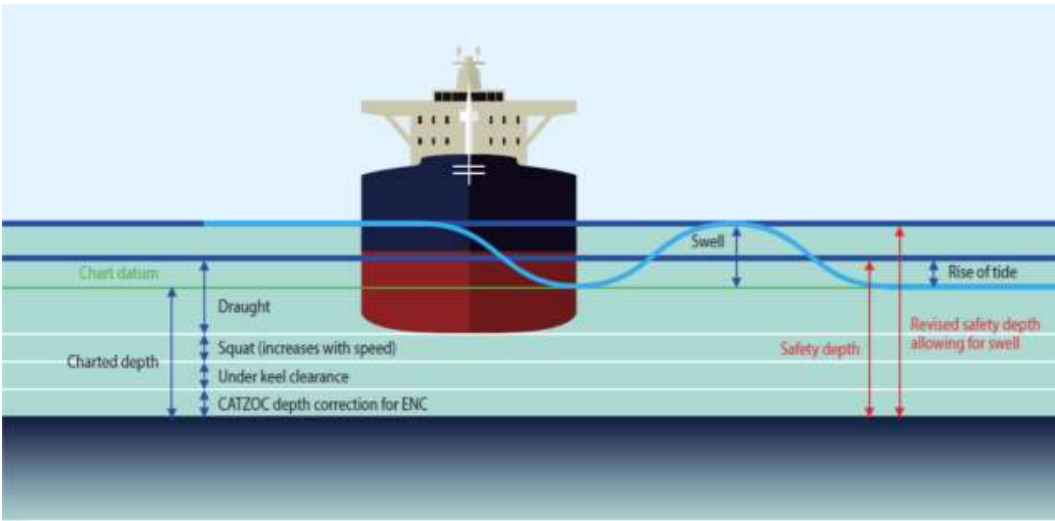


Figure 17 UKC calculation considering all environmental and ENC factors (AWP Marine Consultancy Ltd, 2017)

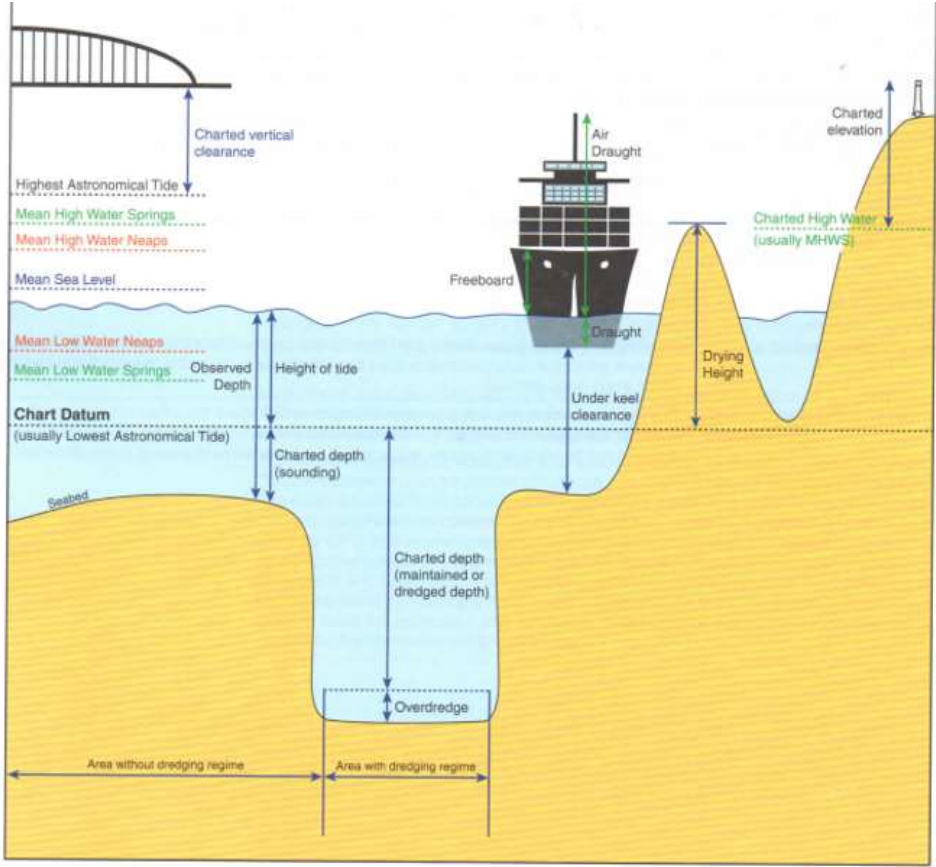


Figure 18 Standardized Depth Terminology (UKHO, 2015)

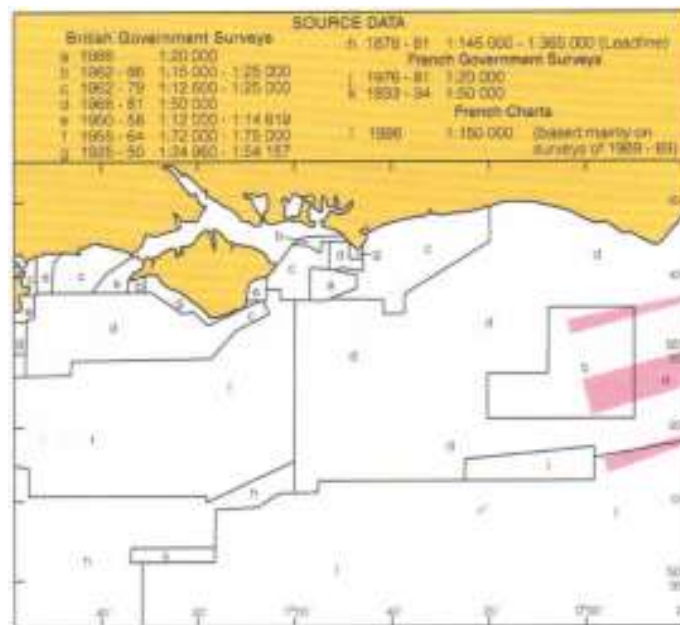


Figure 19 Paper Chart Source Data reference example (UKHO, 2015)

4 Types of Sailing Methods

Doing route planning and therefore calculation, without considering the appropriate type of sailing method to be chosen, will not only significantly affect the resulting distance but also the safety of the voyage itself. Since the middle ages, the evolution of nautical science, geography and geodesy caused the degree of complexity of the subjects to be significantly expanded to help long distance voyages and the resulting changes that these needed such as accuracy, safety and therefore increase the degree of certainty that the vessel would actually arrive at its destination and not get lost or aground.

Depending on the length of the voyage and the curvature of Earth's shape, the main types of sailing in great circle sailing it's illustrated, as a straight line when represented on a gnomonic chart or as a curved line on a mercator chart and in rhumb line sailing, shown as a straight path on a mercator chart or as a curved path on a gnomonic chart.

Rhumb Line sailing is divided into further subcategories with a certain degree of complexity which will cause deviation from the current subject, but for the sake of staying on topic and brevity, these are omitted in this thesis.

4.1 Depiction of Great Circle and Rhumb Line Navigation on Charts

According to National Geospatial Intelligence Agency 2019, *American Practical Navigator*, Chapter 12, plotting a straight course on a mercator map is referred to as rhumb line sailing, but due to the shape and, more precisely, the curvature of the earth, the rhumb line does not represent the shortest distance on its surface. On the contrary, the shortest distance between two pathways on the surface of the earth is the distance between two places on the arc of a great circle sailing, which is portrayed as a curved line on a mercator map but as a straight line on a gnomonic chart due to the form of the earth's curvature. While great circle routes may be drawn more readily on gnomonic charts, they are not conformal, which means that the navigator cannot directly measure directions or distances as he can on a mercator chart. By conformal, we mean that the projection picture preserves every angle between two curves that intersect on Earth (a sphere or an ellipsoid). As a result, the map's distances are not distorted unevenly. A great example of the illustration of great circle tracks on different projections is shown in Figure 20 below.

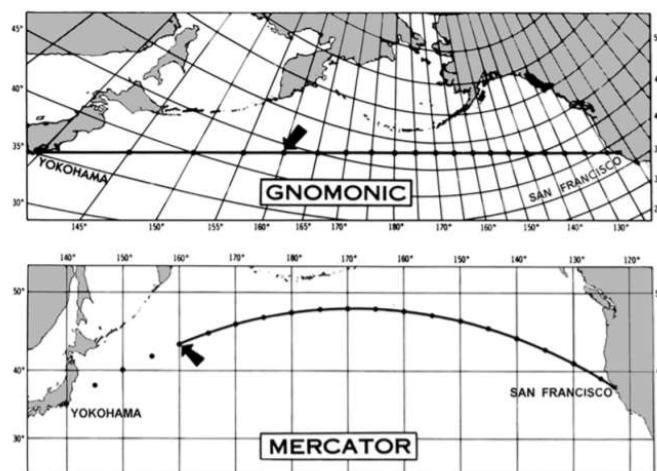


Figure 20 Constructing a great circle track on a gnomonic and a mercator projection (National Geospatial Intelligence Agency, 2019)

According to National Geospatial Intelligence Agency 2019, *American Practical Navigator*, Chapter 12, the most accurate method for numerically calculating any type of course and distance is to use spherical trigonometry and, more precisely, to apply the same methodology used in celestial navigation to sailing calculations. The point of departure is substituted for the observer's position, the geographical position of the

celestial body is substituted for the destination, the difference in longitude is substituted for the meridian or local hour angle, the azimuth angle is substituted for the initial course angle, and the great circle distance is substituted for the zenith distance.

4.2 Great Circle Navigation

According to National Geospatial Intelligence Agency 2019, American Practical Navigator, Chapter 12, by definition, a great circle is a circle on the surface of a sphere whose plane passes through its centre. The most well-known big circles on the earth's surface are the equator ($^{\circ}$ N/S) and the meridians. Thus, any vessel sailing on the equator in a due east/west direction or along a meridian in a due north/south direction is doing great circle sailing. These are not, however, the only big circles on the earth's surface. Any great circle created in any direction on the earth's surface will form a great circle track that a ship may follow.

According to National Geospatial Intelligence Agency 2019, American Practical Navigator, Chapter 12, several considerations to bear in mind while calculating a great circle track for sailing include the fact that the great circle track is always angled away from the equator.

4.3 Factors to consider during Great Circle Navigation

According to National Geospatial Intelligence Agency 2019, American Practical Navigator, Chapter 12, the shortest distance between two points is calculated along the arc of the great circle that links them. Traveling the shortest route between two sites through great circle sailing rather than rhumb line may save several hundred miles. As a result, it is clear that such considerable disparities should be considered while planning an ocean cruise for a vessel. The rhumb line, convenient as it is, should not be used for long voyages except as noted below.

1. For small distances, the rhumb line, and the great circle are nearly coincident.
2. The rhumb line between places near the same meridian is very nearly a great circle.
3. The Equator is both a rhumb line and a great circle. Parallels of latitude near the Equator are very nearly great circles.

According to National Geospatial Intelligence Agency 2019, American Practical Navigator, Chapter 12, Thus, parallel sailing, a kind of rhumb line sailing, is almost as short as a large circle at low latitudes. On the mercator map, the arc of the great circle connecting two locations (unless they are both on the same meridian or both on the equator) will seem to be longer than the rhumb line's shortest path. The rhumb line is the indirect path, and by following the big circle, the vessel is constantly heading directly toward her port, as if it were visible. While the mercator map depicts the path as a straight line, the vessel does not arrive at her port until the very end of her trip. If, when attempting to follow a large circle, the ship is pushed away from it, as could be the case due to inclement weather, it will serve no use to return to the former course at the earliest convenience. Another large circle must be drawn connecting the ship's current location with the port of destination. Except for the equator and constant meridian lines, any great circle intersects each meridian at a different angle. As a result, the path of a ship following a great circle track must be continually adjusted to maintain this angle. Due of the impracticality of continually altering the course of a vessel, this need is satisfied by changing the course at regular intervals. When a ship follows the great circle track, it follows a set of chords connecting the different places along the track's length. A sphere's great circle bisects every other great circle. As a result, the Equator bisects each of the Earth's great circles. As a result, if the great circle is stretched around the Earth, one half will be illustrated in the Northern and the other half in the Southern hemisphere, with the mid-point of either half being the point on the circle farthest from the Equator.

A great circle track may be drawn on a Mercator map by finding the track's number of points, drawing a reasonable curve across them, or connecting the points with a sequence of straight lines called chords. Theoretically, tangents to the great circle are preferable than chords. In practise, the procedures are almost identical. Points are picked along meridians, often at five-degree intervals, since this enables the selecting off and charting of latitudes and longitudes around the great circle track. This approach is often used in practise to identify and draw the path of a great circle on a Mercator map. One of the primary benefits of the solution through great circle chart is that any potential problems become readily visible. Land, ice, or extreme weather, as well as other operational restrictions, may preclude the utilisation of great circle sailing for

portions of or the whole of one's voyage. Pilot charts are very important in this situation. A brief run along the rhumb line is often sufficient to reach a place where the great circle route may be followed. Where a choice is available, the rhumb line picked should be as close to the direct great circle as feasible. Assume the big circle path comes dangerously close to a nautical hazard. In such instance, a great circle to the area of the danger, one or more rhumb lines along the hazard's boundary, and another great circle to the destination may be required. Another possibility is to employ composite sailing; another is to use two great circles, one from the place of departure to a position near the maximum latitude of clear water and another from that point to the destination.

4.4 Rhumb Line Navigation

Any line on the Earth's surface that, due to its constant bearing, cuts all the meridians at the same angle, known as a Rhumb Line course.

According to National Geospatial Intelligence Agency 2019, American Practical Navigator, Chapter 12, a rhumb line intersects all meridians at the same angle and is illustrated as a straight line on a Mercator map. The main benefit of the rhumb line is that it keeps a constant true direction, which means that a ship staying on a Rhumb Line course between two points does not deviate from its true heading. It is suitable for the majority of navigational applications, bearing lines included (except long-range such as the ones obtained by radio bearings). Except at high latitudes, both course lines and rhumb lines are represented on a Mercator map as rhumb lines. The path's spherical shape is illustrated by different formulas for Rhumb line sailings, which produce many alternative outcomes. They will provide varying results for routes, lengths, and locations based on the kind of sailing performed.

5 Factors affecting Route Calculation

The difference between a planned and an executed voyage is environmental and technical factors, potentially affecting the voyage execution. The factors that are affecting the voyage can be distinguished to technical, which are but not limited to the ship's draft, depth, breadth, Length overall, Deadweight, Displacement, Air Draft, Speed

and squat, and the environment such as wind/wave/current/swell direction and speed, barometric pressure, tides, salinity, ocean's depth, and characteristics to name a few.

Then these must be distinguished into static and dynamic so that we can assess and monitor passage planning and execution with the best possible result, according to the below table.

Table 1 List of Dynamic and Static Elements affecting Passage Planning, 2021

Static Elements	Dynamic Elements
Ship's Draft	Speed
Air Draft	Squat
Breadth	Under Keel Clearance (UKC)
Length overall	Wind direction and speed
Deadweight	Wave direction and speed
Displacement	Current direction and speed
Ocean's depth and bottom characteristics	Swell direction and speed
Metacentric Height (GoM)	Air & Water Temperature
Depth	Barometric pressure
	Tides
	Water salinity
	Ice Coverage and thickness
	Voyage Distance and Destination ETA (Estimated Time of Arrival)

(Source: Author, 2021)

Due to the dynamic nature and being affected by the morphology of the surrounding landscape and the other environmental factors, the environmental factors have a seasonal pattern that can make a significant difference in the long-distance voyage.

This example is shown in the Weather Routeing Charts of UKHO (United Kingdom Hydrographic Organization) and the World Sailing Routes by NGIA (National Geospatial Intelligence Agency) like in the Figure 21 below. They depict the optimal route to be

followed depending on the month sailing the area, and the average weather conditions met in these areas.

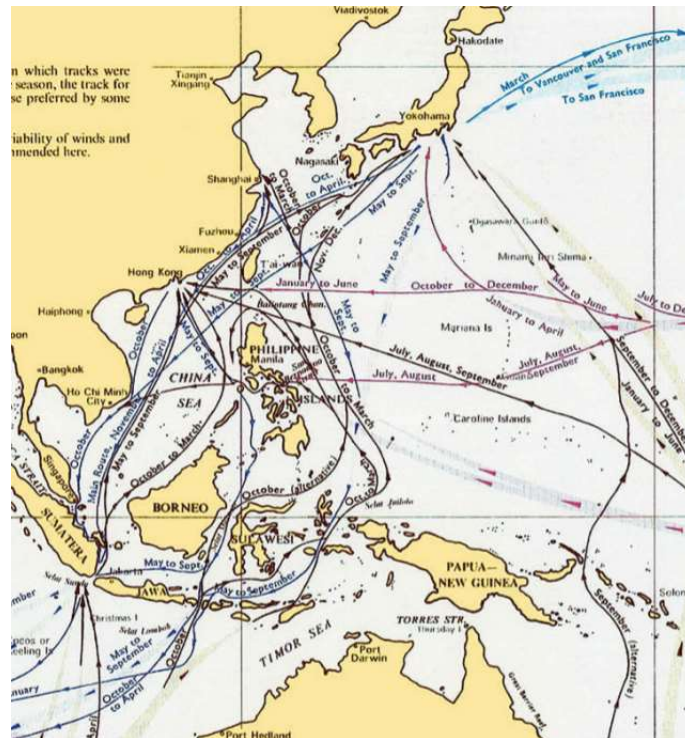


Figure 21 The World Sailing Ship Routes (National Geospatial Intelligence Agency, 2019)

6 History, Present and Future of Electronic Navigational Charts

In May 1992, Electronic Navigational Charts (ENC) officially accepted the IHO (International Hydrographic Organization) S-57 standard. They had, however, been created unofficially since the 1980s for military uses by the US and Canadian Navy.

In November 2000, after drafting the IHO Standard S-57 3.1, the IHO decided to defer its implementation in order to provide ECDIS (Electronic Chart Display Information System) manufacturers further time to produce equipment that complies with the IHO requirements.

Due to the demanding operational challenges inherent in maritime transportation and the ever-increasing pace of global trade, it was decided to create a new ENC standard that would incorporate the benefits of data available from a variety of sources (meteorological, hydrographic, geographic information science (GIS), etc.) and

advancements in maritime technology. Thus, IHO began developing S-100 in 2009 and completed it in January 2010 with the announcement of S-100 standard 1.0.0 and its implementation timeframe. S-100 edition 3.0.0 and S-101 edition 1.0.0 were created in 2017.

Advances in coding and digital hydrography, such as the new Hydrographic standards such as S-100, may be utilised to automate this procedure while taking into consideration the weather, depth, navigational hazards, and other pertinent information, much like a real-world Navigator would. The real-time incorporation of meteorological, hydrographic, and geographic information system (GIS) data will aid in the practical usage and requirement of ENC's in everyday marine operations.

S-100 will be critical in developing and establishing electronic navigation and MSDI (Marine Spatial Data Infrastructure) as essential capabilities of maritime logistics and operations.

According to the International Hydrographic Organization's (IHO) Marine Spatial Data Infrastructures Working Group (MSDIWG), a Marine Spatial Data Infrastructure (MSDI) is that component of a Spatial Data Infrastructure (SDI) that is focused on marine input in terms of governance, standards, information and communications technology (ICT), and content (IHO, 2017). The fact that 71% of the earth's surface is covered by water (USGS, 2018) underscores the crucial need of a maritime data gathering. According to the International Hydrographic Organization's Marine Spatial Data Infrastructures Working Group (MSDIWG), MSDI is gaining traction as a method for integrating disparate data sources for efficient analysis across a variety of disciplines, including spatial planning, environmental management, and emergency response. This demands generic data storage, rather as storage for a single product, a local user group, or a specific purpose. An MSDI is a collection of hydrographic products and an infrastructure that enables data exchange at all levels.

An MSDI can be described as a framework comprising of the key components shown in Figure 22 below.



Figure 22 Policy & Governance, Technical Standards, Information Systems and Geographic Content comprising the Four Pillars of MSDI, (IHO,2017)

Spatial Data is information or data that indicates the geographic location of features and limits on Earth and Space, including natural or man-made features, oceans, and space. It comprises information about these objects and boundaries, such as their properties, observations, and other metrics.

Spatial data is often represented using coordinates and topology and is therefore mappable. Geographic Information Systems are often used to manage, access, and analyse geographical data.

At the 1992 United Nations Conference on Environment and Development in Rio de Janeiro, a landmark resolution was adopted emphasising the need of reversing the effects of environmental degradation (GSDI, 2004). The Agenda 21 resolution establishes measures to tackle deforestation, pollution, the loss of fish stocks, and hazardous waste management, to name a few. At the 1992 Rio Summit and at a United Nations General Assembly special session called in 1997 to examine Agenda 21 implementation, the critical significance of geographical information in helping decision-making and management was underlined. In 2003, during the World Summit on Sustainable Development in Johannesburg, South Africa, a pioneering endeavour was made to illustrate the capabilities, benefits, and possibilities of employing online digital geographic information for sustainable development.

Geographic information is crucial for making informed decisions on a local, regional, and global scale. Crime management, business development, flood mitigation, environmental restoration, community land use assessments, and disaster recovery are

just a few of the areas in which decision-makers benefit from geographic information and the underlying infrastructures (i.e., Spatial Data Infrastructure or SDI) that facilitate information discovery, access, and use.

However, information is a valuable commodity. As a result, sufficient information and the resources necessary to effectively exploit it are not always easily accessible, especially in underdeveloped countries. By developing New Products with Commercial Potential, such as S-100, the most efficient and sustainable way to fund the production of Maritime Spatial Data without delay is to agree on a synchronous international implementation of the Dataset's regular production, validation/qualification, and dissemination. This necessitated coordination among multiple industry players to ensure that end-user devices were suitable for purpose. Legal recognition of new data products is necessary under the IMO framework, and IHO's duty is to guarantee that all technical and operational issues are considered in order to develop an internationally harmonised approach.

Table 2 ENC Global Coverage statistics 2008-2017,2018

ENC Global Coverage	
2008 ENC Global Coverage	2017 ENC Global Coverage
Small Scale Charts 94%	Small Scale Charts 100%
Medium Scale Charts 68%	Medium Scale Charts 93%
Large Scale Charts 65%	Large Scale Charts 98%

(Source: UKHO, 2018)

Currently, less than 15% of the world's ocean waters have been measured directly, and only about 50% of the world's coastal waters (waters 200m deep) have been surveyed. Increased seabed survey coverage is crucial for the sustainable use of the oceans, seas, and marine resources, and therefore for achieving the United Nations Sustainable Development Goal 14 (A. Wölfel, J. Jencks, and C. Devey, Hydro International, 2021).

On August 27, 2021, an international mission comprised of representatives from the United States (NOAA), the United Kingdom Hydrographic Office (UKHO), and the Canadian Hydrographic Service participated in the S-100 sea trial in Busan, Republic of Korea., KHOA (Korean Hydrographic and Oceanographic Office) used an S-100 prototype ECDIS as shown in Figure 23 below.

According to the report contained on the page "S-100 sea trials: working toward standardised navigation products," NOAA, 2021, the primary objective of the S-100 sea trial was to evaluate a prototype S-100-based Electronic Chart Display and Information System (ECDIS) capable of ingesting and displaying a variety of datasets including 1) S-101 – Electronic Navigational Charts, 2) S-102 – High Resolution Bathymetry, 3) S-111 – Surface Currents, and 4) Throughout the route, test scenarios were conducted to check 1) how the system integrated the various kinds of datasets and 2) how the high-resolution bathymetry layer can be used to offer a more accurate safety contour and depict go/no-go situations based on underkeel clearance management data. Following the sea testing, the crew was able to provide suggestions to the IHO's S-100 working group on how to enhance the currently developing interoperability standard. Additionally, gaps were discovered that must be addressed as the IHO continues to refine product requirements. For instance, it was determined that, while data such as marine protected areas (MPA) are necessary for planning purposes, they may not need to be displayed on the front-of-bridge navigation system because the regulatory information associated with the MPA is already included in the base electronic navigational chart (ENC). Another critical feature that became obvious during the sea testing was the need for a consistent representation of data inside the navigation system. If each depiction of the product is produced independently, there is too much information when they are combined in a single system, rendering the system ineffective from a navigational standpoint. This will need enhanced coordination among IHO working groups to ensure that main navigation depiction is actually consistent.



Figure 23 S-100 prototype ECDIS displaying S-101 electronic navigational chart in conjunction with S-129 Underkeel Clearance management data. (NOAA, 2019)

7 S-100 ISO Standards

This new type of ENCs will bring innovations in the Maritime Navigation domain by not developing new components in isolation but together with other related ones, extensive and active feature catalogue registry, symbology, and software enhancements such as Layer Interoperability Catalogue (IC), more straightforward use of hydro data beyond Hydrographic Organizations and ECDIS users, S-100 being built using Unified Modelling Language (UML) thus offering more straightforward service through different platforms and through these innovations the IHO strives to bring Maritime Hydrography and Navigation into the mainstream GIS.

The IHO S-100 Data Framework is supported on various ISO 19100 Standards for the following purposes, as listed in the figure 24 below, in the Appendix II “List of the individual parts, their associated part numbers, and ISO 19100 Conformance standards”.

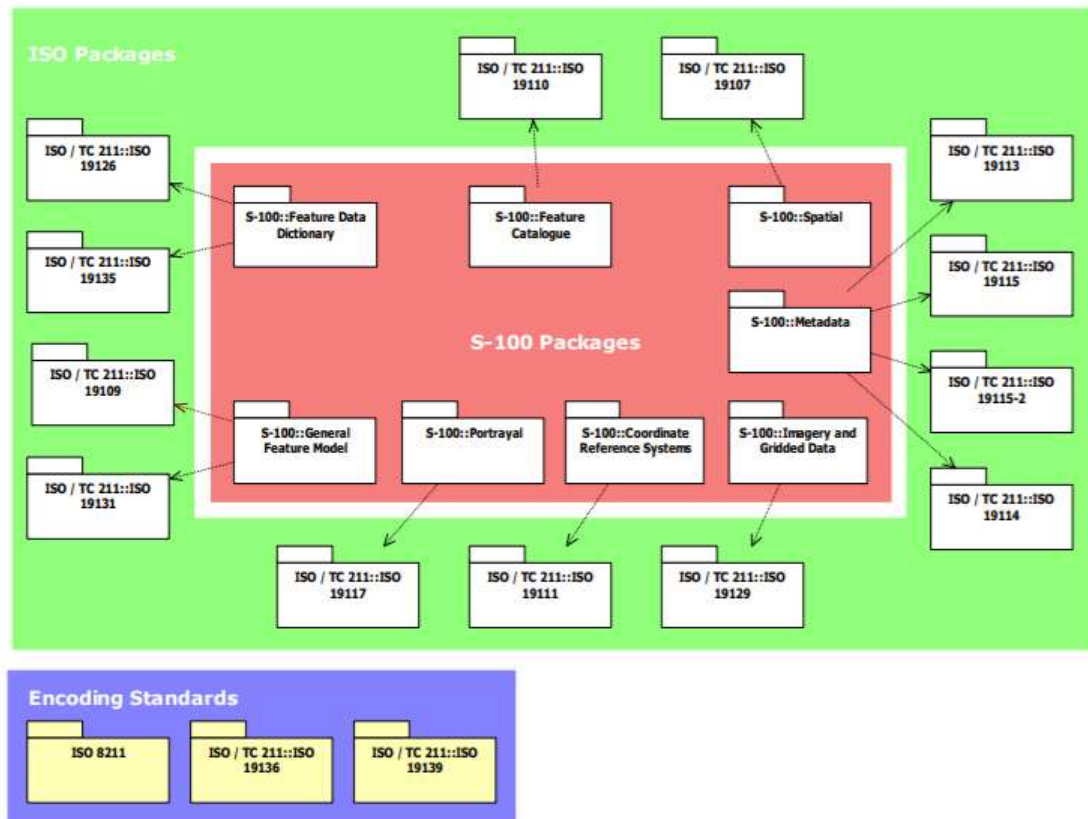


Figure 24 IHO S-100 Components and their Associated ISO Standards (Robert Ward, Lee Alexander, Barrie Greenslade, Anthony Pharaoh, 2008)

According to IHO (2019), the S-100 Standard's primary objectives are to enhance navigational safety, innovate standardised service delivery by decoupling data content from encoding format, enabling format-neutral product specifications, unlock the potential of marine geospatial data by supporting a broader range of marine or hydrographic-related digital data products and customers, and provide dynamic data for environmental intelligence.

To that end, IHO (2019) intends to allow product specifications to evolve through extension without the need to publish new versions of existing product specifications, to provide an ISO-compliant registry managed by the IHO that contains flexible and expandable registers such as feature concept dictionaries and product feature

catalogues, and to provide separate registers for different user communities. The Plug and play updating capability is one of the new standard's primary advancements. This eliminates the requirement for new product specifications or system changes. To make it more user-friendly, the majority of features and attributes will stay intact, but they will be implemented in a structured and methodical manner, as seen below in Figures 25 and 26 below. Nevertheless, the inclusion of additional layers will need a modification in ECDIS operating methodologies. This is the function of the Interoperability Catalogue (IC). The benefits of the S-100 standard to we will present include a new loading strategy for ENC Cells, advanced pick reports for ENC Data, notice-accompanied updates, a new information type, improved text placement, a new method of calculating and displaying bathymetric quality information, and additional information on the ENC.

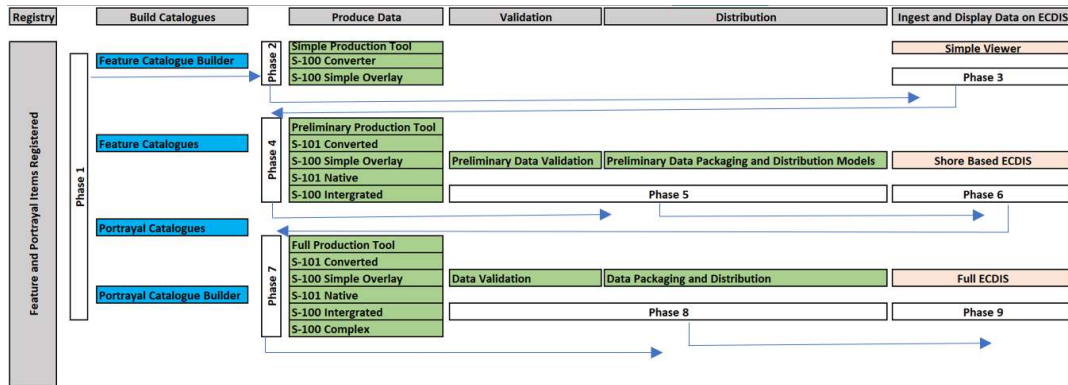


Figure 25 S-100 and S-101 Implementation Plan Phases (KHOA, 2017)

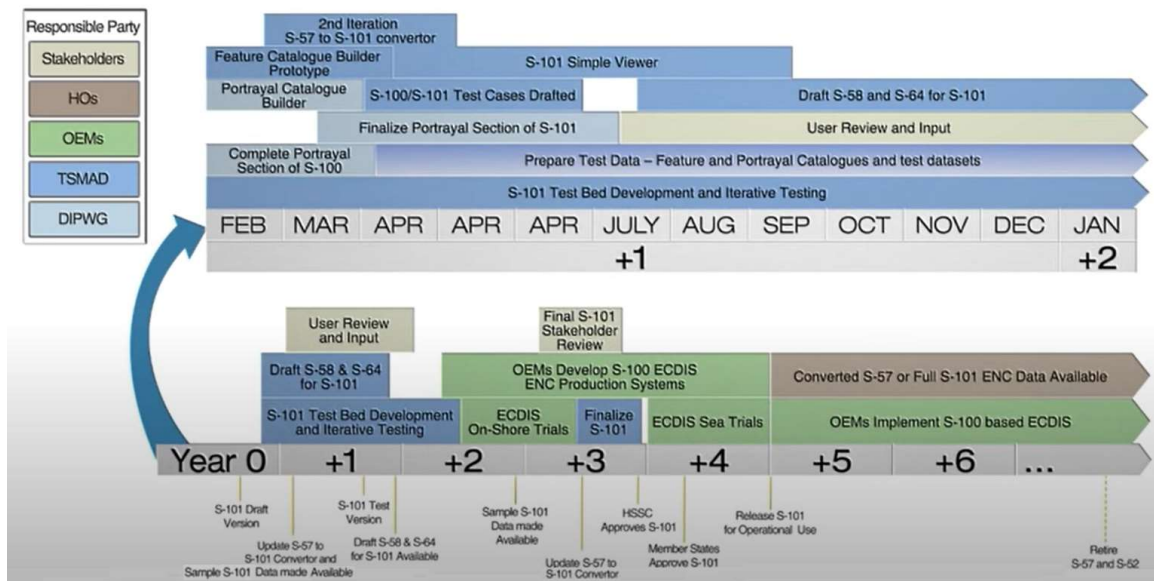


Figure 26 S-100 and S-101 Implementation Plan Timeline (KHOA, 2018)

The S-100 IHO Standard is comprised of the following ten parts:

- 1) Establish the S-100 Registry / Register Mechanism
- 2) Provide Guidance on creating application Schemas
- 3) Metadata
- 4) Creating Feature Catalogues
- 5) Coordinate Reference Systems
- 6) Spatial Properties
- 7) Imagery and Gridded data
- 8) Creating Portrayal Catalogues
- 9) Encoding Formats
- 10) Creating Product Specifications

7.1 Types of S-100 Layers

The S-100 IHO standard introduces various kinds of interchangeable layers, each serving a different purpose, which can suite the navigator depending on the type of data or combination of data he needs. These can be summarized depending on the Scale of Navigation, Type and Issuing authority in Appendix III” ENC Layers types depending on Issuing Authority and purpose of use”, and in conjunction with Figures 27 to 28 contained in this chapter.

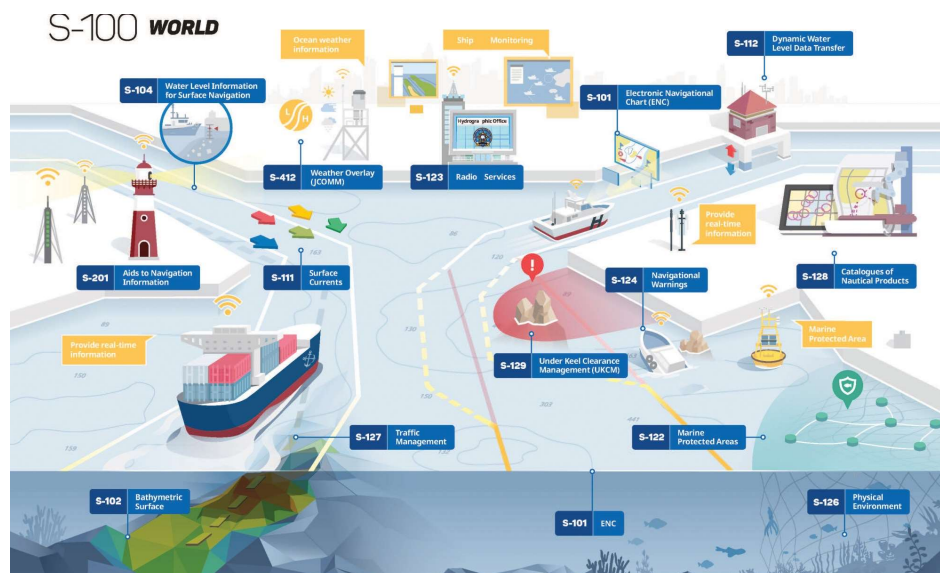


Figure 27 Visual Portrayal of S-100 ENC Types and Uses (KHOA, 2018)

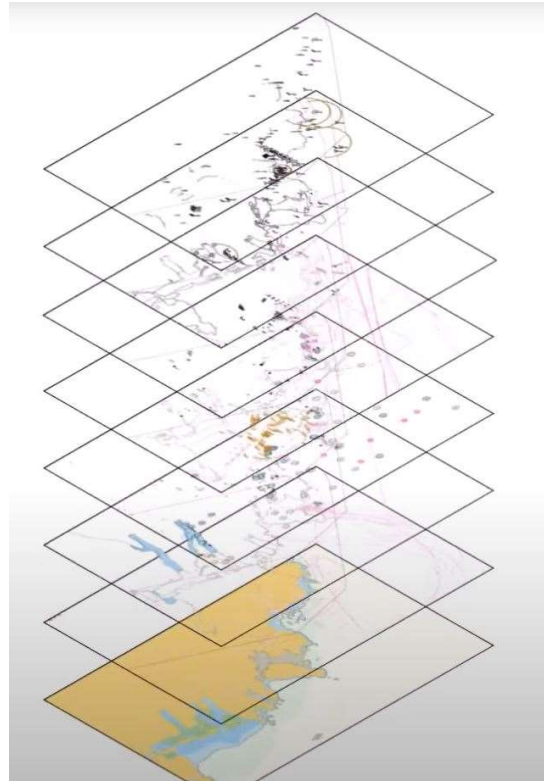


Figure 28 Portrayal of S-101 ENC various object layers (KHOA, 2018)

In addition, one of the differentiating but subtle attributes of S-101 is the difference of scale ranges depending on the navigational purpose of the ENC as shown in the table below.

Table 3 ENC Usage Bands, 2019

Navigational Purpose	Radar	S-101	S-57		Current NOAA ENCs	
	Standard Ranges	Display Scale Ranges	Larger Scale Limit	Smaller Scale Limit	Larger Scale Limit	Smaller Scale Limit
1. Overview	3 000 000 1 500 000	10 000 000 1 500 000	1 500 000		1 500 001	
2. General	1 500 000 700 000 350 000	1 500 000 700 000 350 000	350 000	1 499 999	600 001	1 500 000
3. Coastal	350 000 180 000 90 000	350 000 180 000 90 000	90 000	349 999	150 001	600 000
4. Approach	90 000 45 000 22 000	90 000 45 000 22 000	22 000	89 999	50 001	150 000
5. Harbor	22 000 12 000 8 000 4 000	22 000 12 000 8 000 4 000	4 000	21 999	5 001	50 000
6. Berthing	4 000	4 000 3 000 2 000 1 000		3 999		5 000

(Source: NOAA, 2019)

7.2 Data structures and attributes used in S-100

The S-100 Data Framework is divided into a) Feature Concept Dictionary Register, b) Portrayal Register, c) Metadata Register, e) Product Specifications Register and f) Data Producer Code Register

The process of building product specifications based on S-100 can be divided into the following phases according to the below table.

Table 4 Phases of S-100 ENC Product Specification Assembly, 2015

Phases of S-100 ENC Product Specification Assembly	
Phase 1	User Requirements Assessment
Phase 2	Data Modelling (Application Schema)
Phase 3	Data Classification and Encoding
Phase 4	Create a Product Specification Document
Phase 5	Registry / Register Work
Phase 6	Product Verification

(Source: IHO, 2015)

Due to the large number of layers used in S-100 and the potential for information overload, symbol cluttering can potentially be a source of error when used, particularly for mission-critical tasks such as transiting shallow water, navigating in bad weather, or performing other tasks with a similar level of risk.

To this end, KHOA has recommended to IHO (Paper for Consideration by TSM5, Procedures for S-100 Interoperability Catalogue, KHOA, 2017) that the Interoperability Catalogue (IC) be created. The IC is used to de-clutter displays, reduce information overload, resolve information and symbol conflicts, and improve the overall quality and understandability of information presented to mariners when multiple S-100-based data products are displayed simultaneously. One may argue that this functionality is a game changer for ENCs and maritime geographic information systems in general. If applied

properly, it may provide novel methods for doing Maritime Navigation-related activities or Operational Planning that take into account all possible environmental aspects.

The Interoperability Catalogue has been separated into five (5) levels, each of which corresponds to the proper level of operation and layer count, as seen in the table below.

Table 5 Table of Interoperability Catalogue (IC) Levels, 2018

Level	Definition
Level 0	All interoperability processing is turned off. In this case, feature data pass unchanged through standard portrayal processing.
Level 1	Feature types from different products, including S-101, are interleaved as specified by the display plane and drawing priority information in the interoperability catalogue.
Level 2	If feature types in other products are determined to be superior in terms of accuracy and quality to specific ENC feature types, then these ENC feature types are suppressed.
Level 3	The ENC is treated as one of the components of the data stack, and selected feature instances from other products may be treated as being superior to or enhancing selected ENC feature instances.
Level 4	This level is the same as level 3 but permitted spatial queries to determine related subsets. Operations to define the interoperation results are explicitly defined using an adequate set of spatially capable rules.

(Source: KHOA, 2018)

Level 0 displays the S-101 as layers by first sorting them according to their qualities and then combining all S-101 layers to generate the final output. Then, additional S-10x products, such as S-102 and S-111 in the following example, are superimposed.

At Level 1, the Interoperability Display Plane is applied to permit viewing of the S-101 ENCs, and then all of the S-10x ENCs' layers are combined into a single final layer with all of their properties.

To allow viewing of the S-101 ENCs at Level 2, an Interoperability Suppressed Layer with just the required properties is deployed. The S-10x ENCs' layers are then blended into a single final layer with just the necessary properties.

To explain how the Interoperability Catalogue works, three practical examples illustrated in figures 29 to 32, will be presented below utilising a Nautical S-101 ENC, a Bathymetric S-102 ENC, and a Current S-111 ENC.

Scenario_01 : S-101 + S-102 + S-111

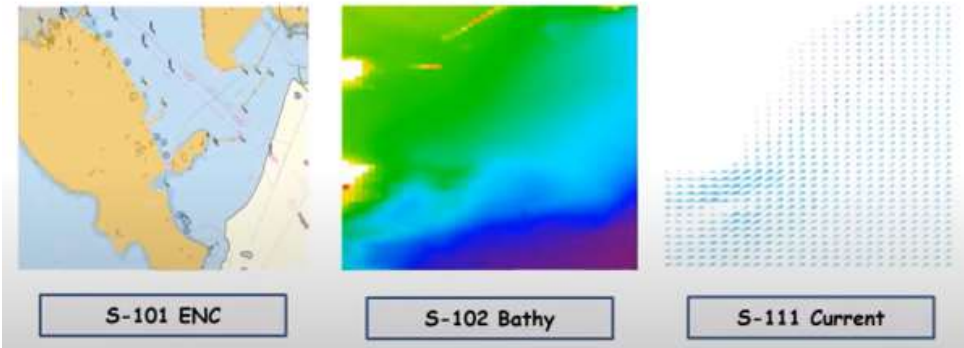


Figure 29 S-10x ENC layers used for the Demonstration (KHOA, 2018)

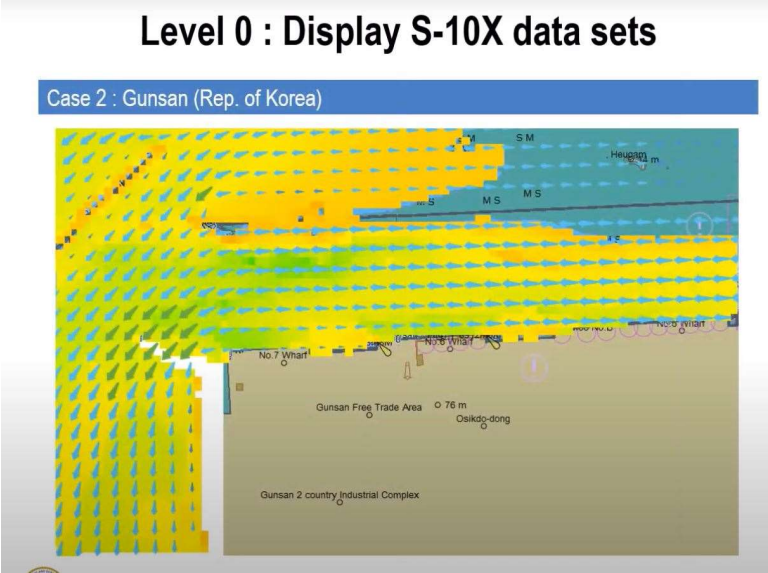


Figure 30 Level 0 final depiction of the S-10x ENCs (KHOA, 2018)

Level 1 : Display S-10X data sets

Case 1 : Gunsan (Rep. of Korea)

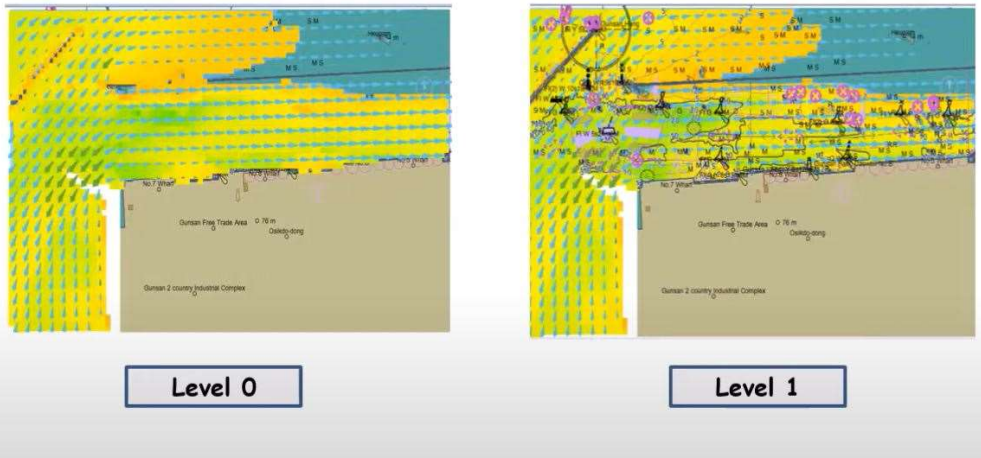


Figure 31 Level 1 final depiction of the S-10x ENC (KHOA, 2018)

Level 2 : Display S-10X data sets

Case 1 : Gunsan (Rep. of Korea)



Figure 32 Level 1 and Level 2 final depiction of the S-10x ENC (KHOA, 2018)

7.3 Display and data integrity of S-100 ENCs

As the S-100 are currently, as of May/2021, still under development, there will be no mention of said standards. More specifically, for Validation checks, there is a

placeholder in the S-101 document under Annex C – S-101 Validation Checks, which doesn't have any specific information as of now.

As for display standards, in the same document, it's mentioned that the Portrayal Catalogue contains the mechanisms for the system to portray information found in S-101 ENCs. More specifically, the S-101 Portrayal Catalogue information are contained in the `portrayal_catalogue.xml` of the S-101 ENC Standard.

8 Pathfinding / Wayfinding Algorithms

A prerequisite to examining the different types of pathfinding algorithms is to review their history and their basic features or characteristics briefly. At the foundational level, pathfinding describes the process of identifying the shortest route between two end points, for instance, in route planning, maze navigation, planning of robot paths, and transit planning. Mathew (2015) also posits that pathfinding involves plotting, using computer applications to identify the optimal (shortest) route between the start and end nodes. According to the researcher, finding a path between points is highly complicated, and a distinction is identified between a path and the shortest path. In a different study, Sidhu (2020) describes pathfinding as a process of generating the optimal route that exists in a map between the start and end nodes, wherein artificial intelligence (AI), the algorithm is designed as a kind of graph search. However, the researcher reiterates that despite being widely adopted in gaming and artificial intelligence, pathfinding is also utilized in applications such as navigation systems, robotics, and networks.

A similar finding is also emphasized by Abd Algfoor, Sunar, and Kolivand (2015). They report that pathfinding has emerged as an essential component in diverse applications such as GPS, robotics, video games, and crowds' simulation. The researchers also observe that pathfinding can be implemented in various types of environments, including real-time, static, and dynamic. Abd Algfoor, Sunar, and Kolivand (2015) further add that pathfinding techniques have improved accuracy and efficiency over the years as diverse researchers have focused on developing high-performance and realistic paths for users. Directly, this indicates that pathfinding seeks to create natural and high-performance paths at a basic level.

8.1 Pathfinding problem variations

According to Botea, A., Bouzy, B., Buro, M., Bauckhage, C. and Nau, D. (2013), diverse variations of the pathfinding problem have emerged over the years, for instance, single-agent pathfinding search, multiagent pathfinding search, adversarial pathfinding, heterogeneous terrain, mobile units, incomplete information, and dynamic changes in the environment. Despite the diversity in pathfinding problems, the researchers, however, argue that pathfinding essentially involves two phases or timesteps: graph generation and algorithms for pathfinding. The steps that help solve the shortest path and optimal path problems are completed in different pathfinding algorithms, including Dijkstra, A* algorithm, genetic algorithms, and ant colony (Rafiq, Kadir, and Ihsan, 2020). The researchers further posit that A* and Dijkstra algorithms are used in finding the shortest paths in a brief explanation. In contrast, the genetic and ant colony algorithms are utilized as search techniques to identify the minimum cost in a graph. The four different pathfinding algorithms are examined in detail in section 15.1, while this section discusses the variations of pathfinding problems further. In particular, three variants are examined: single-agent, multi-agent, and adversarial search pathfinding.

To begin with, single-agent pathfinding problems are defined as problems that involve finding paths between two graph vertices (Sharon, G., Stern, R., Felner, A. and Sturtevant, N.R., 2015). The researchers argue that such problems have been widely researched in diverse applications such as robot routing, GPS navigation, network routing, planning, and combinatorial problems. In a different research, Standley and Korf (2011) further distinguish the single-agent from multiple-agent pathfinding algorithms by highlighting that the single-agent problems contain only one agent and are typically solved using algorithms for graph search-based on the A* algorithm. On the contrary, the researchers argue that multi-agent problems have multiple agents in the same problem space. When computing a solution, there is a need to be careful to avoid conflict between the agents. Sharon, G., Stern, R., Felner, A. and Sturtevant, N.R. (2015) reiterate the findings by observing that the multi-agent kind of pathfinding problems can be generalized as the single-agent pathfinding problems where there is more than one agent ($k > 1$).

Further review by Bulitko Björnsson, Y., Sturtevant, N.R. & Lawrence, R. (2011) outline how the pathfinding algorithms work, whereby the researchers highlight that a requirement for the single-agent problems is that, before an agent can move, the algorithm requires to generate a complete and possibly an abstract path for each agent. As a result, this mode of operation causes them to scale poorly.

The evaluation of the studies (Sharon, Stern, R., Goldenberg, M. & Felner, A. 2013; Bulitko, Björnsson, Y., Sturtevant, N.R. & Lawrence, R., 2011) indicate two differences exist between the single-agent and multi-agent pathfinding problems. First, as the authors posit, the single-agent problem comprises only one agent in the problem space, and its solution involves identifying a path between two graph vertices. In contrast, the multi-agent problem involves multiple agents in the same problem space, whereby each agent is allocated a unique start and end goal state (Sharon, Stern, R., Felner, A. & Sturtevant, N.R., 2015). Secondly, the multi-agent search algorithm must identify paths between all agents from the beginning to end goal states while constrained by the fact that the agents should not collide during their movement in the problem space (Standley and Korf, 2011). Kem, Balbo, and Zimmermann (2017) further add that in other instances, the multi-agent pathfinding problem also aims to minimize the cumulative cost function, for example, the total number of time steps needed to ensure every agent can attain their goal. However, with single-agent algorithms, such constraints are not imposed on the problem space agent. A third difference between the single-agent and multi-agent pathfinding problems also arises, like their associated applications. For instance, Kem, Balbo, and Zimmermann (2017) posit that single-agent applications typically include the traveling salesman or proving theorem, while multi-agent applications comprise robotics, aviation, video games, and traffic control.

The third kind of pathfinding problem is the adversarial search problem (Ivanová and Surynek, 2014). According to the authors, the adversarial cooperative pathfinding problem (ACPF) generalizes cooperative pathfinding (CPF), whereby an extension is made by adding an adversarial component. In explanation, Ivanová and Surynek (2014) posit that with a fundamental CPF problem, the objective is to identify routes that do not collide for agents, allowing them to traverse from their initial start nodes to unique one's disjoint destinations. However, an additional feature with these problems is centralized control, whereby the agents are placed in static environments that are fully

observable and controlled. The agents do not make any decisions (Yannakakis and Togelius, 2018). The authors posit that the ACPF problem advances the CPF by adding an adversarial element outside the central planning mechanism control and acts against it. Ivanová and Surynek (2014) further explain that the classical CPF problems are challenged by their limited expressive power in the real world, as such environments are usually not fully cooperative. Subsequently, the adoption of ACPF pathfinding is justified due to the complexity of the dynamic real-world environment, making it more challenging to control.

Based on the brief review of pathfinding problems, the multiple agent pathfinding (MAPF) searches is the most popular approach. It has been adopted in diverse applications in the real world, including aviation, robotics, video games, and traffic control (Sharon, Stern, R., Felner, A. & Sturtevant, N.R., 2015). The review also showed that in real-world situations, the environments are not fully cooperative. As a result, classical cooperative pathfinding may not be optimal in finding solutions in the problem space. As a result, the need for adversarial cooperative pathfinding was emphasized (Ivanová and Surynek, 2014). In the next section, a brief discussion of these applications is undertaken to understand how MAPF is utilized in real applications such as video games, robotics, and ship routing. The justification for reviewing such applications is that this provides insights that further understand route planning, which is the main focus of this research.

8.1.1 Pathfinding algorithms adoption in video games

According to Cui and Shi (2011), usage of pathfinding in video games is quite common. Different roles are played, and real-time strategies employed, for instance, where characters are sent to complete missions from one location to another. In this regard, pathfinding is utilized to allow the game characters to cleverly avoid obstacles and identify the most efficient path over different terrain. In another research, Mathew (2015) postulates that pathfinding in video games concerns how objects find paths around obstacles, such as how players lead a group of units in the play area containing the different obstacles.

Bulitko, Björnsson, Y., Sturtevant, N.R. & Lawrence, R. (2011) further report that a common problem in video games regards identifying an existent path between two

nodes or locations, whereby agents need to act rapidly in response to player commands and actions by other agents. The researchers add that due to the requirement, most game developers require a time limit that is constant for the path planning amounts per move, for instance, a millisecond for each movement by simultaneous agents in the player space. However, an unintended consequence of such requirements is that pathfinding actions consume significant CPU (central processing unit) resources (Mocholi, Jaen, J., Catala, A. and Navarro, E., 2010). The authors postulate that as modern games feature more dynamic worlds and realistic characters, the problem sizes scale significantly, requiring more computing resources. To tackle the issue, Mocholi, Jaen, J., Catala, A. and Navarro, E. (2010) advocate for ant colony algorithms that consider characters' emotions in pathfinding applications. Bulitko, Björnsson, Y., Sturtevant, N.R. & Lawrence, R. (2011) share a similar view, whereby they argue that conventional algorithms based on A* employed in pathfinding applications in games are challenged because they scale poorly. In explanation, the researchers say that the use of static search algorithms, including A*, Iterative Deepening Algorithm (IDA*), Anytime Algorithms (ARA*), PRA*, DA* and D*, is not efficient as the various algorithms required to generate a solution that is complete before taking the first action. Subsequently, the increase in the size of the problems leads to an increase in planning time and a risk of exceeding prior finite upper bounds (Bulitko, Björnsson, Y., Sturtevant, N.R. & Lawrence, R., 2011).

To alleviate the shortcomings of static search pathfinding algorithms based on A*, real-time heuristic search algorithms are advocated (Lawrence and Bulitko, 2010). The researchers argue that such algorithms are justified in video games due to the requirement for rapid response time and the time limitation in the environments. In explanation, Bulitko, Björnsson, Y., Sturtevant, N.R. & Lawrence, R. (2011) report that search algorithms that are real-time in nature can address the problems in that, instead of computing all actions before deciding which one the agent ought to take, only a few steps are processed by undertaking a lookahead search of a fixed depth in regard to the agent's current status. Thereafter, a heuristic algorithm is used to compute the rest of the actions and update the function over time. However, Lawrence and Bulitko (2010) argue that the shortcoming associated with real-time heuristic algorithms is that they do not view the end goal state. As a result, the agent has a risk of hitting a dead-end or

selecting suboptimal actions. The authors also highlight a second disadvantage because the real-time heuristic search agents are further observed to repeatedly visit the same state space, a phenomenon known as scrubbing (Lawrence and Bulitko, 2010). The implication is that the solution quality tends to be low while the scrubbing behaviour is considered irrational.

Based on the broad application of real-time heuristic algorithms and static search variants, many researchers have focused on enhancing the algorithms in order to improve overall performance. With the heuristic algorithms, the focus is directed towards ensuring they deliver complete solutions. In contrast, with the A* algorithms, there is a focus on improving their scaling capacity with increased problem sizes (Bulitko, Björnsson, Y., Sturtevant, N.R. & Lawrence, R.,2011). For instance, Huang (2020) advocated for an approach to enhance A* algorithms by adding heuristics components to ensure the search for redundant paths was optimized. Thereafter, binary heap would be utilized to store the open list in the A* algorithm and further optimize the weight of paths in the map. The generated path was also further smoothed, and optimization of the A* algorithm done using the zonal search method (Huang, 2020). Findings obtained showed that the pathfinding efficiency improved, and player experience further enhanced.

8.1.2 Pathfinding algorithms adoption in robotics

In the robotics field, Abd Algfoor, Sunar, and Kolivand (2015) posit that since robots move in unpredictable, complex, unknown, and cluttered environments, the robots' movement should ensure they can detect obstacles and avoid them to reach their destinations successfully. As such, robots operate under two key constraints; avoiding obstacles in their environments and ensuring they reach their end goal destinations successfully. In another study, Standley and Korf (2011) add that robots employ cooperative pathfinding algorithms. They facilitate the plan motions of multiple robotic arms, whereby each can accomplish different tasks without collision. However, the limitation with these studies (Standley and Korf, 2011; Abd Algfoor, Sunar, and Kolivand, 2015) is that they only focus on the deployment of pathfinding in scenarios where robots operate singularly. They explain how the agents navigate in environments and avoid obstacles where they are on their own.

Further study by Mai and Mostaghim (2020) nonetheless reveals that pathfinding algorithms are also practical in scenarios where there are multi-robot systems. The algorithms have to mitigate collisions between robots and obstacles and other robots as they attempt to navigate from start to end nodes. There is an additional need to avoid collisions between the robots that travel within the same space simultaneously. Wei, Hindriks, and Jonker (2015) add that multi-robot cooperative pathfinding problems are challenging. The authors further argue that identifying an optimal solution for the problems is intractable and NP-hard as solutions that are centralized and based on a global search can ensure completeness but cannot scale optimally in large teams of robots and do not perform well in real-time.

Likewise, Luna and Bekris (2011) report that adopting decoupled decentralized solutions, where the robotic systems are provided with an extra level of autonomy, is also challenged because they do not guarantee completeness despite working well in real-time applications. As a result, the robots may end up getting stuck in deadlocked situations. One strategy to alleviate the challenge was proposed by Wei, Hindriks, and Jonker (2015), whereby instead of allowing the robots to access a shared database, similar to the decoupled scenario, the robots would be allowed to communicate with each other to keep track of their states and intentions.

Other researchers have further advocated for novel techniques to guide robots in identifying optimal paths to facilitate goal attainment. For instance, Upadhyay, Shrimali, and Shukla (2018) demonstrated unmanned aerial vehicles (UAV) in leading n robots to n different destinations or end goals. The UAVs acted as the leaders of the robots by having a bird's eye environmental view. In the study, an algorithm was deployed to create a relationship between the UAVs and the robots. In turn, the UAV would help create an efficient path for each robot, thereby avoiding collision between robots and obstacles and other robots. A similar study was also conducted by Luo, Espinosa, A.P., De Gloria, A. & Sgherri, R. (2011). They revealed that teamwork between UAVs and robots could be harnessed in undertaking rescue plans in different kinds of hazardous accidents. In the study, the researchers argued that the relationship was efficient in scenarios where outdoor robots could not obtain GPS information from satellites to traverse different terrain. In such a case, micro aerial vehicles (MAV) with cameras

would search the damaged area and communicate with the target robots to undertake the rescue operations.

8.1.3 Pathfinding algorithms adoption in ship routing

Similar to video games and robotic movement, ship routing also adopts pathfinding algorithms to identify optimal sailing courses and speed of the ocean voyage, which are based on nautical charts, captain experiences, sea conditions that are forecasted, and the individual ship features in a particular route (Grifoll, Martorell, L., Castells, M. & de Osés, F.Xavier.M.,2018). In a different study, Shin, Y.W., Abebe, M., Noh, Y., Lee, S., Lee, I., Kim, D., Bae, J. & Kim, K.C. (2020) report that various pathfinding algorithms have been developed to identify ship routes to guarantee economical ship operation by harnessing data on marine climate and Automatic Identification System (AIS). The researchers argue that optimization of shipping routes is crucial as ship owners aim to minimize fuel consumption while also ensuring that minimal emissions are generated to mitigate climate change effects. Shin, Y.W., Abebe, M., Noh, Y., Lee, S., Lee, I., Kim, D., Bae, J. & Kim, K.C. (2020) nonetheless argue that the algorithms used are not efficient in providing optimal routes as they fail to account for experimental conditions of operation such as ocean and weather. In the study, the researchers advocated for adopting an enhanced A* algorithm using weather data and AIS to mitigate the limitations of the conventional A* algorithm.

The suggestion from the evaluation of the studies is that pathfinding in virtual (video game) and natural (robotic or ship routing) environments involve the identification of the shortest path from the start to end nodes as well as avoiding obstacles in the terrain as well as other agents within the same environments. However, several noteworthy differences were observed among the various applications.

8.2 Attributes of Pathfinding / Wayfinding Algorithms important to the process

Pathfinding / Wayfinding algorithms are being used in our daily lives in a wide variety of applications, ranging from Logistics to IT and Finance. The theory behind each algorithm is on its own a wide subject to be discussed in detail in this Thesis, so in this chapter analysis of the components / attributes of the algorithms found in Chapter 7.2, will only

be discussed, which in my opinion are the most pertinent to Automatic Route Calculation.

8.2.1 Dijkstra algorithm

According to Shu-Xi (2012), the Dijkstra algorithm is a popular shortest-path algorithm known as the labelling algorithm. The author reports that shortest-path problems are prevalent in different management, organization, and production settings. For instance, in the production process, shortest path solutions are necessary to complete production tasks quickly and with high efficacy (Shu-Xi, 2012). Zingaro (2021) also holds a similar view, who reports that rational plans are required to ensure significant gains can be made at minimal costs in management processes. Likewise, there is also a further need to ensure a large number of goods can be transported at minimal costs in transport settings. Aside from transport, management, and production applications, shortest-path problems are also widely adopted in diverse fields, including computer network routing algorithms, route navigation, game design, and robot Pathfinder, among others (Shu-Xi, 2012).

In another study, Javaid (2013) reports that the Dijkstra algorithm tackles the challenge of identifying the shortest path between a node in the graph to a destination node at the foundational level. In further explanation, the author also adds that with the algorithm, one can identify the shortest path from one location to all graph points simultaneously, hence being termed as a single-source shortest-path problem. Further review by Neapolitan (2015) also reveals that the Dijkstra algorithm closely resembles the breadth-first search algorithm, which is utilized in identifying the shortest path between nodes in an unweighted graph. However, in contrast, the Dijkstra algorithm is used to determine the shortest path between nodes in a weighted graph. Despite the difference, Zingaro (2021) argues that the Dijkstra algorithm can function like the breadth-first algorithm (BFS), where the edges in the graph are assigned a weight of 1. As a result, by identifying the shortest path in the unweighted graph, the algorithm can minimize the number of edges, similar to the BFS algorithm.

In order to understand how the Dijkstra algorithm works, there is also a further need to examine its underlying basics. For instance, as Nalepa (2020) reports, the algorithm starts at a selected node. It proceeds to analyze the entire graph in order to identify the

existent shortest path between a node and all other graphical nodes. Secondly, Neapolitan (2015) adds that the algorithm maintains the shortest distance between each node and the source node currently known and constantly updates the node values to find a more straightforward path. A third feature is that when the algorithm identifies the shortest path between the source node and other nodes, the node is identified as being visited and further added to the optimal path (Javaid, 2013). By iterating the three steps, the Dijkstra algorithm can add all nodes to the track in the graph, thereby allowing the source node to connect to other nodes by following the shortest path possible (Nalepa, 2020).

An example of solving the shortest path to different nodes using the Dijkstra algorithm is further detailed by Navone (2020), as described in the figure below.

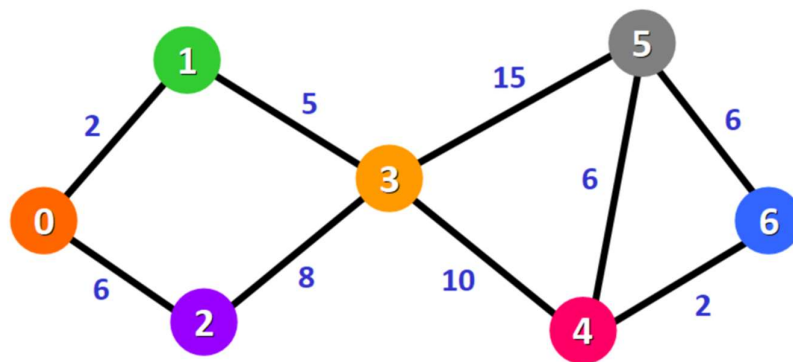


Figure 33 Solving for the shortest path using the Dijkstra algorithm (Navone, 2020)

From figure 33, the main objective is to identify the shortest path from the start node (0) to each of the nodes indicated by numbers (1 to 6). To tackle the problem using the algorithm, there is a need to create a list of both visited and unvisited nodes each time the algorithm calculates the shortest path between two nodes (Navone, 2020).

From the detailed description of the Dijkstra algorithm, a further need arises to examine the advantages and disadvantages it is associated with. To begin with, Lee (2012) highlights that a critical advantage of the Dijkstra algorithm stems from the fact that it is pretty straightforward due to its linear nature. As a result, the algorithm can compute the shortest path from a single node to all other network nodes and from a single source to another single destination node by halting the algorithm whenever the shortest route

is attained. Thirdly, Venkat (2014) also adds that once its output has been generated with the algorithm, it becomes easier to identify the least weight path to all nodes that are marked permanently. In this regard, therefore, a new diagram is not required when each pass has to be developed.

The author further adds that the Dijkstra algorithm has a time complexity of $O(n^2)$. In explanation, Nalepa (2020) posits that the algorithm has to make $n-1$ iterations for a graph with n nodes. After each iteration, a node is added as a marker to the corresponding node set, leading to the iteration experience. Following the iteration, the algorithm processes information from n_i nodes, the block weight, and other information. In turn, this leads to the processing of n network nodes and the time complexity of $O(n^2)$. A fourth advantage highlighted by Nalepa (2020) is that the algorithm has an order of n^2 and, as such, can be applied in solving relatively large problems.

Despite the advantages, the Dijkstra algorithm is associated with various disadvantages, a key aspect being that the algorithm is associated with a slow search speed and low time-consuming efficiency because the algorithm focuses on finding all optimal paths at an accuracy of 100% (Forišek and Steinová, 2013). Therefore, the researchers argue that the algorithm is challenged in terms of the search speed as it undertakes a blind search when selecting the optimal path in the network. Secondly, as Zingaro (2021) observes, the algorithm cannot work on negative weights or edges, unlike other variant algorithms such as the Bellman-Ford. Thirdly, the author also argues that the algorithm also introduces an additional constraint as there is a need to maintain the tracking of vertices visited (Zingaro, 2021).

The evaluation of the studies indicated that the Dijkstra algorithm is considered highly efficient in finding all optimal paths in a given network (Forišek and Steinová, 2013; Nalepa, 2020). This finding implies that the algorithm has been widely adopted in diverse real-world applications. For instance, in traffic information routing, the Dijkstra algorithm is adopted in tracking the source and destinations of the information (Venkat, 2014). A second application area is in routing computer network information. The link-state routing protocol relies on the Dijkstra algorithm to compute the shortest path to the network and populates the information in a routing table (Oliveira and Pardalos,

2014). In further explanation, Srikant and Ying (2014) offer that the Dijkstra algorithm is a link-state routing algorithm, whereby each node has knowledge on the states of all nodes in the same network, and as a result, the entire network topology and all link costs. Subsequently, based on the global link states, the node can compute the minimum cost paths from itself to all other network nodes (Oliveira and Pardalos, 2014). On the same note, Talukder, Garcia, and Jayateertha (2014) highlight the open shortest path first (OSPF) algorithm as a link-state algorithm that relies on the Dijkstra algorithm in internet routing.

In addition to real-world applications such as internet routing and traffic information systems tracking. For instance, Wang, Mao, and Eriksson (2019) proposed a three-dimensional Dijkstra optimization algorithm to optimize ship voyages, improving energy efficiency and safety. A three-dimensional weighted graph was generated in the study, and the Dijkstra algorithm was used for optimization. Findings reported showed that the algorithm was able to lead to optimal routing, which enabled ships to encounter less harsh environments at sea and, as a result, reduced fuel consumption by about 5%. In another study, Liu, S., Jiang, H., Chen, S., Ye, J., He, R. & Sun, Z. (2020) further integrated Dijkstra's algorithm in deep inverse reinforcement learning for route planning used by food delivery agents. In the study, the researchers argued that due to the high growth in food delivery in China, delivery men using e-bikes did not utilize system recommended routes in making deliveries due to the outdated or incomplete road network information in most areas. Therefore, to capture the delivery routes preferred by the delivery men, the Dijkstra algorithm was adopted to characterize them. Deep reinforcement learning was further adopted to recommend preferred routes. Findings showed that the Dijkstra algorithm and reinforcement learning adoption improved the F1-score distance by about 8% where road information was made available.

Mirahadi and McCabe argued that building evacuation strategies impacted the success of an emergency response plan in reaction to disastrous events in the building. However, shortcomings were observed with conventional building evacuation strategies seeking to identify the shortest paths when exiting from buildings in emergency cases. Mirahadi and McCabe (2021) further proposed EvacuSafe, a model for facilitating building evacuation based on the Dijkstra algorithm. By adopting a modified Dijkstra algorithm in the Active Dynamic Signage System (ADSS) model, the safety level of building

evacuation was enhanced significantly compared to the traditional plan for evacuation based on the shortest path. With the ADSS model, the building was monitored in real-time. In an unexpected scenario, changes in the evacuation strategy were communicated to building occupants, thereby facilitating their evacuation. In a different study, Rosita, Rosyida, and Rudiyanto (2019) also proposed implementing the Dijkstra algorithm and multi-criteria decision-making for optimal route distribution for other products.

Based on the evaluation of various studies in this section, the Dijkstra algorithm's functionality and features were identified and further examined. Core advantages and shortcomings were also emphasized and real-world applications in internet routing, traffic flow, building evacuation, and ship routing.

8.2.2 Breadth-First Search (BFS)

According to Miller and Ranum (2011), the breadth-first search algorithm is used to traverse or generate a tree in a breadth ward motion while utilizing a queue to remember the next vertex to start a search when the iteration cannot proceed. Hurbans (2020) further adds that the algorithm begins at a specific node (root) and proceeds to explore every node at the given depth before exploring the next depth of nodes. As such, the author reports that the algorithm visits all children of nodes at a particular depth before visiting the next depth of the child before finding a goal leaf node (Neapolitan, 2015). The author explains that the breadth-first algorithm is the most popular approach for traversing graphs. One starts at the selected node (source) and traverses the graph in a layerwise approach, thereby exploring neighbour nodes directly connected to the source node. After that, the traversal moves onto the next-level neighbour nodes. Subsequently, the review of the different studies indicates that two core rules guide the breadth-first algorithm

- i. Move horizontally and visit all nodes in the current layer. Mark all unvisited nodes as visited and insert them in a queue.
- ii. Move to the next layer. Where no adjacent node is identified, remove the first vertex from the queue.

To facilitate further understanding of the breadth-first search algorithm (BFS), the following example in figure 34 is detailed.

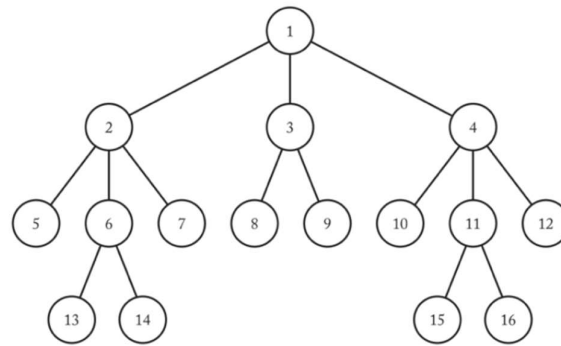


Figure 34. Illustration of BFS algorithm (Neapolitan, 2015)

In figure 34, the starting node for the algorithm is node 1, which initializes the algorithm. The algorithm starts at node 1 and marks it as visited. After that, it moves to the next layer, where it visits node 2 and adds it to the queue. The graph is traversed in a layerwise manner, whereby nodes in the horizontal layer are then visited. In this case, node 3 is visited and enqueued. The algorithm, after that, visits the next adjacent node 4 in the same horizontal layer. As this layer is left with no unvisited nodes, the algorithm dequeues node 2 and traverses its children nodes. The same procedure is repeated iteratively until all nodes in the graph are visited. The flowchart in figure 35 below illustrates the flow of the BFS algorithm.

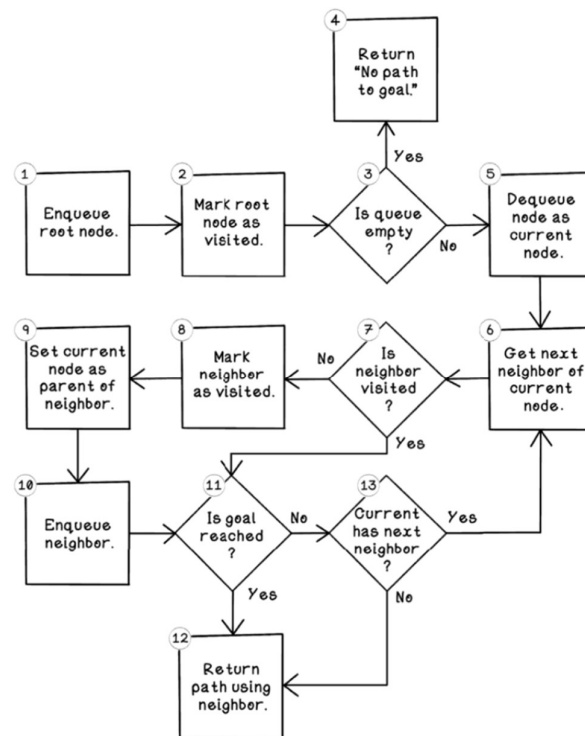


Figure 35. Flowchart of the BFS algorithm (Hurbans, 2020)

The flowchart above summarises the approach described earlier regarding the traversal of the graph. As such, the following stepwise flow is identified:

- i. Enqueue the root node using the first-in-first-out (FIFO) algorithm where objects are processed based on their accessed order.
- ii. Mark the root node as visited.
- iii. Is the queue empty? The algorithm can either return to “no path to the goal” or dequeue the node as the current one.
- iv. Dequeue node as the current node.
- v. Get the next neighbour of the current node.
- vi. Mark the neighbour as visited and set the current node as the parent of the neighbour.
- vii. Enqueue the neighbour node.
- viii. If the goal is yet to be reached, return the path to the neighbour.

The BFS algorithm is associated with several advantages. For instance, it can identify the shortest path between the start node and any other reachable node in the graph (Sedgewick and Wayne, 2016). The authors add that the algorithm is also simple to implement as it involves a small number of steps. Neapolitan (2015) adds that the algorithm is complete and optimal and does not suffer from any potential infinite loop problems. However, several disadvantages are also identified. For example, it is a blind search and delivers poor performance where the search space is large (Sedgewick and Wayne, 2016). Secondly, the algorithm consumes significant memory compared to alternatives such as the depth-first search (DFS).

Nevertheless, despite the few disadvantages of the BFS algorithm, diverse researchers have employed it in tackling different path-finding problems. For instance, Zhang F., Lin, H., Zhai, J., Cheng, J., Xiang, D., Li, J., Chai, Y. & Du, X. (2018) proposed an adaptive breadth-first search algorithm for integrated CPU and GPU (Graphic Processing Unit) architectures. The researchers argued that the BFS algorithm was also becoming more important as a representative algorithm for data analysis with the increased popularity in big data applications. In the research, BFS was employed to identify the traversal order and device for each performance level. Findings obtained showed that the algorithm led to the best energy efficiency and achieved higher performance over other integrated architectures. The analysis of the different studies (Wang, Li, and Fang, 2012; Zhang F., Lin, H., Zhai, J., Cheng, J., Xiang, D., Li, J., Chai, Y. & Du, X., 2018) indicated that the use of BFS algorithms led to high performance and predictive accuracy levels.

Further study by De Carvalho and Soma (2015) also showed that the breadth-first search could also be applied in the minimization of open stacks (MOSP) problems. Results obtained showed that the proposed heuristic BFS algorithm had higher robust behaviour in comparison to the best heuristic for MOSP despite the lower accuracy recorded. As such, the researchers argued that using the BFS algorithm was a cost-effective alternative useful in solving or attaining good upper bounds for tackling MOSP problems. In another study, Rakhee and Srinivas (2016) also developed an efficient routing protocol to facilitate the selection of optimal paths for continually monitoring vital patient signs in body area networks (BANs) in hospital environments. The constraining conditions identified included the high patient numbers and significant traffic that was generated that changed rapidly. In the research, Rakhee and Srinivas (2016) combined

the BFS with the ant colony algorithm, with the results showing that the approach led to enhanced results compared to traditional methods. The review of the studies emphasized that BFS algorithms could be employed in solving path-finding problems, especially in combination with other algorithms such as the ant colony algorithms.

8.2.3 Depth-First Search (DFS)

According to Riansanti, Ihsan, and Suhaimi (2018), the depth-first search algorithm traverses the graph from the root node. It explores the graph to the farthest possible distance along each branch (for a tree graph) before backtracking. Backtracking in this regard implies that when the algorithm moves forward, and no more nodes are left along the current paths that are yet to be traversed, the algorithm then moves backwards along the same path to find nodes that are yet to be traversed. The authors further add that due to its recursive nature, the DFS algorithm is implemented in terms of stacks that maintain the unvisited nodes. As a result, all nodes in the current path will be visited until traversal of all unvisited nodes is undertaken, and thereafter, the path is then selected (Li and Ueno, 2017).

In another study, Kozen (2012) reports that in implementing the DFS algorithm, each vertex of the graph is categorized either as visited or not visited. The authors also posit that the algorithm is implemented in a series of four steps as listed below:

- i) Put any one of the graph's vertices on top of a stack
- ii) Take the top item of the stack and add it to the visited list
- iii) Create a list of the adjacent nodes for the particular vertex. All nodes that are unvisited then placed at the top of the stack
- iv) Repeat steps (ii) and (iii) until the stack is empty.

Regarding its advantages, Miller and Ranum (2011) report that the DFS algorithm uses less memory and resources as it maintains only a list of nodes from the root node to the current node. The authors also add that the algorithm also takes less time to reach the goal node than other algorithms that adopt a similar approach, for instance, the BFS. However, on the contrary, Kozen (2012) argues that there is a likelihood of an infinite

loop occurring as the algorithm searches deep down the graph. Secondly, the author also argues that the algorithm does not guarantee to find the optimal solution as many states of the node keep re-occurring (Li and Ueno, 2017).

Various researchers have further undertaken diverse empirical studies that demonstrate the use of depth-first search algorithms in solving path-finding problems. For instance, Li and Ueno (2017) proposed an extended depth-first search algorithm to undertake optimal triangulation in Bayesian networks. In the study, the researchers extended the algorithm by adding a new dynamic clique maintenance algorithm which computed cliques that had a new edge and also introduced a new pruning rule known as pivot clique pruning. The generated results emerged that the extended DFS algorithm-generated optimal triangulation faster than the conventional algorithms.

8.2.4 Iterative Deepening Depth First Search (IDDFS)

According to Russell and Norvig (2016), the IDDFS algorithm combines the BFS and DFS. The authors postulate that the premise of the algorithm is that it eliminates the disadvantages present in both BFS and DFS by creating a hybrid algorithm with the unique advantages of both. In explanation, Kozen (2012) reports that two main limitations challenge the DFS algorithm; the inability to identify the shortest path to a node and the likelihood of the DFS algorithm reaching the node very late even if it is close to the root node. The author also argues that the shortcoming of the BFS algorithm is that it is memory-intensive despite its fast exploration speed. Subsequently, by developing the IDDFS algorithm, the fast exploration speed of the BFS is coupled with the space efficiency of the DFS algorithm (Russell and Norvig, 2016).

In regard to how the algorithm works, Needham and Hodler (2019) add that the IDDFS algorithm first undertakes a limited depth-first search up to a particular depth which is limited, and after that, the depth limit is incremented through iterating the procedure unless the goal node is identified, or the entire tree is traversed. From an analytical perspective, the IDDFS makes different calls for the DFS algorithm at different depths beginning from the initial value. In this regard, the algorithm implements the DFS algorithm in a BFS fashion. To further understand how the algorithm works, figure 36 below illustrates an example of a graph with four different depth levels.

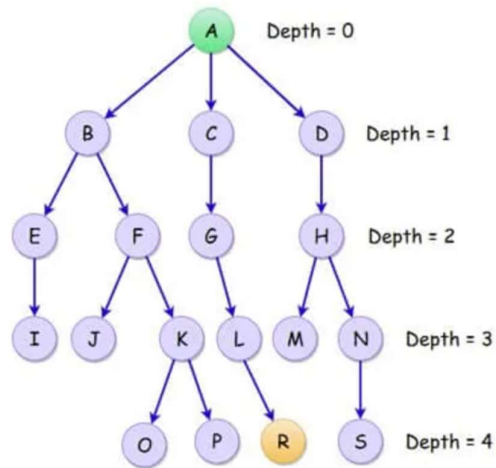


Figure 36. IDDFS graph with 4 depths (Pedamkar, 2020)

As illustrated in the graph, each of the depths is then traversed via the DFS algorithm leading to the solution space outlined in figure 37 below.

The tree can be visited as: A B E F C G D H

$DEPTH = \{0, 1, 2, 3, 4\}$

DEPTH LIMITS

IDDFS

0

A

1

A B C D

2

A B E F C G D H

3

A B E I F J K C G L D H M N

4

A B E I F J K O P C G L R D H M N S

Figure 37. IDDFS graph traversal (Pedamkar, 2020)

Diverse advantages and disadvantages of the algorithm have also been identified, and for instance, it is argued to be complete as it can identify a solution provided it exists in the graph (Pedamkar, 2020). A second advantage is that the algorithm is efficient where solutions are identified at a depth of the tree (Konar, 2018). On the same note, the researcher adds that the algorithm has quick responsiveness as it generates early results that are refined multiple times through different iterations. Pedamkar (2020) further adds that the algorithm is also highly efficient in in-game tree search as it improves depth definition, heuristics, and scores of search nodes. Despite its efficiency, the author argues that it is considered wasteful due to the time and wasted calculations which occur at the same depth (Pedamkar, 2020). In further explanation, Russell and Norvig (2016) report that with the IDDFS algorithm, all work done in the previous phase or

depth is repeated. A second disadvantage also arises from the fact that the failure of the BFS algorithm leads to the eventual failure of the IDDFS (Konar, 2018).

8.2.5 Bi-directional Search

According to Korf (2003), the bi-directional search algorithm is a brute-force algorithm that requires an explicit end goal state instead of creating a test for a given goal condition. The author reports that with the standard graph searches such as the breadth-first and depth-first, the search usually begins from one direction: the root or source node for the BFS algorithm or the end node for the DFS variant. However, with the bi-directional algorithm, the search begins from both directions simultaneously (Russell and Norvig, 2016). The author further adds that one algorithm begins the search from the initial node while the second begins from the goal vertex. As a result, the two algorithms intersect somewhere at the middle of the graph, with the path that traverses from the initial node to the final node being considered the shortest path. The Figure below summarises the bidirectional search algorithm.

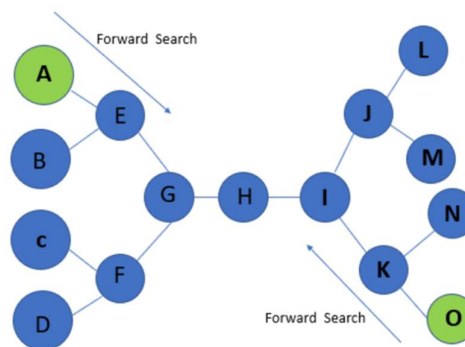


Figure 38. Bidirectional search algorithm (Russell and Norvig, 2016)

In another study, Qi, Shen, and Dou (2013) report that the main objective of the algorithm is to minimize the time required to search for the shortest path in a graph. The time for the search (exploration time) is reduced to a fraction of the time needed by other search algorithms as the two intersect at the middle of the graph. Needham and Hodler (2019) further compare the bidirectional search to the single search algorithm such as BFS/DFS regarding the searching complexity. According to the researchers, single search algorithms (BFS/DFS) have a search complexity of $O(bd)$ for a branching factor b and where the distance from the goal to source vertex is d (Needham

and Hodler, 2019). However, with the bidirectional search, two search operations are executed simultaneously, leading to a search complexity of $O(bd/2)$ for each algorithm and total complexity of $O(bd/2 + bd/2)$, which is less than the overall complexity of single search algorithms $O(bd)$ (Needham and Hodler, 2019). Therefore, two advantages of the bidirectional search include the fast exploration time and reduced memory and resource utilization (Needham and Hodler, 2019). The authors also add that the algorithm is optimal and complete as an optimal solution is always identified. The argument advanced is that the algorithm has a reduced exploration time because it undertakes simultaneous searches.

However, despite such advantages, Russell and Norvig (2016) argue that the algorithm is challenged by the fact that a user needs to be aware of the goal state to be attained in the graph search, thereby decreasing its use cases significantly. An additional disadvantage regards the complexity in implementing the code and instructions for the algorithm, as care needs to be taken when implementing the node and search (Qi, Shen, and Dou, 2013). The authors also add that it is not possible to perform a backward search through all states in some instances. Similarly, there is also a need to ensure the algorithm is robust enough to understand the intersection and where the search should come to an end to avoid the possibility of an infinite loop (Needham and Hodler, 2019). An illustration of the working of the algorithm is detailed below.

Saux and Claramunt (2014), who implemented a bidirectional dynamic routing algorithm for maritime routing based on hexagonal meshes and iterative deepening A* (IDA*) that facilitated data accessibility, showed that the bidirectional search generated better performance in terms of average execution time. Results obtained revealed a reduction in computation time and distance that was traversed in the graph. In a different study, Kumar (2019) also implemented bidirectional algorithms for the A*, breadth-first, and best-first search alternatives and compared performance to the conventional algorithms. Findings reported showed that the bidirectional versions could generate better results compared to the traditional algorithms.

8.2.6 Uniform Cost Search

According to Chandra (2014), the uniform cost search algorithm is a brute-force algorithm that is used to find solutions in a directed weighted graph with a minimum

cumulative cost. In explanation, the author reports that with the uniform cost search algorithm, the goal is to identify the path to the goal node which has the lowest cumulative cost as the nodes are each associated with a cost. Rothlauf (2013) further adds that with the algorithm, the lowest cumulative cost is identified for a weighted graph as the nodes are expanded based on traversal costs from the root node. To understand how the algorithm works, the example graph in the figure below is detailed.

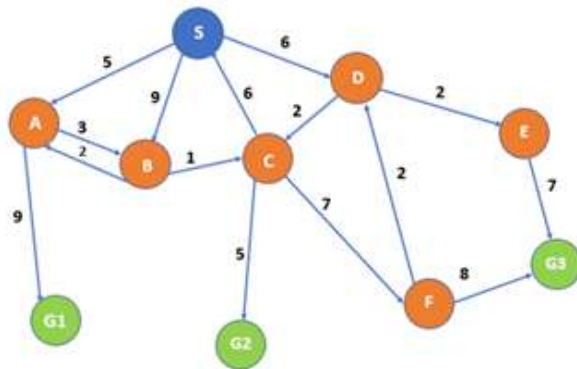


Figure 39. Uniform cost search weighted graph (Jagga, 2020)

As detailed in the graph, the aim is to identify the shortest path with the lowest cumulative cost from the start node (S) to each of the end nodes (G1), (G2), (G3).

One advantage of the uniform cost search algorithm is that it helps identify the lowest cumulative cost path in a weighted graph from the source to the root node (Rothlauf, 2013). Jagga (2020) adds that the solutions derived from the algorithm are also complete and optimal for each state. Other disadvantages identified include high storage consumption and the likelihood of the algorithm being stuck in an infinite loop as it derives possible paths from the root to destination nodes (Chandra, 2014). On the same note, Jagga (2020) reports that the algorithm has to maintain a sorted open list in the priority queue.

8.2.7 Breadth-First Search with Heuristic Function

The breadth-first search with heuristic function has been described as an algorithm that expands a graph's nodes in breadth-first order but employs a heuristic function when pruning the search space (Zhou and Hansen, 2006). The authors note that no node is inserted into the available list with the algorithm where the f-cost is higher than the

upper bound of the optimal solution's costs as the nodes are not on an optimal path. Over the years, breadth-first search with heuristic function algorithms has also been utilized in solving diverse path-finding problems. For instance, Wang, Li, and Fang (2012) utilized the heuristic breadth-first search algorithm to undertake robust tumor classification. The authors argued that finding tumor-related genes with the highest accuracy was vital to serving as tumor biomarkers.

The BFS algorithm with heuristic search was employed to select the best gene subsets, where findings obtained showed that the algorithm led to better generalization performance and identified many tumor-related genes (Wang, Li, and Fang, 2012). The indication from the study was that adoption of a heuristic breadth-first search algorithm led to higher prediction accuracy compared to other methods and utilized fewer genes. In another study, Zhou and Hansen (2006) further demonstrated the effectiveness of breadth-first heuristic search algorithms in tackling complex graph-search problems. In the study, the researchers revealed that breadth-first search was better than best-first strategy as it prevented node generation and was much easier to implement. Two variants of the breadth-first heuristic search algorithms were discussed; a memory-efficient breadth-first branch-and-bound search and a breadth-first iterative deepening A* algorithm which was based on the breadth-first branch-and-bound search. Findings showed that the breadth-first algorithms outperformed other systematic search algorithms when tackling complex graph-search problems based on the computational results.

8.2.8 A* algorithm

The A* (A-star) algorithm is a highly ubiquitous search algorithm highly utilized in pathfinding research problems (Foad, Ghifari, A., Kusuma, M.B., Hanafiah, N. & Gunawan, E. 2021). The researchers posit that the popularity of the algorithm arises from its simplicity, modularity, and efficiency. As a result, the algorithm has been widely utilized in broad and strategy-based games owing to its stability, efficiency, and accuracy. In another study, Iordan (2018) reports that the A* algorithm is a sophisticated breadth-first search (BFS) algorithm whose main aim is to search for shorter than longer paths between the start and end nodes. Ping and Shuai (2013) further describe the algorithm as being complete and optimal. In explanation, the authors posit that

completeness describes the capability of the algorithm to identify the least cost from the source node to the destination node. In contrast, the optimal nature of the algorithm refers to the notion that the algorithm can identify all available paths from the source to the destination nodes (Ping and Shuai, 2013).

In another study, Roy (2019) adds that the optimality of the A* algorithm implies that it is assured of finding the solution that is best while its completeness further suggests that provided there exists a solution for a problem, the A* algorithm is always guaranteed to find it. The author further highlights that the algorithm's values are calculated using the function below;

$$f(n) = g(n) + h(n) \quad (4)$$

(Roy, 2019)

where:

$f(n)$ is the lowest cost of the neighbourhood node n

$g(n)$ is the exact cost of the path from the start node to a node n

$h(n)$ is the heuristic estimated cost from a node n to the goal node

In further explanation, Foad, Ghifari, A., Kusuma, M.B., Hanafiah, N. & Gunawan, E. (2021) argue that whenever the A* algorithm traverses a node, it calculates the cost, $f(n)$, of travel to the nodes that neighbour it and, after that, enters the node which has the lowest $f(n)$ value. Akash (2020) adds that $h(n)$, which is the heuristic cost, is not a cost in reality but instead is a guess cost used to find the cost that could lead to most optimal paths between source and source-destination nodes.

By adopting an approach to identify the least-cost path to the end node, Ghaffari (2014) argues that the A* algorithm utilizes a best-first search approach to identify the optimal path from the initial to destination nodes. Various advantages of the algorithm have also been identified. For instance, since the algorithm relies on path distance as a criterion, it can be adopted in diverse applications (Duchoň F., Babinec, A., Kajan, M., Beňo, P., Florek, M., Fico, T. & Jurišica, L., 2014). A second advantage arises from its ease of comprehension, given that it solves the path-finding problem in a stepwise manner and relies on a definite procedure (Denden, Essalmi, and Tlili, 2016). In further explanation,

the authors report that the A* algorithm is also easy to convert to an actual program as it breaks down problems into smaller sub-parts or components. In another study, Zidane and Ibrahim (2017) argue that the algorithm is associated with diverse disadvantages, including its time-consuming nature and difficulty in putting big tasks into algorithms.

Nevertheless, despite such disadvantages, diverse researchers have employed the A* algorithm in different applications, whereby path-finding problems are solved. For instance, Ghaffari (2014) utilized the A* algorithm to develop a routing protocol that was energy-efficient for wireless sensor networks. In the study, the researcher argued that due to the need to deliver critical information in a multi-hop and energy-efficient manner in wireless sensor networks, there was a further need to extend the lifetime of the networks using energy-efficient routing protocols. To tackle the problem, Ghaffari (2014) proposed an energy-efficient protocol, which improved the network's lifetime by forwarding packets through the shortest path that delivered optimal results. In another study, Yu Y., Wang, J., Xue, X. & Zou, N. (2019) utilized the A* algorithm for developing a route navigation system based on visible light communication and used in an underground garage. In the study, the researchers aimed to tackle the difficulty in the parking lot, security of communications, and low efficiency of the garage by using visible light communication. Findings showed that by using the A* algorithm for route navigation, the needs of vehicle navigation were adequately met, and signal transmission of a range of 4m was attained.

Bagheri S.M., Taghaddos, H., Mousaei, A., Shahnavaz, F. & Hermann, U. (2021) also employed the A* algorithm in crane optimization locations and modular construction configuration. The researchers argued that while the utilization of on-site heavy mobile cranes in construction was inevitable, their high rental costs made it difficult for companies to utilize them. The A* algorithm was adopted to develop a framework that facilitated optimization of the pre-determined lifting sequence to tackle the problem. Results obtained showed that the use of the algorithm was advantageous as it led to lower costs compared to previously used algorithms for lift planning. In another study, Cai and Ji (2018) further employed the A* algorithm in the adaptive routing of network-on chips. The researchers employed the algorithm and coupled it with a deadlock-free adaptive routing algorithm to tackle network congestion. The proposed solution utilized routing table information to select output channels that were not congested for

forwarding packets using the A* algorithm. Results obtained showed that the use of the algorithm led to improved throughput and average latency.

The analysis of the various reviewed applications (Ghaffari, 2014; Cai and Ji, 2018; Yu Y., Wang, J., Xue, X. & Zou, N., 2019; Bagheri S.M., Taghaddos, H., Mousaei, A., Shahnavaz, F. & Hermann, U., 2021) reveals that A* has been adopted in diverse applications and settings. The applications range from wireless sensor networks to route navigation and optimization of crane configurations. Based on the observed diversity, the findings emphasize that the A* algorithm is highly versatile and can be applied in different applications. Nevertheless, other researchers have further modified the A* algorithm and employed it in diverse applications, as detailed in the next section.

8.2.9 Genetic algorithms

At the foundational level, genetic algorithms (GA) are described as optimization algorithms that facilitate the search of potential solutions in a given space (Nagib and Gharieb, 2004). The authors further add that such algorithms are meta-heuristic in nature and are based on evolution models. As a result, genetic algorithms can solve complex problems within a short period of time. In reiteration, Lamini, Benhlima, and Elbekri (2018) highlight that GAs were developed in 1960 by John Holland and inspired by Darwin's natural evolutionary principles. In explanation, the authors report that the genetic algorithms are associated with several evolutionary processes, including initialization, fitness, natural selection, cross over and mutation.

Further examination by Jebari (2013) reveals that GAs generate a random initial population representing all likely solutions to the particular optimized problem at the initialization phase. Xin J., Zhong, J., Yang, F., Cui, Y. & Sheng, J. (2019) further explain that the GA maintains a population of candidate solutions encoded as a binary string referred to as a chromosome. The authors emphasize that the chromosomes are generated at random. An adaptation function assesses each possible solution (chromosome) to ascertain the quality of the potential solution.

The second phase involves selection, whereby generic operators are further applied to the solutions to create new progeny. Xin J., Zhong, J., Yang, F., Cui, Y. & Sheng, J. (2019) add that selecting the parents will be subjected to the actual reproduction based on

their adaptation values. The third phase is the exact reproduction of new progeny from the selected parents using mutation or crossover operators (Nagib and Gharieb, 2004). The authors also add that bits are randomly exchanged between the two strings in the chromosomes of the intermediate population in the crossover operation. In contrast, the mutation process randomly alters some of the bits in the chromosomes. In a reiteration of the findings, Lamini, Benhlime, and Elbekri (2018) further report that crossover is later applied to the new product offspring by recombining data from the two parents selected in the reproduction step. The authors also highlight that mutation is adopted to ensure population diversity by changing some population members' genetic structures based on a mutation rate.

Nelson, Barlow, and Doitsidis (2009) also observe that genetic algorithms repeatedly undertake the evolutionary cycle until the stopping criteria are satisfied. In explanation, the authors report that the stopping criteria may be satisfied where the number of generations is fixed, where the population does not evolve rapidly enough, or where the pre-determined iteration limit is attained.

The GA algorithm has three inputs; population size, crossover rate, and mutation rate. However, only one output is generated, the best chromosome or most optimal solution.

Regarding the advantages of GA algorithms, Achour and Chaalal (2011) report that the algorithms can solve complex problems within a short time period. An explanation for their efficiency highlights that the algorithms can explore the search space in parallel and thereby do not require the function being optimized to be differentiable or have smooth properties. However, despite such advantages, Achour and Chaalal (2011) argue that GAs are still limited as they face the exploration-exploitation dilemma where a significant problem is the capability to select the initial population.

Nevertheless, diverse researchers have, over the years, adopted genetic algorithms in solving path planning problems in different settings. In mobile robot path planning, in particular, researchers argue that GAs are efficient. They enable the robotic agent to identify an optimal path that is also collision-free from the source to the final target, either single or multiple (Sarkar, Barman, and Chowdhury, 2020). To begin with, Lamini, Benhlime, and Elbekri (2018) employed GAs in robotic path planning where an improved crossover operator was utilized. Results obtained showed that the proposed crossover

operator helped avoid premature convergence and offered feasible paths with more optimal fitness values than its parents, thereby converging more quickly. Simulation results also revealed that the use of GAs led to better performance and optimal solutions in different environments. In a second study, Giardini and Kalmár-Nagy (2011) employed the genetic algorithms coupled with heuristic local search in developing a combinatorial planner for autonomous systems. In the study, the researchers solved a subtour problem, a variant of the classical traveling salesman. Results obtained showed that combining genetic operators, which had a double crossover with a mutation, with local boosting technique provided an efficient solver for combinatorial problems that were otherwise considered too difficult.

Further study by Ma J., Liu, Y., Zang, S. & Wang, L. (2020) also revealed the combination of genetic algorithms and continuous Bezier optimization to facilitate path planning for mobile robots. Findings obtained showed that combining the two approaches led to a more practical approach in generating a shorter, smoother, and safer path for the robots in comparison to conventional techniques. In particular, the researchers employed the genetic operations to obtain control points for Bezier curves while shorter paths were selected from the optimization criterion, which determined the Bezier curve's length. Ma J., Liu, Y., Zang, S. & Wang, L. (2020) also revealed that the solution also enhanced the smoothing of the robot's path by adding safety distance to the fitness function to allow it to be updated dynamically based on the distance between the path and obstacles to ensure safe and efficient robotic movement. In a different study, Tu and Yang (2003) had also proposed a genetic algorithm to facilitate path planning for mobile robots where the chromosome had a variable length. Findings reported showed that the robot was capable of finding optimal paths for mobile robots that were collision-free. The simulation results underscored the effectiveness of genetic algorithms in mobile robot path planning. Hu and Yang (2004) also employed knowledge-based genetic algorithms to facilitate path planning for mobile robots. Results obtained showed that the algorithms could find an optimal or near-optimal robot path in dynamic and static environments. In the study, the researchers utilized knowledge-based genetic algorithms for path planning as opposed to standard GAs. In particular, domain knowledge was incorporated in the specialized operators, leading to a combination of local search techniques. A similar study had also been conducted by Sarkar, Barman, and

Chowdhury (2020). They reported that domain-knowledge-based genetic algorithms helped address path planning problems that had one or multiple targets. Based on the results obtained, the researchers argued that reliance on domain knowledge for the genetic algorithms led to better performance. Only one target was considered instead of methods that relied on evolutionary algorithms.

The examination of the various reviewed empirical studies indicates that genetic algorithms have been widely adopted in path planning applications for mobile robots. Further analysis also showed that researchers enhanced the genetic algorithms by improving individual components in the different simulations. For instance, Lamini, Benhlima, and Elbekri (2018) proposed an enhanced crossover operator, while Hu and Yang (2004) used knowledge-based genetic algorithms. Giardini and Kalmár-Nagy (2011) further combined genetic algorithms with local heuristic search. As a result, the genetic algorithms could display better performance in path planning compared to conventional or traditional approaches that relied only on GAs.

8.2.10 Ant Colony algorithms

The final category of the path planning algorithms is the ant-colony algorithms (ACA) based on the ant colony theory (Ye, Ma and Fan, 2005). According to the researchers, the ant colony theory explains the behaviour of real ants while searching for food, whereby they can communicate with one another through the release of pheromones or aromatic essence, which is left on the trail or on paths that the ants travel in search of food. Zhishui (2011) adds that the ants release the pheromones as a way to cooperate with one another in finding the shortest path to the food source. However, in the absence of the pheromones, the author argues that the ants have to walk for longer paths, thereby undertaking more random walks.

Dai X., Long, S., Zhang, Z. & Gong, D. (2019) further argue that upon finding pheromones, ants follow them, thereby reinforcing the trail. However, the author emphasizes that the ants must make several decisions before selecting a given trail with pheromone. For instance, Dai X., Long, S., Zhang, Z. & Gong, D. (2019) report that the ants may be forced to select several paths with pheromone essence. In this regard, the author reveals that ants select the path with the highest concentration of pheromones due to higher probability. Secondly, as Zhishui (2011) reported, the ants are also limited by the

evaporation of the pheromones on the paths travelled, thereby requiring the ants to secrete the pheromones constantly. As a result, the ants also have to decide which paths have a higher pheromone concentration before selecting the path to the food source. A dilemma thus arises in terms of selecting a longer or shorter path to the food source, especially where all paths are associated with pheromones. However, as Wen and Cai (2006) argue, the result is that, over time, more ants will be concentrated in the shorter paths to the food source than the longer paths.

Porting this insight to path planning, ant colony algorithms can be developed for artificial ants, describing the mutual collaboration between different individuals to achieve a particular objective (Wang Y., Chen, J., Ning, W., Yu, H., Lin, S., Wang, Z., Pang, G. & Chen, C. 2020). On the same note, the authors report that the artificial ants would be focused on identifying the optimal and shortest path between the source and destination targets while also relying on pheromones or an artificial feedback system to determine the paths they will traverse. In a reiteration of the argument, Zhishui (2011) report that ant colony algorithms are associated with four core aspects; i) a local strategy that guides the movement of agents in the search space, ii) ant's internal state, which maintains information about the ant's past, iii) the pheromone track which contains helpful information in decision-making when selecting different paths, iv) the ant's decision table which outlines probabilities and also guides decision-making process (Zhishui, 2011).

In another study, Zhangqi, Xiaoguang, and Qingyao (2011) summarize the ant colony algorithm in a series of steps as follows:

- i. Determine the basic parameters of the ACO algorithm, information inspiration factor, pheromone intensity, evaporation coefficient, hope inspiration factor.
- ii. Select an ant, put it at the map's starting point, and highlight the end node to determine the path to be travelled.
- iii. After each ant travels to the end node, update the pheromone partially and compare it with the current optimal path.

iv. If the path obtained is shorter than the optimal path, replace the optimal path with the current path.

v. After ants traverse the path, update the pheromones globally and iterate the process.

With the fundamental understanding of how ant colony algorithms work, examining their diverse applications is also essential, particularly in path-finding problems. To begin with, Korb (2009) employed the ant colony algorithms for optimizing the structure and ligand-based drug design. In the research, artificial ants were utilized in finding the optimal minimum energy used in the conformation of the ligand in the binding site. The researchers further used the artificial ants to facilitate the search towards low energy confirmations. Results showed that the algorithms displayed effective results about the optimization of energy in ligand-based drug design. In a second study, Kazharov and Kureichik (2010) further employed the ant colony algorithms for solving transportation problems by simulating the ant colony's behaviour after making various modifications. The analysis of the two studies underscored the impact of mimicking ant behaviour in tackling optimization problems in diverse settings such as drug design and transportation problems.

In another study, Qian and Zhong (2019) employed ant colony algorithms to facilitate the planning of optimal individualized tourism routes. The researchers gathered latitude and longitude values of each city and actual inter-city train and air tickets involving optimizing traveling to 34 different cities in China. Ant colony algorithms employed a heuristic search and identified optimal travel routes for travel problems. Findings demonstrated that the algorithms were able to generate positive results. The use of ant colony algorithms in the optimization of travel routes has also been examined in other studies such as Hulianytskyi and Pavlenko (2019). They utilized the algorithms to identify the optimal traveller routes in airline networks by considering the constructed route and user conditions with the time-dependent cost of connections. In the research, the ant colony algorithms were considered essential in solving time-dependent problems represented in extended flight graphs. The generated results emerged that the use of the algorithms led to a higher quality of constructed routes between the various diverse regions.

The ant colony algorithms have also been deployed in solving conventional problems such as the traveling salesman under stationary environments (Mavrovouniotis and Yang, 2010). However, the researchers considered a dynamic travelling salesman problem in the study, whereby cities were replaced by new ones when executing the algorithm. To alleviate convergence challenges in the algorithms, the ant colony optimization (ACO) algorithms were enhanced using a memetic ACO algorithm which carried out a localized search. Findings reported showed that the modified ACO performed better than the traditional and unmodified ACO algorithms. Zhang and Zhang (2018) also utilized ant colony optimization algorithms in finding the shortest path in dynamic traffic networks. In the study, the researchers experimented on an existing traffic network, whereby results obtained showed that the algorithm could find the optimal path in a dynamic traffic network.

The analysis of the different empirical studies on ant colony optimization algorithms revealed that the algorithms helped identify optimal routes in different path-finding applications such as road networks, airlines, and even the conventional traveling salesman problem.

8.3 Comparison of Pathfinding / Wayfinding Algorithms

Finding the right process for route planning is not a straightforward answer. There has been a number of significant strides in the field of graph theory and pathfinding algorithms, so this results in a wide variety of choices depending on the user's needs. The summary of the comparison contained in chapter 7.3 is contained in Appendix V, for easier reference. Dijkstra algorithm efficiently locates the direct and shortest possible route between nodes of a wide range of problems. However, it costs time and resources drain as it performs an unguided search. Moreover, its usage is limited to only cyclic graphs because it cannot analyse the graphs in which nodes cannot be traversed back due to the involvement of opposing edges.

A* algorithm prefers the ideal and less extended path between endpoints while consuming extensive storage. The smooth execution of this algorithm depends on the limited branching factor, stagnant cost, and precision of the functions involved.

In contrast to the A* algorithm, Iterative Deepening works in a bit specific and flexible way by evaluating the flash route between initial and defining nodes. Although it occupies less space, it takes way more time to process the graph as it visits the same nodes multiple times due to its inability to keep track of exploration.

To overcome this problem, a more advanced and guided search algorithm, Dynamic A* algorithm, has been designed, which works in a planned and ordered way using heuristic function and gives the most direct and updated paths by analysing live fluctuations in the graph with high accuracy. It is valid to the problems having mediocre complexity. Unlike the A* algorithm, it fails if there is a limitation to the branching factor and constant costs.

Weighted A* algorithm has a brilliant path searching mechanism to evaluate the minimum distance between nodes. It starts its search on states which has low heuristic values while not ignoring the higher values. It has a significant edge over the conventional A* algorithm, but it struggles with intensive and complex problems.

Lifelong Planning A* algorithm (LPA*) is an advanced form of the A* algorithm, which refreshes the distance values from the previous search. Then adjusts to changes in the graph without analysing and re-evaluating the entire chart. The weighted A* algorithm also gives better performance but fails to reach an efficient result in complex situations.

Genetic Algorithms (GAs) are the evolved form of conventional heuristic search algorithms, which are adaptive, and their foundation lay on the concepts of genetics and natural selection. These intelligent algorithms use historical data, and their search is directly focused on the better-performing regions in the space. Optimization and search problems can be efficiently solved using these intelligent algorithms. They can analyse the multiple space regions simultaneously, reducing the processing time while solving intensive problems.

The ant colony algorithm is utilized for discovering optimal ways that depend on the conduct of subterranean insects looking for food. Initially, the ants meander haphazardly. At the point when they discover a wellspring of food, it strolls back to the province, leaving "markers" (pheromones) that show where the food is. In the initial step of taking care of an issue, every ant produces an answer. Next, the paths found by

the ants are analysed. Finally, path values are refreshed. There are numerous improvement issues where it can utilize ACO for tracking down the ideal arrangement. Time taken by this algorithm for convergence is a bit hard to estimate.

Breadth-First Search (BFS) starts in a structured and sequential manner layer by layer. While moving forward, it can detect the connected nodes and ultimately the idea path when it has reached the end. It does not have any hectic problem revisiting the nodes and is easy to use as it involves a few simple steps. It is an unguided search algorithm and struggles with high-end problems if the space is ample and consumes a more significant portion of the storage.

Depth-First Search (DFS) utilizes the backtracking concept. It starts with the initial node and drives all its nearby nodes into a stack. Finally, it pops a node from the stack, which acts as the stimulus for the next node to explore paths and drive all its connecting nodes into a stack, and this process goes on. Owing to its ability to maintain only lists, it does not require much memory space and time to finally develop a path that gives this a distinction over its predecessor, Breadth-First Search.

Iterative Deepening Depth First Search (IDDFS) performs searches in steps. First, it analyses the graphs to a certain depth and then again performs another depth search to a greater extent till the whole tree has been analysed and traversed, and the algorithm has reached its goal. This algorithm can be used in more complex problems where the solutions are expected to be in greater depth. Moreover, quick results are possible as this algorithm completes the search in steps and then refines the search to come up with the most efficient results. However, it can also consume more time if the solution is obtained after multiple iterations and perform calculations on the previous depth again.

The bi-directional search uses two algorithms simultaneously and starts searching from the initial node and vertex node. Consequently, both algorithms meet somewhere in point of time and suggest the most ideal and shortest path. The only smartness which bi-directional search has to show is that it must understand the intersection point to avoid recalculating and reanalysing the whole web of nodes. If heuristics are not involved, Uniform Cost Search is the best option among all other algorithms. Optimal cost can easily be solved for a general graph using UCS. In contrast to the Depth First

Search, Uniform Cost Search prioritizes the least cumulative cost in the weighted graphs and provides complete solutions to the search problem at the cost of ample memory space. Breadth-First Search with Heuristic Function uses heuristic functions and expansion of nodes in the breadth-first order to speed up the search and achieve a complete result. However, it needs ample storage space to store nodes and even fails in extensive search problems.

9 Automatic Route Calculation

Automatic Route Calculation, due to its complexity, will be divided into two parts, consisting of the Automatic Route Planning and the Automatic Route Safety Check Parts. Automatic Route Planning is the first part of each voyage and starts after defining the departure and Arrival Port and plotting a route based on the ship's maneuvering characteristics and expected weather forecast. The second part is the Automatic Route Safety Check, which in essence is the validation of the voyage after counterchecking based on the set safety parameters in place, ensuring a smooth voyage.

9.1 Automatic Route Planning

Considering the detailed explanation in Chapter 7 and more specific Chapter 7.1.3 about Pathfinding Algorithms, this Thesis will use as an example for automatic route planning the A* Algorithm, because the popularity of the algorithm arises from its simplicity, modularity, and efficiency adds that the optimality of the A* algorithm implies that it is assured of finding the best solution. At the same time, its completeness further suggests that provided there exists a solution for a problem. The A* algorithm is always guaranteed to find a solution. Its main aim is to search for shorter as opposed to longer paths between the start and end nodes and optimally efficient as no other optimal algorithm is guaranteed to expand fewer nodes than A*. However, it comes with certain disadvantages, such as being memory intensive, as it gets exhausted once nodes are stored in memory, so programming languages such as C or Rust can be used, which focus more on memory efficiency and fast processing rather than using Python, JavaScript or other inefficient but straightforward languages for such an operation. An Illustration of the practical application of A* Search Algorithm is shown in Figures 40 and 41 below.

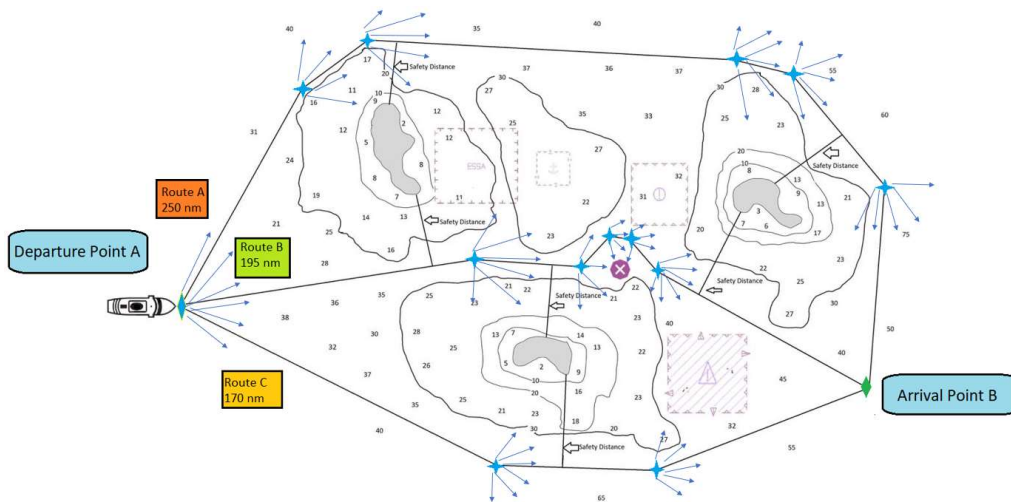


Figure 40 A* Search Algorithm example of finding the optimum route considering various obstacles and safe distance from them (Author, 2021)

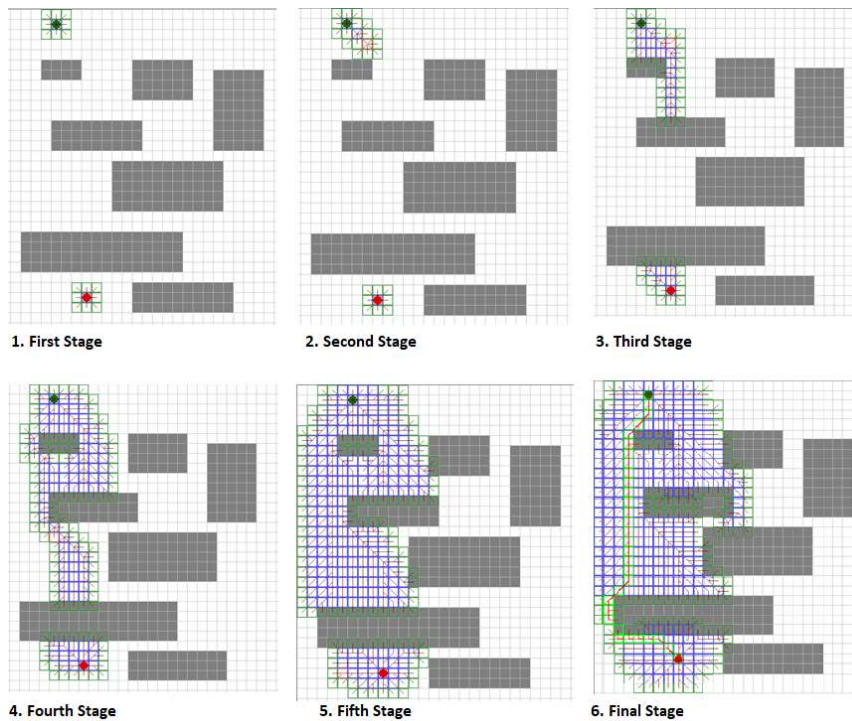


Figure 41 A* search algorithm generic example of finding the route in a cell pattern with obstacles (Author, 2021)

Because this Automatic Route Planning and Calculation software that is discussed in this Thesis, is assumed for standalone use on an Autonomous Vessel without assistance from External Shore Assistance, it must fulfil the following requirements by being able to process any route required in the fastest possible way, be memory efficient, especially

related to Random Access Memory (RAM), because it can cause the system to crash and rebooting it in the middle of the ocean is not a choice, run on a Operating System (OS), which is resource efficient such as Linux, communicate with ship's Sensors and via an Application Programming Interface, with Weather Routeing Software or collect the Raw Weather Data, ensure Redundancy by having simultaneously running another duplicate of the system in standby, so when the primary unit is detected to stop functioning it will immediately start and run in it's place and energy Redundancy by having a sufficient number of Uninterruptible Power Supplies (UPS) sufficient to cover the energy requirements for a three week voyage.

For the initial route, the attributes that will only be taken into account will be object attributed as land and depth contours deeper than the safety contour, considering UKC, squat, and vessel's planned speed. Once the initial route is established between the first and last waypoint, then the software will validate the route and, if needed, will correct it after recognizing the attributes according to the parameters listed in chapter 8.2.

9.2 Automatic Route Safety Check

Route Checking or Route Safety Check currently implemented in ECDIS is done by entering first the ship's characteristics, the Safety Contour, and its voyage waypoints. Based on these, the ECDIS software performs a complete assessment of the voyage and shows every single alarm that is being shown. The Safety Check is made according to IHO S-58 Technical Standard, Section 4.10.2.1 of the IEC 61174, ed. 4 Technical Standard and Circular MSC.1/Circ.1503 IMO Guidelines for ECDIS Good Practice and the Safety Contour. An example of automatic route safety check in ECDIS is shown in the figure below

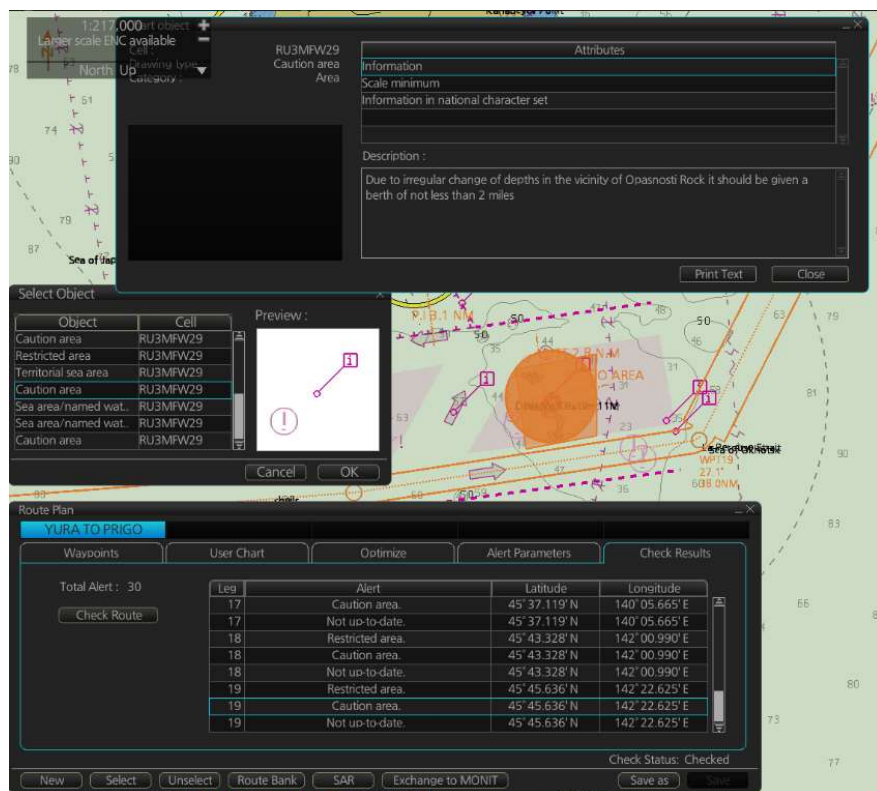


Figure 42 Example of a Route Safety Check by ECDIS using S-57 Attributes (ECDIS screenshot taken by author onboard ship, 2013)

It is a crucial part of the Automatic Route Calculation because it is the distinctive feature that sets apart this route calculation method from other ones. It doesn't blindly look for the shortest route or, in general terms. Still, it provides a more custom and accurate result which depends on the accuracy of the input values and the consistency and accuracy of the dynamic information related to passage planning. The Safety Contour is recommended to be calculated (IHO, 2012) as $\text{Vessel Safety Draft} = \text{Vessel Draft} + \text{Dynamic Squat} + \text{Safety Margin}$, which depending on the shipowner's Navigation Policy, can be altered accordingly so it can more accurately depict the safety contour. It should be noted that if the safety contour in use becomes unavailable due to a change in source data, the safety contour should default to the next deeper contour. The shallow depth, safety depth, and deep-water settings are entered only for the easier visual illustration of the end user and assist the navigator in identifying dangerous shallow and safe navigable areas, so this instance will not be mentioned as we refer to automatic route checking only by software. A practical illustration of how the Safety Contour is shown compared to the actual ship's draft is illustrated below by NOAA.



Figure 43 Example of how the Safety Contour is used in an ENC (NOAA, 2019)

The safety contour can be more easily identified by the software as a no-go area and avoid it rather than using spot soundings, which can prove far more challenging to identify and avoid.

The safety check analysis will be distinguished in the attributes check part and the wayfinding part. The optimum navigable route will be calculated with A* search algorithm by setting the parameters and criteria for considering the parameters mentioned earlier and adding as an additional planning parameter the safety distance from objects such as land, shallow waters (less than safety contour), and the attributes mentioned in the second part. The Safety Distance can be suggested to be 12 nm in Open Waters where traffic, weather, and other parameters allow, and in Shallow waters or Coastal Navigation can be reduced to 5 nm or less depending on the Navigation Risk Assessment of the voyage and the Company's Navigation Policy. The attributes that will be used for the safety check are divided into the following categories according to the table 6 below:

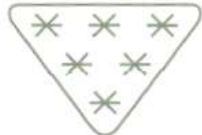
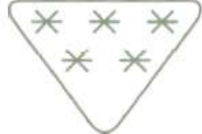
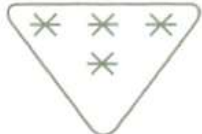



Table 6 Safety Check ENC Attributes and ENC group they belong

Input Ship Values	ENC Attribute group 1 (S-101 ENC)	ENC Attribute group 2 (S-101 ENC)	ENC Attribute group 3 (S-101 ENC)	Input Environmental Factors	UKC Calculation Parameters
1) Ship's Depth,	1) Navigational hazards (Isolated Dangers, wrecks, Reefs, Rocks, Land),	1) Areas for which special conditions exist,	1) Information areas (Anchor berth, Cargo transshipment area, Fishing ground),	1) Wind direction and speed (S-412 Weather / Met-ocean forecasts Overlay).	1) Ship's Squat,
2) Breadth,		2) Caution area,	2) Protected Access areas (Dumping ground, Incineration area, Pipeline area, Precautionary area, Cable area, Fairway, Territorial sea area).	2) Wave direction and speed (S-412 Weather / Met-ocean forecasts Overlay).	2) Tide (S-104 Water Level Information for Surface Navigation).
3) Length overall,	2) Depth contour area with no data,	3) Anchorage area,		3) Current direction and speed (S-412 Weather / Met-ocean forecasts Overlay).	3) Swell Height (S-412 Weather / Met-ocean forecasts Overlay).
4) Deadweight,	3) Category of the zone of confidence (CATZOC),	4) Military practice area,		4) Swell direction and speed (S-412 Weather / Met-ocean forecasts Overlay).	4) CATZOC for determining the minimum CATZOC value on each leg as per Table below (S-101 ENC)
5) Displacement,		5) Marine farm/culture, Seaplane landing area,	3) Ships routing systems and ferry routes (Recommended traffic lane part, Deep water route centerline, Two-way route part, Ferry route, Recommended route centerline, Recommended track, Deep water route part, Traffic separation line, Traffic separation scheme boundary, Traffic separation scheme crossing, Traffic separation scheme lane part, Traffic separation scheme roundabout),	412 Weather / Met-ocean forecasts Overlay).	
6) Keel to Masthead,		6) Inshore traffic zone,		Weather / Met-ocean forecasts Overlay).	
7) Air Draft,		7) Offshore production area,		Weather / Met-ocean forecasts Overlay).	
8) Ship's Draft,		8) Restricted area,		Information for Surface Navigation Ocean).	
9) Ship's Speed,		9) Areas to be avoided,		3) Surface Currents (S-111	
10) Ship's Squat (speed-dependent),		10) PSSA (Particularly Sensitive Sea Area),		10) Ice Coverage and thickness (S-411 Ice Information).	
11) Metacentric Height* (GoM) which is used for calculating parametric rolling,		11) Submarine transit lane, Traffic separation zone		11) Sea Depth and Bottom characteristics (S-101 ENC)	
			4) Miscellaneous (Continental shelf area, Exclusive economic zone)		

(Author, 2021)

Because CATZOC classification has more than a single characteristic, in table 7 below its attributes are shown in full detail. As covered in Chapter 9.2, S-100 have the Interoperability Catalogue (IC) function, so the suitable mode of data display is shown depending on its purpose. This can be used to enhance the accuracy of displayed data, so the end result is the optimal one for the required purpose. For the purpose of Automatic Route Safety Check and for Route Monitoring, Level 2 would be the best one to be used in the S-10x series. At the same time, Levels 3 and 4 are more suitable for use along with the framework of the Marine Spatial Data Infrastructure (MSDI).

Table 7 UKC recommended values in relation to CATZOC Depth and Position Accuracy

ZOC	SYMBOL	DEPTH ACCURACY		POSITION ACCURACY	RECOMMENDED UKC (m)	SEAFLOOR COVERAGE
A1		=0.50m+1%d		±5m	Greater Than	All significant seafloor features detected
		Depth (m)	Accuracy (m)			
		10	0.6			
		30	0.8			
		100	1.5			
A2		=1m+2%d		±20m	2	All significant seafloor features detected
		Depth (m)	Accuracy (m)			
		10	1.2			
		30	1.6			
		100	3.0			
B		=1m+2%d		±50m	2	Uncharted features hazardous to surface navigation are not expected but may exist.
		Depth (m)	Accuracy (m)			
		10	1.2			
		30	1.6			
		100	3.0			
C		=2m+5%d		±500m	3	Depth Anomalies may be expected
		Depth (m)	Accuracy (m)			
		10	2.5			
		30	3.5			
		100	37.0			
D		=2m+5%d		Worse than ZOC C	3	Large depth Anomalies may be expected
		Depth (m)	Accuracy (m)			
		Worse than ZOC C			4	
		Worse than ZOC C			-	
U		Unassessed - The quality of the bathymetric data has yet to be assessed.				

(NOAA, 2019)

9.3 Automatic Route Calculation Process

After combining the Automatic Route Planning and the Automatic Route Safety Check elements of the process as described previously, the Automatic Route Calculation can be successfully implemented. There is no mention of Cross Track Error (XTE) or connection to Navigational Sensors in the below flowchart because this belongs to the domain of autonomous navigation at sea. Due to its complexities and challenges, this thesis won't be enough to tackle it. For reference, the Groups mentioned in the flowchart in Appendix V "Autonomous Route Calculation Flowchart", they can be found in the previous Chapter 8.2.

Due to the risks mentioned in chapter 6.3 about Great Circle sailing and due to the constant course changing, the most common method of sailing is Rhumb Line, so in the Flowchart in Appendix V, the sailing method followed is assumed to be Rhumb Line.

9.4 Potential Applications

Once automatic route calculation is successful, its potential applications within the maritime industry are numerous and diverse, ranging from commercial (i.e., route and distance calculation for merchant ships) to environmental (creation of standard routes through particularly sensitive sea areas / PSSA) to statutory (creation of standard routes through particularly sensitive sea areas / PSSA) (Establishing Commercial Benchmark Routes for Specific Trade of Merchant Ships).

The most straightforward and direct outcomes are cases such as the UKC allowance for port tide optimization. One such experiment was undertaken by NOAA in the Port of Long Beach, Los Angeles, USA. Since 2017, Water levels - Predictions and real-time data are provided through PORTS® utilising additional Maritime Spatial Data gathered by Nearshore Wave Prediction System (NWPS) forecasts for wave and swell conditions (Physical Oceanographic Real-Time System). Wave buoys with real-time data transmission and an Integrated Ocean Observing System Lidar data is gathered along the shoreline and utilised to update nautical charts using latest hydrographic surveys and high-resolution bathymetric inland ENC's.

According to the NOAA's 2017 Port of Long Beach Precision Navigation initiative, Long Beach, is the United States largest port complex and the ninth busiest port in the world, already saves boats an estimated \$10 million per year in lightering expenses. This ground-breaking effort used private sector innovation and NOAA data streams to improve the safety of deep-draft ship navigation. The Port of Los Angeles - Long Beach is exposed to the open ocean and is thus subject to peculiar wave, swell, and water-level conditions. When waves came in long period swells, the new ultra-large oil ships entering the port were prone to groundings. For example, a ship pitching one degree may increase its draught by more than ten feet (03.05 m). The port decreased the maximum permissible ship draught to 65'ft (19.81 metres) as a precaution, despite the

fact that the dredged channel is 78' ft (23.77 meters).



Figure 44 Potential pitch of a vessel entering the Port of Long Beach. (PORT OF LONG BEACH PRECISION NAVIGATION PROJECT, NOAA, 2017)

According to the NOAA-funded Port of Long Beach Precision Navigation initiative, Jacobsen Pilots, which is responsible for all pilotage inside the port, first brought the matter to the notice of NOAA's Office of Coast Survey in late 2012 during a Los Angeles/Long Beach Harbour Safety Meeting. In the years that followed, an industry working group collaborated with a Dutch business that develops a web-based programme known as PROTIDE (PRObabilistic Tidal window DEtermination). PROTIDE optimises the port's accessibility by estimating the optimal times for ships that need tidal data to pass safely. This is accomplished by merging individual ship dimensions and stability information with real channel architecture, up-to-date environmental predictions, and a cutting-edge ship motion analysis engine. All of these computations need a large number of precise and verified real-time measurements of water levels and waves, which were made possible by expanding on an existing initiative with the Southern California Coastal Ocean Observing System (SCCOOS). Due to the project's success, the US Coast Guard Captain of the Port (COTP) has lifted the 65 ft (19.81 metres) draught limit, enabling draughts of up to 66 ft (20.11 metres) to access the port. In April 2017, a 66 ft (20.11 metres) tanker made three transits. After many runs at 66 foot (20.11 metres), the COTP will make a final decision to raise to 67 ft (20.42 metres). The long-term objective is to successfully travel at 69 ft (21.03 metres) draught. Offshore time lightering will be eliminated, resulting in increased operating efficiency and safety, as well as a reduction in environmental risk. Additionally, each additional foot of draught

allows cargo ships to carry an additional 40,000 barrels of crude oil. This corresponds to an extra \$2 million of merchandise loaded for every foot of draught increased. Additionally, in 2020, the National Oceanic and Atmospheric Administration (NOAA) conducted a Socioeconomic Study on the Impact of Precision Navigation, which emphasised the importance of enhancing navigation safety in light of the increasing number of incidents associated with the increasing volume of port traffic (both incoming/outgoing), as illustrated in the charts and figures below in this Chapter.

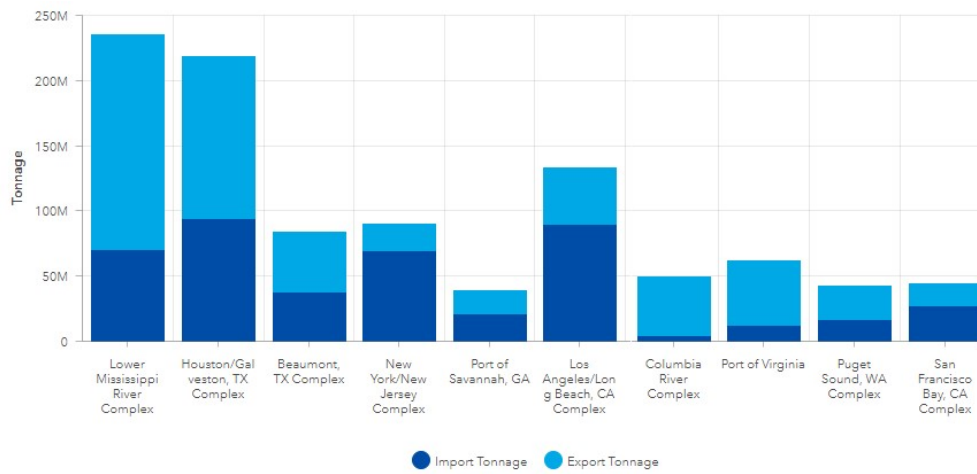


Figure 45 Import vs. Export Tonnage Chart in the 20 Top Ports in the US according to Precision Marine Navigation (PMN) Socio-Economic Study (NOAA, 2020)

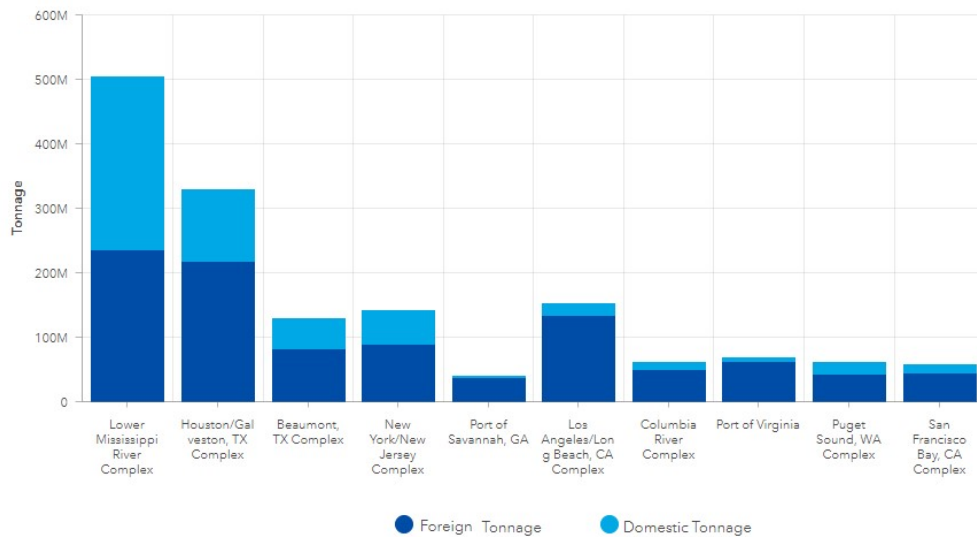


Figure 46 Foreign vs. Domestic Tonnage Chart in the Top 20 Ports in the U.S. according to Precision Marine Navigation (PMN) Socio-Economic Study (NOAA, 2020)

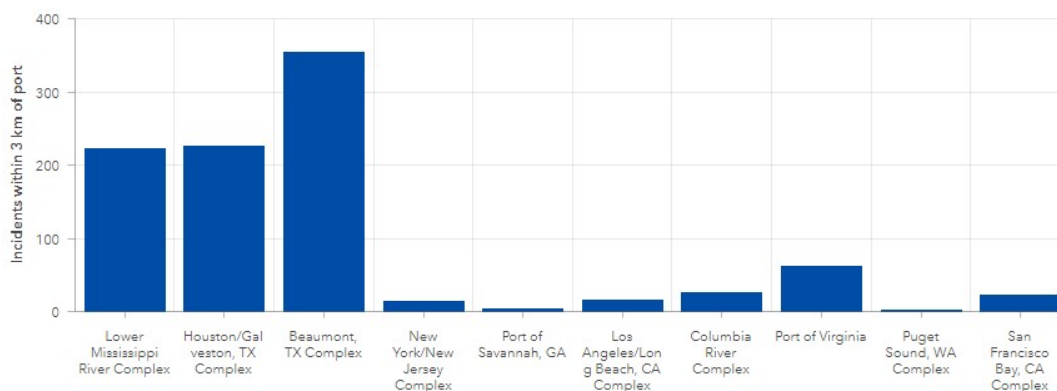


Figure 47 Incidents within 3 km of port Chart in the Top 20 Ports in the U.S. according to Precision Marine Navigation (PMN) Socio-Economic Study (NOAA, 2020)

According to the Precision Socioeconomic Study, NOAA, 2020, ships are already passing through US ports with Keel Clearance, which may be as little as 1 foot in certain circumstances (30 cm). Ships are becoming larger in response to rising demand for commodities and the need to improve the logistic supply chain's efficiency, while the "just-in-time" supply chain concept places increased demands on port operations. Mariners' tools for making safe operating choices have remained largely unchanged over the previous two decades, necessitating greater vessel load and wait periods in ports while depending on manual calculations and theoretical Tidal Tables in the absence of real-time environmental data. Precision Navigation offers sailors with a single data source for all navigational products, rather than forcing them to consult many data sources in order to identify the optimal path through crowded waterways. As indicated in this paper, ports that employ Precision Navigation should experience a gain in efficiency and safety, which benefits a wide range of stakeholders.

According to the Precision Socioeconomic Study, NOAA, 2020, the aim of this initiative is to accomplish three key goals: a) To utilise quantitative data to identify and prioritise the United States Ports that would reap the most benefits from Precision Navigation, as NOAA wishes to determine which additional United States ports would make economic sense to execute future projects. b) Develop valuation approaches for estimating Precision Navigation's major economic advantages. c) Apply those methodologies and consult with local stakeholders to estimate the economic benefits and impacts of implementing Precision Navigation at the Ports of New York/New Jersey and the Lower Mississippi River, which process the most ship tonnage and have some of the highest

incident rates among US ports. Precision Navigation Services are unrelated to S-100 on their own. Once the S-100 is completely developed, they will be effectively integrated into them, demonstrating the potential uses and cost savings that may be realised via the S-100's revolution of Maritime Spatial Data. Once Precision Navigation Services are deployed effectively, advantages will include pilots feeling more secure flying with less under-keel clearance and producing annual savings of between \$200 million and \$472 million owing to decreased operational expenses and faster turnaround. Delays will be reduced as a result of improved weather and tide forecast, and pilots will be able to operate more safely as a consequence of the real-time data from Precision Navigation, lowering the danger of allisions, accidents, and groundings. That benefit alone would save between \$176,000 and \$1,030,000 per year in physical damage and injuries, or around \$3.7 to \$9.8 million per year in more comprehensive economic damages.

Additionally, let us not forget the March 2021 grounding of the container ship Ever Given (see figure below) for reasons that are still unknown but have been attributed to Under Keel Clearance levels, Squat, and Wind Speed, all of which could have been avoided significantly by using more accurate real-time Maritime Spatial Data. Due to the continuing claims and pending proceedings relating to the event, the full effect of the particular grounding has not been determined.



Figure 48 Ever Given grounded in Suez (modified Copernicus Sentinel data [2021], processed by Pierre Markuse March 24th, 2021)

10 Artificial Intelligence

AI (Artificial Intelligence) can be defined either broadly or specifically depending on the type of use we refer to or the degree of familiarity with the subject. If a broad definition could be used, it would refer to it as the concept of a software based agent either trained or left to learn unsupervised, to being operated autonomously from Humans in accordance with either a set of defined goals or make decisions well under uncertainty, without any set goals. Simplifying AI as merely a system that thinks and acts like humans is an oversimplification, which can create certain ambiguity to arise, as AI operated systems have the potential to work outside of the scope of definition of Human Intelligence. This doesn't mean that Human Intelligence is inferior to AI, but rather it can be more proficient in figuring out problems that humans don't have the cognitive and sensory capacity to do so. This way it can be considered as supplementary to Human Intelligence.

10.1 Artificial Intelligence and application of new technological innovations in Route Finding

In this section, the review of machine learning and deep learning in route finding is examined. However, to begin with, a brief overview of machine and deep learning is undertaken to understand the underlying operations between them. At the foundational level, artificial intelligence is described as the science and engineering of making machines intelligent, whereby software thinks intelligently, similar to humans, while exploiting the greater speed and processing power of computers (Hiesboeck, 2018). In this regard, artificially intelligent applications mimic the cognitive functions of humans, such as problem-solving or learning (Oppermann, 2020). The author further postulates that the association between the three concepts is illustrated in figure below.

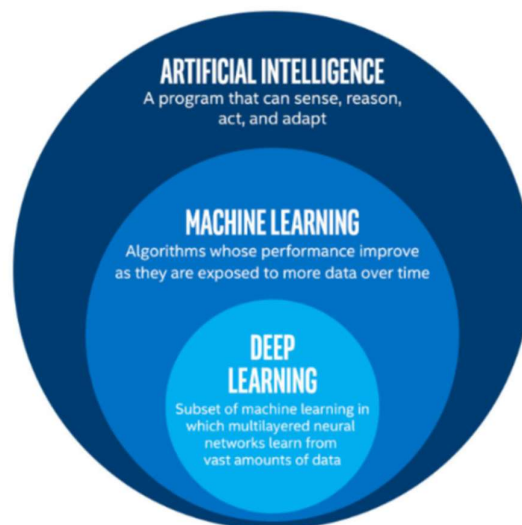


Figure 49. Relationship between AI, machine, and deep learning (Oppermann, 2020)

As illustrated in figure above, both machine and deep learning are subsets of artificial intelligence that focus on equipping machines with intelligence. The following section briefly discusses machine learning and further reviews its applications in route planning and pathfinding.

10.2 Machine Learning

According to Awad and Khanna (2015), machine learning describes a field that focuses on teaching computers to learn without being constantly programmed for specific tasks. Hiesboeck (2018) shares a similar view and further posits that machine learning also describes the ability to learn without explicit programming. However, to facilitate such learning, the algorithms require to be trained using data (Oppermann, 2020). In further explanation, the author reports that machine learning models focus on minimising the error between predictions made and actual existent ground truth values during the training process. In a different study, Shalev-Shwartz and Ben-David (2014) add that there exist diverse types of machine learning algorithms which include classification algorithms used in data classification such as Naïve Bayes Classifier and Support Vector Machines (SVM) and cluster analysis algorithms such as K-means and tree-based clustering.

The researchers argue that each of the different algorithms is associated with an objective or error function (Shalev-Shwartz and Ben-David, 2014). For instance, with the

classification algorithms such as SVM, the researchers argue that the algorithm's objective function would be to categorize data into one of two types, for example, cats and dogs. In such a case, the SVM algorithm would compare the prediction of the model against the ground truth values and parameters adjusted until the error rates between the predictions and actual values is as minimal as possible (Oppermann, 2020). In explanation, Oppermann (2020) further argues that the machine learning algorithms are essentially optimization algorithms as they focus on comparing ground truth values and the predicted model in a bid to reduce the error between them as much as possible.

Different researchers have employed the different algorithms in tackling path finding and route planning problems with this understanding of machine learning. To begin with, Snoeck, Merchán, and Winkenbach (2020) utilized machine learning algorithms to solve routing optimization problems in transport and logistics. In the study, the researchers argued that route optimization was a crucial challenge in the supply and logistics industry owing to data unavailability on endogenous and exogenous customer constraints. As a result, deviations from planned routes were a common challenge reported. A probabilistic Metropolis-Hastings within Gibbs algorithm was further utilized in making inferences from delivery transactions. Results obtained revealed that the method outperformed other existent traditional solutions that focused on simply counting occurrences.

In a second study, Basso, Kulcsár, and Sanchez-Diaz (2021) employed machine learning algorithms to facilitate route planning for electric vehicles in order to save energy. In the study, the researchers argued that the routing of electric vehicles required considering their limited driving range as it was affected by uncertain factors, such as traffic conditions. The researchers adopted a probabilistic Bayesian machine learning approach to predict energy consumption as well as the variance for road paths, routes, and links (Basso, Kulcsár and Sanchez-Diaz, 2021). The researchers further utilized data for public routes in Gothenburg-Sweden using realistic simulations for traffic on a 24-hr basis in Luxembourg city. Results obtained show that there was high accuracy for energy prediction as well as energy savings for the reliable routes (Basso, Kulcsár and Sanchez-Diaz, 2021). The findings underscored the importance of Bayesian networks in route and path planning.

10.3 Deep Learning

According to Oppermann (2020), deep learning is a subset of machine learning as its algorithms also rely on data in order to solve different tasks. Hiesboeck (2018) further adds that deep learning aims to enhance machine learning closer to artificial intelligence by creating knowledge from multiple layers of information processing. The author adds that at the foundational level, deep learning algorithms comprise multiple layers of neural networks, which are modelled after the human brain (Hiesboeck, 2018). Figure below illustrates a deep neural network.

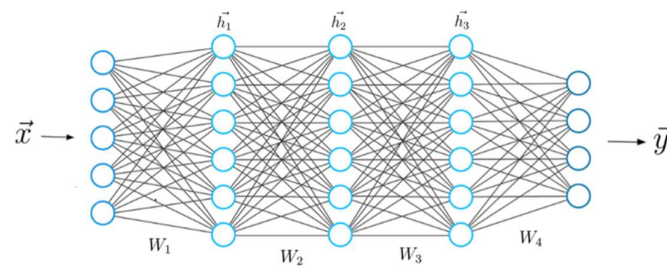


Figure 50 Deep neural network (Oppermann, 2020)

As detailed in figure above, deep learning algorithms are associated with several layers of neural networks (artificial neural networks), which enhance their performance. In another study, Molnar (2019) reports that the improvement in performance for the deep learning algorithms arises from the fact that the artificial neural networks can learn and make decisions on their own from the data, unlike flat algorithms, which parse and learn from data before using the insights to make decisions. Diverse researchers have also employed deep learning algorithms in tackling route and path planning problems. To begin with, Duc, Huu, and Nananukul (2020) developed a dynamic route-planning system that was based on artificial neural networks (ANN) for automated guided vehicles within a warehouse. In the study, the researchers reported that there lacked real-time route-planning algorithms, which were suitable in implementing automated guided vehicles with limited computing resources. To test a real-time route-planning system, Duc, Huu, and Nananukul (2020) developed a simulation of the internal layout of the warehouse and an optimization model based on machine learning ANNs to determine an operational route for the problem. Findings reported showed that the

algorithms were able to generate routes that were 98% accurate for the practical internal layout of the warehouse, which had 18 storage racks and 67 path segments.

In a second study, Blum, Jones, and Yoshida (2019) utilized deep learning reinforcement learning algorithms in real-time path planning for lunar and space exploration robots. In the study, the researchers argued that deep learning was significant in advancing space exploration. It could be applied in high-level tasks such as path planning and low-level tasks such as motion control, which were important in enabling the robots to walk and undertake path planning. To train the robots, Blum, Jones, and Yoshida (2019) proposed deep reinforcement learning with randomized reward function parameters for simulated 8 degree-of-freedom robots that were quadruped and ant-like in shape. The deep reinforcement learning algorithms would enable the robotic agents to travel anywhere within the environment by undertaking both path planning and motion control within the single neural network. Various improvements were reported in the research as the autonomous robots became more liberal in path selection. They could choose real-time paths based on sensor information instead of conservative pre-built graphs (Blum, Jones, and Yoshida, 2019). The research results indicated significant promise regarding the advancement in space exploration robots such as rovers in fast locomotion and legged cave robots prone to rough terrain.

In another third study, Yu, Su, and Liao (2020) further demonstrated the use of neural networks and hierarchical reinforcement learning to facilitate autonomous path-planning in mobile robots. From the study results, it was observed that neural networks and hierarchical reinforcement learning could be effectively used to develop dynamic routes for the mobile robots tested in different kinds of environments. The researchers further compared the performance of various algorithms. The Deep Deterministic Policy Gradient (DDPG) approach was observed to generate the best results, for instance, in terms of shorter path-planning time and reduced number of steps. Furthermore, results obtained showed that the algorithm shortened the convergence time by 91% compared to other algorithms such as Q-learning and further enhanced the smoothness of planned paths by about 44% (Yu, Su, and Liao, 2020). In another study, Lei, Zhang, and Dong (2018) also adopted deep reinforcement learning to facilitate dynamic path planning for robots in unknown environments. From the study results, it was observed that the double Q-network (DDQN), a deep reinforcement learning algorithm, enhanced the

ability of the autonomous robots to avoid obstacles dynamically, thereby successfully reaching the local target position in unknown environments. The argument by the researchers was that deep reinforcement learning solved the curse of dimensionality quickly and further processed inputs that were multidimensional. A different study by Woo, Lee, and Cha (2018) also utilized reinforcement learning and neural networks to develop optimal routes for mobile robots. In the research, the researchers argued that there was a need for mobile robots to dynamically plan their paths, primarily where they worked with other robots and humans. As such, reliance on shortest path algorithms could increase waiting times as other robots blocked optimal paths. Woo, Lee, and Cha (2018) combined neural networks and reinforcement learning to overcome the issues, thereby enabling the robot to design the shortest path by making its own judgment based on available environmental information.

The analysis of the different applications of deep learning underscores the value of neural networks in path-planning problems. Furthermore, from the studies' analysis, deep learning coupled with reinforcement learning was an effective approach in developing dynamic and real-time paths for mobile robotic agents in different environmental contexts. In the study by Duc, Huu, and Nananukul (2020), ANNs were observed to be effective in internal environments such as warehouses, where dynamic real-time route planning paths could be generated for guided vehicles. Blum, Jones, and Yoshida (2019) would further emphasize the value of reinforcement learning in path planning for lunar and space robots. The findings were reiterated in the study by Yu, Su, and Liao (2020), Okereke, Mohamad, and Wahab (2020), and Lei, Zhang, and Dong (2018), where neural networks coupled with deep reinforcement learning led to more optimal results. From an analytical viewpoint, it can be argued that the combination of deep learning and reinforcement learning leads to better results as compared to other algorithms as the autonomous robots are equipped with the capability to be more liberal in path selection as they can choose real-time paths based on sensor information as opposed to conservative pre-built graphs (Blum, Jones and Yoshida, 2019). Furthermore, as Oppermann (2020) argued, such deep learning algorithms do not require any form of human intervention in feature extraction as they automatically handle the process.

10.4 Comparison between Machine and Deep Learning

The comparison between machine learning and deep learning highlights several noteworthy differences. To begin with, Molnar (2019) reports that deep learning algorithms outperform machine learning algorithms by eliminating the need for feature extraction. In explanation, the author argues that machine learning algorithms such as Decision Trees, SVM, Naïve Bayes, and Logistic Regression are considered flat algorithms as they require additional pre-processing through feature extraction before they can be applied to the raw data (Molnar, 2019). Oppermann (2020) also holds a similar view, who also reports that with feature extraction, the raw data is represented in different classes or categories to allow the machine learning algorithms to perform a given task. On the contrary, the author reports that with artificial neural networks, the pre-processing step is not required as the data layers in the algorithms can represent the data independently (Goodfellow, Bengio, and Courville, 2016). This finding implies that with machine learning algorithms, there is a need for human intervention whereby programmers hand-code the different features applied in the data. However, feature extraction is undertaken by the program itself, not requiring additional human intervention with deep learning algorithms. Figure 51 below illustrates the differences between machine and deep learning algorithms.

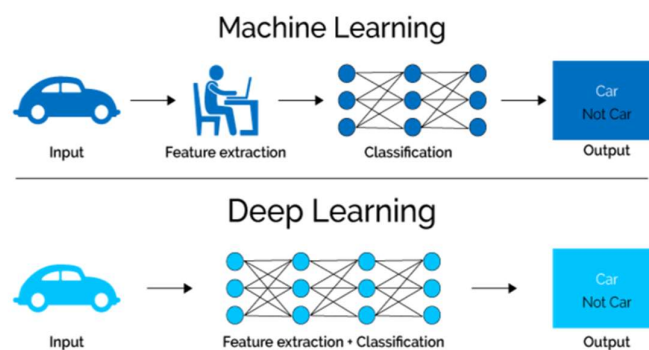


Figure 51. Difference between machine and deep learning (Oppermann, 2020)

As detailed in figure above, machine learning algorithms require the input of programmers to hand-code the different features before the algorithm can be applied to the raw data. As an illustrative example, the author considers the problem of classification of images as either car or not. With machine learning algorithms, a programmer must explicitly state the features that help identify a car, for instance,

windows, wheels, shape, among others (Oppermann, 2020). However, with deep learning, the algorithm can identify the car's unique features without any human intervention, thereby facilitating the analysis process. Conversely, deep learning algorithms already contain the feature extraction process as part of the algorithm's training process. The algorithm can optimize the step-in to obtain the best abstract representation of the data (Oppermann, 2020).

A second difference between machine and deep learning regards the influence of training data on the performance of the algorithms (Goulet, 2020). According to the author, deep learning algorithms can increase their accuracy with an increase in the amount of training data. However, with machine learning algorithms, the accuracy of the algorithms only increases up to a specific saturation point (Goulet, 2020). Thirdly, Goodfellow, Bengio, and Courville (2016) also highlight that further differences also emerge regarding the hardware used by the algorithms. In explanation, the authors report that machine learning algorithms run on less powerful machines such as CPUs with less computing power, unlike deep learning algorithms that require more powerful hardware such as graphical processing units (GPUs). On the same note, Oppermann (2020) observes that a shorter amount of time is required to process machine learning algorithms compared to deep learning algorithms. Likewise, other differences are also observed regarding how the deep learning algorithms can be tuned or performance enhanced, unlike machine learning algorithms associated with a limited tuning capability (Molnar, 2019).

In addition to the discussed differences, other differences have also been observed about their application in route finding and path planning. For instance, from the analysis of diverse empirical studies relying on neural networks and reinforcement learning (Lei, Zhang, and Dong, 2018; Yu, Su, and Liao, 2020; Okereke, Mohamad, and Wahab, 2020), it emerged that higher performance was observed for autonomous agents with regard to path planning and selection of optimal routes in unknown environments. Such advantages were attributed to the capability of the algorithms to select features without human intervention, thereby enabling the agents to be more autonomous in path selection.

11 Cybersecurity concerns

In this section, a review of cybersecurity concerns regarding the use of algorithms for route planning is undertaken. In particular, there is a specific focus on identifying the risks that arise from the use of algorithms in pathfinding and route planning in sea navigation. To begin with, Felski and Zwolak (2020) discuss two core cybersecurity threats that are faced by unmanned surface vehicles (USV) such as ships when autonomous navigational systems are implemented within them. First, the authors argued that there was a risk of threats from using basic equipment for navigational purposes (Felski and Zwolak, 2020). In explanation, the authors argued that reliance on Global Positioning System (GPS) and Global Navigation Satellite Systems (GNSS) for communication was associated with various dangers, such as they would be easily interfered with by malicious users. The finding is further supported by Grant A., Williams, P., Ward, N. & Basker, S. (2009), who reported that GNSS and GPS systems, which were the primary source of navigation for maritime applications, were at risk of interference or signal jamming, which could lead to possible denial of service over large geographical areas.

A second threat identified by Felski and Zwolak (2020) regarded various GNSS vulnerabilities, including spoofing, interference, and cyber-attacks. Figure below displays some of the vulnerabilities associated with GNSS systems.

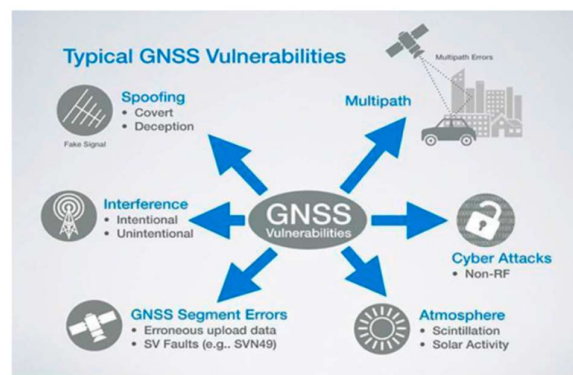


Figure 52 GNSS threats and vulnerabilities (Felski and Zwolak, 2020)

Felski and Zwolak (2020) further reported that signal jamming was a commonly increasing problem as signals could be sent with an intention to interfere with communication from satellites or send false signals to the receivers. However, to

counter the jamming vulnerability, the authors advocated for the use of independent receiving devices that could calculate the ship's position based on multiple satellites. As a result, this would facilitate identifying the source and direction of the jamming signal. A similar finding was also reiterated by Marcos E.P., Caizzone, S., Konovaltsev, A., Cuntz, M., Elmarissi, W., Yinusa, K. & Meurer, M. (2018), who further reported that due to the increased dependence on GNSS systems for communication in sea navigation, this further raised concerns regarding their vulnerability, especially where signals were blocked deliberately through radio frequency interferences. However, in order to further counter cybersecurity threats in ship systems, the adoption of a cybersecurity risk assessment model is recommended. Refer to Appendix B, which shows a hazard analysis model for the cloud, and Appendix C illustrates an assessment model for onboard servers.

12 Conclusions

This research aimed to show the complexities of passage planning by analyzing the factors affecting it and show how to implement the use of the new S-100 IHO standard and pathfinding algorithms in automatic route planning and examine path-finding algorithms in route navigation for various manned and unmanned surface marine vehicles, merchant ships in specific. Still, it can be applied as well to other maritime applications such as Shore Based operations. Based on the examination of diverse studies, it emerged that pathfinding is limited to vehicle navigation, video games and robotics. The findings indicated that a similar end goal was common in the different scenarios as different agents focused on reaching their target destinations. For instance, in both robotics and video games, the various agents focused on avoiding obstacles and successfully reaching their end destinations. However, with ship navigation, path finding is further constrained by a range of environmental and technical factors such as weather, fuel consumption and the need to minimise emissions of greenhouse gases. Therefore, the research underscores the peculiarity of pathfinding in ships as there is a myriad of constraints that must be factored into when selecting the routing algorithms. A direct policy implication that arises in this regard is that ship captains and navigators need to develop a robust and highly diversified criteria when selecting different optimal routes for ship navigation at sea. In this regard, the criteria ought to not only ensure the

ship routing algorithms are effective in avoiding obstacles as well as reaching the end goal but also align to the demands of reducing fuel consumption and greenhouse emissions.

Further analysis in the thesis revealed diverse path-finding algorithms that are important in the process. The algorithms ranged from BFS, DFS, IDDFS, Bidirectional search, Dijkstra, genetic algorithms, ant-colony algorithms, A* and various variations of A* such as D*, Lifelong Planning A* (LPA*), Weighted A*. Comprehensive analysis of the algorithms shown in Chapter 7.3 and Appendix IV, which reveal that each of these algorithms is associated with diverse advantages and disadvantages. Directly, the finding implies that ship captains and navigators have access to a wide array of path-finding algorithms in tackling ship routing problems. Furthermore, from the analysis of diverse empirical studies, it also emerged that different researchers have used different algorithms to solve path-finding problems in ship navigation. Therefore, the policy implication is that there is a need for ship navigators to closely evaluate the different path-finding algorithms in order to ensure they achieve the targeted objectives in ship navigation. An alternative recommendation is reliance on best practices, whereby the navigators can utilise the algorithms that are the most popular by use in other transportation sectors.

Finally, the paper examined the role of artificial intelligence, machine learning and deep learning in route planning. From the analysis carried out in Chapter 9, it emerged that although machine learning is the superset of deep learning, deep learning algorithms have, however, assumed a central role in navigation. In particular, the use of neural networks and deep reinforcement learning algorithms was observed to be a highly popular solution. Nonetheless, as the adoption of autonomous ship navigation is advocated from the use of deep learning algorithms, further cybersecurity concerns were also highlighted, including signal jamming and malicious attacks to the communication equipment in the ships. Nevertheless, diverse solutions were also identified to mitigate the threats and ensure that the ships are able to attain their goals in reaching targeted destinations successfully.

Based on the findings obtained in the research, several future research avenues are identified. First, as the research has revealed that diverse types of path-finding

algorithms ranging from Dijkstra to A*, genetic and ant colony algorithms can be employed in route navigation for ships, there is a need for studies which compare performance of these algorithms in solving ship navigation problems and identifying which algorithms or combination of them, is suitable for ship navigation. Such studies provide important insights regarding the performance of algorithms in route navigation, thereby enabling ship navigators and captains to select the most optimal algorithms for pathfinding in ships. Secondly, the research findings have also revealed that deep learning algorithms have also emerged as an important solution in pathfinding, especially in robotic applications. However, in order to benefit from the algorithms, there is need for further research which utilises them in ship navigation and routing applications and the need to install as many sensors capturing relevant marine environmental data such as tides, sea temperature, salinity, sea current force and direction, firstly in the most frequently visited and infrastructure critical ports. The study has revealed that there is a dearth of research on deep learning applications in ship routing and pathfinding, thereby necessitating future research. A third future avenue is that there is a need for future research into cybersecurity risks that arise from reliance on machine and deep learning algorithms in ship routing activities. However, the study revealed that various cybersecurity risks, including signal jamming and interference of communication in ships, were a commonly reported problem. In future research, more empirical studies which shed light on the shortcomings of reliance on machine and deep learning in pathfinding among ships are required.

Based on the above findings of this Thesis, we can summarise the areas where further research is needed, which can complement the content of this Thesis:

1. How do path-finding algorithms ranging from Dijkstra to A*, genetic and ant colony algorithms compare in solving real life ship navigation problems and identifying which algorithms or combination of them, is suitable for different types and situation of ship navigation?
2. What is the role and importance of marine environmental sensors, installed onboard and in navigable areas of the ship?
3. What are the accuracy and calibration standards applicable to marine environmental sensors used in Precision Marine Navigation applications?

4. What is the role, applications and importance of machine and deep learning path-finding algorithms in ship routing and pathfinding?
5. What are the cybersecurity risks that arise from reliance on machine and deep learning algorithms in ship routing activities?
6. What are the findings of empirical studies on the shortcomings of reliance on machine and deep learning in pathfinding among ships are required?

As stated in Chapter 11 about Research Methodology, due to lack of response for Personal Interviews and Questionnaires not being well suited for this subject, the reliability of this Thesis is based on the literature review and the concepts being introduced. The validity of the Research Question made as stated in Chapter 1.2 is based on the cross examination of the literature review and the quality and reliability of the References used such as the American Practical Navigator Bowditch Almanac and IHO and IMO literature used, which have all considered as the go to references in the field of Maritime Navigation. Finally, the importance of Research Questions posed in this Thesis are dependent on the necessity of Automatic Route Planning in Autonomous Navigation but also for the daily commercial usage in real life applications, which as explained throughout Chapter 8, it's evident that these will be in the forefront of any developments in the future.

13 Bibliography

IHO S-100 Standard, (2015) IHO, <https://youtu.be/6c-tcuVJf7c> [Accessed 02 July 2021]

IHO ECDIS and ENC Standards by British Admiralty, (2017) UKHO, <https://www.admiralty.co.uk/news/blogs/s-57-and-the-latest-iho-standards> [Accessed 03 July 2021].

S-100 Universal Hydrographic Data Model (2019), IHO, https://iho.int/uploads/user/pubs/standards/s-100/S-100_Ed%204.0.0_Clean_17122018.pdf [Accessed 04 July 2021].

S-100–UNIVERSAL HYDROGRAPHIC DATA MODEL, (2018) IHO, https://iho.int/uploads/user/pubs/standards/s-100/S-100_Edition_2.0.0.pdf [Accessed 04 July 2021].

Roh Myung-II, (2013), Determination of an economical shipping route considering the effects of sea state for lower fuel consumption, International Journal of Naval Architecture and Ocean Engineering.

SIRE Vessel Inspection Questionnaire (VIQ), (2019), OCIMF.

S-100 ECDIS Trials (2018), KHOA, <https://youtu.be/Z8FhC2OUdXU> [Accessed 08 July 2021].

Knippers, R. (2009), Geometric aspects of mapping, International Institute for Geo-Information Science and Earth Observation, Enschede, <http://kartoweb.itc.nl/geometrics> [Accessed 09 January 2021].

ESRI ArcMap Manual (2016) <https://desktop.arcgis.com/en/arcmap/10.3/guide-books/map-projections/what-happens-to-features-at-180-dateline-.htm> [Accessed 05 July 2021].

Geokov Map Projections (2014) <http://geokov.com/education/map-projection.aspx> [Accessed 08 January 2021].

American Practical Navigator, (2019) National_Geospatial-Intelligence Agency, https://thenauticalalmanac.com/2019_Bowditch-_American_Practical_Navigator.html [Accessed 05 July 2021].

Admiralty Practical Guide to the use of ENC's, (2012) UKHO.

Douglas D. Nebert (2004), Developing Spatial Data Infrastructures: The SDI Cookbook, Version 2.0.

SPATIAL DATA INFRASTRUCTURES "THE MARINE DIMENSION," Guidance for Hydrographic Offices, Publication C-17, Second Edition, (2017), IHO.

How Much Water is There on Earth?, USGS, (2018), https://www.usgs.gov/special-topic/water-science-school/science/how-much-water-there-earth?qt-science_center_objects=0#qt-science_center_objects [Accessed 08 July 2021].

AWP Marine Consultancy_Ltd (2017), Newsletter Issue 4. <https://awpmarine.com/News/Latest-News/awp-marine-consultancy-ltd-newsletter-4> [Accessed 08 July 2021].

Maritime navigation and radiocommunication equipment and systems Electronic chart display and information system (ECDIS)–61174, Technical Standard ed. 4, (2015), IEC.

ENC Validation Checks, Publication S-58, Edition 6.1.0, (2018) IHO.

IMO Circular MSC.1/Circ.1503 IMO Guidelines for ECDIS Good Practice, (2017), IMO.

Where Do We Map Next, Anne-Cathrin Wöfl, Jennifer Jencks, Colin Devey, (2021), Hydro International.

NAVBasics, Volume 2, Ocean Offshore and Celestial Navigation, Abdul Khalique, (2009), Witherbys Seamanship.

Paper for Consideration by TSM5, Procedures for S-100 Interoperability Catalogue, (2017) KHOA, https://iho.int/mtg_docs/com_wg/S-100WG/TSG5/TSM5-5.3_Procedures_for_S-100_InteroperabilityCatalogue.pdf [Accessed 10 July 2021].

NOAA PORT OF LONG BEACH PRECISION NAVIGATION PROJECT, NOAA, 2017<https://nauticalcharts.noaa.gov/learn/docs/precision-navigation/lalbpresionnav-1pager.pdf> [Accessed 18 July 2021].

Transforming the NOAA ENC®, Implementing the National Charting Plan, (2019) NOAA, <https://nauticalcharts.noaa.gov/publications/docs/enc-transformation.pdf> [Accessed 18 July 2021]

S-100 sea trials: working toward harmonized navigation products, (2019) NOAA, <https://nauticalcharts.noaa.gov/updates/s-100-sea-trials-working-toward-harmonized-navigation-products/> [Accessed 20 July 2021].

Precision Navigation Socioeconomic Study, (2020) Eastern Research Group, Inc. and NOAA, <https://nauticalcharts.noaa.gov/learn/docs/precision-navigation/precision-navigation-socioeconomic-study.pdf> [Accessed 20 July 2021].

IHO S-100: The New Hydrographic Geospatial Standard for Marine Data and Information, Robert Ward, Lee Alexander, Barrie Greenslade, Anthony Pharaoh, (2008), University of New Hampshire, Center For Coastal And Ocean Mapping,

Lee Alexander, Maxim F. Van Norden, Charles M. Fralick, (2001), ECDIS Development Laboratory and Navigation Technology Demonstration Centre, University of New Hampshire, Center for Coastal Mapping.

Abd Alfoor, Z., Sunar, M.S. & Kolivand, H. (2015). A Comprehensive Study on Pathfinding Techniques for Robotics and Video Games. *International Journal of Computer Games Technology*, 2015, 1–11.

Achour, N. & Chaalal, M. (2011). Mobile Robots Path Planning using Genetic Algorithms. In: *ICAS 2011: The Seventh International Conference on Autonomic and Autonomous Systems*.

Akash (2020). A* Algorithm | Introduction to the A* Search Algorithm. [online] Edureka. Available at: <https://www.edureka.co/blog/a-search-algorithm/>. [Accessed 03 August 2021].

Avacheva, T. & Prutzkow, A. (2020). The Evolution of Imperative Programming Paradigms as a Search for New Ways to Reduce Code Duplication. IOP Conference Series: Materials Science and Engineering, 714, 012001.

Awad, M. & Khanna, R. (2015). Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers. Berkeley, Ca: Apressopen.

Bagheri, S.M., Taghaddos, H., Mousaei, A., Shahnavaaz, F. & Hermann, U. (2021). An A-Star algorithm for semi-optimization of crane location and configuration in modular construction. Automation in Construction, 121, 103447.

Basso, R., Kulcsár, B. & Sanchez-Diaz, I. (2021). Electric vehicle routing problem with machine learning for energy prediction. Transportation Research Part B: Methodological, 145, 24–55.

Blum, T., Jones, W. & Yoshida, K. (2019). Deep Learned Path Planning via Randomized Reward-Linked-Goals and Potential Space Applications. [online] Available at: <https://arxiv.org/pdf/1909.06034.pdf> [Accessed 08 August 2021].

Botea, A., Bouzy, B., Buro, M., Bauckhage, C. & Nau, D. (2013). Pathfinding in Games. [online] Available at: <https://drops.dagstuhl.de/opus/volltexte/2013/4333/pdf/4.pdf> . [Accessed 11 August 2021].

Bu, Z. & Korf, R.E. (2019). A*+IDA*: A Simple Hybrid Search Algorithm. [online] Available at: <https://www.ijcai.org/proceedings/2019/0168.pdf> . [Accessed 13 August 2021].

Bulitko, V., Björnsson, Y., Sturtevant, N.R. & Lawrence, R. (2011). Real-Time Heuristic Search for Pathfinding in Video Games. Artificial Intelligence for Computer Games, 1–30.

Cagigas, D. & Abascal, J. (2005). A Hierarchical Extension of the D* Algorithm. Journal of Intelligent and Robotic Systems, 42(4), 393–413.

Cai, Y. & Ji, X. (2018). ASA-routing: A-Star Adaptive Routing Algorithm for Network-on-Chips. Algorithms and Architectures for Parallel Processing, 187–198.

- Carreira, P., Amaral, V. & Vangheluwe, H. (2020). Foundations of multi-paradigm modelling for cyber-physical systems. Cham, Switzerland Springer Open.
- Chandra, M.T. (2014). Classical approach to artificial intelligence. New Delhi: Khanna Book Publishing.
- Cui, X. & Shi, H. (2011). A*-based Pathfinding in Modern Computer Games. *International Journal of Computer Science and Network Security*, 11(1), 125–130.
- Dai, X., Long, S., Zhang, Z. & Gong, D. (2019). Mobile Robot Path Planning Based on Ant Colony Algorithm With A* Heuristic Method. *Frontiers in Neurorobotics*, 13.
- Dakulović, M. & Petrović, I. (2011). Two-way D* algorithm for path planning and replanning. *Robotics and Autonomous Systems*, 59(5), 329–342.
- De Carvalho, M.A.M. & Soma, N.Y. (2015). A breadth-first search applied to the minimization of the open stacks. *Journal of the Operational Research Society*, 66(6), 936–946.
- Denden, M., Essalmi, F. & Tlili, A. (2016). A 3-D Educational Game for enhancing learners' performance in A star Algorithm. *Innovations in Smart Learning*, 29–32.
- Duchoň, F., Babinec, A., Kajan, M., Beňo, P., Florek, M., Fico, T. & Jurišica, L. (2014). Path Planning with Modified a Star Algorithm for a Mobile Robot. *Procedia Engineering*, 96, 59–69.
- Duc, D.N., Huu, T.T. & Nananukul, N. (2020). A Dynamic Route-Planning System Based on Industry 4.0 Technology. *Algorithms*, 13(12), 308.
- Ebendt, R. & Drechsler, R. (2009). Weighted A* search – unifying view and application. *Artificial Intelligence*, 173(14), 1310–1342.
- Edelkamp, S. & Schrödl, S. (2012). Heuristic search: theory and applications. Amsterdam ; Boston: Morgan Kaufmann, C.
- ElHalawany, B.M., Abdel-Kader, H.M., Eldien, A.S.T., Elsayed, A.E. & Nossair, Z.B. (2013). Modified A* algorithm for safer mobile robot navigation. In: U2013

Proceedings of International Conference on Modelling, Identification & Control (ICMIC)Cairo, Egypt, 31st Aug.- 2nd Sept.

Erke, S., Bin, D., Yiming, N., Qi, Z., Liang, X. & Dawei, Z. (2020). An improved A-Star based path planning algorithm for autonomous land vehicles. *International Journal of Advanced Robotic Systems*, 17(5), 172988142096226.

Felski, A. & Zwolak, K. (2020). The Ocean-Going Autonomous Ship—Challenges and Threats. *Journal of Marine Science and Engineering*, 8(1), 41.

Foad, D., Ghifari, A., Kusuma, M.B., Hanafiah, N. & Gunawan, E. (2021). A Systematic Literature Review of A* Pathfinding. *Procedia Computer Science*, 179, 507–514.

ForišekM. & SteinováM. (2013). Explaining algorithms using metaphors. London: Springer.

Ghaffari, A. (2014). An Energy Efficient Routing Protocol for Wireless Sensor Networks using A-star Algorithm. *Journal of Applied Research and Technology*, 12(4), 815–822.

Goodfellow, I., Bengio, Y. & Courville, A. (2016). Deep learning. Cambridge, Massachusetts: The MIT Press.

Goulet, J.-A. (2020). Probabilistic machine learning for civil engineers. Cambridge, Massachusetts: The MIT Press.

Grant, A., Williams, P., Ward, N. & Basker, S. (2009). GPS Jamming and the Impact on Maritime Navigation. *Journal of Navigation*, 62(2), 173–187.

Grosan, C. & Abraham, A. (2013). Intelligent Systems A Modern Approach. Berlin Springer Berlin.

Giardini, G. & Kalmár-Nagy, T. (2011). Genetic Algorithm for Combinatorial Path Planning: The Subtour Problem. *Mathematical Problems in Engineering*, 2011, 1–31.

- Grifoll, M., Martorell, L., Castells, M. & de Osés, F.Xavier.M. (2018). Ship weather routing using pathfinding algorithms: the case of Barcelona – Palma de Mallorca. *Transportation Research Procedia*, 33, 299–306.
- Guo, J. & Liu, L. (2010). A study of improvement of D* algorithms for mobile robot path planning in partial unknown environments. *Kybernetes*, 39(6), 935–945.
- Hamadi, Y. & Schoenauer, M. (2012). *Learning and intelligent optimization 6th international conference; revised selected papers*. Berlin Heidelberg Springer.
- Huang, H. (2020). Intelligent Pathfinding Algorithm in Web Games. *Advances in Intelligent Systems and Computing*, 250–257.
- Hulianytskyi, L. & Pavlenko, A. (2019). Ant Colony Optimization Algorithms with Diversified Search in the Problem of Optimization of Airtravel Itinerary. *Cybernetics and Systems Analysis*, 55(6), 978–987.
- Hurbans, R. (2020). *Grokking artificial intelligence algorithms*. Shelter Island, Ny: Manning.
- Hiesboeck, M. (2018). The Difference Between Artificial Intelligence, Machine Learning, and Deep Learning. [online] Futurist. Available at: <https://martinhiesboeck.com/2018/11/07/beyond-the-hype-the-difference-between-artificial-intelligence-machine-learning-and-deep-learning/> [Accessed 5 May 2021].
- Hu, Y. & Yang, S.X. (2004). A knowledge based genetic algorithm for path planning of a mobile robot. *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04*. 2004.
- Iordan, A.E. (2018). A comparative study of the A* heuristic search algorithm used to solve efficiently a puzzle game. *IOP Conference Series: Materials Science and Engineering*, 294, 012049.
- Ivanová, M. & Surynek, P. (2014). Adversarial Cooperative Path-finding: Complexity and Algorithms. [online] Available at: http://surynek.net/publications/files/Ivanova-Surynek_ACPF_ICTAI-2014.pdf.

- Javaid, M.A. (2013). Understanding Dijkstra Algorithm. SSRN Electronic Journal.
- Jebari, M. (2013). Selection methods for genetic algorithms. *International Journal of Emerging Sciences*, 3(4), 333–344.
- Jagga, S. (2020). Uniform Cost Search | Algorithm of Uniform Cost Search. [online] EDUCBA. Available at: <https://www.educba.com/uniform-cost-search/>. [Accessed 15 May 2021].
- Kang, N.K., Son, H.J. & Lee, S.-H. (2018). Modified A-star algorithm for modular plant land transportation. *Journal of Mechanical Science and Technology*, 32(12), 5563–5571.
- Kazharov, A.A. & Kureichik, V.M. (2010). Ant colony optimization algorithms for solving transportation problems. *Journal of Computer and Systems Sciences International*, 49(1), 30–43.
- Kem, O., Balbo, F. & Zimmermann, A. (2017). Collaborative Search for Multi-goal Pathfinding in Ubiquitous Environments. *Multiagent System Technologies*, 72–88.
- Korb, O. (2009). Efficient ant colony optimization algorithms for structure- and ligand-based drug design. *Chemistry Central Journal*, 3(S1).
- Korf, R.E., Reid, M. & Edelkamp, S. (2001). Time complexity of iterative-deepening-A*. *Artificial Intelligence*, 129(1-2), 199–218.
- Koenig, S., Likhachev, M. & Furcy, D. (2004). Lifelong Planning A*. *Artificial Intelligence*, 155(1-2), 93–146.
- Konar, A. (2018). *Artificial intelligence and soft computing: behavioral and cognitive modeling of the human brain*. New York: CRC Press.
- Korf, R.E. (2003). Search Techniques. *Encyclopedia of Information Systems*, 31–43.
- Kozen, D.C. (2012). *The design and analysis of algorithms*. New York: Springer-Verlag, [Post], Cop.

Kereki, F. (2017). *Mastering JavaScript functional programming: in-depth guide for writing robust and maintainable JavaScript code in ES8 and beyond*. Birmingham, UK: Packt Publishing.

Kmetiuk, A. (2018). *Mastering functional programming: functional techniques for sequential and parallel programming*. Birmingham, UK: Packt Publishing Ltd.

Kumar, N. (2019). Bidirectional Graph Search Techniques for Finding Shortest Path in Image Based Maze Problem. [online] Available at:
https://www.researchgate.net/publication/332371044_Bidirectional_Graph_Search_Techniques_for_Finding_Shortest_Path_in_Image_Based_Maze_Problem .
[Accessed 12 June 2021].

Lamini, C., Benhlina, S. & Elbekri, A. (2018). Genetic Algorithm Based Approach for Autonomous Mobile Robot Path Planning. *Procedia Computer Science*, [online] 127, 180–189. Available at:
<https://www.sciencedirect.com/science/article/pii/S187705091830125X>.
[Accessed 26 June 2021].

Lawrence, R. & Bulitko, V. (2010). Taking Learning Out of Real-Time Heuristic Search for Video-Game Pathfinding. *AI 2010: Advances in Artificial Intelligence*, 405–414.

Le, A., Prabakaran, V., Sivanantham, V. & Mohan, R. (2018). Modified A-Star Algorithm for Efficient Coverage Path Planning in Tetris Inspired Self-Reconfigurable Robot with Integrated Laser Sensor. *Sensors*, 18(8), 2585.

Lei, X., Zhang, Z. & Dong, P. (2018). Dynamic Path Planning of Unknown Environment Based on Deep Reinforcement Learning. *Journal of Robotics*, 2018, 1–10.

Li, C. & Ueno, M. (2017). An extended depth-first search algorithm for optimal triangulation of Bayesian networks. *International Journal of Approximate Reasoning*, 80, 294–312.

Lee, G. (2012). Advances in intelligent systems selected papers from 2012 International Conference on Control Systems (ICCS 2012), March 1 - 2, Hong Kong. Berlin Heidelberg Springer.

Liu, S., Jiang, H., Chen, S., Ye, J., He, R. & Sun, Z. (2020). Integrating Dijkstra's algorithm into deep inverse reinforcement learning for food delivery route planning. *Transportation Research Part E: Logistics and Transportation Review*, 142, 102070.

Luna, R. & Bekris, K.E. (2011). Efficient and complete centralized multi-robot path planning. 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems.

Luo, C., Espinosa, A.P., De Gloria, A. & Sgherri, R. (2011). Air-ground multi-agent robot team coordination. 2011 IEEE International Conference on Robotics and Automation.

Ma, J., Liu, Y., Zang, S. & Wang, L. (2020). Robot Path Planning Based on Genetic Algorithm Fused with Continuous Bezier Optimization. *Computational Intelligence and Neuroscience*, 2020, 1–10.

Mai, S. & Mostaghim, S. (2020). Modeling Pathfinding for Swarm Robotics. *Lecture Notes in Computer Science*, 190–202.

Mathew, G.E. (2015). Direction Based Heuristic for Pathfinding in Video Games. *Procedia Computer Science*, 47, 262–271.

Mocholi, J.A., Jaen, J., Catala, A. & Navarro, E. (2010). An emotionally biased ant colony algorithm for pathfinding in games. *Expert Systems with Applications*, 37(7), 4921–4927.

Marcos, E.P., Caizzone, S., Konovaltsev, A., Cuntz, M., Elmarissi, W., Yinusa, K. & Meurer, M. (2018). Interference awareness and characterization for GNSS maritime applications. 2018 IEEE/ION Position, Location, and Navigation Symposium (PLANS).

- Mavrovouniotis, M. & Yang, S. (2010). A memetic ant colony optimization algorithm for the dynamic traveling salesman problem. *Soft Computing*, 15(7), 1405–1425.
- Mencía, C., Sierra, M.R. & Varela, R. (2013). Intensified iterative deepening A* with application to job shop scheduling. *Journal of Intelligent Manufacturing*, 25(6), 1245–1255.
- Miller, B.N. & Ranum, D.L. (2011). *Problem solving with algorithms and data structures using Python*. Sherwood, Or.: Franklin, Beedle & Associates.
- Mirahadi, F. & McCabe, B.Y. (2021). EvacuSafe: A real-time model for building evacuation based on Dijkstra's algorithm. *Journal of Building Engineering*, 34, 101687.
- Molnar, C. (2019). *Interpretable machine learning: a guide for making Black Box Models interpretable*. Morisville, North Carolina: Lulu.
- Nagib, G. & Gharieb, W. (2004). Path planning for a mobile robot using genetic algorithms. *International Conference on Electrical, Electronic and Computer Engineering, 2004. ICEEC '04*.
- Nalepa, J. (2020). *Smart delivery systems: solving complex vehicle routing problems*. Amsterdam, Netherlands Elsevier.
- Navone, E.C. (2020). Dijkstra's Shortest Path Algorithm - A Detailed and Visual Introduction. [online] Free Code Camp. Available at: <https://www.freecodecamp.org/news/dijkstras-shortest-path-algorithm-visual-introduction/>.
- Neapolitan, R.E. (2015). *Foundations of algorithms*. Burlington, Mass.: Jones Et Bartlett.
- Needham, M. & Hodler, A.E. (2019). *Graph algorithms: practical examples in Apache Spark and Neo4j*. Beijing O'Reilly.

Nelson, A.L., Barlow, G.J. & Doitsidis, L. (2009). Fitness functions in evolutionary robotics: A survey and analysis. *Robotics and Autonomous Systems*, 57(4), 345–370.

Okereke, C., Mohamad, M.M. & Wahab, N.H.A. (2020). A Review of Machine Learning Path Planning Algorithms for Autonomous Underwater Vehicles (AUV) in Internet of Underwater Things (IoUT). In: *The 12th International Conference on Internet (ICONI 2020)* At Jeju Shinhwa World, Korea.

Oliveira, C.A.S. & Pardalos, P.M. (2014). *Mathematical aspects of network routing optimization*. New York: Springer.

Oppermann, A. (2020). Artificial Intelligence vs. . Machine Learning vs. . Deep Learning. [online] Medium. Available at: <https://towardsdatascience.com/artificial-intelligence-vs.-machine-learning-vs.-deep-learning-2210ba8cc4ac> . [Accessed 12 May 2021].

Padamkar, P. (2020). Iterative Deepening Depth-First Search | Advantages and Disadvantages. [online] EDUCBA. Available at: <https://www.educba.com/iterative-deepening-depth-first-search/>. [Accessed 15 July 2021].

Perkins, S., Marais, P., Gain, J. & Berman, M. (2012). Field D* path-finding on weighted triangulated and tetrahedral meshes. *Autonomous Agents and Multi-Agent Systems*, 26(3), 354–388.

Ping, K. & Shuai, L. (2013). A brief introduction of an improved A* search algorithm. *2013 10th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*.

Qi, E., Shen, J. & Dou, R. (2013). *The 19th International Conference on Industrial Engineering and Engineering Management*. Berlin, Heidelberg Springer.

Qian, X. & Zhong, X. (2019). Optimal individualized multimedia tourism route planning based on ant colony algorithms and large data hidden mining. *Multimedia Tools and Applications*, 78(15), 22099–22108.

- Raedt, L.D., Bessiere, C. & Dubois, D. (2012). ECAI 2012: 20th European conference on artificial intelligence, 27-31 August 2012, Montpellier, France including Prestigious applications of artificial intelligence (PAIS-2012) systems demonstrations track. Amsterdam: Ios Press.
- Rafiq, A., Kadir, T.A.A. & Ihsan, S.N. (2020). Pathfinding Algorithms in Game Development. IOP Conference Series: Materials Science and Engineering, 769, 012021.
- Raheem, F.A. & Hameed, U.I. (2018). Path Planning Algorithm using D* Heuristic Method Based on PSO in Dynamic Environment. American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS), 49(1), 257–271.
- Rakhee & Srinivas, M.B. (2016). Cluster Based Energy Efficient Routing Protocol Using ANT Colony Optimization and Breadth First Search. Procedia Computer Science, 89, 124–133.
- Riansanti, O., Ihsan, M. & Suhaimi, D. (2018). Connectivity algorithm with depth first search (DFS) on simple graphs. Journal of Physics: Conference Series, 948, 012065.
- Rosita, Y.D., Rosyida, E.E. & Rudiyanto, M.A. (2019). Implementation of Dijkstra Algorithm and Multi-Criteria Decision-Making for Optimal Route Distribution. Procedia Computer Science, 161, 378–385.
- Rosenberg, J. (2017). Security in embedded systems. Rugged Embedded Systems, pp.149–205.
- Rothlauf, F. (2013). Design of modern heuristics: principles and application. Berlin: Springer.
- Roy, B. (2019). A-Star (A*) Search Algorithm. [online] Medium. Available at: <https://towardsdatascience.com/a-star-a-search-algorithm-eb495fb156bb> [Accessed 24 April 2021].
- Russell, S.J. & Norvig, P. (2016). Artificial intelligence: a modern approach. Upper Saddle River: Pearson.

Saranya, C., Unnikrishnan, M., Ali, S.A., Sheela, D.S. & Lalithambika, Dr.V.R. (2016). Terrain Based D* Algorithm for Path Planning. *IFAC-PapersOnLine*, 49(1), 178–182.

Sarkar, R., Barman, D. & Chowdhury, N. (2020). Domain knowledge based genetic algorithms for mobile robot path planning having single and multiple targets. *Journal of King Saud University - Computer and Information Sciences*.

Sharon, G., Stern, R., Felner, A. & Sturtevant, N.R. (2015). Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219, 40–66.

Sharon, G., Stern, R., Goldenberg, M. & Felner, A. (2013). The increasing cost tree search for optimal multi-agent pathfinding. *Artificial Intelligence*, 195, 470–495.

Shin, Y.W., Abebe, M., Noh, Y., Lee, S., Lee, I., Kim, D., Bae, J. & Kim, K.C. (2020). Near-Optimal Weather Routing by Using Improved A* Algorithm. *Applied Sciences*, 10(17), 6010.

Sidhu, H.K. (2020). Performance Evaluation of Pathfinding Algorithms. [online] Available at:
<https://scholar.uwindsor.ca/cgi/viewcontent.cgi?article=9230&context=etd> .
[Accessed 13 June 2021].

Sipper, M., (2011), *Evolved to win*, Lulu Publishing.

Skiena, S.S. (2021). *Algorithm Design Manual*. S.L.: Springer Nature.

Standley, T. & Korf, R. (2011). Complete Algorithms for Cooperative Pathfinding Problems. In: *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*.

Stentz, A. (1995). The focused D* algorithm for real-time replanning. In: *IJCAI'95: Proceedings of the 14th international joint conference on Artificial intelligence*. 1652–1659.

Sedgewick, R. & Wayne, K. (2016). *Algorithms*. Boston: Addison-Wesley.

Shalev-Shwartz, S. & Ben-David, S. (2014). *Understanding machine learning: from theory to algorithms*. New York, NY, USA: Cambridge University Press.

- Snoeck, A., Merchán, D. & Winkenbach, M. (2020). Route learning: a machine learning-based approach to infer constrained customers in delivery routes. *Transportation Research Procedia*, 46, 229–236.
- Shu-Xi, W. (2012). The Improved Dijkstra's Shortest Path Algorithm and Its Application. *Procedia Engineering*, 29, 1186–1190.
- Srikant, R. & Ying, L. (2014). *Communication networks: an optimization, control and stochastic networks perspective*. Cambridge Etc.: Cambridge U.P.
- Talukder, A.K., Garcia, N.M. & Jayateertha, G.M. (2014). *Convergence through all IP networks*. Singapore: Pan Stanford Publishing.
- Tsatcha, D., Saux, É. & Claramunt, C. (2014). A bidirectional path-finding algorithm and data structure for maritime routing. *International Journal of Geographical Information Science*, 28(7), 1355–1377.
- Tu, J. & Yang, S.X. (2003). Genetic algorithm based path planning for a mobile robot. 2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422).
- Uthus, D.C., Riddle, P.J. & Guesgen, H.W. (2011). Solving the traveling tournament problem with iterative-deepening A*. *Journal of Scheduling*, 15(5), 601–614.
- Upadhyay, A., Shrimali, K.R. & Shukla, A. (2018). UAV-Robot Relationship for Coordination of Robots on a Collision Free Path. *Procedia Computer Science*, 133, 424–431.
- Venkat, R.J. (2014). PATH FINDING - Dijkstra's Algorithm. [online] Available at: <http://cs.indstate.edu/~rjaliparthive/dijkstras.pdf>. [Accessed 12 July 2021].
- Wang, H., Mao, W. & Eriksson, L. (2019). A Three-Dimensional Dijkstra's algorithm for multi-objective ship voyage optimization. *Ocean Engineering*, 186, 106131.
- Wang, A.Q., Hao, B.Y. & Cao, C.J. (2020). Deepening the IDA* Algorithm for Knowledge Graph Reasoning through Neural Network Architecture. *Neurocomputing*.

- Wang, Y., Chen, J., Ning, W., Yu, H., Lin, S., Wang, Z., Pang, G. & Chen, C. (2020). A time-sensitive network scheduling algorithm based on improved ant colony optimization. *Alexandria Engineering Journal*.
- Wen, Z. & Cai, Z. (2006). Global path planning approach based on ant colony optimization algorithm. *Journal of Central South University of Technology*, 13(6), 707–712.
- Wei, C., Hindriks, K.V. & Jonker, C.M. (2015). Altruistic coordination for multi-robot cooperative pathfinding. *Applied Intelligence*, 44(2), 269–281.
- Wang, S.-L., Li, X.-L. & Fang, J. (2012). Finding minimum gene subsets with heuristic breadth-first search algorithm for robust tumor classification. *BMC Bioinformatics*, 13(1), 178.
- Woo, M.H., Lee, S.-H. & Cha, H.M. (2018). A study on the optimal route design considering time of mobile robot using recurrent neural network and reinforcement learning. *Journal of Mechanical Science and Technology*, 32(10), 4933–4939.
- Xin, J., Zhong, J., Yang, F., Cui, Y. & Sheng, J. (2019). An Improved Genetic Algorithm for Path-Planning of Unmanned Surface Vehicle. *Sensors*, 19(11), 2640.
- Yannakakis, G.N. & Togelius, J. (2018). *Artificial intelligence and games*. Cham, Switzerland Springer.
- Ye, W., Ma, D. & Fan, H. (2005). Algorithm for Low Altitude Penetration Aircraft Path Planning with Improved Ant Colony Algorithm. *Chinese Journal of Aeronautics*, 18(4), 304–309.
- Yu, J., Su, Y. & Liao, Y. (2020). The Path Planning of Mobile Robot by Neural Networks and Hierarchical Reinforcement Learning. *Frontiers in Neurorobotics*, 14.
- Yu, Y., Wang, J., Xue, X. & Zou, N. (2019). Route Navigation System with A-Star Algorithm in Underground Garage Based on Visible Light Communication. *Lecture Notes in Electrical Engineering*, 1100–1110.

Zhang, F., Lin, H., Zhai, J., Cheng, J., Xiang, D., Li, J., Chai, Y. & Du, X. (2018). An adaptive breadth-first search algorithm on integrated architectures. *The Journal of Supercomputing*, 74(11), 6135–6155.

Zhang, S. & Zhang, Y. (2018). A Hybrid Genetic and Ant Colony Algorithm for Finding the Shortest Path in Dynamic Traffic Networks. *Automatic Control and Computer Sciences*, 52(1), 67–76.

Zhou, R. & Hansen, E.A. (2006). Breadth-first heuristic search. *Artificial Intelligence*, 170(4-5), 385–408.

Zidane, I.M. & Ibrahim, K. (2017). Wavefront and A-Star Algorithms for Mobile Robot Path Planning. *Proceedings of the International Conference on Advanced Intelligent Systems and Informatics 2017*, 69–80.

Zhangqi, W., Xiaoguang, Z. & Qingyao, H. (2011). Mobile Robot Path Planning based on Parameter Optimization Ant Colony Algorithm. *Procedia Engineering*, 15, 2738–2741.

Zhishui, Z. (2011). Ant Colony Algorithm Based on Path Planning for Mobile Agent Migration. *Procedia Engineering*, 23, 1–8.

Zingaro, D. (2021). *Algorithmic Thinking*. No Starch Press, US.

Allison Kirsop (2021) <https://scienceskills.thinkific.com/> [Accessed 23 October 2021].

Appendixes

This chapter comprises of several tables and figures which were quite extensive to be included in the main text of the thesis, therefore allocated here for easier viewing and referral of the reader.

Appendix I

Route Planning Criteria (Source: Author, 2021)

Priority Order by 1 most important to 11 least	Type of Criteria	Description
1	Verification of Official ENC status	1) ENCs should comply with the latest IHO (International Hydrographic Organization), IMO (International Maritime Organization), IEC (International Electrotechnical Committee), and ISO (International Standards Organization) standards.
2	CATZOC (Category of Zone of Confidence)	CATZOC (Category of Zone of Confidence) ranking in relation to scale, i.e., if an A1 ENC is available on a small scale while a large-scale C, D, or U ENC is available in the interested area, the most accurate will prevail. On the contrary, if an A2 or A3 ENC is available on a large scale while a small scale A1 ENC is available in the interested area, the one with the largest scale will prevail, but the object search results will be cross-

		checked against the A1 small scale ENC (see below figure 17, 18 and 19).
3	Navigational dangers	Such can be wrecks, isolated dangers, areas with charted depth less than safely navigable considering CATZOC Depth Accuracy (see below figure 17, 18, and 19).
4	Navigational Warnings	Temporary and Preliminary (T&P), Navtex and INM-C (Inmarsat-C) EGC (Enhanced General Calling) Navigational Notices, which can be sourced directly from the issuing authorities' websites.
5	Vessel's Static draft	Vessel's draft without the influence of Squat or environmental factors such as rolling because of swell, etc. (see below figure 17).
6	Size and class of Ship	Some vessels might have displacement or breadth restriction in certain navigable areas, such as Kamsarmax bulk carriers.
7	Under Keel Clearance (UKC)	According to OCIMF (Oil Companies International Maritime Forum) SIRE (Ship Inspection Report) Vessel Inspection Questionnaire, 2019 p. 30, Vessel's UKC (Under Keel Clearance), which can be affected by several factors and the Underkeel calculations should include, but not necessarily be limited to (see below figure 17): i) The

		<p>predicted height of the tide (see below figure 17), ii) Changes in the predicted tidal height, which are caused by wind speed and direction and high or low barometric pressure, iii) Nature and stability of the bottom - i.e., sand waves, siltation, etc., iv) Change of water density and the increase in draught due to freshwater allowance, v) The vessel's size and handling characteristics and increase in draught due to heel, vi) Wave response allowance, which is the vertical displacement of the hull due to heave, roll and pitch motions, vii) The reliability of draft observations and calculations, including estimates of hogging and sagging, viii) Reduced depths over pipelines and other obstructions, iv) vessel's Squat.</p>
8	Vessel's Air Draft	It might be obstructed by Bridges and Hanging Cables (see below figure 18).
9	Vessel's maneuvering characteristics	Such can be the vessel's turning circle and Wheel over line speed depending on if the ship is sailing in coastal waters with dense traffic or in open waters.
10	Wind direction/speed, Wave/Swell direction/height	These can be according to user-determined limits and taking into consideration the vessel's GOM (Ships Corrected Metacentric Height) and

	and Current direction/speed	therefore Rolling Period, especially when dealing with tall ships such as Containerships, RoRo (Roll On Roll Off Carriers) or Cruise ships and to minimize vessel bunker consumption by selecting the optimum route.
11	COLREGS (Collision Regulations)	Taking them into consideration when designing routes in case such as when crossing Traffic Separation Schemes, when navigating in TSS and when a vessel can navigate at an inshore traffic zone or any other applicable instance.

Appendix II

List of the individual parts, their associated part numbers, and ISO 19100 Conformance standards (Source: IHO,2015)

Part Title	Part Number	ISO19100 Standard
Conceptual Schema Language S-100	Part 1	ISO 19103:2005, Geographic information - Conceptual schema language ISO
Management of IHO Geospatial Information Registers S-100	Part 2	ISO 19135:2005, Geographic Information - Procedures for registration of items of geographic information
Feature Concept Dictionary Registers S-100	Part 2a	ISO 19135:2005, Geographic Information - Procedures for registration of items of geographic information ISO 19126:2009, Geographic Information – Feature concept dictionaries and registers
Portrayal Register S-100	Part 2b	ISO 19135:2005, Geographic Information - Procedures for registration of items of geographic information ISO 19126:2009, Geographic Information – Feature concept dictionaries and registers ISO 19117:2012, Geographic Information - Portrayal
General Feature Model and Rules for Application Schema S-100	Part 3	ISO 19109:2005, Geographic information - Rules for application schema
Metadata S-100	Part 4a	ISO 19115:2005, Geographic information - Metadata
Metadata for Imagery and Gridded Data S-100	Part 4b	ISO 19115:2005, Geographic information - Metadata
Metadata – Data Quality S-100	Part 4c	ISO 19113, Geographic information - Quality principles ISO 19114, Geographic information - Quality evaluation procedures ISO 19138, Geographic information - Quality measures
Feature Catalogue S-100	Part 5	ISO 19110:2005, Geographic Information - Methodology for feature cataloguing
Coordinate Reference Systems S-100	Part 6	ISO 19111:2007, Geographic information - Spatial referencing by coordinates
Spatial Schema S-100	Part 7	ISO 19107:2003, Geographic information - Spatial schema
Imagery and Gridded Data S-100	Part 8	ISO 19123:2007, Geographic information - Schema for coverage geometry and functions ISO 19129, Geographic information - Imagery, Gridded and Coverage Data Framework
Portrayal S-100	Part 9	
Encoding Formats S-100	Part 10	
ISO/IEC 8211 Encoding S-100	Part 10a	ISO/IEC 8211:1994, Specification for a data descriptive file for information interchange structure implementations
GML Encoding S-100	Part 10b	ISO 19136:2007 Geographic information - Geography Markup Language
HDF5 Encoding S-100	Part 10c	HDF5 Data Model and File Format
Product Specifications S-100	Part 11	ISO 19131:2008 Geographic information – Data product specifications
S-100 Maintenance Procedures S-100	Part 12	

Appendix III

Table of ENC Layers types depending on Issuing Authority and purpose of use (Source: Author, 2021)

ENC Layers types depending on Issuing Authority and purpose of use				
IHO	IALA	Intergovernmental Oceanographic Commission	Other Organizations	International Electrotechnical Commission
From S-101 to 199	From S-201 to 299	From S-301 to 399	From S-401 to 420	From S-421
S-101 ENC	S-201 Aids to Navigation (AtoN)	<u>None at the moment</u>	S-401 Inland ENC	S-421 Route Plan
S-102 Bathymetric Surface	S-210 Inter-VTS Exchange Format		S-411 Ice Information	
S-103 Sub-Surface Navigation	S-230 Application Specific Messages		S-412 Weather / Met-ocean forecasts Overlay	
S-104 Water Level Information for Surface Navigation	S-240 DGNSS Station Almanac			
S-111 Surface Currents	S-245 Loran ASF Data			
S-112 Dynamic Water Level Data Transfer	S-246 eLoran Station Almanac			
S-121 Maritime Limits and Boundaries	S-247 Differential e-Loran Reference Station Almanac			

Appendixes

S-122 Marine Protected Areas				
S-123 Radio Services				
S-124 Navigational Warnings				
S-125 Navigational Services				
S-126 Physical Environment				
S-127 Traffic Management				
S-128 Catalogues of Nautical Products				
S-129 Under Keel Clearance Manager				

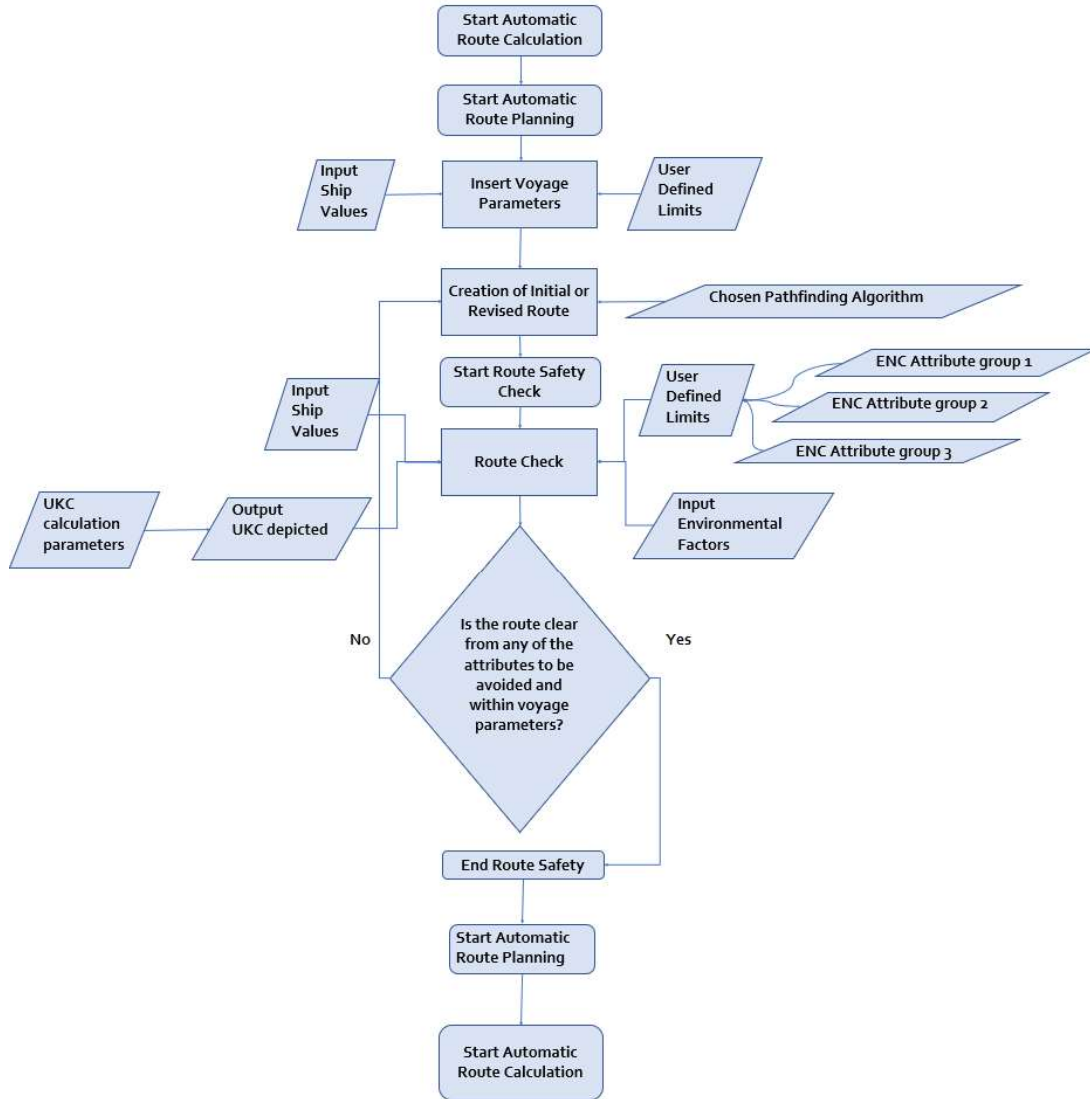
Appendix IV

Table of Comparison of Pathfinding / Wayfinding Algorithms (Source: Author, 2021)

Pathfinding algorithm	Functionality	Advantages	Disadvantages
Dijkstra algorithm	Allows a user to find the shortest path from one source to all points in a graph simultaneously, hence being termed as a	Guaranteed to find the shortest path Efficient in solving relatively large and complex problems Low complexity – the algorithm is almost linear	Undertakes a blind search hence consuming a lot of time and resources Unable to handle negative edges leading to acyclic graphs where it is unable to identify the shortest path
A* algorithm	Its main aim is to search for shorter instead of longer paths between the start and end nodes.	Completeness and optimality Optimally efficient as no other optimal algorithm is guaranteed to expand fewer nodes than A*	Memory intensive as it is exhausted since nodes are stored in memory The algorithm's completeness depends on whether the branching factor is finite and where all actions have a fixed cost. Accuracy of heuristic function $h(n)$ influences the speed of execution of the A* algorithm
Iterative Deepening A* algorithm (IDA*)	Utilized in finding the shortest path between the start nodes and any member of a set of goal nodes in a weighted graph.	Completeness and optimality Heuristics are added to enhance the speed of the process Utilizes less memory Diverse heuristics can be integrated into the algorithm without changing the basic code	Unable to keep track of visited nodes and, as a result, explores such nodes again. Slower as it repeats the exploration of visited nodes. Requires more processing time and power than A*
Dynamic A* algorithm (DA*)	An informed search algorithm which relies on an evaluation and heuristic function to undertake search or path planning activity	Generate shortest paths in a graph in real-time with change or updates to arc costs. Completeness and optimality Helpful in solving highly complex problems Optimally efficient	Where a finite branch is considered, and each action has a fixed cost, the algorithm is likely to fail Challenged by complexity problems The execution speed of A* depends on the accuracy of the heuristic function.
Weighted A* algorithm	A performance-acceleration extension of the A* algorithm whereby the heuristic	Completeness and optimality Provides better performance than A*	Complexity problems
Lifelong Planning A* algorithm (LPA*)	An incremental version of A* identifies the shortest paths from a given source node vertex to a goal vertex while edge costs of the graph are added or deleted.	Completeness and optimality Provides better performance than A*	Complexity problems
Genetic algorithms	Optimization algorithms facilitate the search for potential solutions in a given space and are inspired by natural evolutionary principles advocated by	Able to solve complex problems within a short time period. Able to explore the search space in parallel, thereby not requiring the function to be optimized to be differentiable or have smooth properties.	Limited as they face the exploration-exploitation dilemma where a significant problem faced regards the capability to select the initial population.
Ant Colony algorithms	They are algorithms that mimic natural ants' behaviour and are delineated by a local strategy that guides the movement of agents in the search space.	Can undertake a search among the population in parallel Able to discover optimal solutions Able to adapt to new changes such as distance Has a guaranteed convergence	Difficult to undertake theoretical analysis Uncertain time for convergence Dependent sequences of random decisions Probability is likely to change for each iteration
Breadth-First Search (BFS)	One starts at the selected node (source) and traverses the graph in a layerwise approach, thereby exploring neighbour nodes directly connected to the source node. After that, the traversal moves onto	Able to find the shortest path between the starting point and any other node reachable in the graph. Simple to implement as it involves a small number of steps. Complete and optimal Do not suffer from any potential infinite loop problems.	It is a blind search and, as such, delivers poor performance where the search space is large. Secondly, the algorithm consumes significant memory compared to alternatives such as the depth-first search (DFS).
Depth-First Search (DFS)	The algorithm traverses the graph from the root node and explores the graph as far as possible along each branch (for a tree graph) before backtracking. As a result, all nodes on the current path will	It uses less memory and resources as it maintains only a list of nodes from the root node to the current node. The algorithm also takes less time to reach the goal node than other algorithms that adopt a similar approach, for instance, the BFS.	There is a likelihood of an infinite loop occurring as the algorithm searches deep down the graph. The algorithm does not guarantee to find the optimal solution as many states of the node keeps re-occurring.
Iterative Deepening Depth First Search (IDDFS)	The algorithm first undertakes a limited depth-first search up to a particular depth that is limited. The depth limit is incremented by iterating the procedure unless the goal node is identified or the entire tree is traversed.	Complete as it can identify a solution provided it exists in the graph. The algorithm is efficient where solutions are identified at a depth of the tree. The algorithm has quick responsiveness as it generates early results that are refined multiple times through different iterations. The algorithm is also highly efficient in in-game tree search as it improves depth definition, heuristics, and scores of search nodes.	It is considered wasteful due to the time and wasted calculations that occur at the same depth. Failure of the BFS algorithm leads to eventual failure of the IDDFS
Bi-directional Search	Search begins from both directions simultaneously, whereby, one algorithm begins the search from the initial node while the second begins from the goal vertex. As a result, the two algorithms intersect somewhere at the middle of the graph, with the path that traverses from	Fast exploration time Reduced memory and resource utilization The algorithm is also optimal and complete	A user needs to be aware of the goal state to be attained in the graph search, thereby decreasing its use cases significantly. The complexity in implementation of the code and instructions. It is not possible to perform a backward search through all states. There is also a need to ensure the algorithm is robust enough to understand the intersection and where the search should come to an end to avoid the possibility of an infinite loop.
Uniform Cost Search	A brute-force algorithm which is used to find solutions in a directed weighted graph with a minimum cumulative cost.	It helps identify the lowest cumulative cost path in a weighted graph from the source to the root node. The solutions derived from the algorithm are also complete and optimal for each state.	High storage consumption The likelihood of the algorithm being stuck in an infinite loop as it derives possible paths from the root to destination nodes. The algorithm has to maintain a sorted open list in the priority queue
Breadth-First Search with Heuristic Function	An algorithm that expands the nodes of a graph in breadth-first order employs a heuristic function when pruning the search space.	Complete as it can identify a solution provided it exists in the graph. Heuristics are added to enhance the speed of the process.	Complexity problems. Memory intensive as it is exhausted since nodes are stored in memory

Appendix V

Autonomous Route Calculation Flowchart (Source: Author, 2021)



List of Tables

This chapter comprises of the list of tables contained in this Thesis as follows:

Table 1 List of Dynamic and Static Elements affecting Passage Planning, 2021 (Source: Author, 2021),

Table 2 ENC Global Coverage statistics 2008-2017,2018 (Source: UKHO, 2018)

Table 3 ENC Usage Bands, 2019 (Source: NOAA, 2019)

Table 4 Phases of S-100 ENC Product Specification Assembly, 2015 (Source: IHO, 2015)

Table 5 Table of Interoperability Catalogue (IC) Levels, 2018 (Source: KHOA, 2018)

Table 6 Safety Check ENC Attributes and ENC group they belong (Author, 2021)

Table 7 UKC recommended values in relation to CATZOC Depth and Position Accuracy (NOAA, 2019)

List of Figures

This chapter comprises of the list of figures contained in this Thesis as follows:

- Figure 1 Routeing Publications Covers and Admiralty Chart Catalogue (UKHO, 2020)
- Figure 2 Using a touchscreen Back of Bridge Navigation software for planning routes (Navtor, 2015)
- Figure 3 Modern Weather Routeing Software showing the weather affecting a specific route (SPOS, 2016)
- Figure 4 A graphic ENC Cell Catalogue for a specific area as shown in all modern ECDIS (ESRI, 2015)
- Figure 5 S-57 ENC Raw Data display in Arc Map Maritime Module during construction of an ENC. (ESRI, 2015)
- Figure 6 All Year 2019 Traffic Density Map (MarineTraffic, 2021)
- Figure 7 Handysize Bulk Carrier Year 2019 Traffic Density Map (MarineTraffic, 2021)
- Figure 8 Handymax Bulk Carrier Year 2019 Traffic Density Map (MarineTraffic, 2021)
- Figure 9 Capesize+ Bulk Carrier Year 2019 Traffic Density Map (MarineTraffic, 2021)
- Figure 10 Large Container Ships >3,000 GT Year 2019 Traffic Density Map (MarineTraffic, 2021)
- Figure 11 Handymax Bulk Carrier Traffic (green) overlaid on top of Handysize Tankers (pink) Year 2019 Traffic Density Map (MarineTraffic, 2021)
- Figure 12 Panamax Bulk Carrier Traffic (green) overlaid on top of Handysize Tankers (pink) Year 2019 Traffic Density Map (MarineTraffic, 2021)
- Figure 13 Capesize+ Bulk Carrier Traffic (green) overlaid on top of Handysize Tankers (pink) Year 2019 Traffic Density Map (MarineTraffic, 2021)
- Figure 14 Capesize + Bulk Carrier Traffic (green) overlaid on top of Aframax /LR2 + Tankers (pink) Year 2019 Traffic Density Map (MarineTraffic, 2021)
- Figure 15 Capesize + Bulk Carrier Traffic (green) overlaid on top of Aframax /LR2 + Tankers (pink) vs. Large Container Ships >3,000 GT (yellow) Year 2019 Traffic Density Map (MarineTraffic, 2021)
- Figure 16 S-100 Main ENC layers (KHOA, 2017)
- Figure 17 UKC calculation considering all environmental and ENC factors (AWP Marine Consultancy Ltd, 2017)
- Figure 18 Standardized Depth Terminology (UKHO, 2015)
- Figure 19 Paper Chart Source Data reference example (UKHO, 2015)

Figure 20 Constructing a great circle track on a Mercator projection (National Geospatial Intelligence Agency, 2019.)

Figure 21 The World Sailing Ship Routes (National Geospatial Intelligence Agency, 2019)

Figure 22 Policy & Governance, Technical Standards, Information Systems and Geographic Content comprising the Four Pillars of MSDI, (IHO,2017)

Figure 23 S-100 prototype ECDIS displaying S-101 electronic navigational chart in conjunction with S-129 Underkeel Clearance management data. (NOAA, 2019)

Figure 24 IHO S-100 Components and their Associated ISO Standards (Robert Ward, Lee Alexander, Barrie Greenslade, Anthony Pharaoh, 2008)

Figure 25 S-100 and S-101 Implementation Plan Phases (KHOA, 2017)

Figure 26 S-100 and S-101 Implementation Plan Timeline (KHOA, 2018)

Figure 27 Visual Portrayal of S-100 ENC Types and Uses (KHOA, 2018)

Figure 28 Portrayal of S-101 ENC various object layers (KHOA, 2018)

Figure 29 S-10x ENC layers used for the Demonstration (KHOA, 2018)

Figure 30 Level 0 final depiction of the S-10x ENCs (KHOA, 2018)

Figure 31 Level 1 final depiction of the S-10x ENCs (KHOA, 2018)

Figure 32 Level 1 and Level 2 final depiction of the S-10x ENCs (KHOA, 2018)

Figure 33 Solving for the shortest path using the Dijkstra algorithm (Navone, 2020)

Figure 34. Illustration of BFS algorithm (Neapolitan, 2015)

Figure 35. Flowchart of the BFS algorithm (Hurbans, 2020)

Figure 36. IDDFS graph with 4 depths (Pedamkar, 2020)

Figure 37. IDDFS graph traversal (Pedamkar, 2020)

Figure 38. Bidirectional search algorithm (Russell and Norvig, 2016)

Figure 39. Uniform cost search weighted graph (Jagga, 2020)

Figure 40 A* search algorithm example of finding the optimum route considering various obstacles and safe distance from them (Author, 2021)

Figure 41 A* search algorithm example of finding the route in a cell pattern with obstacles (Author, 2021)

Figure 42 Example of a Route Safety Check by ECDIS using S-57 Attributes (ECDIS screenshot taken by author onboard ship, 2013)

Figure 43 Example of how the Safety Contour is used in an ENC (NOAA, 2019)

Figure 44 Potential pitch of a vessel entering the Port of Long Beach. (PORT OF LONG BEACH PRECISION NAVIGATION PROJECT, NOAA, 2017)

Figure 45 Import vs. Export Tonnage Chart in the 20 Top Ports in the US according to Precision Marine Navigation (PMN) Socio-Economic Study (NOAA, 2020)

Figure 46 Foreign vs. Domestic Tonnage Chart in the Top 20 Ports in the U.S. according to Precision Marine Navigation (PMN) Socio-Economic Study (NOAA, 2020)

Figure 47 Incidents within 3 km of port Chart in the Top 20 Ports in the U.S. according to Precision Marine Navigation (PMN) Socio-Economic Study (NOAA, 2020)

Figure 48 Ever Given grounded in Suez (modified Copernicus Sentinel data [2021], processed by Pierre Markuse March 24th, 2021)

Figure 49. Relationship between AI, machine, and deep learning (Oppermann, 2020)

Figure 50 Deep neural network (Oppermann, 2020)

Figure 51. Difference between machine and deep learning (Oppermann, 2020)

Figure 52 GNSS threats and vulnerabilities (Felski and Zwolak, 2020)