HEURISTIC METHODS FOR WEATHER ROUTING

by

Pengcheng (Louis) Bu

Submitted in partial fulfillment of the requirements
for the degree of Master of Science

at

Dalhousie University
Halifax, Nova Scotia
August 2024

© Copyright by Pengcheng (Louis) Bu, 2024

Table of Contents

List of Tables

List of Figures

Abstract

Weather routing involves optimizing routes for vessels to reduce travel time and fuel consumption by utilizing meteorological data. It is an interdisciplinary problem that requires insights ranging from naval engineering to mathematics.

This thesis proposes three optimization algorithm for weather routing: Hybrid Search, Super A^* Search, and Bézier Evaluation with Evolutionary Strategy (BEES), paired with a discrete Newton-Jacobi smoothing method. Each method leverages different mathematical principles: differential equations, graph theory, and evolutionary algorithms. We will evaluate their performance in various meteorological conditions and analyze their output across different sailing speeds, ocean regions, and times of the year on a common benchmarking platform.

Acknowledgements

This research project was conducted in the Department of Mathematics and Statistics, Faculty of Science, at the Dalhousie University, in collaboration with IE university in Madrid, Spain. I have been collaborating with the brilliant team from Canonical Green since the summer of 2022, the start of my Honours program.

First I would like to express my heartfelt appreciation to my supervisor, Dr. Robert Milson, for his unwavering support and guidance along the way, in both my coursework and research. His patience, expertise, and trust in my abilities have been invaluable to me. I am grateful for the opportunity to work with him and learn from him.

I would also like to thank Dr. David Gómez-Ullate for facilitating a research stay at IE university, his geneorosity and hospitality were essential to my stay in Madrid. His insightful guidance, dedication towards the project, and invaluable feedback have been instrumental in the success of this project. I am grateful for his mentorship and support.

During my time there, I worked with a brilliant team of scientists, including Dr. Daniel Precioso, Javier Jiménez, Dr. Rafael Ballester. The project and this thesis would not be possible without their collaboration, support, and patience. I would like to thank them for having me on the team, and becoming close friends.

I would like to thank MITACS Accelerate International Program (Ref. IT26380) for funding my research travels and stay in Madrid, Spain. The financial support from the Faculty of Graduate Studies, Dalhousie University has been essential to my journey as a graduate student. TAships from the department of Mathematics and Statistics were influential to my development as a teacher and a researcher.

Finally, I would also like to express my appreciation to all the friends and colleagues I have met along the way, who have contributed to my growth and learning, in the past 2 years. Thank you for your support, encouragement, and inspiration.

Pengcheng (Louis) Bu

Halifax, Canada, 2024

Chapter 1

Introduction

1.1 Motivation

The major driving factor of weather routing is **reducing emissions**. Maritime transport handles approximately 80% of the world's trade by volume [1]. 'The Fourth International Maritime Organization GHG Study' estimated that shipping vessels emitted 1.06 billion metric tonnes of CO_2 in 2018, representing 2.89% of total global anthropogenic CO_2 emissions [2]. If this is left unchecked, this number is bound to increase [3], [4]. The road plan includes reducing the CO_2 emissions per transport work of international shipping by at least 40% by 2030 and reducing total annual GHG emissions from international shipping by at least 50% by 2050. Ultimately, the plan aims to phase out GHG emissions from international shipping in this century [5].

The **economy** is another contributing factor to the development and research in weather routing. The cost of fuel accounts for up to 60% of the total operation costs of maritime transport, making it a significant concern for the industry. Unit cost of bunker fuel is extremely sensitive to geopolitical developments [6], making it even more challenging for the industry to invest in sustainable technologies. Adverse weather conditions, such as cyclones, high tides, and strong wind gusts, account for approximately 48% of maritime casualties [7], making **safety** a significant concern for the industry.

As such, despite its importance in world economy, the sector faces significant challenges, including rising fuel costs, stringent environmental regulations, and the need for safe navigation [8], [9]. In this context, weather routing appears as an efficient solution to alleviate these problems.

1.2 Weather Routing

In recent years, the field of weather routing has gained significant attention from both academia and industry. This research area addresses the complex mathematical problem of optimising maritime routes under varying weather conditions. Weather routing involves interdisciplinary approaches, encompassing various fields ranging from Mathematics to Computer Science.

The routing problem is about finding efficient routes for ships, considering ocean conditions, and vessel performances. This complex challenge impacts fuel use, emissions, costs, and travel time. The state of the art in weather routing includes several methodological approaches, each contributing unique insights and advantages. One such approach is the **isochrones method** [10], initially applied by Hanssen in 1960 [11] to optimise routes under stationary weather conditions. The isochrones method constructs sets of connected points that a vessel can reach within a specific time by departing from a given point and travelling in all feasible directions. While this method has seen significant developments [12]–[14], it relies on exploration and simulation, making it computationally expensive, especially when considering time-dependent weather conditions.

Another optimisation approach is **variational optimisation**, which utilises a Lagrangian framework to optimise routes in a continuous space using gradient methods. Initially proposed by Haltiner in 1962 [15] to minimise time in static settings, variational optimisation has been applied to weather routing by incorporating factors such as wave height, direction, and ocean current vector fields [16], [17]. This approach is exploitative, iteratively modifying an initial route to reduce a given cost function such as fuel consumption or travel time. However, it heavily relies on the quality of the initial guess to achieve an optimal route.

Dynamic programming has been widely utilised in weather routing, leveraging Bellman's concept of optimality [18]. Dynamic programming divides the routing problem into smaller sub-problems, finding optimal solutions for each segment and chaining them together to obtain the optimal solution for the entire journey. The solution for each segment can be achieved using various optimisation methods or a

set of predefined rules [19], [20].

Additionally, **graph optimisation** techniques, such as Dijkstra's search algorithm [21], [22], have been employed in weather routing. This approach involves constructing a discrete graph that represents possible vessel locations on the ocean as nodes. Edges between nodes are weighted to represent the cost of travelling between them, considering factors such as time or fuel consumption. Graph algorithms then search for the path between two non-adjacent nodes with the minimum total cost. Heuristics, such as the A*search algorithm [23]–[25], can be used to expedite the graph search process.

Moreover, **evolutionary algorithms** have been applied in weather routing, involving the evolution of a set of routes through random perturbations until an optimal criterion is achieved [24], [26]–[28]. Unlike variational optimisation, which employs gradient methods to modify the initial route, evolutionary algorithms rely on random modifications, potentially guided by a set of rules. However, the random nature of evolutionary algorithms does not guarantee optimality and can result in significant computational costs.

Despite the various optimisation approaches utilised in weather routing, achieving standardised comparisons and benchmarks across different studies remains challenging. The definition and measurement of savings, whether in terms of fuel consumption, sailing time, or both, require standardisation for effective comparison. To address this challenge, this study focuses on developing reproducible benchmarks using historical meteorology and oceanography data from open sources. These benchmarks are designed to reflect realistic scenarios along commercial trading routes. We will only focus on reducing sailing time.

The contribution of this study lies in the comprehensive evaluation and comparison of three optimisation methods for weather variational optimisation, graph optimisation, and evolutionary algorithms. By implementing these methods on the defined benchmarks, we aim to gain valuable insights into their strengths, weaknesses, and applicability in different scenarios. This study provides benchmarking examples and facilitates future research and standardisation efforts in the weather routing community.

1.3 Objectives

Research on weather routing typically targets specific scenarios for optimization. It is essential to understand three key concepts. First, an **Origin-Destination Pair (ODP)** identifies the start and end points of a journey, which, in practical applications, are maritime ports. Initially, weather routing assumed ocean conditions were static, making ODPs define complete optimization problems. However, current approaches to weather routing account for changing weather conditions. An **itinerary** builds upon the ODP by adding a specific departure date, and possibly a time. A **route**, then, specifies the path to follow between the origin and destination. In weather routing, a route is a solution for an itinerary based on factors such as safety, efficiency, and optimal utilization of oceanography and weather conditions.

Table 1.1 presents some of the most recent studies in weather routing, detailing the algorithms they used and the ODPs they examined. A closer look at Table 1.1 reveals a significant challenge: each study operates under its unique set of ODPs, complicating the comparison of algorithms on equal footing. On the rare occasions when two studies share an ODP, they differ in departure dates, resulting in distinct itineraries between works.

The lack of standardized itineraries is a big gap in the research field of weather routing, as it has been recently noticed by [29]. When assessing the efficiency of a new methodology, researchers can only score it against basic solutions, such as the orthodromic or geodesic route [30]–[33]. In some rare cases, they have access to recorded data from a vessel journey [35]–[37]. Even when some algorithms are validated against real data, there is still a lack of comparison between different studies. This observation emphasizes the need for establishing standardized itineraries in weather routing research to facilitate more direct comparisons and benchmarking of algorithms. Such standardization could accelerate progress in the field by enabling researchers to build upon a common foundation of scenarios and data sets.

One notable example of the importance of standardized benchmarks can be seen in the field of Natural Language Processing (NLP) with the development of benchmarks for Large Language Models (LLMs). Benchmarks such as the MMLU [38]

Table 1.1: Examples of recent weather routing studies, indicating the algorithms they used and the Origin-Destination Pairs (ODPs) they tested on.

Reference	**Regions and ODPs**	**Algorithm**
[29]	Mediterranean Sea (Porto Torres to Toulon, Monemvasia to Marmaris)	Graph Search VISIR-2
[30]	North Atlantic Ocean (Charleston to Azores), Indian Ocean (Somalia to Myanmar), Caribbean Sea (Cancun to Charleston, Panama to Houston)	Hybrid Search
[31]	Mediterranean Sea (Barcelona to Limassol, Barcelona to Thessaloniki, Barcelona to Alexandria)	Probabilistic roadmaps
[32]	North Atlantic Ocean (Cap Lizard to New York)	Evolutionary algorithm
[25]	North Atlantic Ocean (Boston to Plymouth), Mediterranean Sea (Tunis to Nice, Palma de Mallorca to Barcelona), others	A* graph search
[33]	Indian Ocean (Singapore to Cape Town), Pacific Ocean (Shanghai to Los Angeles)	Particle swarm
[34]	Atlantic Ocean (New York to Paramaribo), Indian Ocean (Cape Town to Mumbai), Mediterranean Sea (Trieste to Alexandria, Algeciras to Alexandria, Rotterdam to Marseille), others	Genetic algorithm
[35]	Pacific Ocean (Taipei to Los Angeles, Tacoma to Kaohsiung)	*Non disclosed*
[36]	North Atlantic Ocean and Mediterranean Sea (Gulf of Guinea to Marseille)	A* graph search
[37]	Atlantic Ocean (Portugal to Azores)	Evolutionary algorithm

and HellaSwag [39] have been essential in the evaluation of LLMs like GPT-3 [40], Mixtral [41], and LLaMA [42]. It has been one of the main components that allowed the popularity and growth of this field and will help improve many aspects, such as reasoning, understanding, and text generation.

Another recent example closer to weather routing is Google's WeatherBench II [43]. It has set a standard for evaluating weather prediction models by offering a common dataset and evaluation metrics. This standardization has advanced weather forecasting by enabling direct comparisons and highlighting the strengths and weaknesses of various models, such as IFS HRES [44], ERA5 Forecast [45], and Graphcast [46] - a new method based on machine learning that outperforms traditional models in some aspects. These kinds of systems also encourage competition, which drives improvements in the field.

One of this paper's contribution is the development of benchmarking standards. We introduce **`WeatherRouting Bench 1.0`**. These benchmarks specify origin and destination ports, and include data on ocean currents and waves from the Copernicus Maritime Service [47], [48]. The necessary Python code for data acquisition is made available in a public repository[1]. Additionally, a website will soon launch, allowing researchers to submit their algorithmic solutions for each benchmark. These submissions will undergo scoring and ranking, offering a platform for comparative analysis against leading methodologies in a standardized setting. Initially, the evaluation will focus on reducing travel times, taking into account only the effect of waves and currents, with plans to later incorporate wind effects and fuel consumption metrics.

Another critical aspect of our contribution addresses the lack of variety in weather conditions across research. Traditional studies often limit their focus to a single departure date per ODP, overlooking the inherent adaptability of weather routing to varying conditions. Our benchmarks propose multiple departure dates for each ODP. This allows for the evaluation of algorithms across 52 different weeks for the same ODP, providing comprehensive insights into their performance and showcasing their full potential.

Finally, serving as the first use-cases of `WeatherRouting Bench 1.0`, this thesis

[1] https://github.com/Weather-Routing-Research/weather-routing-benchmarks

also introduces 3 novel heuristic-based Weather Routing algorithms, Hybrid Search (`Hybrid Search`), Super A*, and Bezier Evaluation on Evolutionary Strategy (BEES). Each one is unique in their own right, `Hybrid Search` [30] solves the Zermelo's Navigation Equations [49] with a shooting method, Super A* applies the popular A* algorithm [25], [29], [36], [50] shortest paths on a hexagonal grid [51], and allows jumping over multiple nodes to increase possible course changes [52], and BEES uses a evolutionary strategy that minimizes costs of batches of Bézier curves. These algorithms offers a good initial guess to a close-to-optimized solution.

In addition, a refinement step is added after `Hybrid Search` and A*, based on the Ferraro-Martín de Diego-Sato algorithm (FMS) [17], [53], which modifies the solution given by the A* using calculus of variations, moving its way-points to a local optimum while visually smoothing the whole route. The FMS is an exploitative algorithm that has been recently applied in conjunction with a shooting method for weather routing [30] and showed great potential. This paper proves that A* provides a good initial solution for the FMS.

These three optimization methods utilizes knowledge from distinct areas of mathematics, including calculus of variations, differential geometry, combinatorics and graph theory, and genetic algorithms; despite this, all these methods uses a heuristic, which gives although not completely optimized but a 'good enough' solution to the weather routing problem. The algorithms are evaluated on the `WeatherRouting Bench 1.0` set, and we analyze their performances across different ODPs, seasons, and vessel speeds.

The paper is structured as follows. First, we introduce all the necessary components of tackling the Weather Routing problem in 2, where we address information required, such as, benchmarks and ODP selection in 2.1, modelling, acquisition, and interpolating weather data in 2.2. Next, we introduce the 3 optimization algorithms in Chapter 3, `Hybrid Search`in Section 3.1, A* in Section 3.2, and BEES in Section 3.3. We also perform a hyperparameter search to ensure we use an efficient configuration of A*. We then briefly explain the FMS algorithm in 3.4. Finally, in section 4, we give a cross comparison on each algorithm's performance on some example routes and study optimization results generated by A*+FMS, analyze the influence

of the benchmark characteristics, such as the ODP, vessel speed, the relative impact of currents and waves, and seasons.

Chapter 2

Materials

2.1 Benchmarks

An *instance* of a weather routing problem is defined by the following components:

- A pair of departure and destination ports (ODP),
- a specified departure date and time,
- vessel characteristics,
- a prescribed cruising speed,
- relevant meteorological information (e.g., waves, currents),
- a cost function that depends on the aforementioned parameters,
- decision variables available to the algorithm to minimize the cost function.

Weather routing algorithms are each designed for a specific kind of optimization problem. A robust optimization algorithm should be capable of handling all optimization problems of a same type, provided the set of available decisions remains unchanged. For `WeatherRouting Bench 1.0`, we assume that vessels sail at a constant speed over water v_{wtr}, implying that the vessel's engine power does not vary along the route.

A more realistic optimization problem would be to minimize fuel consumption subject to a prescribed maximum travel duration, allowing both changes in course and engine power along the route. Future versions of our weather routing benchmark will incorporate more realistic optimization problems and cost functions such as the one mentioned above, and possibly extending it also to safety parameters. For this first version, we believe the proposed optimization problem is already rich enough to constitute a valid starting point.

2.1.1 Cost function

As mentioned above, we assume that vessels travel at a prescribed speed over (calm) water, but their actual speed with respect to ground is affected by ocean waves and currents. The total travel time will depend thus on these navigation conditions and the chosen route, and it will be the cost function we aim to optimize in this first version of `WeatherRouting Bench 1.0`. To calculate this time, we need to understand how environmental factors affect vessel speed, especially speed loss due to wave resistance and changes caused by ocean currents.

First, we will analyze the impact of wave-induced speed reduction. Since the 1970s, both academia and industry have put a lot of effort into studying this effect. Various methods have been developed, ranging from statistical and regression models based on wave tank experiments [54]–[57] to advanced fluid dynamics and potential flow models [58]. Recently, data-driven methods, including Machine Learning and Neural Networks [59]–[61], have further improved our ability to predict wave resistance and the resulting speed losses.

We have chosen the Townsin-Kwon[57] model in Molland's Ship Resistance and Propulsion [62], as it has valid results for a relatively small set of environmental parameters: significant wave height (in meters) and wave incidence angle θ (in degrees). It also depends on the speed over water v_{wtr} (in meters per second), its beam length L (in meters), displacement ∇ (in cubic meters) and block coefficient c_B.

Kwon [56] later updated the Townsin-Kwon equations of percentage speed loss over water:

$$c_w \coloneqq \frac{\Delta v_{\text{wtr}}}{v_{\text{wtr}}} \cdot 100\% = c_\beta c_u \alpha \tag{2.1}$$

where c_β is the weather reduction coefficient, c_u is the speed reduction coefficient, α is the correction factor, and we denote the percentage speed loss by c_w. These coefficients are all unit-less, and they are calculated and adapted from Molland [62] as the following.

The weather reduction coefficient, c_β, depends on the wave incidence angle θ and the Beaufort number BN:

$$2c_\beta = \begin{cases} 2 & 0^\circ \leq \theta < 30^\circ \\ 1.7 - 0.03 \cdot (\mathrm{BN} - 4)^2 & 30^\circ \leq \theta < 60^\circ \\ 0.9 - 0.06 \cdot (\mathrm{BN} - 6)^2 & 60^\circ \leq \theta < 150^\circ \\ 0.4 - 0.03 \cdot (\mathrm{BN} - 8)^2 & 150^\circ \leq \theta \leq 180^\circ \end{cases} \tag{2.2}$$

Beaufort number is a dimensionless quantity usually derived from wind speed [63], but also related to significant wave height (h_{wav} in meters) by using the relation between wind speed and wave formation [64] (since the most important factor causing waves is surface wind):

$$\mathrm{BN} = (\kappa h_{\mathrm{wav}})^{2/3}; \quad \kappa = 3.83\ \mathrm{m}^{-1}. \tag{2.3}$$

The speed reduction coefficient, c_u, depends on the Beaufort number BN and displacement of the vessel ∇ (in cubic meters), according the following expression:

$$c_u = 0.7\ \mathrm{BN} + \frac{\mathrm{BN}^{13/2}}{\theta \nabla^{2/3}}; \quad \theta = 22\ \mathrm{m}^{-2}, \tag{2.4}$$

Lastly, the correction factor α depends on the vessel's Froude Number Fr. The formula also depends on its block coefficient c_B. For the purpose of our simulation, we take the formula derived from [62] for $c_B = 0.6$:

$$\alpha = 2.2 - 2.5\ \mathrm{Fr} - 9.7\ (\mathrm{Fr})^2 \tag{2.5}$$

Froude number is a dimensionless quantity defined as:

$$\mathrm{Fr} = \frac{v_{\mathrm{wtr}}}{\sqrt{gL}}, \tag{2.6}$$

where v_{wtr} is the vessel speed over water, L is the beam length of the vessel and $g \approx 9.81\mathrm{m/s}^2$ is the gravitational acceleration.

A requirement to the optimization algorithm we present in this paper (Section 3.4) is to take first- and second- order derivatives of the cost function. c_β in equation.

(2.2) is a piece-wise, discontinuous function with respect to θ, even numerical differentiation methods creates artifacts and instability in the optimization phase. To prevent this, we interpolated the output values of c_β to fit a smooth function seen in 2.1, given by

$$2\hat{c}_\beta = c - b \cdot (\mathrm{BN} - a)^2 \tag{2.7}$$

where

$$a = 6\sin^{(2/3)}(\alpha/2) + 2 \tag{2.8}$$

$$b = \frac{1}{40}(1 + \sin(1.2\alpha) - \cos(1.2\alpha)) \tag{2.9}$$

$$c = 2 - 1.6\sin(\alpha/2) \tag{2.10}$$

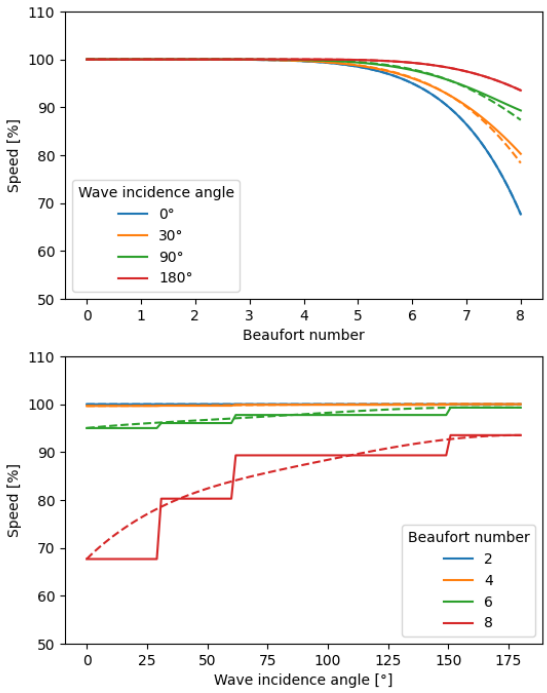


Figure 2.1: Comparison of the interpolated values (dotted line) versus the computed according to [62] (solid line). The top graph shows the speed reduction at increasing Beaufort numbers, and the bottom graph shows the speed reduction before and after applying interpolated c_β at various wave incidence angles θ.

From [56], [62] models wave effect as a percentage reduction in vessel's speed over water. The other component of speed reduction comes from currents, unlike the effects of waves, it is a change in velocity that is determined by the velocity of the currents along a route, which could work against or with the vessel along the route.

To model the effect of currents on the vessel's speed, we simply consider vector addition:

$$\overrightarrow{v}_{\text{grd}} = \overrightarrow{v}_{\text{vessel}} + \overrightarrow{v}_{\text{cur}}.$$

If $v_{\text{cur},\parallel}$ and $v_{\text{cur},\perp}$ are the components of the current velocity parallel and perpendicular to the vessel's direction, we have:

$$v_{\text{grd}} = v_{\text{cur},\parallel} + \sqrt{(c_w \cdot v_{\text{wtr}})^2 - v_{\text{cur},\perp}^2} \tag{2.11}$$

A solution of the optimization problem will be given as a polygonal curve defined by a sequence of waypoints $\{q_i\}$, where $q_i = (x_i, y_i, t_i),\ i \in [N]$, and N is the total number of waypoints along a route. Here, q_1 is the starting location, and q_N is the destination. Each waypoint is a triple, where each entry represents the longitude, latitude, and timestamp at the ith waypoint, respectively. We will adhere to the convention of specifying the route by providing the position of the vessel at equally fixed time intervals $\Delta t = \{15, 30, 60\}$ min for the three cases $v_{\text{wtr}} = \{24, 12, 6\}$ knots.

The combined effect of waves (2.1) and currents (2.11) determine the actual speed v_{grd} at a given position and time. Let $v_{\text{grd},i}$ be the speed of the vessel at position q_i and time t_i. We are assuming that a value of the waves height and direction and the ocean currents are available at every position q_i in the ocean. Weather and ocean data as provided by forecasting services (see Section 2.2) are given on a grid as a NetCDF file, so interpolation functions are necessary to retrieve weather data at an arbitrary position. We discuss the interpolation methods employed in Section 2.2.2.

We assume that the weather conditions between q_i and q_{i+1} remain constant, and thus the polygonal curve that defines the route is traversed in a piecewise linear way. For a more robust numerical scheme, we consider that the speed of the vessel in covering the segment from q_i to q_{i+1} is given by the average between $v_{\text{grd},i}$ and

$v_{\mathrm{grd},i+1}$, and thus all waypoints in the route should satisfy the condition

$$\Delta d_i = \frac{1}{2}\left(v_{\mathrm{grd},i} + v_{\mathrm{grd},i+1}\right)\,\Delta t, \quad i = 1, \ldots, N-1, \tag{2.12}$$

where Δd_i be the haversine distance between two consecutive waypoints q_i and q_{i+1}. A route with waypoints $\{q_i = (x_i, y_i, t_i)\}_{i=1}^{N}$ must satisfy (2.12) to be compatible with the vessel cruise speed v_{wtr} and the weather conditions along the route. In the Github repository we provide functions to test the feasibility of every possible route. We also provide functions to reparametrize polygonal trajectories so that the reparametrized waypoints are equally spaced in time while preserving the feasibility of the route.

The decision variables for the optimization problem are thus to determine the course heading at every Δt time interval.

This completes the definition of the optimization problem to be tackled in this thesis and in `WeatherRouting Bench 1.0`. Future versions of the benchmark will include other cost functions, such as fuel consumption, GHG emissions or safety at sea, and other decision variables (freedom to change both engine power and direction).

2.2 Meteorological Data and Processing

2.2.1 Weather data

The analysis of the vessel motion to calculate the travel time needs to have access to weather data, specifically ocean currents, waves and wind. This Section introduces the real data used in `WeatherRouting Bench 1.0`, which uses NEMO 2.2.1 model to obtain currents data and MFWAM 2.2.1 for waves.

Ocean General Circulation Models (OGCMs)

Ocean General Circulation Models (OGCMs) models simulate the physical interactions of the world's oceans and atmospheres, capturing the dynamics of ocean currents, temperature, salinity, and ice cover over time [65]. By mathematically modelling the fundamental laws of fluid dynamics and thermodynamics, OGCMs

can predict changes in the oceanic circulation and its interaction with the atmosphere, land, and sea ice. OGCMs can be classified based on the grid type used to distribute the data. They consists of two grids, a horizontal grid covers the surface of the earth, while the vertical grid covers the depth levels of the ocean or height levels of the atmosphere. Most commonly used horizontal grids are the following:

1. **Finite Differences:** These are the most used grids and discretized the space evenly, (aka, a regular grid by fixed distance or degrees of arc, both covers the longitudes and latitudes of the earth). The distribution of data usually follows Arakawa's structure [66] that have 5 options, from A to E, increasing by complexity and accuracy.

2. **Finite Element:** This is the second most popular choice for ocean modelling, especially for coastlines and offshore regions. In a finite element grid, weather variables are discretized by triangular regions. You can adjust the sizes of triangles according to the weather variable, or the complexity of the shoreline due to its 'fractal' nature. This dynamic nature offers great flexibility and ease to adjust the detail of the model where you need it most [67].

3. **Spectral:** These are the least used grids in ocean models, but they are widely used by atmospheric ones [68]. They are harder to use because of the boundaries generated by land.

There are various vertical discretization methods that accurately modelling the depth of the ocean, the most common ones are Z-coordinates, Sigma(S)-coordinates, and Isopycnal coordinates. All of these methods offers a great amount of accuracy on the surface of the ocean, in the domain that we are most interested in.

Nucleus for European Modelling of the Ocean (NEMO)

The Nucleus for European Modelling of the Ocean [69] represents an important component for the global oceanographic research. Developed through a collaborative effort by several leading European research institutions, it is a versatile modelling

framework designed to study the ocean and its interactions with the lower atmosphere, sea ice, and bio-geochemical processes.

Referring to the grid system, NEMO employs an orthogonal curvilinear grid for horizontal representation and combines Z and S coordinates for the vertical dimension, and distribution of output variables is arranged in a three-dimensional Arakawa C-type grid [66].

NEMO's framework offers a holistic approach to ocean modelling by integrating multiple key components. It simulates oceanic physical processes using advanced numerical methods to understand currents, temperature, salinity, and sea level changes [69]. Additionally, NEMO handles the complex interactions between the ocean and sea ice, including formation and melting, and their effects on circulation [70]. It also explores marine ecosystems and bio-chemical cycles to study the ocean's contribution to the carbon cycle and environmental responses [71].

Météo-France Wave Model (MFWAM)

The Météo-France Wave Model (MFWAM) is an advanced, flexible modelling framework designed to simulate the generation, propagation, and dissipation of ocean waves. It serves both as a scientific tool for understanding wave dynamics and as a core element in operational wave forecasting systems.

MFWAM employs the ECWAM-IFS-38R2 [72] computing code, incorporating dissipation terms developed by [73]. This foundation ensures robust performance in simulating wave dynamics. In November 2014, the MFWAM model received significant upgrades, thanks to advancements from the European research project "My Wave" [48], [74].

Operationally, the MFWAM model is driven by 6-hourly analysis and 3-hourly forecast winds from the ECMWF-IFS atmospheric system. The wave spectrum is discretized into 24 directions and 30 frequencies, ranging from 0.035 Hz to 0.58 Hz, providing detailed wave information. Additionally, it utilizes partitioning to separate the swell spectrum into primary and secondary swells, allowing for more nuanced and precise wave predictions.

Data processing

The maintenance of waves (MFWAM) and currents (NEMO) data are undertaken by [75], a program initiated by the European Union designed to enhance European informational services through the utilization of satellite Earth Observation and in situ (non-space) data.

The primary aim of Copernicus is to provide comprehensive monitoring and forecasting of the environmental state across terrestrial, marine, and atmospheric domains. This project supports a broad spectrum of objectives, including aiding climate change mitigation and adaptation strategies, fostering efficient emergency management practices, and enhancing the security and well-being of European citizens.

`GLOBAL_ANALYSISFORECAST_PHY_001_024` is a Copernicus product that provides a comprehensive dataset including over 30 variables such as salinity, potential temperature, and currents, among others [47]. Within this project, the focus is primarily on the current data, which is analyzed in terms of its vertical (v_0) and horizontal (u_0) components. The dataset is structured on a regular grid with a resolution of 1/12°, spanning from 180°W to 179.92°E and 89°S to 90°N (4320 × 2041 resolution). This product features 50 depth levels, arranged on an Arakawa C type grid [47]. However only the surface level is utilized for this analysis. Considering that the usual draught of a container vessel is 12m [76], the currents vary 0.01 m/s on average between the surface layer and at 12m depth, which has a negligible effect compared with a vessel's typical speed.

`GLOBAL_ANALYSISFORECAST_WAV_001_027` is another product offered by Copernicus, which provides users the sea surface significant wave *height* and *direction*, along with a comprehensive list of variables [48]. The spatial resolution of this dataset is 1/12°, spanning 180°W to 179.92°E and 89°S to 90°N, the same data dimensions as the product mentioned previously; however the temporal resolution is 3 hours.

In this model, currents are recorded on a daily basis due to their relatively stable nature, with significant changes occurring over longer periods, and waves are stored every 3 hours. Consequently, the dataset comprises a NetCDF (Network

Common Data Form) file for each day of the year, adhering to the naming convention: `YYYY-MM-DD.nc`. NetCDF is a set of software libraries and machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data [77]. This arrangement results in a collection of 365 files[1], each with near 9 million data points for each variable, creating a dataset of 16 GB containing currents data and 128 GB of waves data.

2.2.2 Interpolation

Given the dynamic nature of vessel locations at sea, it is imperative to estimate the ocean current magnitudes at these positions accurately. To achieve this, we employ bicubic interpolation on meteorological datasets, including ocean currents speed in northward and eastward directions, and wave height and northward direction.

The most typical interpolation methods in a 1- or 2-D setting are Nearest Neighbour, Bilinear, and Bicubic. Although computationally, Nearest Neighbour and linear methods are the simplest, both methods have a shortcoming: they produce a discontinuous field. This is detrimental to variational methods that expect a continuous field, such as the FMS algorithm that we will introduce in section 3.4. For this reason, We use bicubic interpolation.

Let $s_{i,j}$ where $i = 0, 1, \ldots, 2147$ and $j = 0, 1, \ldots, 4319$, represent any meteorological condition, indexed by the grid point along the latitudinal and longitudinal coordinates. The (i, j) indices are determined by the grid solution, which is available through netCDF files from Copernicus product [75]. We construct a bicubic polynomial like the following:

We build a bicubic polynomial for each $1/12^\circ \times 1/12^\circ$ square, this polynomial can be represented as:

$$f_{(i,j)}(x, y) = \sum_{m=0}^{3} \sum_{n=0}^{3} a_{i,j} \cdot x^m \cdot y^n$$

[1]Weather Data: Google Drive file

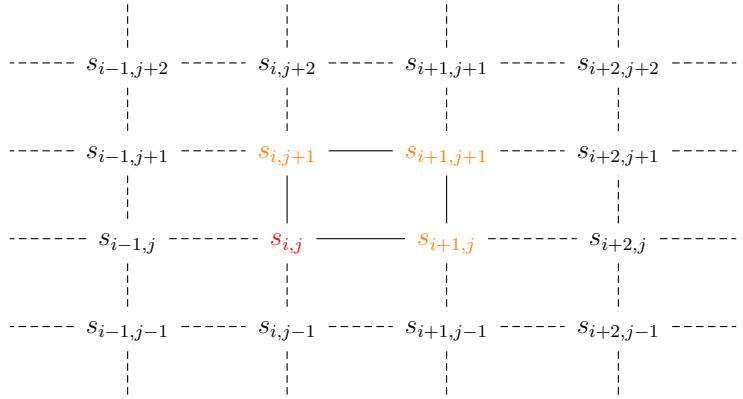


Figure 2.2: Interpolation Grid at index (i, j). To compute the bicubic polynomials valid for square $s_{i,j}, s_{i+1,j}, s_{i+1,j+1}, s_{i,j+1}$ (coloured in red and orange), we need to access all the values surrounding and including the square (the ones labeled in black).

In matrix form, it can also be expressed as:

$$f_{(i,j)}(x, y) = \begin{bmatrix} 1 & x & x^2 & x^3 \end{bmatrix} \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} 1 \\ y \\ y^2 \\ y^3 \end{bmatrix}$$

To compute the coefficients $a_{i,j}$, it is necessary to have a minimum of 16 linearly independent equations. These equations can be obtained by initially considering the immediate 4×4 square that surrounds the grid square of interest, indexed by the lower left corner (i, j) (highlighted in red in the diagram below). Subsequently, the values at these 16 points are taken into account, as depicted in the following diagram:

We arrange the above values into a length 16 vector $\overrightarrow{s}$, and we define the interpolation matrix $\mathbf{M}$ as a block matrix like the following:

$$\mathbf{M} = \begin{bmatrix} \mathbf{0} & 2\mathbf{A} & \mathbf{0} & \mathbf{0} \\ 2\mathbf{B} & -\mathbf{A} & 2\mathbf{A} & \mathbf{B} \\ \mathbf{A} & -2\mathbf{A} & \mathbf{A} & \mathbf{0} \\ \mathbf{B} & \mathbf{A} & -\mathbf{A} & -\mathbf{B} \end{bmatrix} \tag{2.13}$$

where

$$\mathbf{A} = \frac{1}{3}\begin{bmatrix} 0 & 6 & 0 & 0 \\ -2 & -3 & 6 & -2 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix} \tag{2.14}$$

and

$$\mathbf{B} = \frac{1}{9}\begin{bmatrix} 0 & -6 & 0 & 0 \\ 2 & 3 & -6 & 1 \\ -3 & 6 & -3 & 0 \\ 1 & -3 & 3 & -1 \end{bmatrix} \tag{2.15}$$

As these matrix coefficients are defined, notice that this matrix $\mathbb{M}$ is a 16 by 16 matrix. We also ensure that these polynomials agrees on the boundaries, i.e.,

$$\begin{cases} f_{(i,j)}(-,0) = f_{(i,j-1)}(-,1) \\ f_{(i,j)}(-,1) = f_{(i,j+1)}(-,0) \end{cases} \tag{2.16}$$

$$\begin{cases} f_{(i,j)}(0,-) = f_{(i-1,j)}(1,-) \\ f_{(i,j)}(1,-) = f_{(i+1,j)}(0,-) \end{cases} \tag{2.17}$$

to ensure that no sudden discontinuity is introduced in interpolation.

Then coefficients vector $\overrightarrow{a}$ can be determined by multiplying $\overrightarrow{s}$ with the above

defined matrix $\mathbf{M}$ as follows:

$$\mathbf{M}\overrightarrow{s} = \overrightarrow{a}$$

Where s and a are defined as follows:

$$\overrightarrow{s} = \begin{bmatrix} s_{i-1,j+2} \\ s_{i,j+2} \\ s_{i+1,j+1} \\ s_{i+2,j+2} \\ s_{i-1,j+1} \\ s_{i,j+1} \\ s_{i+1,j+1} \\ s_{i+2,j+1} \\ s_{i-1,j} \\ s_{i,j} \\ s_{i+1,j} \\ s_{i+2,j} \\ s_{i-1,j-1} \\ s_{i,j-1} \\ s_{i+1,j-1} \\ s_{i+2,j-1} \end{bmatrix} \qquad \overrightarrow{a} = \begin{bmatrix} a_{00} \\ a_{01} \\ a_{02} \\ a_{03} \\ \hline a_{10} \\ a_{11} \\ a_{12} \\ a_{13} \\ \hline a_{20} \\ a_{21} \\ a_{22} \\ a_{23} \\ \hline a_{30} \\ a_{31} \\ a_{32} \\ a_{33} \end{bmatrix} \tag{2.18}$$

Note that these coefficients $\overrightarrow{a}$, is only valid within the square with corners $s_{i,j}$, $s_{i+1,j}$, $s_{i+1,j+1}$, $s_{i,j+1}$.

It is evident that the resulting vector $\overrightarrow{a}$ is also of length 16. To facilitate further analysis, we reshape this vector into a 4×4 matrix denoted as $\mathbf{A}$:

$$\mathbf{A} = \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix}$$

The above matrix of 16 coefficients are dependent on the grid point of interest (i, j).

Using this matrix, we can construct the bicubic polynomial $f_{(i,j)}(x, y)$ at any point (x, y) in the square.

This process is very similar to striding in convolution in signal and image processing or Convolutional Neural Networks (CNNs), which we can utilize `Numpy`'s strengths in multi-dimension matrix algebra and computing to interpolate the entire data grid all at once instead of a nested for-loop, which drastically decreases compute time, trading off system Random Access Memory Usage, which could be a limitation in some hardware configurations.

To address this limitations, we use an approach also used in CNNs , known as chunking. With chunking, depending on the ODPs of interest, we isolate the area on the surface of earth to apply interpolation. This significantly reduces the amount of data points involved in interpolation, by at most 90% in shorter voyages, and around 50% in cross pacific or atlantic voyages.

2.3 `WeatherRouting Bench 1.0`

Having introduced what constitutes a weather routing problem, we can now define the set of instances that make the first version of `WeatherRouting Bench 1.0`. We will need to define the ODPs, departure dates, vessel parameters and operational characteristics. The cost function has already been introduced in Section 2.1.1.

Selecting a good set of ODPs for weather routing requires attention, as it first need to be representative of real world applications. In Table 1.1, all ODPs reflects real shipping routes. It also need to be of interest to test various weather routing algorithms to their full capacity. This not only requires weather routing algorithms to find an optimal path according to some criterion, but also consider obstacle avoidance. These ODPs should therefore appear in different levels of complexity, including land presence and difficult meteorological conditions.

To meet the first criteria, we can take a look at maritime networks. A maritime network is a system of maritime flows and connections between ports and other locations [78]. Maritime networks reveal various aspects of global shipping, including mapping traffic densities to show the spatial patterns of vessel movements [79]. Via

the density of maritime networks, we can identify the major hubs: the ports that are most visited. We have chosen a set of instances based on these major hubs, as they will represent ODPs relevant for the shipping industry. A literature review has been conducted to extract the major hubs in recent maritime shipping [80]–[82].

Next step is to ensure that our instances include ODP with several levels of complexity. The ODPs for the five chosen shipping routes are shown in Table 2.1. Each of these five routes will be considered in direct and reverse direction. As for the departure dates, we consider a departure for every Sunday of 2023, for a total of 52 departure dates per ODP. This ensures a variety of scenarios, as the ocean conditions change with the seasons.

Table 2.1: List of Origin-Destination Pairs chosen, grouped by regions connected.

Ocean	**Port 1 (Code)**	**Port 2 (Code)**
Atlantic Ocean	Hamburg (DEHAM)	New York (USNYC)
Atlantic Ocean	New York (USNYC)	Colón (PAONX)
Pacific Ocean	Balboa (PABLB)	Callao (PECLL)
Indian Ocean	Kuala Lumpur (MYKUL)	Hurghada (EGHRG)
Mediterranean Sea	Said (EGPSD)	Algeciras (ESALG)

For the vessel type, in this first version of the benchmark we will use a typical container ship. Its parameters are listed in table 2.2. We assume the vessel can achieve constant power delivery for the entire duration of the journey, and as such, the vessel sails at constant speeds over water, v_{wtr}. We considered three different speeds, at very slow (6 knots), slow (12 knots), and normal speed (24 knots).

Together with the 52 departure dates per ODP, ten pairs of ODPs, and at three different speeds, we have a total of 1,560 instances, providing a comprehensive basis for the purposes of this study.

The goal for all instances will always be to minimize the travel time, accounting for the effect of waves and water currents as explained in Section 2.1.1. Solutions for every route will be reported as polygonal curve with includes waypoints specifying the vessel's position (latitude, longitude) at constant time intervals of $\Delta t = \{15, 30, 60\}$ min when the vessel has speed $v_{\text{wtr}} = \{24, 12, 6\}$ kn, respectively. With this choice, the average distance between waypoints is around 10km, which is of the same order

Table 2.2: Parameters of every instance of `WeatherRouting Bench 1.0`.

Journey	
ODP	See Table 2.1 (10 ODPs).
Departure Date	Every Sunday of 2023, starting on 00:00 01-01-2023 UTC and finishing on 00:00 24-12-2023 UTC (52 dates).
Vessel	
Length	L = 220 m
Displacement	$\nabla = 36500$ m^3
Block Coefficient	$c_B = 0.6$
Speed over Water	$v_{\text{wtr}} = \{6, 12, 24\}$ kn (constant)
Optimization Problem	
Cost Function	Travel time affected by waves and water currents.
Solution	Polygonal curve with waypoints (latitude, longitude) specifying vessel position at constant time intervals.

as the grid spacing of weather data. This ensures that the hypothesis that weather conditions remain constant over each segment of the route is sound.

For an easier interpretation of the results, all computed times will be compared with a reference standard route which is given with the instance. This standard route is chosen to be the orthodromic or shortest distance route. Thus, rather than giving the total time of a given candidate solution, we will report the relative reduction (or increase) in travel time of the candidate route with respect to the shortest distance route departing on the same moment.

Chapter 3

Heuristic Methods

3.1 Zermelo's Navigation Problem and Hybrid Search

3.1.1 Zermelo's Navigation Problem on the plane

This problem was proposed in 1931 by Ernst Zermelo [49], is a classic time-optimal control problem, where its aim is to find **time minimum trajectories** under the influence of a drift vector

$$\vec{w}(x_1, x_2) = \langle w_1(x_1, x_2), w_2(x_1, x_2) \rangle$$

where x_1, x_2 are local coordinates, and where w_1, w_2 are the vector components chosen relative to a local frame. This drift vector can be interpreted as wind or water current. In small scale simulations, the coordinates and the vector components can be taken to be Euclidean. Once we pass to larger scale simulations that take into account the curvature of the Earth, the coordinates (x_1, x_2) indicate longitude and latitude (in degrees), while the vector components are taken relative to a local east-north framing (in meters).

The goal is to navigate from a specified initial point along a path that minimizes time, under the influence of $\vec{w}$, assuming the vessel provides constant thrust V (speed over water) and has a heading angle (over water) α w.r.t. the x_1-axis. Thus, the velocity components over ground can be expressed as:

$$\begin{aligned} \frac{dx_1}{dt} &= V \cos\alpha + w_1(x_1, x_2) \\ \frac{dx_2}{dt} &= V \sin\alpha + w_2(x_1, x_2) \end{aligned} \tag{3.1}$$

Using the Calculus of Variations, one can show that such a path necessarily obeys

the following differential equation, first derived by Zermelo [49]

$$\frac{d\alpha}{dt} = \sin^2(\alpha)\, w_{2,1} + \sin(\alpha)\cos(\alpha)\ (w_{1,1} - w_{2,2}) - \cos^2(\alpha)\, w_{1,2} \tag{3.2}$$

where for the sake of brevity, we write $w_{i,j} = \partial w_i / \partial x_j$.

Equation (3.2) is known as the **Zermelo differential equation**. Together with (3.1) it gives the form for time-optimal trajectories as a dynamical system in the 3-dimensional space parameterized by (x_1, x_2, α). We will refer to the initial value problem for this 3-dimensional dynamical system as the ZIVP. This means that given a current vector field $(w_1(x_1, x_2), w_2(x_1, x_2))$, an initial position $(x_1^{(0)}, x_2^{(0)})$ and an initial heading α, the trajectory defined by the initial value problem is guaranteed to be time optimal, i.e. each point in that trajectory cannot be reached in shorter time by a vessel with constant speed over water V starting from $(x_1^{(0)}, x_2^{(0)})$.

For the interest of completeness, the derivation of this last equation is fully explained in Appendix .2.

3.1.2 Zermelo's Navigation Problem on the Sphere

We now modify the above equations to the case where the ship is traveling on the surface of the Earth - idealized here as a perfect sphere. We first adopt spherical coordinates $x_1 = \theta$ (longitude) and $x_2 = \phi$ (latitude) measured in units of κ radians. In particular, it may be convenient to take $\kappa = \pi/180$ if we wish to measure in degrees. The velocities of currents will be given relative to a east-north framing, which we represent as the following 2×2 matrix

$$F(\theta, \phi) = \begin{bmatrix} K\cos\theta & 0 \\ 0 & K \end{bmatrix}$$

where K is the conversion scale from the units used to measure θ, ϕ and the units used to measure local velocities. For example, if global position is measured using degrees of arc, and local velocities are measured in kilometers, then letting R be the Earth's radius in kilometers ($R \approx 6367$ km), we have $K = \kappa R = \pi R/180 \approx 111.1$ kilometers per 1 degree of arc.

With these conventions in place, the velocity over ground of a vehicle moving at a speed of V over water is given as

$$\begin{aligned} K\cos(\kappa\phi)\frac{d\theta}{dt} &= V\cos(\kappa\alpha) + w_1(\theta,\phi) \\ K\frac{d\phi}{dt} &= V\sin(\kappa\alpha) + w_2(\theta,\phi) \end{aligned} \tag{3.3}$$

where α is the vessel's heading measured relative to an East-North framing, $\vec{w}(\theta,\phi) = \langle w_1(\theta,\phi), w_2(\theta,\phi)\rangle$ with w_1 being the component of displacement relative to east, and w_2 the component of displacement relative to north.

Using the Calculus of Variations once again, now on the sphere, one can show that such a path necessarily obeys the following differential equation,

$$\begin{aligned} \kappa K\frac{d\alpha}{dt} = &\begin{bmatrix}\cos(\kappa\alpha) & \sin(\kappa\alpha)\end{bmatrix}\begin{bmatrix}\sec(\kappa\phi)w_{1,1} & w_{1,2}\\ \sec(\kappa\phi)w_{2,1} & w_{2,2}\end{bmatrix}\begin{bmatrix}\sin(\kappa\alpha)\\ -\cos(\kappa\alpha)\end{bmatrix} \\ &-\cos(\kappa\alpha)\tan(\kappa\phi)(V+\cos(\kappa\alpha)w_1+\sin(\kappa\alpha)w_2) \end{aligned} \tag{3.4}$$

Equation (3.4) can be considered the analogue of the Zermelo differential equation for motion on a sphere. For the sake of completeness, the derivation of this equation is fully explained in Appendix .2 [30].

3.1.3 Hybrid Search method

Hybrid Search (HS) is the three-step algorithm proposed in this paper for solving the ZNP in either Euclidean or Spherical background. The three steps are (i) exploration, (ii) refinement, and (iii) smoothing. The output of the exploration and refinement phases is a piece-wise optimal trajectory that connects a starting location with a desired destination.

In effect, exploration is a shooting method based on the Zermelo's initial value problem (ZIVP). The exploration algorithm formulates multiple instances of a ZIVP with a given initial position and a cone of directions aimed towards the target. The trajectories are then evolved using RK4 numerical solutions to the Zermelo Differential Equation with dynamic termination conditions. The most obvious termination

condition is to select the trajectory that minimizes the distance to the target. In practice, it turns out that a better heuristic is to terminate each trajectory when the difference between the heading angle and the direction to target exceeds a certain pre-set threshold. The algorithm is greedy, in that a single "winner" trajectory is selected from the list of dynamically terminated trajectories. This selection is performed on the basis of minimum distance to target.

The refinement phase is just a second run of the exploration algorithm, but this time the cone of initial directions is more narrow and centered on the winner shot angle of the previous exploration phase. The candidate trajectories are then evolved using the same heuristic and a winner is selected based on proximity to the target. Then a new exploration phase begins, starting at the final waypoint of the winner segment. The precise details of the exploration and refinement sub-algorithms are detailed in sections 3.1.3 and 3.1.3 respectively.

The third phase consists of smoothing the output of the refinement using the FMS algorithm [17], [53]. This algorithm is a numerical Boundary Value Problem scheme that works by iteratively shifting a given discretized trajectory towards a time-minimizing route. The approach is based on the discrete Calculus of Variation and can, in principle, be utilized with any given Lagrangian. In our case, we select the time-minimizing Lagrangian such that the corresponding Euler-Lagrange equations are precisely the Zermelo Differential Equation. We then discretize the time-minimizing Lagrangian using a pre-selected time-step and begin the iteration with the piece-wise optimal solution generated by the exploration and refinement sub-algorithms. Because the initial trajectory is piece-wise optimal, the overall effect is that of smoothing the sharp turns present in the initial trajectory and converting the piece-wise smooth and piece-wise optimal solution to a smooth, near optimal solution of the Zermelo problem. The relevant details of the FMS algorithm are specified in section 3.4.

Exploration step

Given a start point $\boldsymbol{x_A} = (x_{A,1}, x_{A,2})$, and a goal point $\boldsymbol{x_B} = (x_{B,1}, x_{B,2})$, we can first centre a search cone in the direction of $\Lambda_{A,B}$, following equation (3.5) (assuming

an euclidean space). The amplitude for the search cone is γ. If the vector field was null and we started a trajectory with heading $\alpha = \Lambda_{A,B}$, the vessel would eventually arrive to $\boldsymbol{x_B}$. Thus, by taking this search cone, we are assuming that the optimal route will always point close to the destination and that the vector field will have a small effect on the vessel trajectory. However, this assumption can be relaxed by increasing the amplitude of the cone, γ (up to 2π, covering all directions).

$$\Lambda_{i,j} = \Lambda(\boldsymbol{x_i}, \boldsymbol{x_j}) = \arctan\left(\frac{x_{j,2} - x_{i,2}}{x_{j,1} - x_{i,1}}\right) \tag{3.5}$$

Equation (3.5) defines the angle $\Lambda_{i,j}$ from point $\boldsymbol{x_i}$ to point $\boldsymbol{x_j}$. This equation is applicable in Euclidean space, and can be generalized to spherical geometry for short distances. However this does not hold for our study as distances between start and end points are significant, so when working in spherical space it is better to replace equation (3.5) by the following:

$$\Lambda_{i,j} = \arctan\left(\frac{-c_j \cdot s_i + c_i \cdot s_j}{-\left(c_i \cdot c_j + s_i \cdot s_j\right) \cdot \sin(x_{i,2}) + (c_i^2 + s_i^2) \cdot \sin(x_{j,2})}\right) \tag{3.6}$$

where $c = \cos(x_1) \cdot \cos(x_2)$; and $s = \sin(x_1) \cdot \cos(x_2)$.

Next, we generate N initial shooting angles, namely

$$\alpha_n(0) \in [\Lambda_{A,B} - \gamma/2,\ \Lambda_{A,B} + \gamma/2]\,.$$

To do so we N-sect the search cone into $\alpha_0, \ldots, \alpha_N$, evenly spread across the whole cone, and use each of these α_n as an initial condition to solve the system of ODE via the Fourth order Runge-Kutta method (RK4). We will use these shooting angles to generate N local paths, or trajectories $q_n(t) = (x_{n,1}(t), x_{n,2}(t), \alpha_n(t)),\ n \in [0, N]$.

The N generated trajectories evolve using RK4, in iterations of time $\tau > \Delta t$ (where Δt is the time step of RK4). After every iteration τ, each trajectory n is checked individually to assert whether it meets any one of three stopping conditions. If it does, trajectory n is left out of the RK4 loop and will not evolve further. These three rules are:

1. Trajectory n is stopped at time T if

$$D\left(\boldsymbol{x_n}(T), \boldsymbol{x_B}\right) \leq d,$$

being $D(\boldsymbol{x_a}, \boldsymbol{x_b})$ the distance metric between two points, defined according to the space we are operating on, and d a certain distance threshold. This implies the vessel has reached its goal.

2. Trajectory n is stopped at time T if its heading $\alpha_n(T)$ deviates too much from the goal. To assert this, we take point $\boldsymbol{x_n}(T)$, and compute its angle to $\boldsymbol{x_B}$, named $\Lambda(\boldsymbol{x_n}(T), \boldsymbol{x_B})$, see equations (3.5) and (3.6). Otherwise, the trajectory keeps evolving while the following condition is met:

$$\left(\Lambda(\boldsymbol{x_n}(T), \boldsymbol{x_B}) - \gamma_d/2\right) \leq \alpha_n(T) \leq \left(\Lambda(\boldsymbol{x_n}(T), \boldsymbol{x_B}) + \gamma_d/2\right),$$

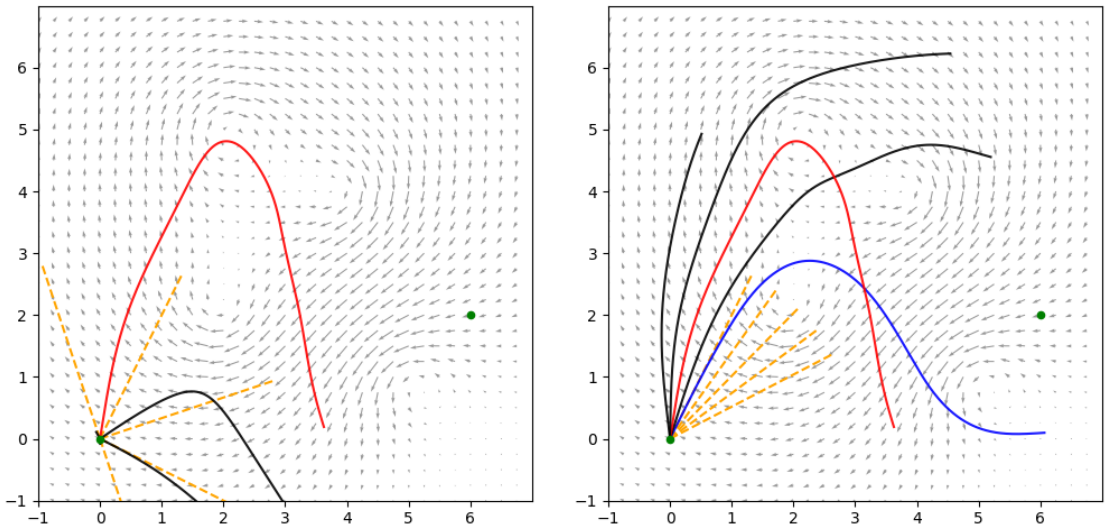
where γ_d is the maximum deviation allowed from the goal, typically equal or lower than the search cone $\gamma_d \leq \gamma$. The higher γ_d, the more exploratory is this method, but it will take more iterations to converge.

3. Trajectory n is stopped at time T if any of its points $\boldsymbol{x_n}(t), t \in [0, T]$ are located in land. In addition to stopping the trajectory, the algorithm discards all the way-points $q_n(t),\ t \geq t_{\text{land}}$, being $\boldsymbol{x_n}(t_{\text{land}})$ the first point located in land. The trajectory $q_n(t),\ t < t_{\text{land}}$ is kept, as it may still be the optimal route and just needs a course correction, that will be done in a later step.

Note that each trajectory may be stopped at a different time T. For this reason, we denote as T_n the last moment of trajectory n, i.e. its waypoints are $q_n(t),\ t \in [0, T_n]$.

One can argue that the second rule is too strict for small γ_d, as the vessel can be heading "wrongly" for a negligible amount of time before turning "correctly" again, and that the resulting route might be optimal. However, when working with real scenarios, the influence of the vector field is small enough to justify that a vessel going in a "wrong" direction will not turn "correctly" on time to compensate this

deviation.



(a) (b)

Figure 3.1: First two steps of the Hybrid Search (HS) method: (a) exploration and (b) refinement. Each trajectory is generated from a different shooting angle (in orange) and evolves using Fourth order Runge-Kutta method (RK4) method iteratively with $\tau = 0.1$, until their heading deviates more than $\gamma_d = \pi/2$ radians from the goal. After all local paths are computed, the one that got closer to the destination is chosen as best (highlighted in the graph). The search cone had an amplitude of $\gamma = \pi$ radians in the exploration step and was centred on the direction of the goal. During refinement, the search cone was centred on the shooting angle of the best route found in the exploration step, and its amplitude is narrower, $\gamma = \pi/5$.

Figure 3.1a shows a visualization of this exploration step, highlighting the one which got closest to the goal. RK4 method ensures that all trajectories are time optimal. After all N trajectories stop, if none of them reached the goal $\boldsymbol{x_B}$ (i.e. none met the first stopping rule), we choose the trajectory

$$m : D\left(\boldsymbol{x_m}(T_m), \boldsymbol{x_B}\right) \leq D\left(\boldsymbol{x_n}(T_n), \boldsymbol{x_B}\right) \forall n \in [0, N] \tag{3.7}$$

We denote this trajectory m as the "best trajectory", then move to the refinement step (section 3.1.3).

Refinement step

In the exploration step, we assumed that the optimal route should be heading closely towards the goal $\boldsymbol{x_B}$, and evolved trajectories defined by the points

$$q_n(t),\ t \in [0, T_n],\ n \in [0, N].$$

with initial shooting angles $\alpha_n(0) \in [\Lambda_{A,B} - \gamma/2,\ \Lambda_{A,B} + \gamma/2]$.

We now generate a narrower search cone, with amplitude $\gamma_b << \gamma$ (for instance, $\gamma_b = \gamma/5$) and we center it on α_m, where m is the "best trajectory" from the exploration step. Thus, the newly generated initial shooting angles are evenly spread across

$$\alpha_n(0) \in [\alpha_m - \gamma_b/2,\ \alpha_m + \gamma_b/2]\,,\ n \in [0, N]$$

We now re-run the exploration algorithm for this last segment. At some point, all the trajectories will have stopped due to the three rules we set. If no trajectory reached the goal $\boldsymbol{x_B}$ (i.e. no trajectory meets the first stopping criteria), we update the "best trajectory" m following equation (3.7). The algorithm goes back to the exploration step (section 3.1.3) using $\boldsymbol{x_A} = \boldsymbol{x_m}(T_m)$ as the starting point.

This loop between exploration-refinement continues until the first stopping rule happens, i.e. one trajectory gets close enough to the destination $\boldsymbol{x_B}$. Figure 3.10a displays one possible result of this process. One issue is apparent: the vessel takes sharp turns in the connections between local paths. This happens because each segment (except the last one) is stopped due to deviating from the goal, so the vessel is forced to correct its course by turning sharply to reach its destination.

3.2 Graph Optimization: A Super Star Search

Graph optimization is a powerful mathematical technique widely used in the field of weather routing [25], [50]. These algorithms represent the ocean as a graph and aim to find the path that minimizes a specific objective function. The objective function can be travel time, fuel consumption, or other operational costs, depending on the requirements of the application. One advantage of graph search algorithms is their

ability to easily incorporate constraints, such as obstacle avoidance, by disallowing certain nodes in the graph. Nevertheless, this flexibility necessitates discretizing the search space, resulting in routes with abrupt turns, which are impractical for real-world applications. However, these routes serve effectively as initial solutions for more refined optimization methods, as is the case for the FMS algorithm that we will introduce in Section 3.4.

3.2.1 Grid Resolution

To implement graph optimization, we need to discretize the world map using a grid. Each node of the grid corresponds to a coordinate in latitude and longitude. The grid avoids points on land, as there are no currents or waves reading on land. A sensible graph would be to consider the data grid provided in our meteorological data; the distance between points in currents data being $1/12° \simeq 0.083°$ (approximately 10 km at the equator). This results in a grid of size 4320×2041, containing near 9 million nodes. This grid would then transformed into an undirected weighted graph, where each node represents a coordinate, the edges connect adjacent squares, and weights are computed with the cost function T defined in Section 2.1.1. However using square grids, (aka, Mercator Projection) has limitations due to the Earth's spherical shape, which requires significant distortion to fit the grid.

To address this issue, we instead utilize an hexagonal grid provided by the H3 library [51]. As shown in Figure 3.2, hexagons together with 12 appropriately shaped pentagons accurately cover the round shape of the Earth. There are different sizes of hexagons ranging from 4×10^6 km^2 to 1 m^2 with 15 resolution levels. Our A* graph search algorithm will be tested **grid resolutions** 3 to 5. The specifics of these resolutions are shown in table 3.1.

3.2.2 Neighbours Exploration

Each hexagon of the H3 grid has six neighbours, allowing for two more possible directions per node compared to a traditional square grid. This is important for real applications of weather routing, as vessels take smooth turns. A square forces 90°

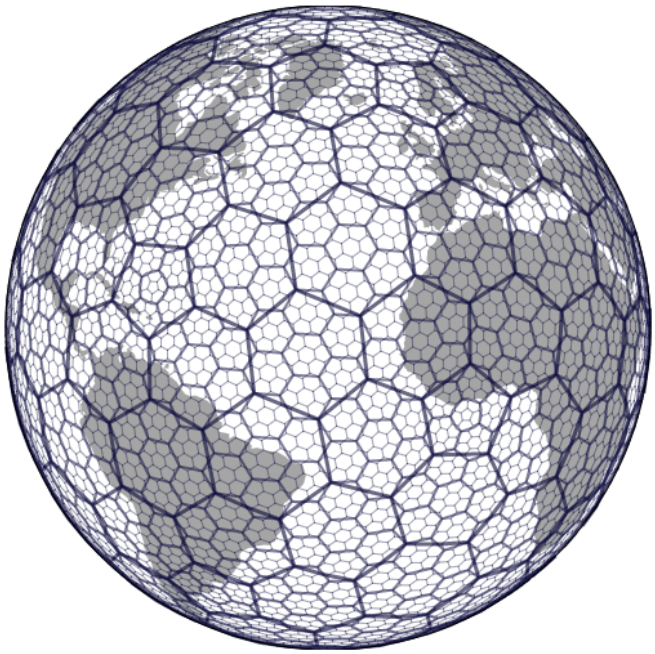


Figure 3.2: Partition of the Earth into hexagonal grids of different sizes, using H3. Image from [51].

Table 3.1: H3 resolutions and Hexagon sizes. Data from [51].

H3 Res.	**Hexagon Area (km^2)**	**Hexagon Edge Length (km)**	**Number of unique indexes**
3	12,392	59.8	41,162
4	1,770	22.6	288,122
5	253	9.8	2,016,830

turns while an hexagon reduces that to 60°. However, that is still a sharp turn for maritime standards. To address this challenge, we connect each node to its ***N*-order neighbours**, effectively providing $6 \cdot (1 + \sum_{k=0}^{N-1} k)$ possible directions and reducing the angle of course changes, as depicted in Table 3.2.

One of the main challenges faced by weather routing algorithms is avoiding land. To ensure land avoidance, we implement two rules when building the graph: (1) hexagons located directly on land are removed and (2) edges that cross land are also removed, preventing any jumping over the land. This is shown in Figure 3.3.

Table 3.2: Graph characteristics dependence on the N-order neighbours.

N	Maximum number of available neighbours	Minimum course correction
1	6	60°
2	18	30°
3	36	20°

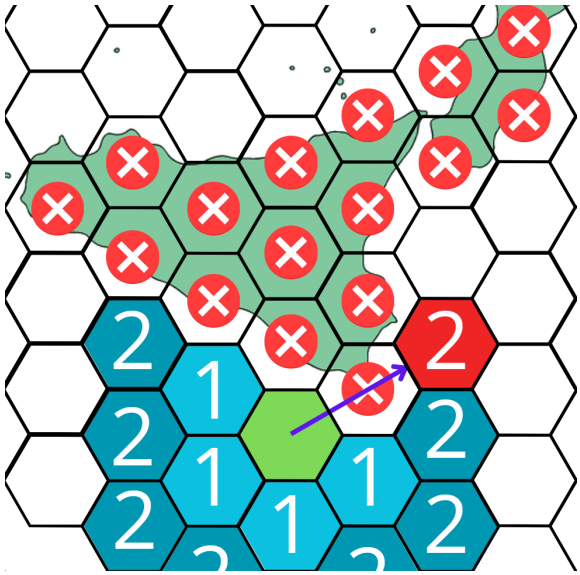


Figure 3.3: Demonstration of how the algorithm can jump over several neighbours, enhancing the flexibility of the route. It also shows the two techniques to ensure land avoidance: Hexagons located directly on land are removed, ensuring that every path on the pruned graph lies entirely on water. Note that the hexagons here are for illustrative purposes only.

3.2.3 Heuristic

With the graph constructed, the goal of our weather routing algorithm is to find the minimal path between two nodes $(n_{\text{start}}, n_{\text{end}})$. The minimal path is defined as the one with the least travel time. The most popular method used in graph optimization problems is Dijkstra's algorithm [83]. This algorithm computes the minimal path between each pair of nodes, requiring exploration of the entire graph. However, in this case, exploring the entire graph is not feasible due to its size (over 10 million nodes). An alternative is the A* algorithm [84], which unlike Dijkstra's algorithm,

does not need to explore the entire graph to find the minimal path between two nodes, thanks to the use of an heuristic inside the cost function. Thus, the cost function for node n is defined as:

$$f(n) = g(n) + h(n), \tag{3.8}$$

where $g(n)$ is the cost of the path from n_{start} to n, and $h(n)$ is the heuristic cost of the path from n to n_{end}. Following Huang [85], we choose

$$h(n) = \frac{d(n, n_{\text{end}})}{\bar{v}_{\text{short}}}, \tag{3.9}$$

where $d(n, n_{\text{end}})$ is the Haversine (or great circle) distance between nodes n and n_{end} and $\bar{v}_{\text{short}}$ is the average ground speed of the vessel computed along the route of shortest distance (orthodromic).

A* takes into account both the explored path and an approximation of the remaining path. Thus, A* provides an optimal solution as long as the heuristic is admissible, i.e. it does not overestimate the real cost from n to n_{end}.

A* takes into account both the explored path and an approximation of the remaining path. Thus, A* provides an optimal solution as long as the heuristic is admissible, i.e. it does not overestimate the real cost from n to n_{end}. Although A* is guaranteed to find an optimal route if the heuristic is admissible, the quality of the heuristic affects the number of nodes explored by the algorithm, so the complexity of A* depends on the heuristic chosen. In the worst-case scenario, a poor heuristic will force A* to explore every node in the graph, similarly to Dijkstra's algorithm.

There are several variations of A* that can help the algorithm converge faster. One of them is Weighted A* [86], where the cost function is defined as:

$$f(n) = g(n) + w \cdot h(n) \tag{3.10}$$

Here, w is a weight that multiplies the heuristic component, affecting the number of nodes explored by the algorithm. For example, setting $w = 0$ cancels the heuristic component, making A* behave like Dijkstra's algorithm. Conversely, assigning a

very high value to w causes the path from n_{start} to n to be disregarded, resulting in a Greedy Best First Search [87]. Higher values of w are expected to speed up the algorithm, but do not guarantee the optimal result, even when using admissible heuristics.

3.2.4 Hyper-parameter Search

We have run experiments for a number of different configurations for the A* algorithm, namely

- Grid resolution: 3, 4, 5
- N-order neighbours: 1, 2, 3
- Weight of the heuristic w: 0.5, 1.0, 1.25

This amounts to 27 different configurations, run across all ten ODPs (five pairs of ports in both directions) at three different velocities (6, 12 and 24 knots), for a total of 810 different experiments. For each experiment we registered the computation time of the A* algorithm and the time it takes the vessel to reach its destination - the target to be optimized. We then compared that travel time with the time taken by following the minimum distance route (circumnavigation), computing the percentage gain of the A* with respect to it. All experiments where conducted on the same machine[1].

It is worth noting that A* was not able to find a route for 90 out of the 810 experiments. The main source of this issue were the ODPs between EGHRG and MYKUL. Due to the narrow Suez canal, some grid configurations did not present any feasible connection between the start and end nodes. Particularly, grid resolution 4 with 1st-order neighbours and grid resolution 3 with 1st- and 2nd-order did not reach a feasible solution. This challenge is not a limitation of the A* algorithm itself but rather a consequence of the H3 graph structure's resolution constraints. To address this, one potential improvement is to implement an adaptive multi-scale resolution

[1]Intel Core i9-14900K, with 128GiB RAM.

Table 3.3: Pearson correlation coefficient (PCC) between the gains produced by A* (compared with the minimum distance), its computation time and the different hyper-parameters of this algorithm.

PCC	**Gain**	**Compute time**
Vessel speed	-0.258	0.030
***N*-order neighbours**	0.678	0.146
Grid resolution	0.211	0.269
Weight of the heuristic	-0.253	-0.242
Explored nodes	0.150	0.757
Wave height	0.107	0.030

strategy that increases grid granularity in the presence of complex land masses like archipelagos or straits.

We computed the Pearson correlation [88] between hyper-parameters and results, to better understand their impact. These Pearson coefficients are shown in Table 3.3, comparing instance's parameters such as speed, explored nodes, and wave height. In relation to problem instances, it is evident that higher vessel speeds result in less gains. On the other hand, an increased number of nodes and the presence of strong currents and high waves increase the potential gains achieved by A*. Among A* hyper-parameters, increasing the N-order neighbours significantly improves optimization, because it adds more nodes to explore at each step, as shown in Table 3.2. Additionally, the weight of the heuristic and grid resolution greatly impact computation time, which is crucial for deployment and implementation of this system.

Figure 3.4 groups the A* gains by N-order neighbours and grid resolution, validating our expectations: a finer grid resolution and a bigger neighbour order opens more paths to explore, and thus improves the overall results. We also observe, however, that a grid resolution of 5 increases the computation time by at least an order of magnitude while only managing to net gains similar to resolution 4. To balance gains with a reasonable computation time, we will choose a grid resolution of 4 with 3rd-order neighbours. Next we decide a weight for the heuristic.

Looking at Table 3.4 we conclude that an heuristic weight of 0.5 offers the best results without a significant cost in computation time. We also observe, however, that A* still struggles to achieve gains over the minimum distance routes in some

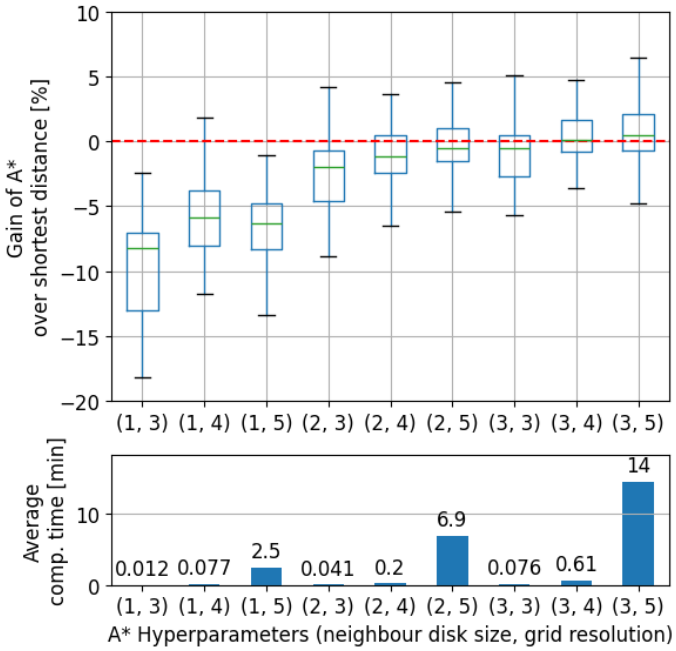


Figure 3.4: Hyper-parameter settings for A*, grouped by N-order neighbours and grid resolution. The gain of A* over circumnavigation is shown, and computation times are plotted below.

Table 3.4: Gain and computation time of A*, showing mean (std) across configurations. Grid resolution is fixed to 4, and 3rd-grade neighbours.

Speed (knots)	**Weight of Heuristic**	**Gain (%)**	**Compute time (min)**
6	0.5	3.79 (2.43)	1.51 (1.73)
6	1.0	3.00 (1.83)	0.36 (0.65)
6	1.25	-0.56 (2.46)	0.14 (0.23)
12	0.5	1.34 (2.03)	1.03 (1.06)
12	1.0	0.88 (1.72)	0.37 (0.46)
12	1.25	-1.89 (1.74)	0.09 (0.17)
24	0.5	0.01 (0.84)	1.48 (1.82)
24	1.0	-0.16 (0.88)	0.42 (0.52)
24	1.25	-2.75 (1.41)	0.11 (0.21)

scenarios. To improve its results, we will apply a FMS refinement discussed in Section 3.4 to every A* output. This ensures that FMS will always output a solution at least as good as the seed route provided by A*.

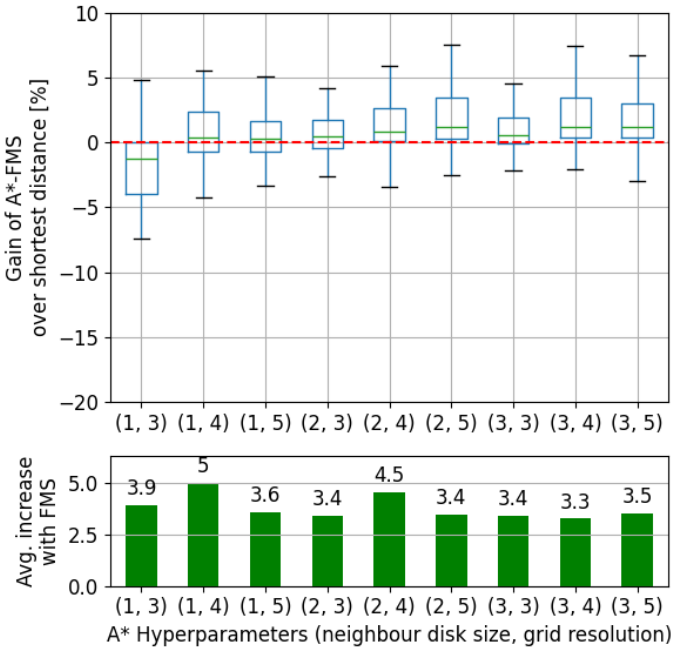


Figure 3.5: The box-plot show the gain of A*+FMS over shortest distance route, for every A* configuration. Below them, the bars show the average increase (in gain %) that FMS achieves when applied to A*.

FMS is able to improve any A* configuration greatly, as illustrated in Figure 3.5. FMS is able to neutralize the disadvantages resulted by sub-optimal A* configurations, to the extent where the average gain in each A* configuration is always greater than 0. As such, worse A* configurations benefit more from FMS, witnessing an increase of around 8% in gains for 1st-grade neighbours. The choice of A* hyperparameters is not so crucial after applying the FMS algorithm, whose addition is a great improvement over a pure graph optimization method.

3.3 Bézier Evaluation on Evolutionary Strategy (BEES)

We also propose combining a black-box optimiser with a solution space consisting of a parameterisation of the vessel trajectories going from origin to destination. This

approach is similar to variational optimisation in the sense that full routes are generated and iteratively adapted so as to minimise a functional, instead of progressively assembled leg by leg. We choose the Covariance Matrix Adaptation Evolution Strategy (CMA-ES)[89] as our optimiser, while we parameterise the trajectories using an K-degree Bézier curve.

3.3.1 Parameterisation

Two-dimensional Bézier curves of degree K use a set of control points

$$\boldsymbol{c}_0, \ldots, \boldsymbol{c}_K \in \mathbb{R}^2$$

We fix $\boldsymbol{c}_0$ and $\boldsymbol{c}_K$ as the source and destination points, respectively, and leave $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_{K-1}$ free.

Although the general Bézier formula reads

$$f(t) = (x(t), y(t)) = \sum_{k=0}^{K} \binom{K}{k} (1-t)^{K-k} t^k \boldsymbol{c}_k, \tag{3.11}$$

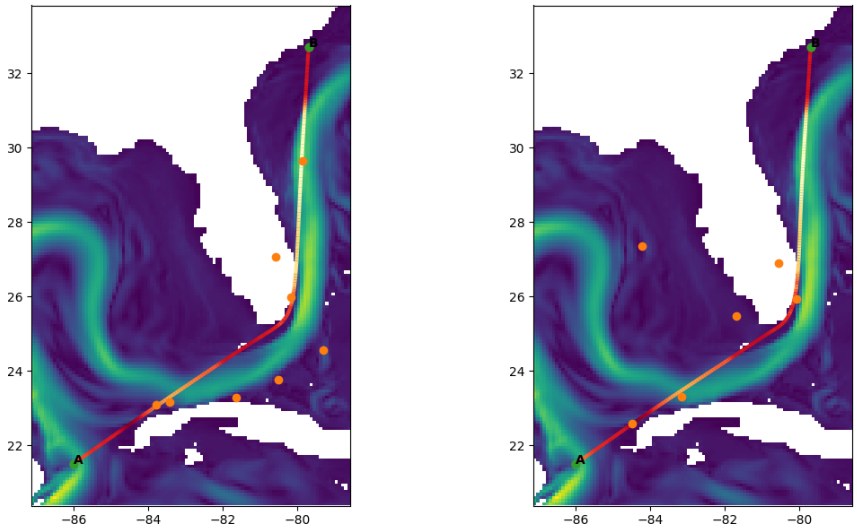
we use De Casteljau's recurrence [90] instead, which is known to be more numerically stable: $f(t) = \boldsymbol{c}_0^{(n)}$, where

$$\boldsymbol{c}_k^{(i)} \coloneqq \begin{cases} \boldsymbol{c}_k^{(i-1)}(1-t) + \boldsymbol{c}_{k+1}^{(i-1)} t \\ \boldsymbol{c}_k^{(0)} \coloneqq \boldsymbol{c}_k. \end{cases} \tag{3.12}$$

Bézier's method yields smooth (infinitely differentiable) curves, which can be very expressive with few degrees of freedom (see Fig. 3.6).

3.3.2 Optimizer

CMA-ES is an evolutionary optimization algorithm that proposes candidate solutions using an adaptive multivariate Gaussian distribution. We start with an initial solution $\boldsymbol{\mu}_0$ which is a straight path connecting source with destination. At each iteration t, the distribution follows $\mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$ and is defined so as to approximate



(a) (b)

Figure 3.6: Different sets of control points (shown in orange) can yield almost identically shaped Bézier curves, however velocity along the curve varies

the geometry of the score function around $\boldsymbol{\mu}_t$. Then, a sample $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_P$ is drawn from it. The function is evaluated on each $\boldsymbol{x}_p$ and the values are used to define the next distribution $\mathcal{N}(\boldsymbol{\mu}_{t+1}, \boldsymbol{\Sigma}_{t+1})$. The sequence $\boldsymbol{\mu}_t$ can be understood as a stochastic version of gradient descent, whereby noisy samples around the point are taken at each iteration (the *mutations*) so that the algorithm has higher chances of escaping local minima. In contrast, the optimizer is derivative-free, which makes it applicable to a wide range of scenarios. At the same time, the fact that P samples are evaluated in batch at each t makes it easy to parallelize: as we will show, our algorithm can score more than 1000 vessel trajectories per second.

Instead of discarding paths that cross land, we include a penalization term. The cost we add is equal to the total distance traversed on land times a large enough factor (we chose 10^6 in our experiments). This progressively guides the optimizer out of invalid routes into routes that fully circumnavigate the land. Although this strategy can result in local minima for certain shore lines (see Fig. 3.7), this is not a problem in practice if the CMA-ES hyperparameter σ_0 (initial mutation rate) is chosen high enough.

Figure 3.8 illustrates the convergence of this method over different random seeds.

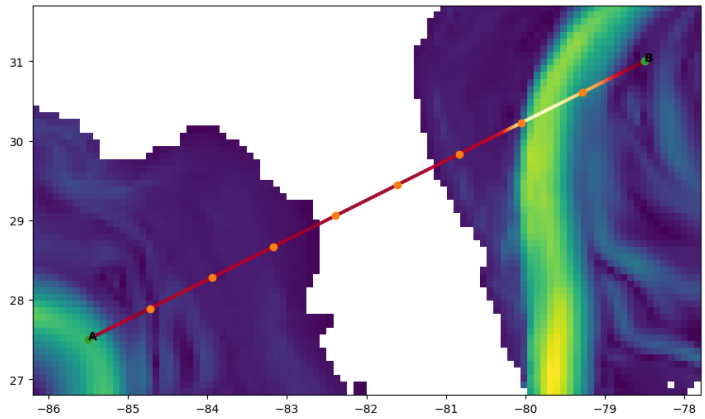


Figure 3.7: Trajectories crossing hourglass-shaped land may lie at a local minimum of the land penalization function. Still, the CMA-ES optimizer was often able to escape such minima in our experiments.

3.4 Smoothing Method - Ferraro-Martín de Diego-Sato (FMS) Algorithm

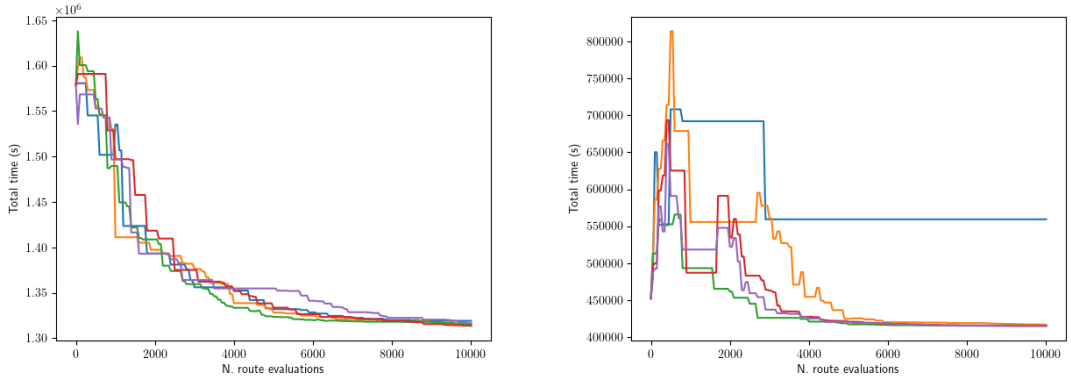
We employ a Ferraro-Martín de Diego-Sato (FMS) algorithm [17], [53], which is able to, firstly, smooth out sharp course corrections and, secondly, find further savings in the route. We do this by numerically solving a Boundary Value Problem of the Zermelo's Navigation Problem. This approach is based on Calculus of Variations and Lagrangian Mechanics on a discrete setting.

Let us quickly review the Newton-Jacobi iterative algorithm for solving nonlinear equation. Consider an equation of the form $0 = f(x)$ where $f(x)$ is a differentiable function of one variable. Newton's method proposes that we pick an approximate solution $x = x^{(0)}$ and then solve the linearized system

$$f(x^{(0)}) + f'(x^{(0)})(x^{(1)} - x^{(0)}) = 0$$

to obtain an $x^{(1)}$. If $x^{(0)}$ is sufficiently close to a root of $f(x) = 0$, one can show that $|f(x^{(1)})| < |f(x^{(0)})|$ and we can iterate to produce a sequence $x^{(0)}, x^{(1)}, x^{(2)}, \ldots$ by solving, at each stage the linearized system

$$f(x^{(i)}) + f'(x^{(i)})(x^{(i+1)} - x^{(i)}) = 0.$$



(a) Panama → New York

(b) New York → Hamburg

Figure 3.8: Convergence of CMA-ES Bézier for five different random seeds and two different (source, destination) pairs. In open sea (left), CMA-ES Bézier converges in a fairly stable way. In contrast, areas with frequent land obstacles (right) yield a harder optimization landscape, with routes occasionally failing to reach the global optimum.

The Newton-Jacobi method generalizes Newton's method to the case of an $n \times n$ system of nonlinear equations $F(q) = 0$ where q is a point in n-dimensional space and F is a transformation of n-dimensional space; i.e., $F(q) = (F_1(q), \ldots, F_n(q))$ is an n-vector of functions. As above, we begin with an initial guess q_0 and then construct a sequence of approximate solutions by solving the linearized equations

$$F(q_i) + DF(q_i)(q_{i+1} - q_i) = 0$$

for q_{i+1}. Under suitable assumptions, one can show that the sequence $q_0, q_1, q_2, \ldots$ converges to a zero of F.

The key idea introduced in [17] is to apply tr apply the NJ method iteratively to primitive 3-point trajectories, i.e. trajectory trajectories path consisting of q_{k-1}, q_k, q_{k+1}. For each such trajectory we freeze q_{k-1}, q_{k+1} and seek for the optimal placement of q_k. This amounts to a solution of the discrete Euler-Lagrange equation

$$D_2 L_d(q_{k-1}, \bar{q}_k) + D_1 L_d(\bar{q}_k, q_{k+1}) = 0$$

for an unknown $\bar{q}_k$. The complete derivation of the system of discrete Euler-Lagrange equations is included in Appendix .3.

We now apply the NJ method by taking

$$F(q) = D_2L_d(q_{k-1}, q) + D_1L_d(q, q_{k+1})$$

and apply one iteration of the method to solve the linearized system

$$F(q_k) + DF(q_k)(q_k^* - q_k) = 0$$

for the unknown q_k^*. Fully written, the system for q_k^* is then

$$\begin{aligned} &D_2L_d(q_{k-1}, q_k) + D_1L_d(q_k, q_{k+1}) + \\ &\qquad + \left(D_{22}(q_{k-1}, q_k) + D_{11}L_d(q_k, q_{k+1})\right)(q_k^* - q_k) = 0 \end{aligned}$$

We now apply the same one-step iteration to all the primitive trajectories

$$(q_{k-1}, q_k, q_{k+1}),\ k = 1, \cdots, N-1$$

to obtain a new trajectory $q^* = (q_k^*)_{k=0}^N$ with $q_0^* = q_0$ and $q_N^* = q_N$. If the initial trajectory $q^{(0)}$ is well chosen, then the iterated sequence of trajectories $q^{(i)}, i = 0, 1, \ldots$ where $q^{(i+1)} = q^{(i)*}$ converges to a solution of the discretized Euler-Lagrange equations. Moreover, this solution is ensured to be time optimal, since the conditions based on the positivity of the Hessian matrix hold generically for the Lagrangian of the Zermelo problem (see Theorems 10 and 14 in [53]). By locally time optimal, we mean that any neighboring trajectory will employ a larger time to reach the target.

In the original Zermelo problem seen in Section 3.1.1, we deal with a constrained optimization problem whose Lagrangian function has the form

$$L = \dot{t} + \lambda_1(\dot{x}_1 - (V\cos\alpha + w_1)\dot{t}) + \lambda_2(\dot{x}_2 - (V\sin\alpha + w_2)\dot{t}) \tag{3.13}$$

A full explanation of the derivation of the above Lagrangian is given in Appendix .2. The constraints associated to that Lagrangian are

$$\begin{aligned} \dot{x}_1 &= (V\cos\alpha + w_1)\dot{t} \\ \dot{x}_2 &= (V\sin\alpha + w_2)\dot{t} \end{aligned} \tag{3.14}$$

We will apply the FMS algorithm to the Euclidean Zermelo problem after suitably transforming (3.13) into a non-constrained optimization problem. It is possible to extend the FMS methodology to spherical backgrounds and to constrained optimization, but we do not pursue these directions in the present paper.

We begin by combining (3.14) into the single constraint

$$(\dot{x}_1 - w_1\dot{t})^2 + (\dot{x}_2 - w_2\dot{t})^2 = V^2\dot{t}^2. \tag{3.15}$$

Setting

$$\begin{aligned} X &= \sqrt{\dot{x}_1^2 + \dot{x}_2^2} \\ W &= \sqrt{w_1^2 + w_2^2} \end{aligned}$$

we rewrite (3.15) as the following quadratic equation in $\dot{t}$:

$$(V^2 - W^2)\dot{t}^2 + 2XW\cos\beta - X^2 = 0,$$

where β is the angle between $\dot{x}$ and w. The solution gives us the following unconstrained Lagrangian:

$$\hat{L} = \dot{t} = \frac{X}{V^2 - W^2}\left(-W\cos\beta + \sqrt{V^2 - W^2\sin^2\beta}\right)$$

As given, the above $\hat{L}$ is not a regular Lagrangian, and the corresponding $\hat{L}_d$ will not give a convergent FMS algorithm. This difficulty can be remedied by observing that $\hat{L}^2$ **is** regular, and so we take $\hat{L}_d^2$ as the discrete Lagrangian for our implementation of the FMS algorithm.

Figure 3.10b shows the results of FMS after 10 000 iterations, applied to the route generated at the end of the exploration and refinement loop.

The algorithm iteratively moves a finite number of points along a route, say $\{q_i\}_{i\in[n]}$, to a set of points $\{q_i'\}_{i\in[n]}$ that gives a lower cost compared to the previous iteration, i.e., $L(\{q_i'\}) < L(\{q_i'\})$. The FMS algorithms contains 3 hyper-parameters: *damping*, *maximum iterations*, and *early stopping*, so we can determine how much every point along a route is moved in one iteration, and how many iterations should be applied before we reach local minima. This can be best explained by applying an analogy to Neural Networks; damping is similar to *learning rate*, it is applied to determine the step size taken to reach a local minima at each iteration. This hyper-parameter offers a trade off between rate of convergence and overshooting [91]. A number of maximum iterations is set to 2,000. Before reaching that limit, the early stopping may stop the FMS if the cost function does not decrease for 20 consecutive iterations. This hyper-parameter avoid unnecessary computation time once the FMS reaches the local minima. More details are provided in [30] and references therein.

In Figure 3.9, we see the sudden change in directions along the route. After applying few hundred iterations of the FMS algorithm, we can see that these sudden changes in direction have been smoothed out. Not only does FMS provide a smoother trajectory, but also FMS is guaranteed to converge to a locally optimal solution of the variational problem, as proved in [17], [53].

The shortcoming of FMS is that this locally optimal trajectory might be far from the global optimum, as it happens with many gradient descent methods. In [30] an exploration phase Our proposed algorithm is a concatenation

FMS is able to improve any A* configuration greatly, as illustrated in Figure 3.9. FMS is able to neutralize the disadvantages resulted by sub-optimal A* configurations, to the extent where the average gain in each A* configuration is always greater than 0. As such, worse A* configurations benefit more from FMS, witnessing an increase of around 8% in gains for 1st-grade neighbours. The choice of A* hyper-parameters is not so crucial after applying the FMS algorithm, whose addition is a great improvement over a pure graph optimization method.

Similarly, we apply this algorithm to `Hybrid Search`,

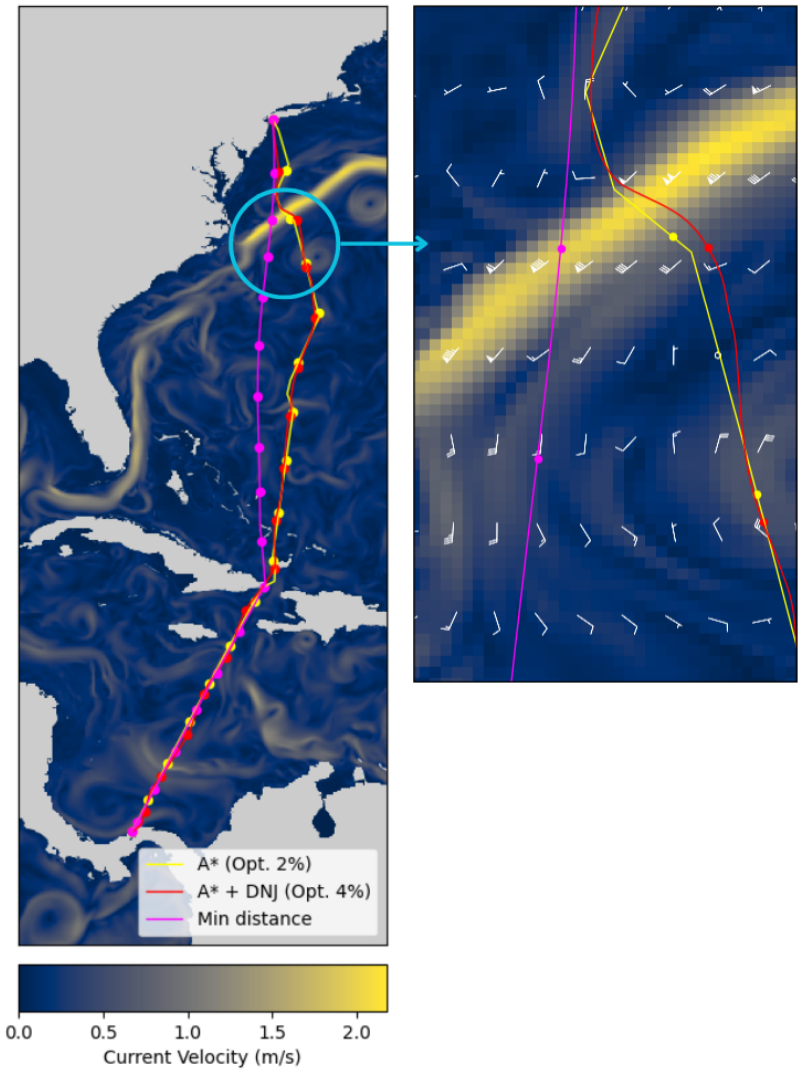
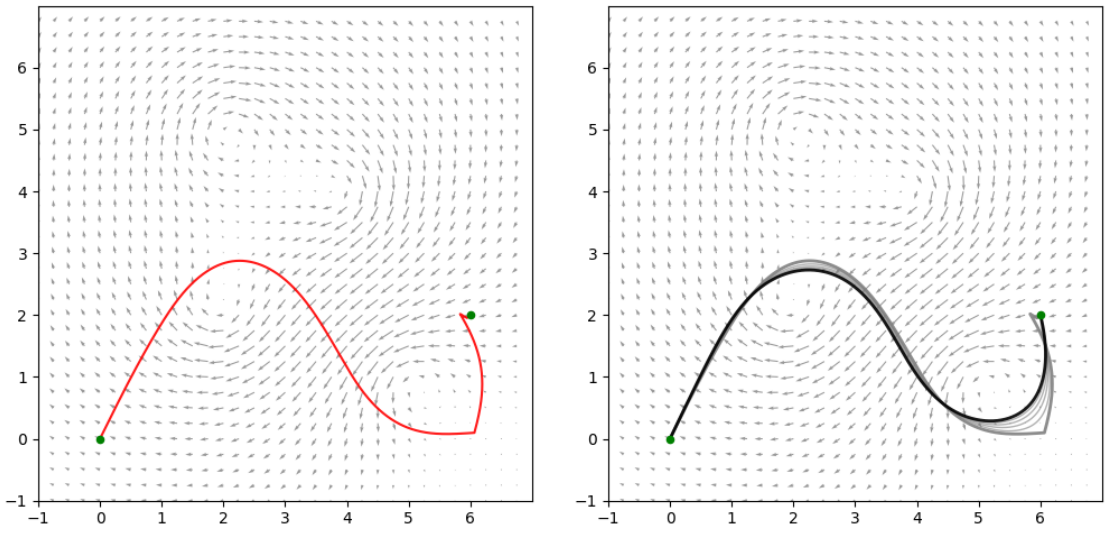


Figure 3.9: Route comparison between the shortest distance route (purple), A* (yellow) and A*-FMS (red). One can see that after applying FMS, the 'jagged' turns from A* are smoothed out, as we move from the discrete space of A* to a continuous space.



(a) (b)

Figure 3.10: (a) Optimized route obtained by alternating the first two steps of HS method. The segments are locally optimal (thanks to RK4) but are joined by sharp turns. (b) The whole route is then smoothed with FMS method for 10 000 iterations.

Chapter 4

Results

4.1 Comparison

These algorithms have been tested and compared in three different benchmarks, previously described in Section 2.1. The study was conducted at three different ship speeds, as the ship's speed directly influences the extent to which meteorological data affects the results. Three different variables have been used to study the behaviour of the algorithms:

- Travel time in hours,
- Total distance of the route in kilometers, and
- Computation time.

The results are presented in Tables 4.1, 4.2, and 4.3. The tables show the mean results of the three algorithms, `min_distance`, `Astar`, and `Bézier`, and the `HybridSearch` algorithm. The results are presented for three different ship speeds: 6, 12, and 24 kt.

Furthermore, the study was carried out weekly for one year, starting on Jan 1, 2023. This approach allows us to study the seasonal variability of the weather and how this changes the optimal route.

4.2 A*-FMS Results

After performing a hyper-parameter search for the first week of each benchmark, we selected the configuration of our algorithm by considering the trade-offs between compute time and relative route savings. The A* algorithm is configured with the following parameters: grid resolution of 4, 3rd-order neighbors, and a heuristic weight of 0.5. The output is then smoothed by the FMS algorithm with no damping, for

Speed (kt)	**Method**	**Travel time** (h)	**Distance** (km)	**Comp. time** (s)
6	min_distance	419.8	**4533.38**	69.85
	Astar	**401.4**	4801.86	99.03
	Bézier	403.6	4609.60	96.20
	HybridSearch	418.5	4554.27	**12.41**
12	min_distance	209.0	**4533.38**	59.09
	Astar	205.7	4646.74	100.95
	Bézier	**204.5**	4630.49	97.88
	HybridSearch	209.4	4551.95	**3.12**
24	min_distance	125.7	**4533.38**	57.73
	Astar	125.2	4593.63	101.97
	Bézier	**124.5**	4569.99	91.54
	HybridSearch	125.7	4536.92	**2.08**

Table 4.1: Mean results on the journey Charleston - Azores during a year. We compare the route of minimum distance (geodesic) with the output from our routing methods. Computation time is provided for illustration purposes.

Speed (kt)	**Method**	**Travel time** (h)	**Distance** (km)	**Comp. time** (s)
6	min_distance	575.4	**6201.00**	93.51
	Astar	571.21	6352.43	155.05
	Bézier	**564.8**	6249.33	109.05
	HybridSearch	572.0	6231.82	**17.46**
12	min_distance	286.5	**6201.00**	77.60
	Astar	289.4	6363.43	158.09
	Bézier	**284.8**	6220.52	108.50
	HybridSearch	286.8	6235.55	**8.19**
24	min_distance	172.0	**6200.00**	66.26
	Astar	174.7	6351.27	158.97
	Bézier	**171.5**	6211.00	88.55
	HybridSearch	172.6	6233.60	**4.62**

Table 4.2: Mean results on the journey Somalia - Myanmar during a year. We compare the route of minimum distance (geodesic) with the output from our routing methods. Computation time is provided for illustration purposes.

Speed (kt)	Method	Travel time (h)	Distance (km)	Comp. time (s)
6	min_distance	257.7	**2601.86**	58.31
	Astar	269.8	2800.52	34.85
	Bézier	**255.2**	2638.77	66.95
	HybridSearch	260.2	2661.81	**6.26**
12	min_distance	124.6	**2601.86**	52.44
	Astar	132.0	2800.52	34.65
	Bézier	**124.3**	2608.68	65.68
	HybridSearch	126.8	2660.27	**2.82**
24	min_distance	**73.6**	**2601.86**	45.95
	Astar	78.6	2800.52	34.62
	Bézier	73.7	2603.82	61.67
	HybridSearch	75.4	2666.88	**1.97**

Table 4.3: Mean results on the journey Panama - Houston during a year. We compare the route of minimum distance (geodesic) with the output from our routing methods. Computation time is provided for illustration purposes.

a maximum of 2,000 iterations, and is subject to early stopping if no improvements are observed in the previous 20 consecutive iterations.

There are five pairs of ports, introduced in Table 2.1, that can be traveled in both directions, totaling ten ODPs. Departures occur every Sunday of 2023, starting on January 1st, resulting in 52 departures per benchmark. We assume ocean data is available for the entire journey duration. The vessel model remains constant, but it travels at three different speeds over water: 6 knots, 12 knots, and 24 knots. This setup results in 1,560 experiments. To facilitate further analysis and discussion, we group the experiments to study different effects.

4.2.1 Vessel Speed

The first study conducted with these results examines how the gains of our A*-FMS algorithm are affected by vessel speed. Figure 4.1 shows a histogram of gains across all benchmarks based on vessel speed over water. Results indicate that lower vessel speeds achieve greater gains with weather routing. At a vessel speed of 6 knots, time savings average 3.60% (with a standard deviation of 2.61). For 12 knots, savings are

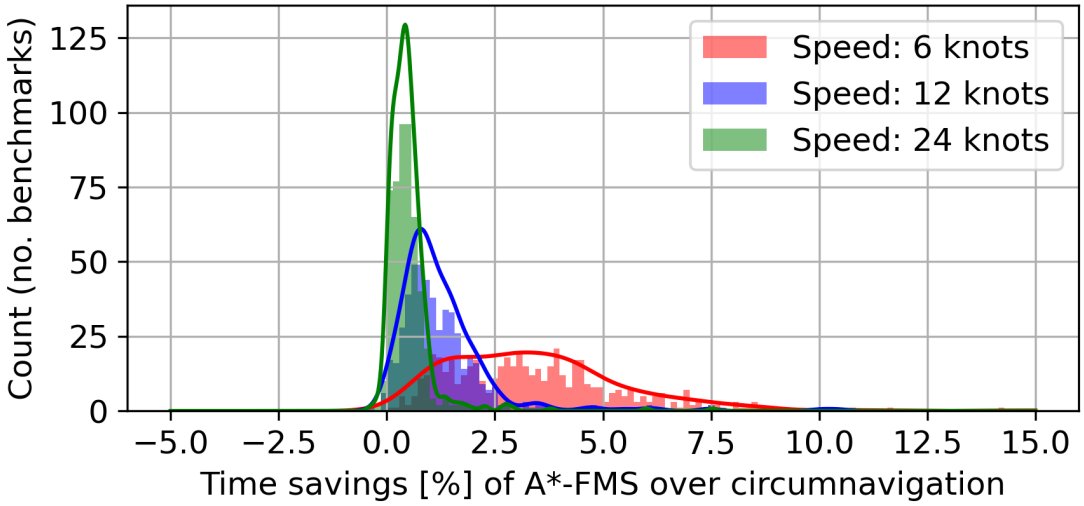


Figure 4.1: Time savings histogram and best-fit normal distribution for each vessel speed of our A*-FMS algorithm compared to circumnavigation.

1.36% (1.61), and for 24 knots, they decrease to 0.51% (0.58).

It is relevant to note that at higher speeds, there are benchmarks in which the optimized route does not improve upon the circumnavigation time. Specifically, when sailing at 12 knots, seven out of 520 experiments show negative gains. At 24 knots, 14 experiments yield worse results with A*-FMS compared to circumnavigation. These losses are marginal, the lowest being -0.4% (20 minutes longer than circumnavigation for an 84-hour journey). However, these cases are still worth investigating as they represent very challenging scenarios and/or shortcomings of our A*-FMS algorithm. In contrast, at lower speeds, A*-FMS can save up to 27% of travel time in some scenarios. We will explore these extreme cases in detail later.

Overall, these results highlight the significant impact of vessel speed on the effectiveness of weather routing algorithms. The reduced effectiveness at higher speeds suggests that external conditions and the vessel's interaction with the environment become more challenging to optimize, underscoring the importance of algorithmic improvements for high-speed scenarios.

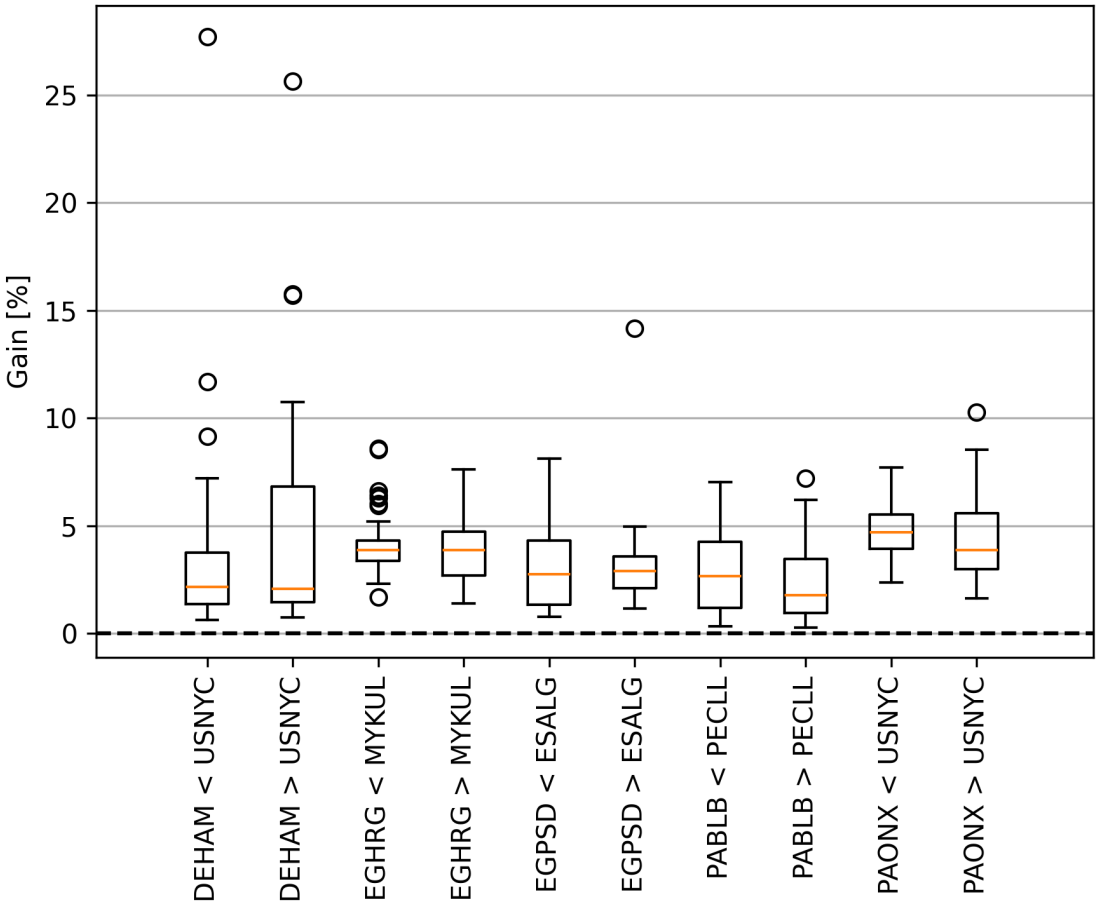


Figure 4.2: Percent reduction in travel time achieved by the A*-FMS algorithm compared to the circumnavigation route at 6 knots

4.2.2 ODP Effect

Our next discussion focuses on how the gains from weather routing depend on the Origin-Destination Pair (ODP). As noted in Table 1.1, each paper employs a different set of ODPs, making it essential to assess whether this choice affects the results reported by the algorithm. In Figure 4.2, we compare the overall gains (travel time reduction relative to circumnavigation) of our A*-FMS algorithm for each set of ODPs, sorted by direction of travel, when sailing at 6 knots. The general trend of these boxplots is similar for vessel speeds of 12 and 24 knots, with gains inversely proportional to speed.

We first observe that median gains differ between pairs of ports. From lowest to highest gains:

- 1.9% for Balboa (PABLB) - Callao (PECLL),
- 2.1% for Hamburg (DEHAM) - New York (USNYC),
- 2.5% for Port Said (EGPSD) - Algeciras (ESALG),
- 3.7% for Hurghada (EGHRG) - Kuala Lumpur (MYKUL),
- 4.2% for Colón (PAONX) - New York (USNYC).

The variance also differs significantly between ODPs, confirming our hypothesis that the choice of itineraries plays a crucial role in the results reported by weather routing algorithms. The presence of strong oceanographic conditions, such as the Gulf Stream in PAONX-USNYC, increases the potential gains of weather routing. We also observe that gains tend to increase with the distance between ODPs, as this allows the algorithm to explore a broader space. Indeed, the longest routes present the largest outliers, demonstrating that the algorithm can exploit certain favourable conditions when the weather is advantageous. We will explore the effect of seasonality later on.

In section 4.2.1, we noted that the algorithm could not outperform the circumnavigation route at higher speeds. We now see that this effect is dependent on the ODP: at 24 knots, PABLB-PECLL shows eight instances out of 52 with negative gains, PAONX-USNYC shows four, while EGPSD-ESALG and DEHAM-USNYC have only one each, and none in EGHRG-MYKUL.

These findings emphasize the importance of carefully selecting ODPs when evaluating the performance of weather routing algorithms. Variations in oceanographic conditions, route length, and external factors significantly influence the potential gains, highlighting the need for a diverse set of benchmarks to accurately assess algorithm performance.

4.2.3 Seasonal Study

We have assessed how vessel speed and the ODP affect the overall gains achievable with weather routing. Next, we discuss the seasonal effect on the gains of a journey. We group our benchmarks by seasons in the Northern Hemisphere, as shown in Table 4.4.

Table 4.4: Gains of A*-FMS algorithm over circumnavigation, grouped by each season of 2023 in the Northern Hemisphere.

Season	**Avg. Gain % (Std.)**		
	6 kn	**12 kn**	**24 kn**
Winter	5.02 (3.69)	2.09 (2.61)	0.71 (0.93)
Spring	4.06 (1.75)	1.42 (0.75)	0.53 (0.31)
Summer	2.74 (1.84)	0.96 (0.94)	0.39 (0.36)
Autumn	2.53 (1.87)	0.99 (1.28)	0.39 (0.49)

We identify greater savings across all speeds during winter, and less so in spring, compared to summer and autumn. In fact, gains in winter almost double the ones in other seasons. The standard deviation of winter is also the highest, implying that this is the season with the biggest weekly differences, i.e., the most variability.

To better asses the impact of seasonality, we will focus on how the circumnavigation time changes across the year. We compute the average travel time and compared each week's circumnavigation time against the average. The percent difference is shown in the bars of Figure 4.3, while the lines depict the gains of A*-FMS for every week.

Between DEHAM and USNYC, in winter, we see a significant increase in circumnavigation time due to harsher weather conditions. In particular, during the third and fourth week, we observe an increase in travel time of around 35% compared to the yearly average. With such a great discrepancy in time, increased savings of nearly 25% due to A*-FMS can be observed. This is only evident in the first few weeks of the year. In spring, summer, and fall, most circumnavigation routes perform consistently with similar savings. These variations are largely due to the fact that the routes between DEHAM and USNYC are in the North Atlantic, where studies have observed extreme wave climates in winters [92], [93].

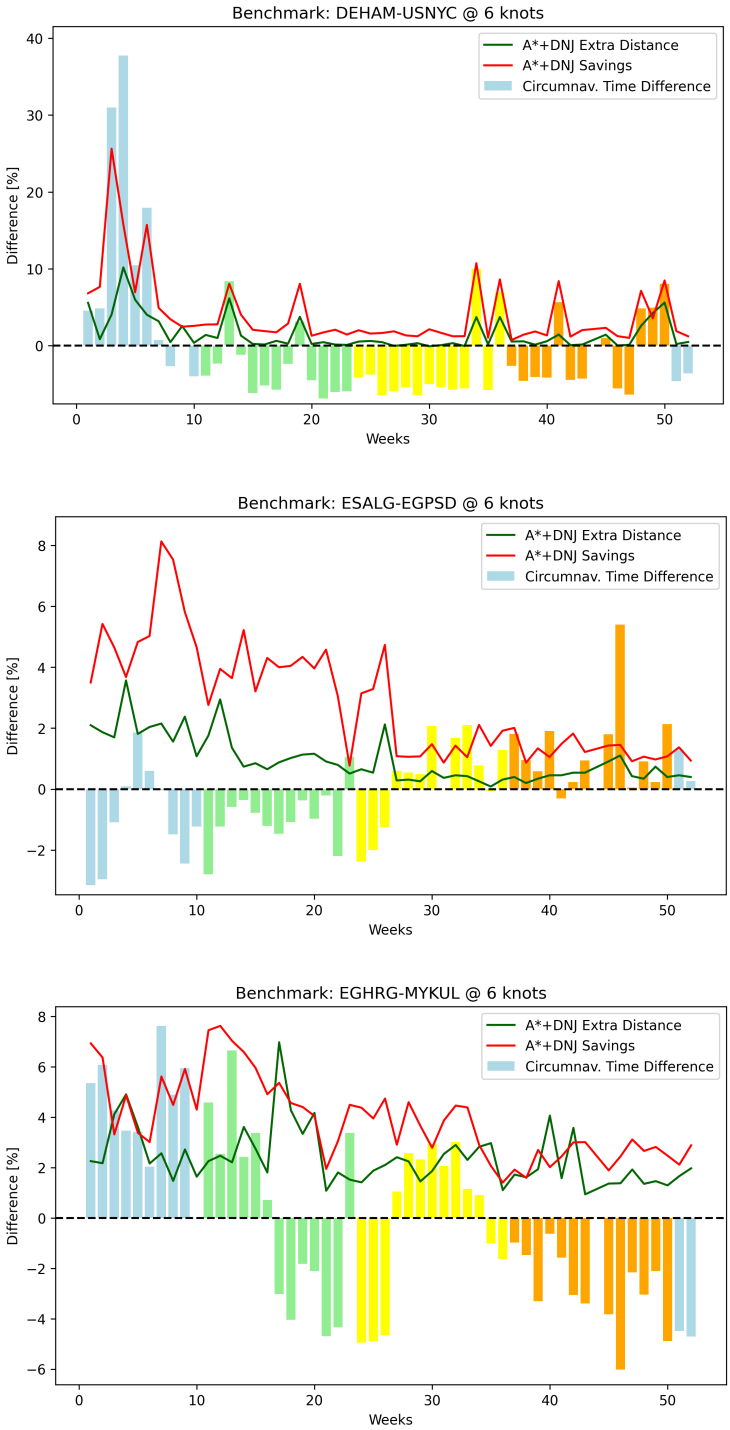


Figure 4.3: Percent difference between circumnavigation route per week to yearly average, and A*-FMS compared to circumnavigation with respect to time and distance. Each week is coloured by season, starting with winter.

Such extreme wave climates are also observed in the Mediterranean Sea, although this is not reflected in our benchmarks, as seen in subfigure 3 of Figure 4.3.

In contrast, routes between USNYC and PAONX show small fluctuations in travel time across the year, without a clear seasonal tendency.

4.2.4 Weather Variables

To conclude this discussion, it is relevant to study which weather variable has a greater impact on weather routing. We know from the equations in section 2.1.1 that waves can reduce the vessel speed by up to 70% in adverse conditions. Likewise, strong ocean currents can reach speeds of 5 knots (https://oceanservice.noaa.gov/facts/gulfstreamspeed.html). Following these prior assumptions, we expect waves to have a bigger impact, with currents becoming more relevant for slow steaming.

We ran our experiments again, considering only waves or only currents. We compared the gains of both experiments in Table 4.5. Currents play a bigger role in terms of gains; the spread of gains with only currents is not only tighter than with only waves, but also consistently higher, especially at lower speeds.

Analyzing the routes, we see that around 35% of the route has a BN of 3 and 4, with a maximum at 5, as shown in Figure 4.4, with an average BN of 3. According to our speed reduction data in Table 2.1, this results in a maximum of 5% speed reduction. At a nominal sailing speed of 24 knots, this gives a net speed reduction of around 1.8 knots. The vessel does not encounter extremely high waves along these routes, with occurrences of BN greater than 5 being less than 3%, and only in outlier cases.

Table 4.5: Comparison between considering different weather variables, percentage gains (standard deviation) with only currents, only waves, and total.

Velocity	**Gains avg. (std.) (%)**		
(knots)	**Only Currents**	**Only Waves**	**Both**
6	3.10 (1.72)	0.47 (2.03)	3.55 (2.58)
12	1.06 (0.59)	0.35 (1.53)	1.34 (1.59)
24	0.42 (0.25)	0.12 (0.55)	0.50 (0.58)

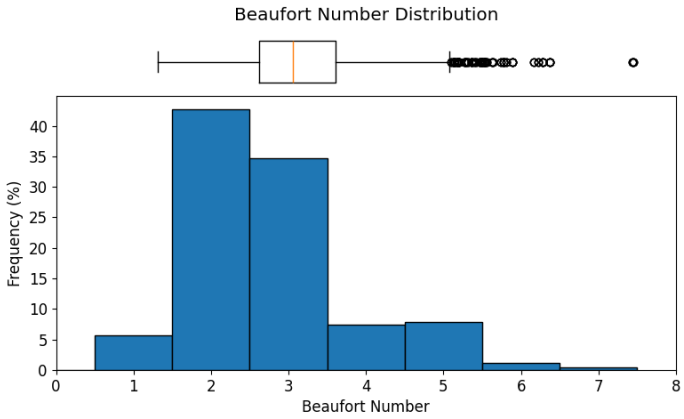


Figure 4.4: Distribution and boxplot of BN along the circumnavigation route.

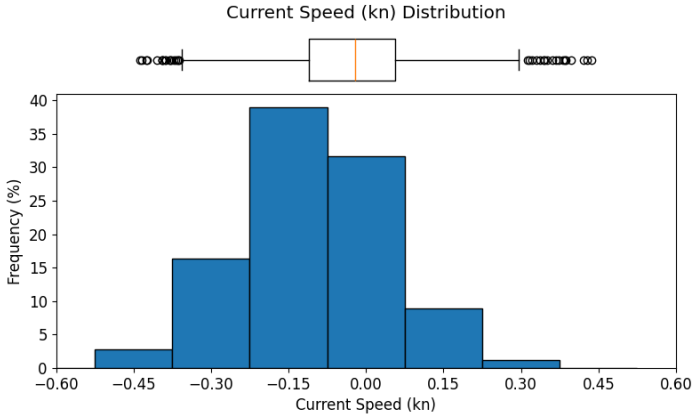


Figure 4.5: Distribution and boxplot of current speeds along the circumnavigation route, relative to the speed of the vessel. The sign of current speed indicates whether the current's direction is along or against the vessel's direction of travel.

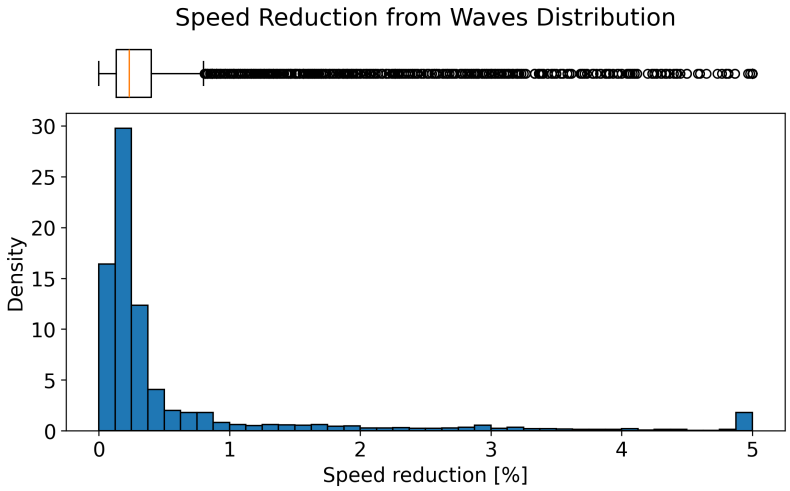


Figure 4.6: Distribution and boxplot of speed change from waves and BN of vessel speed along the circumnavigation route.

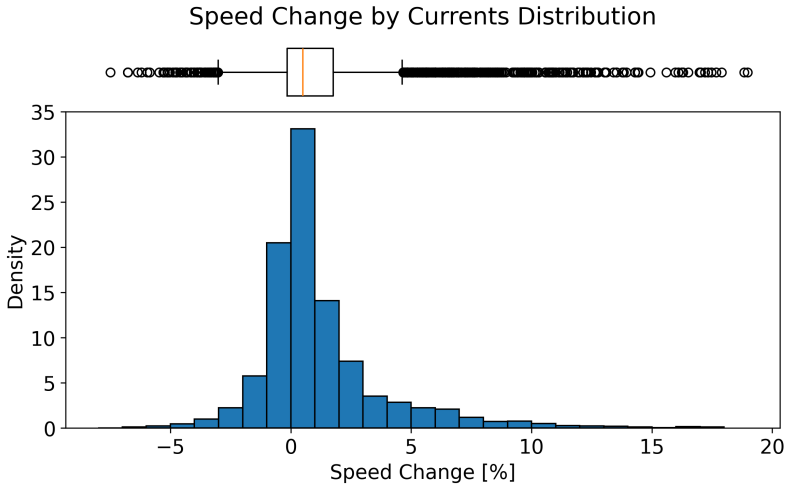


Figure 4.7: Distribution and boxplot of speed change from currents along the circumnavigation route.

Notice that in Figure 4.6, as we have modelled our wave effect reduction, it always results in a decrease in speed. As discussed above, the average speed reduction due to waves is minimal, around 0.3%. However, compared to the speed change from currents, it results in around a 0.6% increase in the vessel's speed, while reaching 5% at its maximum. This matches our results in Table 4.5, as gains over the circumnavigation route are higher on average and more consistent (smaller standard deviation) with currents than with waves.

Chapter 5

Discussion

In this thesis, we introduced a common benchmarking platform for weather routing algorithms, 3 heuristics based, optimal route finding algorithms, and a route smoothing algorithm to address and alleviate some shortcomings of the first 2 algorithms.

In `Hybrid Search`, it employs a technique of shooting different guesses centered around the direction towards the destination by solving the Zermelo's Navigation Equations. We alternate between and exploration and exploitation phases, together with some stopping criteria, `Hybrid Search` is able to output a chain of trajectories connecting the origin and destination ports. Each segment are locally optimal given the weather conditions, and the final trajectory is then smoothed by the Ferraro-Martín de Diego-Sato FMS algorithm, which converges to a time optimal trajectory [30].

In A*, we discretize the earth's surface into a hexagonal grid, and apply the famous A* algorithm to find the optimal path between two points. We modify the A* algorithm with different heuristic weights, and allowing neighbour jumping to provide more options to explore. We then refine the path with the FMS algorithm to smooth out sharp turns and find further savings in the route. We compare the performance of A*-FMS with `Hybrid Search` and CMA-ES across a common benchmarking platform, `WeatherRouting Bench 1.0` [94].

Lastly in BEES, we use Bézier curves to generate a route, as a few control points can generate very flexible and complex Bézier curves. We then apply an evolutionary algorithm, CMA-ES, to the control points of the Bézier curve to optimize the route, as Bézier curves are smooth, and reached local optimal, we did not see the need to apply FMS to the route.

Amongst these three algorithms, in the case of travel time, the best algorithm has consistently been BEES, closely followed by Hybrid Search. On the other hand, A* Graph Search yields poor results due to the limitations of working on a grid, resulting

in strange artifacts. This is clearly shown in routes which are not in open sea, like Panama-Houston. Moreover, to optimize for more precise scenarios would require a thinner mesh that would produce a huge number of nodes and a more complex computation.

On the other hand, our optimization algorithms have been shown to travel up to 9% more distance in exchange of reducing the total time. It is also worth to note that Hybrid Search finds shorter routes but makes less efficient use of currents, in contrast to CMA-ES which is the best for efficient routing.

In terms of computation time, Hybrid Search is the fastest, while A* is the slowest. This makes sense due to the extensive graph exploration required by the A* algorithm, considering the large number of points it needs to examine. CMA-ES is an evolutionary algorithm, so its computation time can be reduced by adjusting the stopping criteria to be more flexible, yet it still yields excellent results with a run time of around a minute and a half. Finally it can be seen how these results only depend on the algorithm and the distance, the computation time is not affected by the speed of the ship. This is normal, because the larger the space of solutions the longer it takes to explore them.

We also introduce a benchmarking platform for other academics and industry partners to contribute to the weather routing problem. Creating easily accessible weather data files to download, provided in our repository https://github.com/Weather-Routing-Research/weather-routing-benchmarks, identifiable, realistic sets of ODPs, and a realistic, but simple cost function to compute resistances due to waves and speed changes from currents, one can easily optimize and test routes to the benchmarks we computed with A*-FMS. Also providing comprehensive forecasting data and models from trusted meteorological sources, one can understand and optimize routes with greater accuracy and score the routes' performance with a comprehensive list of weather variables to identify strengths and weaknesses of their optimization algorithm.

We also introduce a new optimization algorithm, A*-FMS, that we utilize a hexagonal grid that tiles and discretize the surface of earth to apply A*, then move to a refining process by applying the FMS algorithm. This new optimization algorithm

serves as the first use case of the above-mentioned benchmark platform, to evaluate A*-FMS' performance across our benchmarks.

In addition, creating a comprehensive list of ODPs creates variability in weather conditions, considering global seasonal changes and heat transfers between water bodies, landmasses, and atmosphere. This evens the playing ground instead of utilizing special current streams in different regions of the ocean that may introduce inconsistency or bias to different algorithms. This would benefit our future benchmark versions, as projects such as wind-powered cargo ships [95] would require more accurate wind forecasts, while projects like slow-steaming [96], would benefit from more accurate currents and waves forecasts. Furthermore, with all-year-round departure dates, we can infer certain weather patterns and characteristics from optimized routes, reinforce weather forecasts with past seasonal experiments, and even taking advantage of foresight in weather patterns to set sail on certain periods of the year to achieve greater savings or avoid extreme sea conditions.

However our consumption models can be improved, not only from ship designs, but also better understanding and simulation of other friction and resistance terms would be crucial to a more detailed and accurate power prediction. In the aspects of ship designs, we are only taking into account of length, displacement and block coefficient of the vessel, however there are more variables that impact the performance of a vessel, such as the beam, drought, Prismatic coefficient, Midships Coefficient Factor, presence and design of Bulbous bow, and longitudinal centre of buoyancy [54], [55]. All of the previously mentioned is taking into account of vessel hull designs, and all of them are required to simulate and compute resistances, such as resistance of appendages, wave-making and wave-breaking resistance, bulbous bow pressure near water surface, and etc. [54]; these resistance terms would not only better inform future ship designs and refitting, but also offers an opportunity to use main engine power output, or equivalently, fuel consumption, which directly influences fuel and operational cost of a shipping vessel, as a viable cost function, instead of using time spent on route. This would better align us towards the sustainable development strategies set out by the IMO and the UN [97], [98]. As an added benefit, we can model ship motions and identify hazardous conditions that might put crew or cargo

in danger, which, in some cases, is of an higher concern for high sensitivity cargo that needs specialized handling.

Particularly, one challenge we face is uncertainties in weather forecasts. We assumed that we have access to the ocean forecast for the whole duration of the journey, as our benchmark is performed on past data reanalysis. However, when operating in real conditions, one only has access to up to 10 days in the future [75], with less accuracy the further ahead the forecast is. So uncertainties in weather forecasts demands an even more robust benchmarking platform.

As we have developed a platform to accelerate research and implementation of weather routing, we plan to address these above-mentioned shortcomings in our next version, adequately named Weather Routing Bench 2.0, we will introduce more accurate ship dynamics and power output modelling, as well as improved cost functions that better reflect real world costs and operation challenges. With these additions, we aim to provide a even more comprehensive set of benchmarks to better evaluate and score different weather routing algorithms.

Bibliography

[1] UNCTAD, *Review of maritime transport 2021*. UN, 2021.

[2] J. Faber *et al.*, *Fourth greenhouse gas study 2020*.

[3] J. Hüffmeier and M. Johanson, "State-of-the-art methods to improve energy efficiency of ships," *Journal of Marine Science and Engineering*, vol. 9, no. 4, 2021. DOI: 10.3390/jmse9040447.

[4] K. Wang, X. Yan, Y. Yuan, X. Jiang, X. Lin, and R. Negenborn, "Dynamic optimization of ship energy efficiency considering time-varying environmental factors," *Transportation Research Part D: Transport and Environment*, vol. 62, pp. 685–698, 2018. DOI: 10.1016/j.trd.2018.04.005.

[5] T.-H. Joung, S.-G. Kang, J.-K. Lee, and J. Ahn, "The imo initial strategy for reducing greenhouse gas(ghg) emissions, and its follow-up actions towards 2050," *Journal of International Maritime Safety, Environmental Affairs, and Shipping*, vol. 4, no. 1, pp. 1–7, 2020. DOI: 10.1080/25725084.2019.1707938. eprint: https://doi.org/10.1080/25725084.2019.1707938.

[6] S. Kotcharin and S. Maneenop, "Geopolitical risk and corporate cash holdings in the shipping industry," *Transportation Research Part E: Logistics and Transportation Review*, vol. 136, p. 101 862, 2020. DOI: https://doi.org/10.1016/j.tre.2020.101862.

[7] Z. Zhang and X.-M. Li, "Global ship accidents and ocean swell-related sea states," *Natural Hazards and Earth System Sciences*, vol. 17, no. 11, pp. 2041–2051, 2017. DOI: 10.5194/nhess-17-2041-2017.

[8] S. E. Roberts, D. Nielsen, A. Kotłowski, and B. Jaremin, "Fatal accidents and injuries among merchant seafarers worldwide," *Occupational medicine*, vol. 64, no. 4, pp. 259–266, 2014.

[9] T. P. Zis, H. N. Psaraftis, and L. Ding, "Ship weather routing: A taxonomy and survey," *Ocean Engineering*, vol. 213, p. 107 697, Oct. 2020. DOI: 10.1016/J.OCEANENG.2020.107697.

[10] R. W. James, *Application of wave forecasts to marine navigation*. New York University, 1957.

[11] G. L. Hanssen and R. W. James, "Optimum ship routing," *The Journal of Navigation*, vol. 13, no. 3, pp. 253–272, 1960.

[12] H. Hagiwara, "Weather routing of (sail-assisted) motor vessels," *Delft University of Technology, Delft*, 1989.

[13] M.-I. Roh, "Determination of an economical shipping route considering the effects of sea state for lower fuel consumption," *International Journal of Naval Architecture and Ocean Engineering*, vol. 5, pp. 246–262, Jun. 2013. DOI: 10.2478/IJNAOE-2013-0130.

[14] Y.-h. H. Lin, M.-C. C. Fang, and R. W. Yeung, "The optimization of ship weather-routing algorithm based on the composite influence of multi-dynamic elements," *Applied Ocean Research*, vol. 43, pp. 184–194, May 2013. DOI: 10.1016/j.apor.2013.07.010.

[15] G. J. Haltiner, H. D. Hamilton, and G. Arnason, "Minimal-time ship routing," *Journal of Applied Meteorology (1962-1982)*, pp. 1–7, 1962.

[16] N. A. Papadakis and A. N. Perakis, "Deterministic minimal time vessel routing," *Operations Research*, vol. 38, pp. 426–438, May 1990. DOI: 10.1287/opre.38.3.426.

[17] S. J. Ferraro, D. M. de Diego, and R. T. S. M. de Almagro, "Parallel iterative methods for variational integration applied to navigation problems," *IFAC-PapersOnLine*, vol. 54, pp. 321–326, Jan. 2021. DOI: 10.1016/J.IFACOL.2021.11.097.

[18] R. Bellman, "On the theory of dynamic programming," *Proceedings of the national Academy of Sciences*, vol. 38, no. 8, pp. 716–719, 1952.

[19] W. Shao, P. Zhou, and S. K. Thong, "Development of a novel forward dynamic programming method for weather routing," *Journal of Marine Science and Technology*, vol. 17, pp. 239–251, May 2012. DOI: 10.1007/s00773-011-0152-z.

[20] D. Sidoti *et al.*, "A Multiobjective Path-Planning Algorithm with Time Windows for Asset Routing in a Dynamic Weather-Impacted Environment," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, pp. 3256–3271, May 2017. DOI: 10.1109/TSMC.2016.2573271.

[21] C. P. Padhy, D. Sen, and P. K. Bhaskaran, "Application of wave model for weather routing of ships in the north indian ocean," *Natural Hazards*, vol. 44, pp. 373–385, May 2008. DOI: 10.1007/s11069-007-9126-1.

[22] K. Takashima, B. Mezaoui, and R. Shoji, "On the fuel saving operation for coastal merchant ships using weather routing," *Marine Navigation and Safety of Sea Transportation*, pp. 457–462, 2009.

[23] G. Mannarini, G. Coppini, P. Oddo, and N. Pinardi, "A prototype of ship routing decision support system for an operational oceanographic service," *TransNav : International Journal on Marine Navigation and Safety of Sea*

Transportation, vol. Vol. 7, pp. 53–59, Mar. 2013. DOI: 10.12716/1001.07.01.06.

[24] J. Szlapczynska, "Multi-objective weather routing with customised criteria and constraints," *Journal of Navigation*, vol. 68, pp. 338–354, May 2015. DOI: 10.1017/S0373463314000691.

[25] M. Grifoll, C. Borén, and M. Castells-Sanabra, "A comprehensive ship weather routing system using CMEMS products and A* algorithm," *Ocean Engineering*, vol. 255, p. 111 427, 2022.

[26] K. Kepaptsoglou, G. Fountas, and M. G. Karlaftis, "Weather impact on containership routing in closed seas: A chance-constraint optimization approach," *Transportation Research Part C: Emerging Technologies*, vol. 55, pp. 139–155, May 2015. DOI: 10.1016/j.trc.2015.01.027.

[27] H.-B. Wang, X.-G. Li, P.-F. Li, E. I. Veremey, and M. V. Sotnikova, "Application of real-coded genetic algorithm in ship weather routing," *The Journal of Navigation*, vol. 71, no. 4, pp. 989–1010, 2018.

[28] P. Krata and J. Szlapczynska, "Ship weather routing optimization with dynamic constraints based on reliable synchronous roll prediction," *Ocean Engineering*, vol. 150, pp. 124–137, May 2018. DOI: 10.1016/j.oceaneng.2017.12.049.

[29] G. Mannarini, M. L. Salinas, L. Carelli, N. Petacco, and J. Orović, "Visir-2: Ship weather routing in python," *Geoscientific Model Development*, vol. 17, no. 10, pp. 4355–4382, 2024.

[30] D. Precioso, R. Milson, L. Bu, Y. Menchions, and D. Gómez-Ullate, "Hybrid search method for zermelo's navigation problem," *Computational and Applied Mathematics*, vol. 43, no. 4, p. 250, 2024. DOI: 10.1007/s40314-024-02756-w.

[31] N. Charalambopoulos, E. Xidias, and A. Nearchou, "Efficient ship weather routing using probabilistic roadmaps," *Ocean Engineering*, vol. 273, p. 114 031, 2023. DOI: https://doi.org/10.1016/j.oceaneng.2023.114031.

[32] S. Grandcolas, "A metaheuristic algorithm for ship weather routing," *Operations Research*, vol. 3, p. 35, 2022.

[33] W. Zhao, H. Wang, J. Geng, W. Hu, Z. Zhang, and G. Zhang, "Multi-objective weather routing algorithm for ships based on hybrid particle swarm optimization," *Journal of Ocean University of China*, vol. 21, no. 1, pp. 28–38, 2022.

[34] S. Kuhlemann and K. Tierney, "A genetic algorithm for finding realistic sea routes considering the weather," *Journal of Heuristics*, vol. 26, no. 6, pp. 801–825, 2020.

[35] C.-L. Tsai, D.-T. Su, and C.-P. Wong, "An empirical study of the performance of weather routing service in the north pacific ocean," *Maritime Business Review*, vol. 6, no. 3, pp. 280–292, 2021.

[36] C. Gkerekos and I. Lazakis, "A novel, data-driven heuristic framework for vessel weather routing," *Ocean Engineering*, vol. 197, p. 106 887, 2020.

[37] R. Vettor, J. Szlapczynska, R. Szlapczynski, W. Tycholiz, and C. G. Soares, "Towards improving optimised ship weather routing," *Polish Maritime Research*, vol. 27, no. 1, pp. 60–69, 2020.

[38] D. Hendrycks *et al.*, *Measuring massive multitask language understanding*, 2021. arXiv: 2009.03300 [cs.CY].

[39] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi, "Hellaswag: Can a machine really finish your sentence?" *arXiv*, 2019. arXiv: 1905.07830 [cs.CL].

[40] T. B. Brown *et al.*, "Language models are few-shot learners," *arXiv*, 2020. arXiv: 2005.14165 [cs.CL].

[41] A. Q. Jiang *et al.*, *Mixtral of experts*, 2024. arXiv: 2401.04088 [cs.LG].

[42] H. Touvron *et al.*, *Llama: Open and efficient foundation language models*, 2023. arXiv: 2302.13971 [cs.CL].

[43] S. Rasp *et al.*, *Weatherbench 2: A benchmark for the next generation of data-driven global weather models*, 2024. arXiv: 2308.15560 [physics.ao-ph].

[44] European Centre for Medium-Range Weather Forecasts, *Ecmwf ifs cy41r2 high-resolution operational forecasts*, Boulder CO, 2016. DOI: 10.5065/D68050ZV.

[45] H. Hersbach *et al.*, "The era5 global reanalysis," *Quarterly Journal of the Royal Meteorological Society*, vol. 146, no. 730, pp. 1999–2049, 2020. DOI: https://doi.org/10.1002/qj.3803. eprint: https://rmets.onlinelibrary.wiley.com/doi/pdf/10.1002/qj.3803.

[46] R. Lam *et al.*, *Graphcast: Learning skillful medium-range global weather forecasting*, 2023. arXiv: 2212.12794 [cs.LG].

[47] E.U. Copernicus Marine Service Information (CMEMS), *Global Ocean Physical Analysis and Forecast.* https://data.marine.copernicus.eu/product/GLOBAL_ANALYSISFORECAST_PHY_001_024/description, Accessed on 20-02-2024, Marine Data Store (MDS). DOI: https://doi.org/10.48670/moi-00016.

[48] E.U. Copernicus Marine Service Information (CMEMS), *Global Ocean Waves Analysis and Forecast.* https://data.marine.copernicus.eu/product/

GLOBAL_ANALYSISFORECAST_WAV_001_027/description, Accessed on 30-05-2024, Marine Data Store (MDS). DOI: https://doi.org/10.48670/moi-00017.

[49] E. Zermelo, "Über das navigationsproblem bei ruhender oder veränderlicher windverteilung," *ZAMM - Journal of Applied Mathematics and Mechanic*, vol. 11, no. 2, pp. 114–124, 1931. DOI: https://doi.org/10.1002/zamm.19310110205. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/zamm.19310110205.

[50] T. P. Zis, H. N. Psaraftis, and L. Ding, "Ship weather routing: A taxonomy and survey," *Ocean Engineering*, vol. 213, p. 107 697, 2020.

[51] I. Uber Technologies, "H3 - Hexagonal Hierarchical Geospatial Indexing System," *Software*, 2018.

[52] D. Precioso, "Applications of machine learning and data science to the blue economy: Sustainable fishing and weather routing," Ph.D. dissertation, Universidad de Cádiz, 2023.

[53] S. J. Ferraro, D. Martín de Diego, and R. T. Sato Martín de Almagro, "A parallel iterative method for variational integration," *arXiv preprint arXiv:2206.08968*, 2022.

[54] J. Holtrop, G. Mennen, *et al.*, "An approximate power prediction method," *International shipbuilding progress*, vol. 29, no. 335, pp. 166–170, 1982.

[55] O. M. Faltinsen, "Prediction of resistance and propulsion of a ship in a seaway," in *13th Symposium on Naval Hydrodynamics, Tokyo*, 1980, pp. 505–529.

[56] K. Young-Joong, "A research on the approximate formulae for the speed loss at sea," *Journal of Ocean Engineering and Technology*, vol. 19, no. 2, pp. 90–93, 2005.

[57] R. Townsin and Y. Kwon, "Approximate formulae for the speed loss due to added resistance in wind and waves," *Engineering, Environmental Science*, 1983.

[58] M. Kim, O. Hizir, O. Turan, S. Day, and A. Incecik, "Estimation of added resistance and ship speed loss in a seaway," *Ocean Engineering*, vol. 141, pp. 465–476, 2017.

[59] M. Mittendorf, U. D. Nielsen, and H. B. Bingham, "Data-driven prediction of added-wave resistance on ships in oblique waves-a comparison between tree-based ensemble methods and artificial neural networks," *Applied Ocean Research*, vol. 118, p. 102 964, 2022. DOI: https://doi.org/10.1016/j.apor.2021.102964.

[60] Y. Yang *et al.*, "Research on ship resistance prediction using machine learning with different samples," *Journal of Marine Science and Engineering*, vol. 12, p. 556, Mar. 2024. DOI: 10.3390/jmse12040556.

[61] Y.-R. Kim and S. Steen, "Application of machine learning algorithms for predicting added resistance in arbitrary wave headings of a ship," *Ocean Engineering*, International Conference on Offshore Mechanics and Arctic Engineering, vol. Volume 5B: Ocean Engineering; Honoring Symposium for Professor Günther F. Clauss on Hydrodynamics and Ocean Engineering, V05BT06A026, Jun. 2022. DOI: 10.1115/OMAE2022-78433.

[62] A. F. Molland, S. R. Turnock, and D. A. Hudson, *Ship resistance and propulsion: practical estimation of ship propulsive power*, en. New York: Cambridge University Press, 2011.

[63] National Oceanic and Atmospheric Administration (NOAA), *Beaufort wind scale*, https://www.weather.gov/mfl/beaufort, Accessed on 10-07-2024.

[64] National Oceanic and Atmospheric Administration (NOAA), *Estimating wave height using wind speed during a tropical cyclone*, https://www.vos.noaa.gov/MWL/201512/waveheight.shtml, Accessed on 10-07-2024.

[65] J. C. McWilliams, "Chapter 14 - formulation of oceanic general circulation models," in *General Circulation Model Development*, ser. International Geophysics, D. A. Randall, Ed., vol. 70, Academic Press, 2000, pp. 421–456. DOI: https://doi.org/10.1016/S0074-6142(00)80062-5.

[66] A. Arakawa and V. R. Lamb, "Computational design of the basic dynamical processes of the ucla general circulation model," *Methods in Computational Physics: Advances in Research and Applications*, Methods in Computational Physics: Advances in Research and Applications, vol. 17, J. CHANG, Ed., pp. 173–265, 1977. DOI: https://doi.org/10.1016/B978-0-12-460817-7.50009-4.

[67] Q. Wang *et al.*, "The finite element sea ice-ocean model (fesom) v.1.4: Formulation of an ocean general circulation model," *Geoscientific Model Development*, vol. 7, no. 2, pp. 663–693, 2014. DOI: 10.5194/gmd-7-663-2014.

[68] L. M. Polvani, R. Scott, and S. Thomas, "Numerically converged solutions of the global primitive equations for testing the dynamical core of atmospheric gcms," *Monthly weather review*, vol. 132, no. 11, pp. 2539–2552, 2004.

[69] G. Madec *et al.*, "Nemo ocean engine," *Notes du Pôle de modélisation de l'Institut Pierre-Simon Laplace (IPSL)*, vol. v3.6-patch, no. 27, 2017. DOI: 10.5281/zenodo.3248739.

[70] M. Vancoppenolle *et al.*, *Si3, the nemo sea ice engine*, version 4.2release_doc1.0, Jan. 2023. DOI: 10.5281/zenodo.7534900.

[71] N. T. W. Group, *TOP - Tracers in Ocean Paradigm - The NEMO Tracers engine*, version v4.2.0, Mar. 2022. DOI: 10.5281/zenodo.1471700.

[72] European Centre for Medium-Range Weather Forecasts, "Part vii: Ecmwf wave model," European Centre for Medium-Range Weather Forecasts, Tech. Rep., 2016, Accessed: 2024-06-07.

[73] F. Ardhuin *et al.*, "Semi-empirical dissipation source functions for wind-wave models: Part i, definition, calibration and validation at global scales," *J. Phys. Oceanogr.*, vol. 40, Jan. 2010.

[74] P. Janssen, L. Aouf, A. Behrens, G. Korres, L. Cavalieri, K. Christiensen, O. Breivik, *MyWave Project*, Accessed: 2024-06-07, 2016.

[75] Copernicus, *Copernicus: Europe's eyes on earth*, https://www.copernicus.eu/es, Accessed: 2024-06-07.

[76] B. Rinauro, E. Begovic, F. Mauro, and G. Rosano, "Regression analysis for container ships in the early design stage," *Ocean Engineering*, vol. 292, p. 116 499, 2024.

[77] Unidata Program Center, *Network Common Data Form (NetCDF) User's Guide*, University Corporation for Atmospheric Research (UCAR), Boulder, Colorado, 2023.

[78] Q. Liu, Y. Wang, R. Zhang, H. Yan, J. Xu, and Y. Guo, "Arctic weather routing: A review of ship performance models and ice routing algorithms," *Frontiers in Marine Science*, vol. 10, p. 113 494, 2023. DOI: 10.3389/fmars.2023.1190164.

[79] C. Ducruet, "The geography of maritime networks: A critical review," *Journal of Transport Geography*, vol. 88, p. 102 824, 2020. DOI: https://doi.org/10.1016/j.jtrangeo.2020.102824.

[80] D. L. Alderson, D. Funk, and R. Gera, "Analysis of the global maritime transportation system as a layered network," *Journal of Transportation Security*, vol. 13, pp. 291–325, 2020.

[81] N. G. Álvarez, B. Adenso-Díaz, and L. Calzada-Infante, "Maritime traffic as a complex network: A systematic review," *Networks and Spatial Economics*, vol. 21, no. 2, pp. 387–417, 2021.

[82] J. Ge, Q. Zhang, Z. Wan, *et al.*, "Regional operating patterns of world container shipping network: A perspective from motif identification," *Physica A: Statistical Mechanics and its Applications*, vol. 607, p. 128 171, 2022.

[83] E. W. Dijkstra *et al.*, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[84] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, pp. 100–107, 1968. DOI: 10.1109/TSSC.1968.300136.

[85] Y. Huang, Y. Xiao, H. Wang, and H. Yi, "A rapid globe-wide shortest route planning algorithm based on two-layer oceanic shortcut network considering great circle distance," *Ocean Engineering*, vol. 287, p. 115 761, 2023.

[86] R. Ebendt and R. Drechsler, "Weighted a* search - unifying view and application," *Artificial Intelligence*, vol. 173, pp. 1310–1342, Sep. 2009. DOI: 10.1016/J.ARTINT.2009.06.004.

[87] C. Frasinaru and M. Raschip, "Greedy best-first search for the optimal-size sorting network problem," *Procedia Computer Science*, vol. 159, pp. 447–454, Jan. 2019. DOI: 10.1016/J.PROCS.2019.09.199.

[88] D. Freedman, R. Pisani, and R. Purves, "Statistics (international student edition)," *Pisani, R. Purves, 4th edn. WW Norton & Company, New York*, 2007.

[89] N. Hansen, S. D. Müller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es)," *Evolutionary Computation*, vol. 11, no. 1, pp. 1–18, 2003.

[90] W. Boehm and A. Müller, "On de Casteljau's algorithm," *Computer Aided Geometric Design*, vol. 16, no. 7, pp. 587–605, 1999. DOI: https://doi.org/10.1016/S0167-8396(99)00023-0.

[91] N. Buduma and N. Locascio, *Fundamentals of Deep Learning: Designing Next-generation Machine Intelligence Algorithms*. O'Reilly Media, 2017.

[92] V. Morales-Márquez, A. Orfila, G. Simarro, and M. Marcos, "Extreme waves and climatic patterns of variability in the eastern north atlantic and mediterranean basins," *Ocean Science*, vol. 16, no. 6, pp. 1385–1398, Nov. 2020. DOI: 10.5194/os-16-1385-2020.

[93] E. Gleeson, C. Clancy, L. Zubiate, J. Janjić, S. Gallagher, and F. Dias, "Teleconnections and extreme ocean states in the northeast atlantic ocean," *Advances in Science and Research*, vol. 16, pp. 11–29, 2019. DOI: 10.5194/asr-16-11-2019.

[94] J. J. de la Jara *et al.*, "Weatherrouting bench 1.0: Towards comparative research in weather routing," *Manuscript submitted for publication*, 2024.

[95] European Climate, Infrastructure and Environment Executive Agency, *New wind powered cargo ship sets sail*, Aug. 2023.

[96] L. M. A. Nestor Goicoechea, "Optimial slow steaming speed for container ships under the eu emission trading system," *Energies*, vol. 14, 2021. DOI: {https://doi.org/10.3390/en14227487}.

[97] International Maritime Organization, *IMO and Sustainable Development*, 2017.

[98] International Maritime Organization, *Development of the strategic plan for 2024 to 2029*, 2023.

Appendices

.1 Published and Submitted Papers

The following papers have been published or submitted to journals that I have co-authored:

- D. Precioso, R. Milson, L. Bu, *et al.*, "Hybrid search method for zermelo's navigation problem," *Computational and Applied Mathematics*, vol. 43, no. 4, p. 250, 2024. DOI: 10.1007/s40314-024-02756-w

  - This study was conducted with collaboration with Canonical Green in the summer of 2023, when the first MITACS internship took place.
  - In this work we applied a shooting method as an Initial Value Problem to solve Zermelo's Navigation Equations, both in synthetic vectorfields and real ocean currents.
  - My role was developing the Zermelo's Navigation Equations in Python, parallelizing the numerical differential equations solver, and generating the pipeline for the software package.
  - Section 3.1 is largely based on this work.

- J. J. de la Jara, D. Precioso, L. Bu, *et al.*, "Weatherrouting bench 1.0: Towards comparative research in weather routing," *Manuscript submitted for publication*, 2024

  - This study was conducted during the second MITACS internship, in the summer of 2024, with Canonical Green in Madrid, Spain.
  - In this work we introduced a common testing platform for weather routing algorithms, and applied the A^* search algorithm to solve the optimal path problem in a hexagonal grid, and then refined the path with the FMS algorithm. We compared the various characteristics of the benchmarks and the performance of the algorithm.
  - My role in this work was running simulations on our A^* -FMS algorithm, between different ports, different times of the year, and different vessel speeds. I ran scripts to cross analyze and generate results. I also

contributed to developing the cost function 2.1.1 and the writing of the paper.

– Chapters 2, 4, Sections 3.2 are based on this work.

.2 Derivation of Zermelo's equations

.2.1 Zermelo's Navigation Problem on the plane

We are dealing here with a constrained optimization problem whose Lagrangian function has the form

$$L = \dot{t} + \lambda_1(\dot{x}_1 - (V\cos\alpha + w_1)\dot{t}) + \lambda_2(\dot{x}_2 - (V\sin\alpha + w_2)\dot{t}). \tag{1}$$

The goal is to find trajectories $x(s),\ \dot{x}(s) = x'(s),\ t(s),\ \dot{t}(s) = t'(s) > 0,\ \alpha(s)$ with fixed end-points that minimize $t(s_1) - t(s_0) = \int_{s_0}^{s_1} L ds$, and obey constraints

$$\begin{aligned} \dot{x}_1 &= (V\cos\alpha + w_1)\dot{t} \\ \dot{x}_2 &= (V\sin\alpha + w_2)\dot{t} \end{aligned} \tag{2}$$

The quantities λ_1, λ_2 are known as Lagrange multipliers. As we now show, their form is determined by the Euler-Lagrange equations associated with the above Lagrangian, namely

$$\frac{dL_{\dot{t}}}{ds} = 0 \tag{3}$$

$$L_{x_i} - \frac{dL_{\dot{x}_i}}{ds} = 0 \quad i = 1, 2 \tag{4}$$

$$L_\alpha = 0, \tag{5}$$

Equation (3) gives

$$\frac{d}{ds}\left(\lambda_1(V\cos\alpha + w_1) + \lambda_2(V\sin\alpha + w_2)\right) = 0$$

which implies that

$$\lambda_1(V\cos\alpha + w_1) + \lambda_2(V\sin\alpha + w_2) = C \tag{6}$$

where $C \neq 0$ is a constant. Equation (5) gives

$$\lambda_1 \sin\alpha - \lambda_2 \cos\alpha = 0. \tag{7}$$

Together, (6) (7) determine the form of the Lagrange multipliers, namely:

$$\lambda_1 = \frac{C\cos\alpha}{V + w_1\cos\alpha + w_2\sin\alpha} \tag{8}$$

$$\lambda_2 = \frac{C\sin\alpha}{V + w_1\cos\alpha + w_2\sin\alpha} \tag{9}$$

Going forward, we re-parameterize all curves with respect to time t so that

$$\frac{d}{dt} = \frac{1}{\dot{t}}\frac{d}{ds}.$$

E-L equations (4) give the dynamics of the Lagrange multipliers, namely

$$\frac{d\lambda_1}{dt} = -\lambda_1 w_{1,1} - \lambda_2 w_{2,1} \tag{10}$$

$$\frac{d\lambda_2}{dt} = -\lambda_1 w_{1,2} - \lambda_2 w_{2,2} \tag{11}$$

Rewriting (7) as

$$\tan\alpha = \frac{\lambda_2}{\lambda_1},$$

and taking derivatives, gives

$$\begin{aligned}
\sec^2(\alpha)\frac{d\alpha}{dt} &= \frac{d}{dt}\left(\frac{\lambda_2}{\lambda_1}\right) \\
\left(\frac{\lambda_1^2+\lambda_2^2}{\lambda_1^2}\right)\frac{d\alpha}{dt} &= \frac{1}{\lambda_1^2}\left(-\lambda_2\frac{d\lambda_1}{dt} + \lambda_1\frac{d\lambda_2}{dt}\right) \\
\frac{d\alpha}{dt} &= \frac{\lambda_2^2 w_{2,1} + \lambda_1\lambda_2(w_{1,1}-w_{2,2}) - \lambda_1^2 w_{1,2}}{\lambda_1^2+\lambda_2^2} \\
\frac{d\alpha}{dt} &= \sin^2(\alpha)\, w_{2,1} + \sin(\alpha)\cos(\alpha)\,(w_{1,1}-w_{2,2}) - \cos^2(\alpha)\, w_{1,2}
\end{aligned} \tag{12}$$

.2.2 Zermelo's Navigation Problem on the sphere

The modified Lagrangian takes the form

$$\begin{aligned} L = \dot{t} + \lambda_1 \left(\dot{\theta} - K^{-1} \sec(\kappa\phi) \left(V \cos(\kappa\alpha) + w_1 \right) \dot{t} \right) \\ + \lambda_2 \left(\dot{\phi} - K^{-1} \left(V \sin(\kappa\alpha) + w_2 \right) \dot{t} \right) \end{aligned}$$

The E-L equations (4) now read

$$K \frac{d\lambda_1}{dt} = -\sec(\kappa\phi)\lambda_1 w_{1,1} - \lambda_2 w_{2,1} \tag{13}$$

$$K \frac{d\lambda_2}{dt} = -\lambda_1 \kappa \sec(\kappa\phi)\tan(\kappa\phi)(V\cos(\kappa\alpha) + w_1) - \lambda_1 \sec(\kappa\phi) w_{1,2} - \lambda_2 w_{2,2} \tag{14}$$

In the current setting (5) gives

$$\tan(\kappa\alpha) = \frac{\lambda_2}{\lambda_1}\cos(\kappa\phi)$$

Taking d/dt yields

$$\begin{aligned} \kappa \sec^2(\kappa\alpha)\frac{d\alpha}{dt} &= \frac{\cos(\kappa\phi)}{\lambda_1^2}\left(-\lambda_2 \frac{d\lambda_1}{dt} + \lambda_1 \frac{d\lambda_2}{dt} \right) \\ &\quad - \frac{\kappa}{K}\tan(\kappa\alpha)\tan(\kappa\phi)(V\sin(\kappa\alpha) + w_2) \\ \kappa K \sec^2(\kappa\alpha)\frac{d\alpha}{dt} &= \frac{\lambda_2}{\lambda_1} w_{1,1} + \frac{\lambda_2^2}{\lambda_1^2}\cos(\kappa\phi) w_{2,1} - w_{1,2} - \frac{\lambda_2}{\lambda_1}\cos(\kappa\phi) w_{2,2} \\ &\quad - \kappa\tan(\kappa\phi)(V\cos(\kappa\alpha) + w_1) \\ &\quad\quad - \kappa\tan(\kappa\alpha)\tan(\kappa\phi)(V\sin(\kappa\alpha) + w_2) \\ \kappa K \sec^2(\kappa\alpha)\frac{d\alpha}{dt} &= \sec(\kappa\phi)\tan(\kappa\alpha) w_{1,1} + \sec(\kappa\phi)\tan^2(\kappa\alpha) w_{2,1} - w_{1,2} \\ &\quad - \tan(\kappa\alpha) w_{2,2} - \tan(\kappa\phi)(V\cos(\kappa\alpha) + w_1) \\ &\quad\quad - \kappa\tan(\kappa\alpha)\tan(\kappa\phi)(V\sin(\kappa\alpha) + w_2) \\ \kappa K \frac{d\alpha}{dt} &= \begin{bmatrix} \cos(\kappa\alpha) & \sin(\kappa\alpha) \end{bmatrix} \begin{bmatrix} \sec(\kappa\phi) w_{1,1} & w_{1,2} \\ \sec(\kappa\phi) w_{2,1} & w_{2,2} \end{bmatrix} \begin{bmatrix} \sin(\kappa\alpha) \\ -\cos(\kappa\alpha) \end{bmatrix} \\ &\quad - \cos(\kappa\alpha)\tan(\kappa\phi)(V + \cos(\kappa\alpha) w_1 + \sin(\kappa\alpha) w_2) \end{aligned} \tag{15}$$

.3 Euler-Lagrange equations

.3.1 Continuous Euler-Lagrange equations

Define an action functional along a curve $q(t)$ in n-dimensional space with fixed end points as follows,

$$J(q(t)) = \int_a^b L(t, q(t), \dot{q}(t))dt, \quad q(a) = \alpha, \quad q(b) = \beta. \tag{16}$$

The function $L(t, q(t), \dot{q}(t))$ is called the Lagrangian of the optimization problem. The classical problem in the Calculus of Variations is to minimize J by subjecting $q(t)$ to suitable constraints.

A necessary condition for minimization is that the variation δJ vanishes for all possible variations of the trajectory $\delta q = \epsilon\phi$, where $\phi(t)$ vanishes at the endpoints, and ϵ is the variational parameter. From the functional (16), define

$$h(\epsilon) = J(q + \epsilon\phi) = \int_a^b L(t, q(t) + \epsilon\phi(t), \dot{q}(t) + \epsilon\dot{\phi}(t))dt.$$

Now differentiate and use the smoothness of L to interchange the derivative and the integral to get

$$\begin{aligned} h'(\epsilon) = \frac{d}{d\epsilon}J(q + \epsilon\phi) &= \int_a^b \frac{d}{d\epsilon}L\left(t, q(t) + \epsilon\phi(t), \dot{q}(t) + \epsilon\dot{\phi}(t)\right)dt \\ &= \int_a^b \phi(t)\left[\frac{\partial L}{\partial q}\left(t, q(t) + \epsilon\phi(t), \dot{q}(t) + \epsilon\dot{\phi}(t)\right)\right. \\ &\qquad \left. + \dot{\phi}(t)\frac{\partial L}{\partial \dot{q}}\left(t, q(t) + \epsilon\phi(t), \dot{q}(t) + \epsilon\dot{\phi}(t)\right)\right]dt. \end{aligned}$$

Now setting $\epsilon = 0$ and using our definition of the variational derivative yields

$$\delta J(q)(\phi) = \int_a^b \left[\phi(t)\frac{\partial L}{\partial q}(t, q, \dot{q}) + \dot{\phi}(t)\frac{\partial L}{\partial \dot{q}}(t, q, \dot{q})\right]dt. \tag{17}$$

This functional is known as the *first variation* of J. In order to obtain an explicit formula for δJ, we need the integral on the right side of the above equation to be

linear in $\phi(t)$. We can accomplish this via integration by parts.

$$\int_a^b \dot{\phi}(t)\frac{\partial L}{\partial \dot{q}}(t,q,\dot{q})dt = \left[\phi(t)\frac{\partial L}{\partial \dot{q}}(t,q(t),\dot{q}(t))\right]_{t=a}^{t=b} - \int_a^b \phi(t)\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}(t,q,\dot{q})\right)dt$$

Since $\phi(b) = \phi(a) = 0$, by assumption, we obtain the following formula for the first variation:

$$\delta J(q)(\phi) = \int_a^b \left[\frac{\partial L}{\partial q} - \frac{d}{dt}\frac{\partial L}{\partial \dot{q}}\right]\phi(t)dt.$$

Therefore, in order for $\delta J(\phi)$ to vanish for all ϕ, the critical trajectory $q(t)$ must satisfy the *Euler-Lagrange equations*

$$\frac{\partial L}{\partial q} - \frac{d}{dt}\frac{\partial L}{\partial \dot{q}} = 0. \tag{18}$$

.3.2 Discrete Euler-Lagrange equations

Now consider two positions: q_0 and q_1, and a time step $h > 0$. We discretize a continuous Lagrangian $L(q,\dot{q})$ by assuming that q_1, q_0 are close together so that $\dot{q}$ can be approximated by $(q_1 - q_0)/h$. This allows us to define the following discrete Lagrangian

$$L_d(q_0,q_1;h) := \frac{h}{2}\left(L\left(q_0,\frac{q_1-q_0}{h}\right) + L\left(q_1,\frac{q_1-q_0}{h}\right)\right),$$

which approximates the action integral along a straight trajectory from q_0 to q_1. In the discrete Calculus of Variations, we replace a continuous curve $q(t)$ with a piecewise linear curve determined by a sequence of points $\{q_k\}_{k=0}^N$ with h units of time required to go from q_k to q_{k+1}. We will now calculate the discrete action over this sequence by summing the discrete Lagrangian.

$$J_d = \sum_{k=0}^{N-1} L_d(q_k,q_{k+1};h).$$

We now vary the trajectory by $dq = \{dq_k\}_{k=0}^N$ with $dq_0 = dq_N = 0$ in order to fix the boundary points q_0, q_N. Note that we use dq rather than $\epsilon\phi$ to describe the variation because the discretized system has finite degrees of freedom. The variation of the

discrete action can now be given as

$$\begin{aligned} dJ_d &= \sum_{j=1}^{N-1} \frac{\partial}{\partial \boldsymbol{x_j}} \left(\sum_{k=0}^{N-1} L_d(q_k, q_{k+1}; h) \right) dq_j \\ &= \sum_{k=0}^{N-1} \left[D_1 L_d(q_k, q_{k+1}; h) dq_k + D_2 L_d(q_k, q_{k+1}; h) dq_{k+1} \right] \end{aligned}$$

Recall that each $\boldsymbol{x_j} = (q_{j1}, \ldots, q_{jn})$ is a point in n-dimensional space, so that $\partial/\partial \boldsymbol{x_j}, D_1, D_2$ are actually n-vectors of partial derivative operators. Rearranging the above sum (this corresponds to the integration by parts step in the continuous case) we obtain

$$dJ_d = \sum_{k=1}^{N-1} \left[D_2 L_d(q_{k-1}, q_k; h) + D_1 L_d(q_k, q_{k+1}; h) \right] dq_k.$$

If we require that the variation of the action is 0 for all dq_k, then we obtain the discrete Euler-Lagrange equations

$$D_2 L_d(q_{k-1}, q_k; h) + D_1 L_d(q_k, q_{k+1}; h) = 0, \quad k = 1, \ldots, N-1. \tag{19}$$