# Multi-Objective Ship Weather Routing: An Evolutionary Approach

Master Thesis Econometrics & Management Science

*Operations Research & Quantitative Logistics*


*Author*
J.J. Seuren

*Student number*
482672

*Supervisor*
Prof. dr. ir. R. Dekker

*Co-reader*
Dr. P.C. Bouman

Erasmus University of Rotterdam
Erasmus School of Economics


November 16, 2020

# Abstract

Over the past years, the tramp shipping industry is experiencing an increased awareness of sustainable shipping. Stricter regulations are imposed on ship fuel consumption, leading to higher operational costs. Costs further increase due to adverse environmental conditions, which represent the major causes of delays. Tactical planning of ship routes considering these effects is necessary to maintain a competitive advantage in the tramp shipping industry.

The problem of Multi-objective Ship Weather Routing (MOSWR) is to seek a set of optimum routes for ocean-going vessels that allow a variable engine speed and consider different aspects along the route trajectory, such as (1) dynamic environmental conditions, (2) high-cost areas, and (3) shipping canals. The term optimum means minimum in both fuel cost and travel time. Since these are opposing objectives, a set of Pareto-optimal solutions is required to assist the decision-maker in making an informed selection of a route with an optimal balance between travel time and fuel costs.

Ship route optimization has been the subject of research for more than six decades. Many of these studies follow a single-objective approach, find inaccurate solutions, or have high computational complexity, making them unsuitable for practical applications. For this reason, we present an evolutionary approach for solving the MOSWR problem, capable of handling all its different aspects.

Considering these aspects, our approach effectively reduces both travel time and fuel costs and finds Pareto-optimal routes that

- are initialized on a Geodesic Discrete Global Grid with variable density near obstacles,
- are realistic, smooth, and not restricted to an arbitrary grid,
- take advantage of (avoid) favorable (adverse) ocean currents,
- avoid adverse weather conditions,
- consider Emission Control Areas and shallow waters,
- enable a variable engine speed profile, and
- either pass or avoid major shipping canals, like the Panama or Suez Canal.

Furthermore, we test three well-known multi-objective evolutionary algorithms on their performance on the MOSWR problem: M-PAES, NSGA-II, and SPEA2. SPEA2 proves to be computationally fast but does not find diversified Pareto fronts. NSGA-II has the shortest computation time for several problem instances and performs equally well to M-PAES in terms of Pareto performance.

We present several computation speed improvements, including an improved spatial indexing technique to test for route intersections. Our algorithm is faster than existing multi-objective approaches, requiring 3 min – 5 min computation time on a standard PC for large instances. All the above make the presented approach the best choice for a practical solution method to the Multi-Objective Ship Weather Routing problem.

# Contents

# List of abbreviations

| | |
|---|---|
| **2DDP** | Two-dimensional dynamic programming |
| **3DDP** | Three-dimensional dynamic programming |
| **BFT** | Beaufort number |
| **DGG** | Discrete Global Grid |
| **EA** | Evolutionary algorithm |
| **ECA** | Emission Control Area |
| **GDGG** | Geodesic Discrete Global Grid |
| **GFS** | Global Forecast System |
| **GRIB** | General Regularly-distributed Information in Binary form |
| **GSHHG** | Global Self-consistent, Hierarchical, High-resolution Geography |
| **HFO** | Heavy Fuel Oil |
| **IMO** | International Maritime Organization |
| **KC** | Kuroshio Current |
| **MBR** | Minimum bounding rectangle |
| **M-PAES** | Memetic Pareto Archived Evolution Strategy |
| **MOEA** | Multi-objective evolutionary algorithm |
| **MOSWR** | Multi-Objective Ship Weather Routing |
| **NSGA-II** | Non-dominated Sorting Genetic Algorithm II |
| **SPEA2** | Strength Pareto Evolutionary Algorithm 2 |
| **STR** | Sort-Tile-Recursive |
| **ULSFO** | Ultra-Low Sulfur Fuel Oil |
| **VLSFO** | Very-Low Sulfur Fuel Oil |
| **WBC** | Western boundary current |

# 1 Introduction

Seaborne cargo transportation is responsible for moving over 80 percent of global trade (UNCTAD, 2019), and it is the only cost-effective option for transporting large volumes between the continents. As of 2018, world trade volumes have expanded on average 4.1 % annually since 2000 (WTO, 2019), and an annual growth rate of 3.4 % is projected for the period 2019 – 2024 (UNCTAD, 2019). Over the last decades, demands in the maritime shipping industry are increasing while experiencing an increased awareness of the effect of fuel usage on operational costs and environmental emissions. Given the enormous and growing volume of goods transported by sea each year in combination with global trends in sustainable shipping, accurate ship route optimization becomes more compelling in today's shipping industry.

## 1.1 Problem field

Generally, commercial cargo shipping differentiates into liner shipping and tramp shipping. Typically, in the liner mode, container vessels travel according to published schedules and transport cargo on the associated routes, comparable to a bus line. In tramp shipping, a ship trades on the spot market and does not have a fixed schedule or published port of calls so that for each voyage a route planning is made to order. Remaining with the analogy, this transportation mode compares to a taxi service. In 2018, dry bulk shipments, together with tanker trade shipments (oil, gas, and chemicals), accounted for over 80 percent of total maritime trade (UNCTAD). As these cargo types are mainly transported by tramp shipping, the significance of tramp trade in the world trade is apparent. Due to its size, cost-effectiveness and ease of entry and exit, the tramp market is highly competitive (CRSL, 2015). For this reason, savings through targeted planning of ship routes may provide a leading competitive advantage.

Targeted planning of a tramp ship requires accurate estimations of the duration of port-to-port transits. The accuracy of these estimations of travel time depends mainly on the accuracy of models for weather forecasts and ship performance. The inaccuracy of arrival time estimations requires scheduling idle time at the destination port to avoid delay penalties. Specifically, adverse weather conditions represent one of the significant causes of delay in the shipping industry (Notteboom, 2006).

Environmental conditions affecting the ship navigation are wind, waves, and ocean current. Wind and waves generate additional resistance to a ship, resulting in a loss of ship speed, whereas ocean currents affect both the speed and course of a ship. With an accurate forecast of environmental conditions, a ship routing can be performed in which the estimated travel time is more reliable and is minimized. Accurate estimations of travel time

result in fewer costs for both waiting and delay at the destination, as well as a reduction in voyage cost.

Voyage costs involve fuel costs, canal passing fees, and other costs, such as port-related costs, repair, maintenance, manning, and extra insurances for piracy risks. Despite the steep decrease in fuel prices due to the oil crisis in March 2020, fuel costs typically constitute more than half of the voyage cost. Excluding port charges fixed for a voyage, canal fees are the second-highest contribution to the voyage costs. The total fuel cost depends on the fuel price, fuel consumption rate, and time sailing. The total canal fees depend on the canal, and the ship's dimensions, loading, and type.

Not only the economic impact but also the environmental impact of fuel consumption in maritime trade is tremendous. Shipping activities contribute to 14 % of global nitrogen oxides emissions and 5 % of sulfur oxide (SOx) emissions. With the increasing social interest in slowing down climate change, authorities worldwide have begun to enforce strict regulations regarding air and sea pollution emanating from fuel consumption by ocean-going vessels. Regulations from the International Maritime Organization (IMO) that came into force from January 2020 require these vessels to globally reduce SOx emissions. Many shipping companies prefer to achieve this reduction by switching to Very-Low Sulfur Fuel Oil (VLSFO). At the time of writing, the price of VLSFO is, on average, 28.7 % higher compared to standard Heavy Fuel Oil (HFO)[1]. Consequently, the share of fuel costs in the total voyage cost increased significantly as of 2020.

Even more stringent regulations on SOx emissions are established within Emission Control Areas (ECAs) defined by the IMO (*MARPOL*, 2005). In this research, we assume a ship to apply fuel-switching so that the fuel costs within ECAs are significantly higher than fuel costs outside ECAs. In response to the recently developed spread in fuel prices and an increase in awareness of sustainable development, shipping companies try to find ways to reduce fuel consumption in global regions and specifically within ECAs.

Companies that can afford longer travel times often choose to slow-steam. This concept of reducing the ship's speed by a few knots saves fuel while also reducing SOx emissions. However, for many shipping companies, the reduction in fuel costs that comes with slow-steaming does not outweigh the lost profit due to increased travel time. An optimal balance between short travel time and low fuel cost can be obtained by effectively speeding up and slowing down in specific regions, to avoid, for example, additional costs within ECAs or speed reductions due to storms.

## 1.2 Relevance

Finding routes that are not only cost-efficient but also ensure the on-time arrival of the ship is a complex task that is difficult to do by hand. Especially when considering environmental

[1]Three-month average fuel prices in the period of May 1 to July 31, 2020, are retrieved from https://shipandbunker.com, accessed on August 13, 2020.

conditions, variable engine speed, and greenhouse gas emission regulations. For this reason, an automatic ship routing that minimizes travel time and fuel cost considering earlier mentioned elements is of high value for a tramp shipping company.

Several optimization methods have been developed to optimize ship routes in an ocean-crossing voyage. The first was the "isochrone method", proposed by James (1957) and later modified by Klompstra et al. (1992) and Hagiwara and Spaans (1987) to be more suitable for computerization. Isochrone methods are based on manual navigation techniques and, therefore, have long been used in commercial ship navigation software. However, due to their single-objective approach and constant engine speed requirement, it is not possible to minimize travel time and fuel cost simultaneously.

A few years later, the calculus of variations was applied to the ship weather routing problem (Bleick & Faulkner, 1965; Haltiner et al., 1962). Its popularity lay within its exact, continuous approach to finding an optimal ship route. With the use of second-order derivatives, the method introduces unacceptable errors in the solution values if there are inaccuracies in weather forecast data. For this and other practical issues, the method's popularity decreased in the following years.

Dynamic programming proved to be suitable method for ship routing, as it handles constraints easily and can take three control variables: ship's position (longitude, latitude) and engine speed. Due to its grid dependency, a high grid resolution is required to obtain accurate solution. This increases the computational complexity, requiring simplifications of the route evaluation as proposed in (Shao et al., 2012; Tanaka & Kobayashi, 2019; Zaccone et al., 2018). A similar and more common approach to ship navigation uses graph search methods (Montes, 2005; Padhy, 2008; Sen & Padhy, 2015). These methods are generally fast but less versatile, as they optimize a single-objective and do not return smooth routes.

More modern is a multi-objective evolutionary approach. Hinnenthal and Clauss (2010), Marie and Courteille (2009), and Szapczyska (2015) use a multi-objective evolutionary algorithm to find a set of routes that are Pareto-optimal for different (opposing) criteria. As this approach finds such a set in a single run and it is proved to be flexible in practical applications, multi-objective evolutionary algorithm are the obvious choice for the Multi-Objective Ship Weather Routing problem, presented in this study. Overall, methods proposed earlier have limitations in computational time or do not consider all aspects of the problem presented in this thesis.

## 1.3 Research objectives

With this study, we aim to answer the research question as stated below.

> *Can we develop a solution approach for the Multi-Objective Ship Weather Route problem which considers different routing aspects and has better performance than existing ship routing methods?*

Performance is measured in terms of computation time and indicators designed for multi-objective optimization methods. Moreover, the presented solution method is evaluated based on its flexibility in real-world applications, in terms of handling different sailing regions, environmental conditions, and ship types. To provide a substantiated answer to our research question, we state the following subquestions.

*SQ1. What is the effect of ocean currents on the set of solutions to the MOSWR problem?*

*SQ2. What is the effect of weather on the set of solutions to the MOSWR problem?*

*SQ3. What is the effect of Emission Control Areas on the set of solutions to the MOSWR problem?*

*SQ4. What is the effect of a variable engine speed profile in combination with ocean current, weather effects, or Emission Control Areas on the set of solutions to the MOSWR problem?*

*SQ5. What is the performance of a selection of multi-objective evolutionary algorithms on solving the MOSWR problem considering different environmental conditions?*

The first four subquestions include four different aspects of the MOSWR problem and SQ5 covers the performance assessment of the solution method presented in this thesis. This work comprises two major parts: (1) the formulation of the Multi-Objective Ship Weather Routing (MOSWR) problem, and (2) an evolutionary approach to solve the MOSWR problem.

The MOSWR problem is to seek a set of optimum ship routes under specified departure and destination, with variable engine speed during the voyage and considering time-varying environmental conditions and high-cost areas along the ship trajectory. The term optimum means a minimum of fuel costs and travel time. Since these are opposing objectives, a set of Pareto optimum solutions is required to support the end-user in an informed selection of a route that has an optimal balance of travel time and fuel cost. A Pareto optimum solution implies a feasible solution with minimum objective values, such that an improvement of one objective is only possible by impairing another. Besides variable engine speed and forecasted environments, the MOSWR problem considers high-cost areas and major shipping canals, e.g., the Panama Canal and the Suez Canal. With the inclusion of canals, multiple route options may exist that include or avoid these canals, enabling decision-making on the trade-off between routes with longer travel time and fast routes that include canal fees.

As a framework to our solution approach for the MOSWR problem, we use a multi-objective evolutionary algorithm (MOEA) with specialized mutation and crossover operations for the MOSWR problem. We test and compare the performance of three well-known MOEAs: (1) the Non-dominated Sorting Algorithm II (NSGA-II), an elitist approach presented by Deb et al. (2002), (2) the improved Strength Pareto Evolutionary Algorithm (SPEA2), another elitist approach by Zitzler et al. (2001), and (3) the Memetic Pareto

Archived Evolution Strategy (M-PAES) (Knowles & Corne, 2000), which has a stronger focus on local search.

This work presents a ship routing algorithm that has practical applications in the early voyage planning phase and onboard during a voyage. Our solution approach contains the following novel components:

- A multi-objective routing algorithm for minimum travel time and fuel cost with customizable support for different constraints and optimization criteria
- Time-varying environmental conditions, i.e., weather and ocean currents, affecting the ship course and speed.
- High-cost areas, such as ECAs and shallow waters, which either induce additional cost or incur optimization penalties.
- Variable engine speed during the voyage, allowing for, e.g., slow-steaming.
- Multiple route options, including routes that either pass or avoid major shipping canals.
- A route is smooth and realistic as it is defined in continuous space and does not have preset values for the coordinates of the waypoints along the path.
- Reduced computation time by using a modified spatial indexing technique for determining route feasibility and cost factors.

## 1.4 Thesis outline

The remainder of this thesis is outlined as follows. First, Chapter 2 reviews the existing ship weather routing methods and related work. Then, Chapter 3 provides a description of different elements of the MOSWR problem and formulates the problem such that it is generic for different types of environmental conditions, high-cost areas, and objectives. In Chapter 4, we present our evolutionary approach to solving the MOSWR problem. It includes a description of three selected MOEAs and the metrics used to evaluate the performance of the MOEAs. Also, it presents the initialization procedure, specialized genetic operators, and a fast intersection method to evaluate the cost and feasibility of a route leg. Next, Chapter 5 provides computational results of case studies on the effect of routing in weather, ocean currents, and high-cost areas such as Emission Control Areas and shallow waters. The chapter concludes with a study of three selected MOEAs tested on solving different instances of the MOSWR problem. Finally, Chapter 6 summarizes our findings and provides recommendations for further research.

# 2 Literature review

This chapter provides a review of relevant literature on different subjects related to the research topic. First, Section 2.1 gives a timeline of the methods presented in the literature on solving a ship routing problem. Subsections 2.1.1 to 2.1.4 provide a more detailed review of four general solution methods used for ship weather routing problems: (1) modified isochrone, (2) dynamic programming, (3) graph search, and (4) multi-objective evolutionary algorithms, respectively. Secondly, Section 2.2 reviews the few articles that study a ship routing problem that include the effect of ocean currents. Finally, Section 2.3 gives a comparison of the reviewed methods and enhances the clarity of the positioning of the presented work.

## 2.1 A timeline of solving the ship routing problem

Minimal time and minimal (fuel) cost navigation have been the subject of extensive research work for more than six decades. Figure 2.1 provides an illustration of the development of research on different ship routing methods.

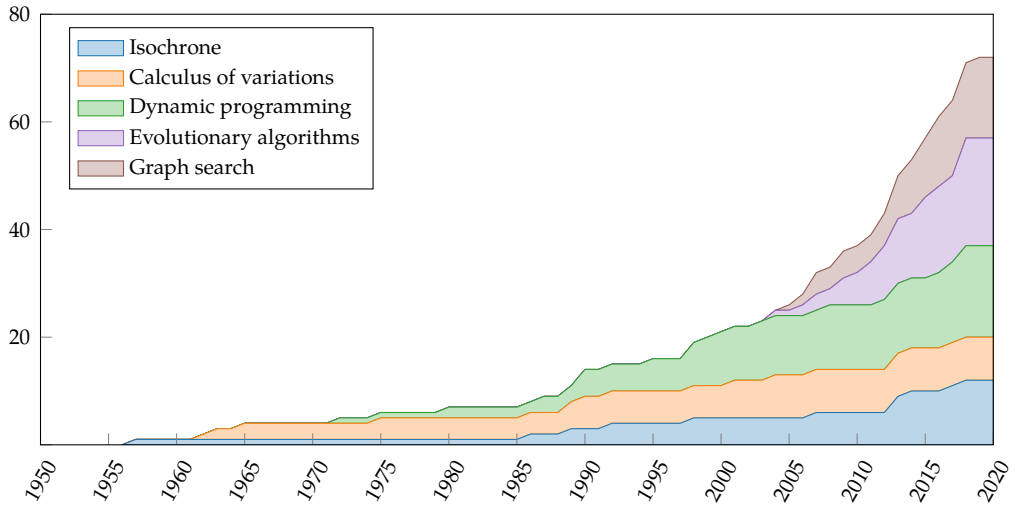



**Figure 2.1.** Research development for five general ship routing methods.
Notable academic articles are accumulated over the years 1957-2020.

The pioneering work of James (1957) lay the foundation for ship routing research with the "isochrone method". This methodology originates from a manual method used by navigators, based on spatial isochrones a ship can reach within a given time. Hagiwara (1989) revised the isochrone method and named it the "modified isochrone method". Later, Klompstra et al. (1992) proposed the "isopone method", a variant on the modified isochrone method for the minimization of fuel consumption. Both the isochrone and isopone method

were found to be less suitable for computerization than the modified isochrone method. Section 2.1.1 elaborates upon the latter method and discusses it applications.

A few years after the introduction of the isochrone method, the calculus of variations was introduced for solving the ship routing problem. First, Haltiner et al. (1962) solved the deterministic minimum time routing problem assuming stationary wave fields with the calculus of variations. Then, Bleick and Faulkner (1965) solved the same problem as Haltiner et al., allowing for time-dependent wave fields. However, the approach of Bijlsma (1975) set a solid basis for the calculus of variations in ship weather routing. He solved the minimal time routing problem using deterministic weather forecasts and Pontryagin's maximum principle. Alternative methods became more popular than the calculus of variations in the past twenty years. This is presumably due to three reasons:

1. since inaccuracies are expected in the environmental data, second-order derivatives lead to unacceptable errors in the solution values (Hagiwara & Spaans, 1987, p. 16),

2. it is often difficult to converge on a route to the destination (Marie & Courteille, 2009, p. 133), and

3. considering a variable ship speed profile is complex and requires significant computational effort (Shao et al., 2012).

Because of these disadvantages, we do not further look into ship routing literature on the calculus of variations.

Fifteen years after the work of James (1957) on the isochrone method, other forms of dynamic programming became popular methodologies for solving the ship routing problem. The first was by Zoppoli (1972), who formulated a constrained dynamic programming, which was also the first approach to the solution of stochastic minimum time routing. Dynamic programming became increasingly popular from 1990, starting with basic utilization of dynamic programming for a grid of points (Lo & McCord, 1998; De Wit, 1990), and more recently, with the utilization of three-dimensional dynamic programming, proposed by Shao et al. (2012) and Zaccone et al. (2018). Section 2.1.2 discusses the usage of dynamic programming in ship routing problems and focuses on the three-dimensional variant.

In recent years, some researchers attempted to solve the ship routing problem using grid-based techniques such as Dijkstra's method and A*. Padhy (2008) dealt with the deterministic minimal time routing based on the WAM wave model using Dijkstra. More recently, Chu et al. (2015) used Dijkstra's algorithm to assess the impact and sensitivity of environmental uncertainties assimilated from ensemble weather forecasts to ship routing optimizations with respect to fuel cost minimization. Similar to Dijkstra's algorithm is A*, of which the application to ship routing was first introduced by Jung and Rhyu (1999) to determining the economical shipping route and later compared to modified isochrone by Roh (2013).

Hinnenthal and Clauss (2010) propose a different methodology from the ones addressed

so far. Using a multi-objective evolutionary algorithm, they solved a two-objective optimization problem by minimizing travel time and fuel consumption. Evolutionary approaches, such as the genetic algorithm, seem very convenient for solving multi-objective ship routing problems. Also Szapczyska (2007) and Marie and Courteille (2009) propose an evolutionary multi-objective optimization approach to search discrete or continuous search space and find a Pareto-optimal set of routes. Section 2.1.4 provides a more detailed review of their work.

### 2.1.1 Modified isochrone

Hagiwara and Spaans improved the isochrone method to be more suitable for computerization (Hagiwara, 1989; Hagiwara & Spaans, 1987), giving it more practical implementations for ship weather routing systems. He named it the "modified isochrone method", which deterministically solves a discrete optimization problem containing a single-objective function and constraints regarding time and the ship's position, course, speed, and motion. This method allows a ship to vary its course at any intermediate waypoint around the reference route, e.g., the shortest (great circle) route. The range and resolution of course variation at each waypoint determine the number of potential subroutes. It is a recursive algorithm, and the subroutes generated by the method increases exponentially with the number of time stages. Sub-lanes are introduced as constraints on the algorithm to reduce the potential subroutes. Therefore, the accuracy and also the speed of this method lies in the chosen width of the sub-lane.

For each waypoint generated by the modified isochrone method, it assumes that a ship operates at a constant engine speed during the voyage. Therefore, the sailing distance of a time stage depends on the environmental conditions that the ship will encounter at these waypoints. The waypoints that lead to too much delayed or total fuel cost at the destination are not considered in subsequent stages.

This method finds the least-time route and is able to find a quasi-minimum fuel route. The latter route is obtained by keeping the engine speed constant during the simulation and then applying the modified isochrone method to determine the minimum travel time. By varying the engine speed in subsequent simulations, this method finds the least fuel-consuming route for a constant engine power providing on-time arrival. It then treats the minimum time route as the quasi-minimum fuel route. Thus, the method does not minimize fuel consumption itself. Moreover, due to a constant engine speed over the entire voyage, this method may find a suboptimal minimum fuel route when environmental conditions are considered.

Based on the modified isochrone method, Klompstra et al. (1992) introduced the "isopone method". The isopone considers the distance that a ship can reach using equal fuel consumption. The method handles single-objective functions such as minimum travel time or minimum fuel consumption. An advantage compared to modified isochrone is that isopone enables a variable engine speed during the voyage. The commercial weather rout-

ing software SPOS (Ship Performance Optimization System) initially adopted the isopone method. However, the isopone method was replaced with the modified isochrone method in the final product (Shao et al., 2012), despite the isopone method theoretically offering better results. The main reason for this switch was that the isopone method seemed to be more difficult to understand for navigators than the modified isochrone method.

For this reason, many other commercial weather routing companies employ modified isochrone as well (Chen, 2013). Unfortunately, due to the method's lack of speed management, these commercial software do not have the ability to change speed to avoid, for example, rough weather areas. The next section reviews dynamic programming, which has the capability of handling multiple control variables, e.g., ship location and speed.

### 2.1.2 Dynamic programming

Dynamic programming is a powerful method for solving various optimization problems and is widely used in many areas. It separates a large, complicated problem into many subproblems and solves the subproblems first. Then, using Bellman's optimality principle (Bellman, 1952), the subproblems are combined to reach an overall solution, and finally, the optimal sequence of decisions can be identified. Many of these subproblems are often similar. The dynamic programming approach tries to solve each subproblem only once, reducing the number of computations. This is especially useful when the number of repeating subproblems is exponentially large.

In dynamic programming for ship routing, the grid system is often constructed based on the great circle path. Along this path, a route is divided into many stages. For each stage, a one-dimensional grid perpendicular to the great circle subroute is generated for path selection. The ship's control vector is chosen at each stage, and subproblems are calculated stage by stage. Information from the previous stage is used to determine the control variables of the next stage. The parameters used to describe a stage must be variables that monotonically increase as the ship's voyage or route optimization progresses. In ship routing problems, time, fuel consumption, and voyage progress increase monotonically during a voyage. All these three variables can thus be used to define a stage.

Dynamic programming in ship routing can be categorized into two types: (1) two-dimensional programming (2DDP) and (2) three-dimensional programming (3DDP). We focus on 3DDP in the following, since in 2DDP the engine speed is kept constant and only the path is optimized.

3DDP allows for engine speed reduction during the voyage; therefore, it also considers the time variable for route planning in addition to the ship's position. Thus, the 3DDP can provide a better result than 2DDP but obviously uses more computing time. Every stage comprises many states, while a state is defined by a grid waypoint and a discretized time. Because the ship's path, i.e., course angles and waypoints, is predefined on the grid system, the only variable that requires optimization is ship speed.

Shao et al. (2012) and Zaccone et al. (2018) both present a 3DDP based ship route optimization method. They both limit their objective function to minimizing fuel consumption while respecting arrival time, safety, and comfort restrictions.

1. The 3DDP presented by Shao et al. is a forward dynamic programming method that uses the voyage progress as a stage variable through voluntary or involuntary speed control. They compare their 3DDP to a 2DDP variant and a selection of multi-objective evolutionary algorithms (MOEAs), including NSGA-II. They find considerable fuel savings compared to 2DDP, but no significant fuel savings between 3DDP and NSGA-II. Although, other MOEAs perform slightly worse.

   Shao et al. report computation time of less than 30 s for 3DDP compared to five minutes used by the MOEAs. However, we note that the performance of MOEAs is heavily dependent on the solution's fitness evaluation and parameter settings. The accuracy of the results found by their 3DDP algorithm is highly dependent on the grid fineness, which directly couples to the required computation time.

2. Zaccone et al. present a 3DDP also taking into account ship motions and comfort due to weather effects. The optimization is performed in a discretized space-time domain by parameterizing the voyage as a multi-stage decision process. Their dynamic programming approach's computation time is significantly affected by the chosen discretization parameters and constraint settings. The more the problem is constrained, the fewer nodes are processed, resulting in a faster computation. They report a computation time required for finding a Pareto set containing 134 routes of 10 min 50 s.

### 2.1.3 Graph search

The Dijkstra algorithm, together with the A* algorithm, is among the popularly algorithms for finding the shortest path between two given nodes in a graph with strictly positive weights. It provides a deterministic method for solving a discrete optimization problem consisting of one objective, e.g., minimum travel time or fuel cost, and only implicitly defined constraints. In most ship routing literature, a graph is assigned positive weights representing a single or sum of objectives in these edges according to weather conditions and a predefined engine speed. A graph search algorithm is then able to find the minimum time route (or minimum fuel route) by minimizing the sum of weights.

Graph search algorithms are generally very fast for solving the ship weather routing problem considering a single objective and a constant engine speed. Modifying the algorithm for handling time-varying environmental conditions and varying ship speeds requires multiple grid dimensions. This complicates the construction of a graph and significantly increases the number of nodes and, therefore, the computation time. To our knowledge, two graph search methods are proposed in ship routing literature that are capable of handling voluntary speed reduction and other ship weather routing aspects.

1. Montes (2005) introduces a network ship routing model that tries to find the least-

time path using a modified version of Dijkstra's shortest path algorithm and a basic ship response function. A binary heap function is used to reduced the size of the multi-dimensional graph, which reduces the computational complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(m \log(n))$, where $n$ is the number of nodes and $m$ is the number of arcs. Computation times in the order of milliseconds are reported for problem instance with a very low graph resolution. Due to restriction to a coarse grid resolution and a single-objective approach, this method is only useful for quick estimates for ship voyage durations.

2. Also Padhy (2008) develops a method based on a modification of the Dijkstra algorithm for finding a least-time route. Next to voluntary speed reduction, this method considers the effect of involuntary speed reduction due to added resistance caused by wind, waves, and ocean current. Hence, it is assumed that ocean currents do not affect the ship's course.

   Sen and Padhy (2015) state that the main disadvantage of this algorithm is that the resulting path is not smooth. This may result in a zigzag path and, consequently, requiring a refinement of the grid. A high-resolution grid, together with a short time interval in environmental forecasts, increases the computational significantly. To maintain an acceptable level of accuracy, Padhy chooses a high grid resolution and assumes a static environment. No computation times are reported.

Graph-based approaches rather consider single-objective optimization problems, e.g., minimum travel time. To optimize multiple objectives, criteria can be represented as a weighted-sum in a single-objective cost function. Although, aggregation of multiple criteria may result in losing detailed information on various possible routes, including the best routes for each criterion. More suitable for handling multiple criteria are evolutionary algorithms. Applications of these algorithms will be discussed in the next section.

### 2.1.4 Multi-objective evolutionary algorithms

Evolutionary algorithms (EAs) are metaheuristic optimization algorithms that develop a population of solutions. One of the most well-known evolutionary algorithms is the genetic algorithm, inspired by natural evolution. It initializes with a first generation of solutions (route legs), each with its own fitness, e.g., travel time and fuel cost. Next generations are iteratively generated by creating offspring, i.e., recombining elements (waypoints) of the solutions of the previous generation. When creating a new solution, certain mutations may occur, affecting one or more solution elements.

In the classical literature for multi-objective optimization problems, there did not exist any optimization methods that could find multiple optimal solutions in a single simulation. EAs are logical candidates to solve these problems, as they work with a population of solutions enabling approximation of the Pareto optimal set in a single run. This set contains solutions with a balanced impact of the criteria, as well as the best solutions according to these criteria. Thus, the concept of EAs was extended by introducing multi-objective evolu-

tionary algorithms (MOEAs). The aim of these algorithms is twofold. In the first place, the algorithm's approximation of the Pareto front (i.e., the Pareto set projected in the objective space) should be as close to the actual Pareto front as possible. Secondly, the obtained solutions should be evenly spread over the Pareto front and capture the front's extreme ends. MOEAs have attracted a lot of research effort during the last twenty years, and they are still one of the hottest research areas in the field of evolutionary computation. This led to better, faster, and more accessible MOEAs. An extensive overview of frequently used MOEAs is given by Konak et al. (2006).

Notable ship weather routing applications of multi-objective evolutionary optimization methods are those proposed in (Hinnenthal & Clauss, 2010; Marie & Courteille, 2009; Szapczyska, 2015). Furthermore, Veneti et al. (2018) test two popular MOEA, SPEA2 and NSGA-II, on the ship weather routing problem. And, Li et al. (2017) test NSGA-II on a simplified multi-objective ship weather routing problem.

1. Marie and Courteille (2009) use MOGA, an MOEA proposed in (Fonseca & Fleming, 1998), to minimize a ship route's fuel consumption and travel time. This study's novelty is a spatial and temporal generation of route variants based on a generic and automatic meshing. Since this technique has a physics-based definition and requires a low number of free variables, the solution method seems reliable in practice and has a limited computation time of about 2 h. Marie and Courteille state that the major disadvantage of the MOGA is mainly related to the number of evaluations necessary to obtain satisfactory solutions. That is, the search extends in all the directions of the solution space, producing a rich database.

2. Hinnenthal and Clauss (2010) seek to find a route that minimizes travel time and fuel consumption, and maximizes robustness regarding weather forecast uncertainties. Similar to Marie and Courteille, they use MOGA as a framework for their solution approach. The stochastic behavior of weather is accounted for with the utilization of probabilistic ensemble weather forecasts. For each route, the ship behavior in waves is simulated using transfer functions. These computationally expensive functions lead to a large calculation time of a single route. As a result, the whole optimization takes nearly 10 h.

3. Szapczyska (2015) presents the Multi-objective Evolutionary Weather Routing Algorithm (MEWRA), a ship routing algorithm utilizing a robust evolutionary algorithm: Strength Pareto Evolutionary Algorithm (SPEA) (Zitzler & Thiele, 1999). The MEWRA simultaneously minimizes travel time, fuel consumption, and the ship's safety risk in rough weather regions. Multiple constraints are considered, such as shallow water, piracy risk areas, and a discrete set of attainable ship speeds.

   Szapczyska reports a computation time of 6 min if all criteria and constraints are considered. The computation time is significantly less compared to methods proposed in (Hinnenthal & Clauss, 2010; Marie & Courteille, 2009). The main reason is likely due to a more simplified method for estimating the involuntary speed reduction. Also,

MEWRA returns much less Pareto-optimal solutions, i.e., approximately two times fewer route evaluations are performed.

4. Veneti et al. (2018) compare an improved version of SPEA, SPEA2 (Zitzler et al., 2001), to NSGA-II (Deb et al., 2002) according to their performance on solving the ship weather routing problem. A route path is defined in discrete space, i.e., a grid structure is used, and route objectives to be minimized are the ship's safety risk and total fuel cost. The travel time is set as a limitation to the voyage. During the voyage, weather forecasts are used to determine the ship's speed reduction and safety risk, but the engine speed is kept constant. Veneti et al. report a slightly lower computation time for NSGA-II, but SPEA2 achieves marginally better solutions.

5. Lastly, Li et al. (2017) use NSGA-II to find ocean voyages that minimize both risk and travel time. Risk is affected by weather, and fuel consumption is added as a constraint to the problem. Using a predefined set of longitudinal coordinates and setting upper and lower bounds for the sailing region, Li et al. keep the total computation 'within minutes'. Additionally, the calculations of ship speed loss and distance between two waypoints are simplified, making this method well-suited for quick estimates of ocean voyages but less suitable for realistic ship routes.

The required computational time of MOEA mainly depends on the number of free variables and objectives, as well as the number of route evaluation points. Both Hinnenthal and Clauss, and Marie and Courteille implement spatial and temporal modeling methods to reduce the number of free variables. However, Hinnenthal and Clauss state that "a genetic algorithm based method will never reach the computational speed of a deterministic or graph theory based method" (Hinnenthal & Clauss, 2010, p. 119).

## 2.2 The ship routing problem including ocean current

Chang et al. (2015) provide a global map of strong ocean currents focusing on western boundary currents (WBCs). Theoretically, WBCs with average speeds of 2 kn – 3 kn can reduce travel time by 2 % – 8 % for ships sailing at 12 kn – 16 kn (Chang et al., 2013; Lo & McCord, 1998). The majority of proposed optimization methods in ship weather routing literature neglects the effect of ocean currents on a ship route. In this section, we review two articles that focus on strong ocean currents. The first considers the Gulf Stream region near the east coast of North-America, and the second addresses the Kuroshio Current in the East Chinese Sea.

1. A ship weather routing problem using ocean current data from the Gulf Stream is examined in (Lo & McCord, 1998). To address the uncertainty caused by a delay in delivering the ocean current information, Lo and McCord formulated a minimum fuel problem as an adaptive, probabilistic dynamic programming using longitudinal progression as a discrete stage variable. This problem includes stochastic ocean currents and uses the ship's heading (i.e., the compass direction) and engine speed as

decision variables. However, with the introduction of a stochastic variable and two control parameters, the size of the search space is significant. To limit the computation time, Lo and McCord developed a so-called adaptive heading-along heuristic, which optimizes for heading only, such that the expected travel time is minimized. Using fine-resolution current estimates of the Gulf Stream, Lo and McCord found a relative average fuel savings of 7.5 % when riding favorable currents and 4.5 % when avoiding unfavorable currents for ships with a nominal velocity of 16 kn.

2. Tanaka and Kobayashi (2019) formulated an optimal fuel routing problem with deterministic ocean currents as a mixed-integer nonlinear optimization problem. They present a 3DDP that solves two sequential subproblems in each iteration. The former is the problem of finding all feasible paths from origin to destination on a graph, with feasibility constraints on the arrival time. The engine speed along these paths are then optimized in the second problem. This algorithm finds the optimal path on a graph. However, if the navigation region or the graph's resolution is large, the number of solution evaluations increases significantly. The algorithm is terminated if the best solution is found or if it exceeds a maximum computation time.

   Tanaka and Kobayashi performed numerical experiments on the East Chinese Sea. Section 5.3.2 compares their numerical results to our findings regarding the same problem instance. For various graph resolutions and arrival time constraints, they generated thirty instances to compute routes between Tokyo, Japan, and Keelung, Taiwan, in two directions. Under loose travel time constraints, the algorithm finds the optimal solution within an hour. For more restrictive arrival times, the algorithm is terminated prematurely after an hour. In most cases, strong, feasible solutions were returned in early iterations, implying that the algorithm likely finds near-optimal solutions in an early stage.

## 2.3 Ship routing methods comparison

Table 2.1 gives an overview of the general methods used in ship routing literature. Regarding the demands for multi-objective ship weather routing, it is expected that objective functions become multimodal when optimization results are governed by environmental constraints. Therefore, the solution method should be able to overcome local optima. In the ship routing literature, MOEAs and 3DDP methods are stochastic and exact approaches, respectively, that serve for the purpose of finding these local optima. Using MOEAs, optimization criteria can be easily extended or disregarded making it a flexible approach. The stochastic nature of the evolutionary algorithms serves well for simultaneously searching multiple Pareto optimum solution over the entire solution space. For this reason and other favorable characteristics mentioned in Section 2.1.4, we choose to use an MOEA as a framework to the solution approach.

Table 2.2 gives an overview of the discussed literature. Several aspects of ship weather routing are researched thoroughly in the past six decades. However, their usage by ship op-

erators and in particular on board navigating officers is still not very clear from a practical application point of view. A possible reason for the limited information on the actual algorithm that is used by ship navigators could be the commercial nature of such algorithms: many of the algorithms currently used are proprietary and not available in the open literature. A detailed comparison of the various methods is difficult because there is no basis for such comparisons. Each work formulates the ship routing problem individually, considering different aspects. And some articles use data sources that are not publicly available.

To our knowledge, there does not exist a ship weather routing algorithm in the literature that simultaneously minimizes travel time and fuel consumption considering the following aspects:

- Time-varying environmental conditions, including weather and ocean currents.
- A variable engine speed profile.
- High-cost areas (e.g., ECAs and shallow waters).
- Finding realistic smooth routes.

A ship routing algorithm that considers these aspects is compelling in today's shipping industry. Especially due to their significant impact on a ship's travel time, fuel consumption, and ultimately, the environment. For applications in the early voyage planning phase and on-board, the routing algorithm requires a short computation time.

**Table 2.1.** General ship routing methods comparison.

| Method | Space | Variable engine speed | Advantages | Disadvantages |
|---|---|---|---|---|
| Isochrone | grid | no | Easily understood<br>Easy constraint handling | Complicated implementation |
| Calculus of variations | continuous | no | Mathematically elegant | Complicated implementation<br>Convergence problems<br>Expected second-order derivate error |
| Dynamic programming | grid | yes | Easily understood<br>Easy constraint handling | Accuracy largely grid-dependent<br>Computationally expensive |
| Graph search | grid | yes | Computationally fast<br>Easy constraint handling | Complicated implementation<br>No smooth routes |
| Evolutionary algorithms | continuous | yes | Easy multi-objective handling<br>Flexible approach | Computationally expensive<br>Not an exact approach<br>Requires parameter tuning |

**Table 2.2.** Ship routing methods comparison.
Objectives T, F, S, and R denote travel time, fuel consumption, ship safety, and weather forecast robustness, respectively. Computation time of a method is listed if it is reported and the article's publication date is in the recent past.

| Author | Solution approach | Weather | Ocean current | Variable engine speed | Continuous space | Objectives | Computation time |
|---|---|---|---|---|---|---|---|
| Isochrone methods | | | | | | | |
| (James, 1957) | isochrone | yes | no | no | no | T | n/a |
| (Hagiwara, 1989) | modified isochrone | yes | yes | no | no | T or F | n/a |
| (Klompstra et al., 1992) | isopone | yes | yes | no | no | T, F | n/a |
| Calculus of variations (CoV) | | | | | | | |
| (Haltiner et al., 1962) | CoV | yes | no | no | yes | T | n/a |
| (Bleick & Faulkner, 1965) | CoV | yes | no | no | yes | T | n/a |
| (Bijlsma, 1975) | CoV | yes | no | yes | yes | T or F | n/a |
| Dynamic programming (DP) | | | | | | | |
| (Lo & McCord, 1998) | 2DDP | no | yes | yes | no | F | n/a |
| (Shao et al., 2012) | 3DDP | yes | no | yes | no | F | $< 30$ s |
| (Zaccone et al., 2018) | 3DDP | yes | no | yes | no | F | 10 min 50 s |
| (Tanaka & Kobayashi, 2019) | 3DDP | no | yes | yes | no | F | 5 min – 1 h |
| Graph search | | | | | | | |
| (Montes, 2005) | modified Dijkstra | yes | no | yes | no | T | 60 ms – 80 ms |
| (Padhy, 2008) | modified Dijkstra | yes | not tested | yes | no | T | not reported |
| Evolutionary algorithm | | | | | | | |
| (Szapczyska, 2007) | SPEA | yes | no | yes | yes | T, F, S | 6 min |
| (Hinnenthal & Clauss, 2010) | MOGA | yes | no | yes | yes | T, F, R | 10 h |
| (Marie & Courteille, 2009) | MOGA | yes | no | yes | yes | T, F | 2 h |
| (Veneti et al., 2018) | NSGA-II, SPEA2 | yes | no | no | no | F, S | 20 s – 60 s |
| (Li et al., 2017) | NSGA-II | yes | no | yes | no | F, S | 'within minutes' |
| Presented method | NSGA-II, SPEA2, M-PAES | yes | yes | yes | yes | T, F | 3 min – 5 min |

# 3 The Multi-Objective Ship Weather Routing problem

The previous chapter reviewed general methods used in ship routing literature. Many ship routing problems have been researched in the past. For the majority of these problems, one aims to optimize a single objective function, for instance, travel time. In recent years, multi-objective optimization applied to the ship weather routing problem became increasingly popular. An evolutionary approach to such problems is found to have several practical advantages over other methods. Therefore, we use a multi-objective evolutionary algorithm as a framework for our solution approach to solving the Multi-Objective Ship Weather Routing (MOSWR) problem

In this chapter, we give a formal definition of the MOSWR problem. First, we describe the problem and outline different components of the ship weather routing structure in Section 3.1. Secondly, we formulate the ship routing problem as a continuous optimization problem in Section 3.2. Elaborating on this mathematical formulation, Section 3.3 provides calculation methods used to obtain the distance over a route leg, and Section 3.4 provides the actual speed derivation in different environmental conditions. Finally, Section 3.5 describes the mathematical concept of Pareto dominance, that we use for the evaluation of solutions containing multiple criteria.

## 3.1 Problem description

The structure of the Multi-Objective Ship Weather Routing problem is shown in Figure 3.1. In principle, the MOSWR problem can be broken down into several comprehensible components. The core part of the problem is a set of *route decision variables*, affected by two preceding components: *path selection criteria* and *navigable area limitations*, shown in orange and blue, respectively. Considering the input of these components, the route decision variables define a *ship route* in green. Following this structure, we first describe the elements of a ship route in Section 3.1.1. Then, Sections 3.1.2 and 3.1.3 describe the different elements comprising the path selection criteria and navigable area limitations, respectively.

### 3.1.1 Ship route

A ship route (green) is characterized by a series of waypoints, starting at the departure and ending at the destination. A waypoint is defined by a pair of longitude and latitude coordinates in degrees. At each waypoint, except at the destination waypoint, a ship route contains information on the ship heading and engine power required to reach the next waypoint. The heading is the compass direction in which a ship's bow is pointed.

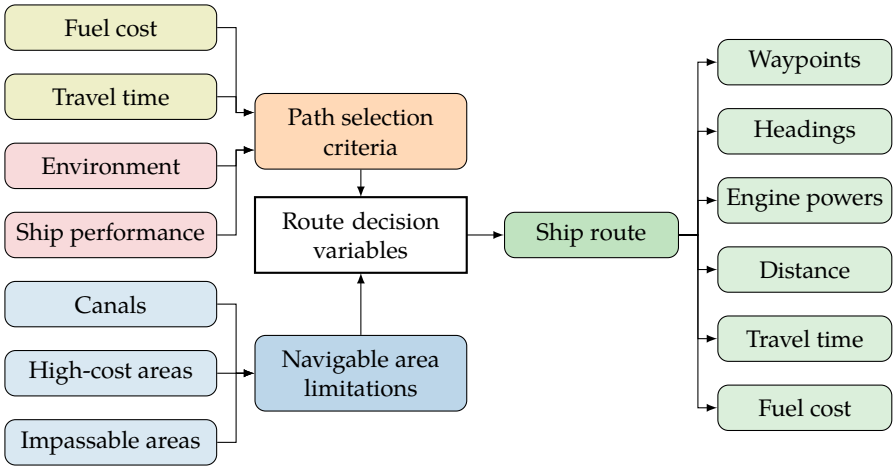



**Figure 3.1.** Overall structure of the Multi-objective Ship Weather Routing problem. Route decision variables are affected by path selection criteria (orange) and navigable area limitations (blue). The first elements comprise the optimization objective (yellow) and ship performance assessment (red). The second element, navigable area limitations, defines the constraints and costs for the navigable area. A solution to the problem is a ship route (green), containing several characteristics.

### 3.1.2 Path selection criteria

Path selection criteria (orange) of the MOSWR problem comprise two main components: (1) objectives and (2) ship performance assessment. The former include minimization of fuel cost and minimization of travel time (yellow). The latter, ship performance assessment, is affected by environmental conditions and the ship performance model (red).

*Objectives*

In commercial shipping, ship navigation is based on a single criterion or a combination of multiple criteria. Generally, these criteria are minimum travel time, fuel consumption, and safety risk of crew and cargo. In this thesis, we consider the problem of simultaneously minimizing travel time and total fuel cost. For completeness, we note that the MOSWR problem also applies in the case of a combination of other objectives.

Minimum fuel cost and minimum travel time are opposing objectives, as total fuel cost has a negative relation with the ship's travel time, i.e., minimizing fuel consumption results in an increase in the sailing duration. Consequently, simultaneously optimizing these two objectives requires multi-objective optimization. It is unlikely that there exists a single optimal solution to a multi-objective problem; it is rather a set of equal solutions. In the MOSWR problem, one aims to find such a set of Pareto optimum routes from any given departure to any given destination. A Pareto optimum route implies a feasible ship route with minimum objective values, such that an improvement of one objective is only possible by impairing another. Section 3.5 describes the mathematical concept of Pareto optimality.

*Environmental conditions*

Besides these objectives, other elements affecting route decisions are weather forecasts for the future sailing area and ship performance, shown in yellow in Figure 3.1. Environmental conditions affecting decisions for a minimum fuel and time route are wind, waves, and ocean currents. James (1957) states that a combination of wind and waves has the most significant effect on ship speed reduction on ocean-going vessels. However, waves (and wind) mainly affect the ship's speed, while ocean currents affect both the ship's speed and its course over ground. Hence, the MOSWR problem considering only wind and waves, is a special case of the same problem that includes exclusively time-varying ocean currents. Both cases are considered in a selection of case studies presented in Chapter 5.

*Ship performance*

Environmental conditions affect the ship performance. A ship performance model predicts ship behavior due to these effects and translates this behavior to ship motions. Consequently, the model approximates the ship's fuel consumption and speed at varying environmental conditions at sea. For the ship routing problem to be generic for different types of ships and environmental conditions, careful attention is to be paid to modeling ship performance.

The accuracy of the optimization results depends heavily on the accuracy of the ship performance model under different environmental conditions and ship controls. However, in this research, we simplify the ship performance model with empirical results to focus on the operations research aspect of the MOSWR problem.

We assume that the actual ship speed over a leg is the sum of the reduced ship speed and ocean current velocity. The reduced ship speed is dependent on the nominal ship speed in calm water given the engine speed and the speed loss due to added resistance caused by wind and waves.

The speed reduction due to wind and waves is dependent on the ship characteristics such as hydrodynamic models and engine specifications. In this study, this involuntary speed reduction is approximated using a semi-empirical model presented by Kwon (2008). A reasonable accuracy of added resistance due to wave and wind is required for all ship headings and a range of speeds under any wave and wind conditions. This is meaningful for obtaining accurate fuel optimization results. This approximate method's effectiveness should be verified in the future, which is out of the scope of this research.

### 3.1.3 Navigable area limitations

Navigable area limitations (blue) imposed on a ship route are categorized into three components: (1) impassable areas, (2) high-cost areas, and (3) major shipping canals.

*Impassable areas*

A ship can navigate freely within the navigable area, a predefined continuous space containing high-cost areas and excluding impassable areas. Impassable areas are handled as constraints, constituted by land obstacles, sea ice, and no-go zones. In this thesis, we include shorelines and potentially the (ant)arctic circles as impassable areas. Shorelines are obtained from high-resolution shoreline data, the Global Self-consistent, Hierarchical, High-resolution Geography (GSHHG) Database (Wessel & Smith, 1996), which is publicly accessible. The Antarctic and Arctic circles construct potentially impassable areas to the south and north of these circles, respectively. The decision to restrict these areas, e.g., for safety or environmental reasons, is left to be made by the end-user.

*High-cost areas*

Within high-cost areas, an additional operation cost rate is incurred, translating to additional fuel costs. For example, these are areas with safety risks, such as piracy zones or seasonal heavy weather, and areas with high operational costs such as ECAs. In this study, we focus on ECAs as high-cost areas, as well as penalizing routes navigating through shallow waters. For the sake of completeness, we note that other high-cost areas could be introduced as well.

Stricter controls are established within ECAs to minimize airborne emissions from ships, as defined by Annex VI of the MARPOL convention under the IMO (*MARPOL*, 2005). These ECAs are the Baltic Sea, the North Sea, and the U.S. coasts. There are mainly two ways shipping companies can achieve compliance with the ECA sulfur regulations, i.e., fuel switching and applying a scrubber. For ships that operate both within and outside ECAs, fuel switching is a straightforward compliance alternative. Thus, ships are assumed to apply fuel switching to comply with ECA regulations. Ultra-Low Sulfur Fuel Oil (ULSFO) with 0.10% m/m (mass by mass) sulfur content is consumed within ECAs and Very-Low Sulfur Fuel Oil (VLSFO) 0.50% m/m is used elsewhere. ULSFO is more expensive than VLSFO, inducing a potential change in ship route and operating speed decisions.

Future fuel price differentials are uncertain. Moreover, the COVID-19 and oil crises that started simultaneously in March 2020 caused even more uncertainty. For simplicity reasons, we define a standard scenario for the case study representing the current market situation described in 5.1.

Fuel costs are highly dependent on the ship's engine speed settings, as the fuel consumption rate is approximately proportional to the third power of ship speed (Fagerholt et al., 2010). Due to the difference in fuel costs, shipping companies that operate both within and outside ECAs face different speed decisions in each area. A possible consequence ofECA regulations is that these companies choose to sail at lower speeds within ECAs and speed up outside to compensate for longer travel time. Besides effects on speed decisions, the regulations may also affect the ship route trajectory. Namely, avoiding ECAs by repositioning sailing legs could lead to lower fuel costs. Thus, we recognize several trade-offs due to relations between speed, distance, and fuel cost.

Besides Emission Control Areas, we introduce an optional penalty for sailing through shallow waters imposing draft limitations on ships. With the introduction of shallow water penalties, a ship route preferably avoids these areas but can still pass these areas at a higher cost. This is practical shallow waters are not clearly defined, but navigation near these areas is required, e.g., near coasts, ports, and through major canals. A penalty incurred by traversing shallow waters does not reflect in the final solution value but only affects the fitness evaluation during optimization.

*Canals*

Finally, the MOSWR problem includes the optional passage major shipping canals. Two busy canals are, for instance, the Panama Canal and the Suez Canal. As canal fees mainly depend on the ship type and its characteristics, the canal fees are set as input value by the end-user. To support the end-user in conscious decision-making of routes concerning travel time and costs, a set of solutions to the MOSWR problem may contain routes passing a canal and alternative routes avoiding the canal.

## 3.2 Mathematical formulation

This section mathematically formulates the MOSWR problem as a partially continuous optimization problem. In doing so, first, we define the sailing region in which a route is represented. The sailing region contains impassable areas and areas in which a penalty or high-cost is incurred, so-called high-cost areas. Next, we present the structure of a ship route on which geographical and ship limitations are imposed. The problem in MOSWR is seeking a set of Pareto optimum routes such that these limitations are satisfied. In this thesis, the selection of Pareto optimum routes is based on two objective values: a route's travel time and total fuel cost. Non-trivial for the calculation of these objective values are a route's distance and the actual ship speed, governed by the ship performance in environmental conditions. Sections 3.3 and 3.4 provide a more detailed description of the calculation of the route distance and ship performance along the route, respectively. The remainder of this section gives a formulation of the MOSWR.

The trajectory of a ship route is defined as a piecewise-smooth curve on an Earth ellipsoid surface, i.e., a smooth curve that changes direction at discrete points, as shown in Figure 3.2. It is defined by a sequence of $n + 1$ waypoints constituting $n$ route legs. Each leg $r_i$, $i \in [n]$[1], contains starting and ending waypoint locations and nominal ship speed constant over the leg.

The first constraint imposed on a ship route is that of the sailing region. Let the sailing region $\Omega_0$ be projected on the surface of an Earth ellipsoid, such that

$$\Omega_0 \in \{(\lambda, \varphi) \mid \lambda_{min} \leq \lambda \leq \lambda_{max}, \varphi_{min} \leq \varphi \leq \varphi_{max}\}, \tag{3.1}$$

where $\lambda$ and $\varphi$ are latitude and longitude in degrees, respectively. Within the sailing region,

[1] $[k] \coloneqq \{1, 2, \ldots, k\}$

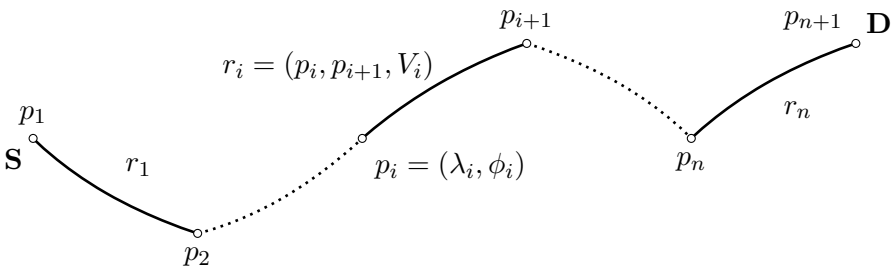



**Figure 3.2.** Schematic representation of a route $\mathbf{r}$.
Departure and destination are denoted by $S$ and $D$, respectively. A leg $r_i$ is defined by its start $p_i$ and end $p_{i+1}$ waypoints, and the nominal ship speed $V_i$. The latitude and longitude of a waypoint $p_i$ are denoted by $\lambda_i$ and $\varphi_i$, respectively.

we define $n_c$ impassable areas $\Omega_C^i$, $i \in [n_c]$ that remain invariant during the sailing period. A ship can navigate freely within the sailing region but cannot intersect impassable areas. Thus, we can express the navigable area $\Omega_a$ as

$$\Omega_a = \Omega_0 \setminus \bigcup_{i=0}^{n_c} \Omega_C^i. \tag{3.2}$$

The second constraint of the ship route is to the characteristics of the ship itself. Let us assume that the nominal ship speed is limited to a discrete interval $[V_{min}, V_{max}]$, where $V_{min}$ and $V_{max}$ are the minimum and maximum nominal ship speed of the interval, respectively.

A solution to the MOSWR problem is defined by the decision vector of route variables $\mathbf{r}$, expressed in the form

$$\mathbf{r} = \Big\langle (p_1, p_2, V_1), (p_2, p_3, V_2), \ldots, (p_{n-1}, p_n, V_{n-1}), (p_n, p_{n+1}, V_n) \Big\rangle, \tag{3.3}$$

where $p_j$, $j \in [n+1]$, is the location of the $j$-th waypoint described by the route trajectory $\mathbf{p} = \mathbf{p}(\mathbf{r})$, with

$$\mathbf{p}(\mathbf{r}) = \Big(p_1, p_2, \ldots, p_n, p_{n+1}\Big) = \Big\langle (\lambda_1, \varphi_1), (\lambda_2, \varphi_2), \ldots, (\lambda_n, \varphi_n), (\lambda_{n+1}, \varphi_{n+1}) \Big\rangle, \tag{3.4}$$

and $V_i$ is the nominal ship speed at leg $r_i$, $i \in [n]$, in knots.

The latitude and longitude pair of each waypoint $p \in \mathbf{p}$ are restricted to lie in $\Omega_a$. Not only the waypoints must be within the navigable area, but also the whole route trajectory $\mathbf{p}$ formed by these waypoints must be within the navigable area, such that

$$\mathbf{p} \in \Omega_a. \tag{3.5}$$

In other words, a ship route is infeasible if there exists an intersection of any route leg $r \in \mathbf{r}$ with any impassable area $\Omega_C^i$, $i \in [n_c]$. Section 4.5 describes the method used to test whether there exists such an intersection.

The nominal ship speed $V_i$ at each leg constitutes the speed variable $\mathbf{v}$. The desired

constant ship speed for each leg of the route is represented by the vector

$$\mathbf{v} = \Big(V_1, V_2, \ldots, V_{n-1}, V_n\Big), \quad V_i \in [V_{min}, V_{max}] \forall i \in [n]. \tag{3.6}$$

Let $L$ denote the total length of the route. The route length can be calculated as the sum of the length of each leg $r_i$ in $\mathbf{r}$,

$$L = L(\mathbf{r}) = \sum_{i=1}^{n} L_i, \tag{3.7}$$

where $L_i$ is the length of the $i$-th leg calculated as the geodesic distance, which will be discussed in the next section.

The total travel time $T$ in hours from departure to destination can be summed by the time $T_i$ in hours taken for each leg and is represented by

$$T = \sum_{i=1}^{n} T_i, \quad T_i = \frac{L_i}{\overline{V_{a,i}}}, \tag{3.8}$$

where $\overline{V_{a,i}}$ is the average, actual ship speed at leg $r_i$, obtained from the geodesic between the start and end waypoint of $r_i$ and the nominal ship speed $V_i$. The derivation of $\overline{V_{a,i}}$ will be discussed in Section 3.4.

Let us assume that the fuel cost rate at leg $r_i$ can be described by the equation

$$FC_i = c_f^i fr(V_i), \quad i \in [n], \tag{3.9}$$

where $fr(V_i)$ denotes the fuel consumption rate in metric tons per hour for a nominal ship speed $V_i$, and $c_f^i$ is the cost of fuel at $r_i$, defined as

$$c_f^i = \begin{cases} C_{f,ECA}, & \text{if } r_i \cap \bigcup_{j=0}^{n_e} \Omega_{ECA}^j \neq \emptyset, \\ C_{f,0} & \text{otherwise,} \end{cases} \tag{3.10}$$

where $C_{f,0}$ is the cost of standard fuel, $C_{f,ECA}$ is the cost for ECA fuel, and $\Omega_{ECA}$ is set of polygons in which ECA fuel is required.

Then, a route's total fuel cost $FC$ can be computed as the sum of the fuel cost of each leg expressed as

$$FC = \sum_{i=1}^{n} FC_i T_i, \tag{3.11}$$

where $T_i$ is the time $T_i$ in hours taken for the $i$-th leg, defined in Equation 3.8.

Each solution is assigned two fitness values, $T$ and $FC$. The dominance of a solution is then obtained according to Pareto dominance, which will be discussed in Section 3.5.

## 3.3 Geodesic distance

Different methods exist in approximating the shortest distance between a pair of latitude and longitude points on a curved surface. To calculate the distance of a route leg represented on the Earth's surface is therefore non-trivial for the MOSWR problem. The shortest path between two points on a curved surface is represented by a geodesic. Section 3.3.1 describes two frequently used methods for calculating the geodesic distance between two points on the Earth's surface. Lastly, Section 3.3.2 describes a practical method to approximate short geodesics with a straight line.

### 3.3.1 Geodesic

The first method to approximate an Earth geodesic is the haversine formula, which treats the Earth as a sphere. The geodesics on a sphere are great circles whose centers coincide with the center of the sphere. Through any two points on a sphere's surface, with latitude and longitude pairs $(\lambda_1, \varphi_1)$ and $(\lambda_2, \varphi_2)$, the geodesic distance $l$ is the great circle distance, computed as

$$l = R\theta, \tag{3.12}$$

where $R$ is the mean Earth radius and $\theta$ is the central angle between the two waypoints in radians. The central angle is computed with the haversine formula first developed by Inman (1849)

$$\theta = \arcsin\sqrt{\sin^2\left(\frac{|\varphi_2 - \varphi_1|}{2}\right) + \cos\varphi_1 \cos\varphi_2 \sin^2\left(\frac{|\lambda_2 - \lambda_1|}{2}\right)}, \tag{3.13}$$

where $(\lambda_1, \varphi_1)$ and $(\lambda_2, \varphi_2)$ are the latitude and longitude pairs of two points on the Earth's surface.

Due to the effect of the Earth's rotation, the shape of the Earth is better approximated by an oblate spheroid, a slightly flattened sphere. Thus, the haversine formula introduces a distance approximation error as it assumes a spherical Earth. This error is within 0.5 % for latitude and 0.2 % for longitude (*Admiralty manual of navigation*, 1987, p. 10).

To address this, Vincenty (1975) developed a method for finding a geodesic on Earth that is accurate up to 0.5 mm. Given the coordinates of the two points $(\lambda_1, \varphi_1)$ and $(\lambda_2, \varphi_2)$, Vincenty's inverse method finds the ellipsoidal distance $d$, as described in Algorithm 2 in Appendix A. Vincenty's inverse method requires the major and minor axis lengths of the Earth's reference ellipsoid. In this study, we use the WGS 87 ellipsoid defined by the NIMA (2000). This reference ellipsoid model of the Earth is the standard for cartography, geodesy, and satellite navigation, including the Global Positioning System (GPS).

Since Vincenty's inverse method is an iterative procedure that aims to minimize the output value, this method is slower in general. On a Core i7 PC, the computation time for calculating distance with Vincenty's inverse method is 5 – 10 µs, which is approximately two times the 2 – 5 µs it takes for distance computation with the haversine formula.

In the final optimization model, we use Vincenty's inverse method for the calculation by default, since multiple experiments showed that the computational overhead of Vincenty's inverse method did not govern the algorithm's total computation time. Nonetheless, we include the haversine method as an optional distance calculation method so that the end-user can choose either method.

### 3.3.2 Rhumb line

A geodesic other than a meridian or the equator is a curved line whose true direction changes continually. Therefore, navigators usually do not attempt to follow it exactly in practice. Instead, they select several waypoints along the geodesic, construct rhumb lines between the waypoints on a Mercator projection, and then steer along these rhumb lines (Bowditch, 1802).

The Mercator projection is a cylindrical map projection assuming the Earth as a sphere. It is the standard for ship navigation because of its unique property of representing any course of constant bearing, i.e., the angle about the north line, as a straight line, known as a rhumb line. Hence, a rhumb line's distance is the Euclidean length of the line on a Mercator projection. In this study, we use the rhumb line to approximate a short geodesic with a straight line.

For the derivation of the coordinates on a Mercator projection, first, we use the inverse Gudermannian function (Zwillinger, 1995), which gives the height $\psi$ on a Mercator projection of a given latitude $\varphi$

$$\psi = \ln(\tan(\frac{\pi}{4} + \frac{\varphi}{2})), \tag{3.14}$$

then we obtain the Mercator projection easting and northing coordinates

$$E = R\lambda, \tag{3.15}$$

$$N = R\psi, \tag{3.16}$$

respectively, from longitude $\lambda$, latitude projected on Mercator $\psi$ and the mean Earth radius $R$.

## 3.4 Actual ship speed

It is necessary to find the average of the actual ship speed at the corresponding leg to compute the travel time between two waypoints. The actual ship speed is assumed to be affected by ocean currents and wind. This section describes the calculation of the actual ship speed by first determining the ship speed loss due to wind and waves and, subsequently, determining the speed change due to ocean current.

### 3.4.1 Involuntary speed reduction due to wind and waves

The nominal ship speed is often reduced as a consequence of the added resistance due to wind and waves. This so-called involuntary speed reduction is dependent on multiple factors such as different types of environmental conditions, ship hydrodynamics, and other ship characteristics. Kwon (2008) presented a semi-empirical method for predicting the involuntary speed reduction in irregular waves and wind under the assumption that the ship engine can provide a constant power output under different weather conditions. As the method only requires information on the main ship characteristics and wind vectors at the ship's location, it makes for a more generic method that is straightforward to implement. Consequently, Kwon's method has been implemented in several ship routing studies to provide an approximate of the ship speed loss and fuel consumption increase due to weather effects. The remainder of this section describes the by Kwon.

First, we define the following notation:

$\Delta V$ ship speed difference in $\mathrm{m\,s^{-1}}$.

$V$ nominal ship speed in $\mathrm{m\,s^{-1}}$.

$C_\beta$ speed direction reduction coefficient. Depending on the true wind direction $TWD$ in degrees and the Beaufort number $BN$, representing the wind speed.

$C_U$ speed reduction coefficient. Varying with the ship's block coefficient $C_b$, loading conditions, and the Froude number $F_n$. The block coefficient of a ship is the ratio of the underwater volume of the ship to the volume of a rectangular block having the same length, breadth, and depth.

$C_{Form}$ hull form coefficient. A function of ship type, the Beaufort number $BN$ and the ship displacement $\nabla$ in $\mathrm{m^3}$.

$V_a$ actual ship speed in the selected weather conditions $\mathrm{m\,s^{-1}}$.

$F_n$ Froude number. A dimensionless number associated with the nominal ship speed $V$ in calm water conditions, the ship length between perpendiculars $L_{pp}$ (m), and the gravitational acceleration $g$ ($\mathrm{m\,s^{-2}}$).

Then, the percentage loss of speed $V_{loss}$ can be expressed by the following (Kwon, 2008)

$$V_{loss} = \frac{\Delta V}{V} 100\% = C_\beta C_U C_{Form}, \tag{3.17}$$

where

$$\Delta V = V - V_a, \tag{3.18}$$

$$V_a = F_n \sqrt{L_{pp} g}. \tag{3.19}$$

Tables B.1, B.2, and B.3 in Appendix B list the empirical formulae to obtain the speed

direction reduction coefficient $C_\beta$, the speed reduction coefficient $C_U$, and the hull form coefficient $C_{Form}$, respectively.

### 3.4.2 Speed over ground affected by ocean currents

Besides an involuntary speed reduction due to wind and waves, the actual ship speed may be affected by ocean currents in both its magnitude and direction. Suppose a ship travels using a constant engine power along the rhumb line between two points $q_1$ and $q_2$ with coordinates $(\lambda_1, \varphi_1)$ and $(\lambda_2, \varphi_2)$, respectively. The magnitude of the nominal ship speed $V$ follows directly from the engine power.

In order to obtain the magnitude of the speed over ground $V_a$, we need to find the direction of the nominal ship speed such that the course over ground is equal to the 'compass bearing' $\beta$ from $q_1$ to $q_2$. The bearing is the clockwise angle between the line segment connecting points $q_1$ and $q_2$ and the north line. The direction of $V$ is equal to the ship's 'heading' $\gamma$, which is the ship's orientation about the north line. Figure 3.3 shows a schematic of the scenario described above.

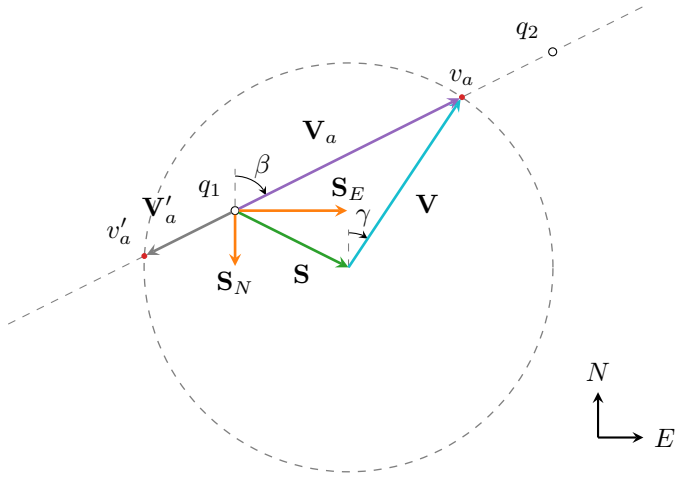



**Figure 3.3.** An illustration of the derivation of the speed over ground $V_a$.
Given are start $q_1$ and end $q_2$ points, nominal ship speed $V$, and eastward $S_E$ and northward $S_N$ components of the current velocity at point $q_1$. The line through $q_1$ and $q_2$ intersects the circle with radius $V$, centered at the ocean current velocity vector $\mathbf{S}$, at points $v_a$ and $v'_a$, where $v_a$ is the intersection point nearest to $q_2$. Then, the actual ship speed $V_a$ is represented by the length of vector $\mathbf{V}_a$, from $q_1$ to $v_a$. All vector lengths are measured in knots.

For the derivation of the speed over ground due to ocean current, we present the following approach. First, we calculate the bearing between points $q_1$ and $q_2$. Secondly, we express the eastward and northward velocity components of the speed over ground as a function of the ship's heading and the ocean current velocity at the ship's location. Then, we express the magnitude of the speed over ground as a function of nominal ship speed, bearing, and ocean current velocity.

A rhumb line is a straight line on the Mercator projection with an angle on the projection equal to the bearing $\beta$ (see Section 3.3.2). Hence, the compass bearing between points

$q_1 = (\lambda_2, \varphi_2)$ and $q_2 = (\lambda_2, \varphi_2)$ can be expressed as

$$\beta = \arctan2(\Delta\lambda, \Delta\psi), \tag{3.20}$$

measured in degrees, where $\Delta\lambda$ is the smallest difference in longitudes, $|\lambda_2 - \lambda_1| \leq 180°$) and $\Delta\psi = \psi_2 - \psi_1$ is the projected latitude difference. The projected latitude difference is obtained with Equation 3.14. The function $\arctan2(y, x)$ returns $\arctan(y/x)$ with adjustments for the signs of $x$ and $y$ so that the angle returned is the angle of the Cartesian point $(x, y)$ in polar coordinates.

Let us assume that the vector of the speed over ground $\mathbf{V}_a$ at a given point in time is the sum of the nominal ship velocity vector $\mathbf{V}$ and the velocity of the location and time-dependent ocean current $\mathbf{S}$. Also, we assume that the ship's heading $\gamma$ is equal to the direction of the nominal ship speed. Then, the eastward and northward velocity components of $\mathbf{V}_a$ are given by

$$V_a \sin\beta = V \sin\gamma + S_E, \tag{3.21}$$

$$V_a \cos\beta = V \cos\gamma + S_N, \tag{3.22}$$

respectively, where $\gamma$ is the heading, $V$ is the nominal ship speed, and $\beta$ is the bearing. $S_E = S_E(t, \varphi, \lambda)$ and $S_N = S_N(t, \varphi, \lambda)$ denote the eastward and northward ocean current velocity components, respectively, as functions of the time $t$, latitude $\varphi$, and longitude $\lambda$.

Now we rewrite Equation 3.22 with respect to $\gamma$

$$\gamma = \arccos \frac{V_a \cos\beta - S_N}{V}, \tag{3.23}$$

so that from substitution in Equation 3.21, it follows that

$$V_a \sin\beta = V \sin\Big(\arccos \frac{V_a \cos\beta - S_N}{V}\Big) + S_E. \tag{3.24}$$

We solve this last expression for $V_a$, outlined in Appendix C, resulting in two roots

$$V_a'' = S_E \sin\beta + S_N \cos\beta \pm \sqrt{V^2 - (S_E \cos\beta - S_N \sin\beta)^2}, \tag{3.25}$$

such that the square root has a real solution. This is true if $(S_E \cos\beta - S_N \sin\beta)^2 \leq V^2$, i.e., if the distance from the ocean current vector $\mathbf{S}$ to the line through $q_1$ and $q_2$ is smaller than the nominal ship speed.

We take the root with maximum value as the speed over ground, which is expressed by the function of time, position, and nominal ship speed

$$V_a = S_E \sin\beta + S_N \cos\beta + \sqrt{V^2 - (S_E \cos\beta - S_N \sin\beta)^2}, \quad \text{s.t. } V_a > 0, \tag{3.26}$$

with

$$S_E = S_E(t, \varphi, \lambda),$$
$$S_N = S_N(t, \varphi, \lambda).$$

To ensure the ship reaches the next waypoint, we require $V_a > 0$. In general, this is the case, as the nominal ship speed is typically larger than the ocean current velocity. Figure 3.3 shows that the vector $\mathbf{V}_a$ can also be interpreted as the sum of the ocean current vector $\mathbf{S}$ and nominal ship speed vector $\mathbf{V}$.

Following from above, the speed over ground at time $t$, between points $q_1$ and $q_2$, denoted as $V_a(t)$, is completely determined by the values of $(\lambda_1, \varphi_1)$ and $(\lambda_2, \varphi_2)$, the time $t_1$ at $q_1$, and by the value of $V$.

### 3.4.3 Average speed at a route leg

At this point, we are able to calculate the speed reduction due to wind and waves and the speed over ground due to ocean currents. If both environmental conditions are considered in the optimization, the actual ship speed at the ship's location is obtained with two steps. First, the reduced speed due to wind and waves is calculated using Equation 3.17. Then, the speed over ground caused by ocean currents is calculated with Equation 3.26 using the reduced speed calculated in the first step as the nominal ship speed $V$.

Let us assume that the geodesic between points $q_1$ and $q_2$ is described by $(\lambda(t), \varphi(t))$ on the time interval $[t_1, t_2]$. Then, the average actual ship speed can be expressed as

$$\overline{V_{a,i}} = \frac{1}{t_2 - t_1} \int_{t_1}^{t_2} V_a\left(t, \varphi(t), \lambda(t)\right) dt, \tag{3.27}$$

where $V_a\left(t, \lambda(t), \varphi(t)\right)$ is the actual ship speed in knots at location $(\lambda(t), \varphi(t))$ at time $t$, calculated using Equation 3.26.

Since time $t_2$ is unknown at this stage, it is not possible to solving the integral in Equation 3.27. To avoid calculation of this equation, we present a recursive procedure that uses the fundamental theorem of calculus and the actual ship speed $V_a(t)$ at time $t$.

Following from Theorem 1 in Appendix D, we can approximate a geodesic by the sum of infinitely small straight line segments. Suppose we split the geodesic $g$ connecting endpoints $q_1$ and $q_2$ into $m$ smaller segments, with each an equal geodesic length. Then, every rhumb line connecting the endpoints of each segment has equal length $D$.

Since the length of each segment becomes infinitely small, we assume that the actual ship speed at a segment is constant. The actual ship speed at a segment is represented by the actual ship speed at the start of the segment.

Then, given a geodesic $g$ connecting endpoints $q_1$ and $q_2$ and subdivided into $m$ segments of equal geodesic length, the average of the actual speed over $g$ can be recursively obtained

using

$$\overline{V_a} \approx \lim_{m \to M} \frac{1}{m} \sum_{j=1}^{m} V_a(t_j, \lambda_j, \varphi_j), \tag{3.28}$$

where

| | |
|---|---|
| $V_a(t_j, \lambda_j, \varphi_j)$ | is the actual ship speed in knots at the start the $j$-th segment, calculated using Equation 3.26, |
| $t_j$ | is the time at the start of segment $j$ in hours, |
| $\lambda_j, \varphi_j$ | are the latitude and longitude coordinates, respectively, of the start of the $j$-th segment in degrees, |
| $M$ | is a very large integer. |

Since route legs are defined as geodesic curves connecting two waypoints, the actual ship speed over a route leg can be calculated using Equation 3.28. The time at the start of a route leg $r_i$, for $r_i \in \mathbf{r}$, is given by

$$t_i = \begin{cases} t_1, & \text{for } i = 1, \\ \sum_{k=1}^{i-1} L_k / \overline{V_a^k}, & \text{otherwise}, \end{cases} \tag{3.29}$$

where $\overline{V_a^k}$ is the average of the actual ship speed at the $k$-th route leg, and $L_k$ is the length of the $k$-th route leg.

Now we can calculate the average of the ship's actual speed at leg $r_i$, $r_i \in \mathbf{r}$, given the nominal ship speed, the leg waypoints, the environmental conditions at the ship's location, and the start time at $r_i$. Hence, we can recursively obtain the average of the ship's actual speed of each leg, given the voyage departure time $t_1$.

## 3.5 Pareto optimality

In multi-objective problems, typically, there does not exist a single feasible solution that minimizes all objective functions simultaneously. Therefore, we treat multiple objectives with Pareto-evaluation. By evaluating a set of solutions according to multiple objectives, such as travel time and fuel cost, a Pareto front can be constructed consisting of Pareto-optimal solutions. Pareto-optimal solutions cannot be improved in any of the objectives without impairing one of the other objectives. In this section, we describe the concept of Pareto optimality. Here we note that it is a purely mathematical concept in which we assume that ordering of solutions is possible in terms of each single objective taken separately. Readers interested in more detailed descriptions should refer to (Ngatchou et al., 2005).

Let us denote $\mathbf{z} = f(\mathbf{r})$ as the vector of objective values of solution $\mathbf{r}$. Also, let us write route $\mathbf{r}_j$ as $\mathbf{j}$ for readability. An underlying definition of Pareto optimality is the notion of

Pareto dominance. For any two individuals $\mathbf{i}, \mathbf{j} \in R$,

$$\mathbf{i} \prec \mathbf{j} (\mathbf{i} \text{ dominates } \mathbf{j}) \Leftrightarrow \mathbf{z}_i < \mathbf{z}_j, \tag{3.30}$$

$$\mathbf{i} \preceq \mathbf{j} (\mathbf{i} \text{ weakly dominates } \mathbf{j}) \Leftrightarrow \mathbf{z}_i \leq \mathbf{z}_j. \tag{3.31}$$

A solution $\mathbf{r}^* \in R$ is called Pareto-optimal if there is no $\mathbf{r} \in R$ such that $\mathbf{z} < \mathbf{z}^*$. If $\mathbf{r}^*$ is Pareto-optimal, $\mathbf{z}^*$ is called efficient. The set of all Pareto-optimal solutions $\mathbf{r}^* \in R$ is $R^*$, the so-called Pareto set. And, the set of all efficient points $\mathbf{z}^* \in Z$ is $Z^*$, the Pareto front. A Pareto front and Pareto dominance relationships of three neighboring individuals are illustrated in Figure 3.4.

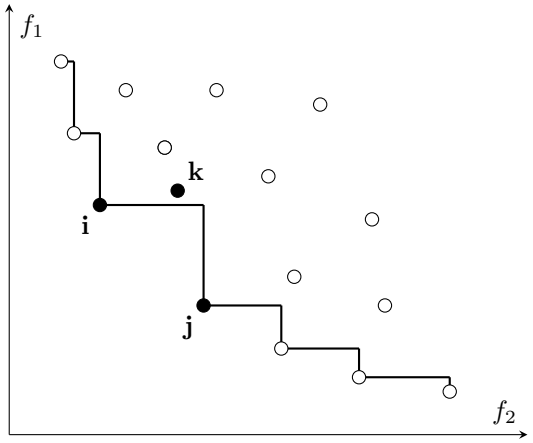



**Figure 3.4.** A sample Pareto front for a two-objective minimization problem.
Solution **k** is dominated by both solutions **i** and **j**; hence, its value is not on the Pareto front. The objective values of **i** and **j** lie on the Pareto front, since both solutions weakly dominate each other.

Since the MOSWR problem is a constrained problem, the classic Pareto dominance needs to be extended to constraint dominance (Deb, 2000). This constrained dominance also considers the feasibility of the individual. That is, an infeasible individual, violating at least one constraint, is said to be dominated by every other individual that is feasible. The following two-step procedure determines the constraint dominance relation between two individuals. An individual **i** constraint dominates another individual **j** if and only if

1. **i** is feasible and **j** is not, or
2. both **i** and **j** are feasible and **i** dominates **j** as in the classic Pareto dominance approach.

# 4 Solution method

In the previous chapter, we formulated the MOSWR problem as a continuous multi-objective optimization problem. Principally, it is the problem of seeking a set of Pareto-optimal routes while considering environmental conditions along the trajectory. The route decision variables are a series of waypoints in continuous space and a constant engine speed between two consecutive waypoints.

In this chapter, we present an evolutionary approach to solving the MOSWR problem. This approach uses an MOEA as a framework. We select three well-known MOEAs from the literature, and describe the performance metrics for the comparison of the selected MOEAs.

An MOEA initializes with a set of initial routes. For fast convergence to a set of optimal routes, we present an initialization procedure that finds a set of feasible initial paths on a graph optimized for each objective taken separately, as well as paths including and excluding major shipping canals and paths avoiding shallow waters. Next, we introduce recombination and mutation operators specialized for ship weather routing. We present a hybridized mutation method that combines uniform mutation and Gaussian variation.

To test whether a route leg intersects with an obstacle or passes a high-cost area, we propose a method for testing whether a line segment intersects a polygon. First, we describe an existing method for two-dimensional line segment intersections. Secondly, we reduce the number of calculations by this method with an indexing method specialized for spatial access methods.

## 4.1 Multi-objective evolutionary algorithm

In multi-objective optimization, a single solution optimizing all objectives generally does not exist. Instead, we seek a set of Pareto optimal solutions. The multi-objective evolutionary algorithms characteristics desirable for performing this task, most notably the handling of multiple candidate solutions. It finds a Pareto set in a single run and is especially useful for expensive multi-objective optimization problems because the solutions in a generation can be evaluated in parallel. In this study, we test three MOEAs on solving the MOSWR problem. Before describing these three algorithms, the remainder of this section discusses the algorithms' general characteristics.

The algorithms considered in this thesis use binary tournament selection to select parent routes. This selection procedure randomly samples (with or without replacement) two routes from the current generation. One of these routes is chosen as a parent based on one or multiple fitness values assigned to these routes by the algorithm. A so-called mating pool is filled with the selected parents.

When the mating pool is filled, pairs of parents are used to create offspring, i.e., new routes. Multiple methods exist to do so. The $k$-point crossover operator, frequently used in EAs, splits the parents at $k$ random points and recombines the obtained slices to form two children. Single-point crossover is special cases with $k = 1$, which we use in this study.

After creating a child route, a mutation may occur, enabling diversification and escaping from local minima. We use four different mutation operators, each selected with a certain probability. Section 4.4.2 describes these mutation operators and introduces the concept of dynamically selecting well-performing operators.

Some MOEAs use an elite preservation mechanism in their procedures. Besides the regular population, this procedure stores certain solutions in a global archive. It updates the archive in each generation by adding new well-performing solutions and possibly removing the worst. In general, the archive is larger than the regular population. Different works have shown that elitism is crucial for the convergence of MOEAs (Bosman & Thierens, 2003; Rudolph, 1994; Zitzler et al., 2000). The inclusion of elitism in an MOEA eliminates the chance of any undesired loss of information during the mutation stage and significantly improves performance. Therefore, it contributes to finding a set of multiple routes that is as close as possible to the true Pareto optimal front.

We selected three MOEAs from the literature based on two criteria:

1. low computational complexity, i.e., few solution evaluations required per generation, and
2. the ability to find a set of multiple routes as close as possible to the true Pareto optimal front.

Table 4.1 lists the selected algorithms and provides a rough comparison of MOEA characteristics. Sections 4.1.1, 4.1.2, and 4.1.3 describe these algorithms in more detail.

**Table 4.1.** Multi-objective evolutionary algorithms considered in this study.

| | M-PAES | NSGA-II | SPEA2 |
|---|---|---|---|
| Fitness assignment | local search: PAES | non-dominated sorting | dominator strength |
| Diversity mechanism | cell-based density | crowding distance | $k$-th nearest neighbor |
| Selection | binary tournament | binary tournament | binary tournament |
| Replacement | generational | elitist | generational |
| Archiving | global and local | none | local |

All of these algorithms implement an elite preservation mechanism in varying ways and are known for their low computation complexity. Both NSGA-II and SPEA2 are proven methods in the ship routing literature (see Section 2.1.4) and are two of the most frequently encountered MOEAs in the multi-objective optimization literature. The central theme in these two algorithms is the fitness assignment according to some kind of non-dominated sorting, and the diversity preservation among solutions in the Pareto set. NSGA-II incorporates a fast non-dominated sorting algorithm and uses Pareto optimality levels as the primary criterion to select solutions. And SPEA2 derives the strength of each solution from

the number of other solutions it dominates. In addition, NSGA-II uses the crowding distance to maintain a diverse set of solutions, whereas SPEA2 applies the $k$-nearest neighbor approach.

M-PAES is an MOEA hybridization using a local search procedure: a so-called memetic algorithm. Compared to general MOEAs, memetic algorithms are able to offer not only better convergence speed, but also a better approximation of the Pareto front (Lara et al., 2010). Being selective in the local search phase, this algorithm is computationally efficient while maintaining diversity. Memetic algorithms are not yet earlier implemented as part of a solution approach to the ship weather routing problem. Therefore, along with M-PAES having favorable characteristics, testing M-PAES on solving the MOSWR problem may provide promising results. The remainder of this section describes the procedures of the selected MOEAs in more detail.

### 4.1.1 NSGA-II

Non-dominated Sorting Genetic Algorithm (NSGA) was first introduced by Srinivas and Deb (1994). Later, NSGA-II was introduced by Deb and Goel (2001), improving the computational complexity and including a diversity preservation strategy based on sharing in NSGA and on crowding in NSGA-II. Furthermore, it uses an elitist principle by emphasizing non-dominated solutions. The selection of individuals is based on their nondomination rank and crowding distance, i.e., the density of the objective space around the solution.

At each generation $t$, the offspring population $Q_t$ of size $N$ is first created by using the parent population $P_t$ and the genetic operators we describe in Section 4.4. The two populations are then combined to form a new population $R_t$ of size $2N$. After that, the algorithm classifies the population $R_t$ into multiple non-dominated fronts using a fast non-dominated sorting procedure. The new population $P_{t+1}$ is then filled by routes of the different non-dominated fronts, one at a time. The first non-dominated front $Z_{t,1} \subseteq R_t$ is filled with the first class, i.e., the routes not dominated by any other strategy in $P_t$. The filling continues with routes of the second non-dominated front $Z_{t,2} \subseteq R_t$, such that it contains the routes not dominated by any other route in the remainder of the population (i.e., $P_t \setminus F_{t,1}$), and so on.

Since the overall population size of $R_t$ is $2N$, not all fronts can be filled in the new population of size $N$. All fronts that could not be accommodated are deleted. When the last allowed front, say $Z_l$, is considered, there may exist more routes in the front than the slots remaining in the new population. This scenario is illustrated in Figure 4.1. Instead of arbitrarily discarding some members from the last front, the algorithm chooses the routes that will make the diversity of the selected routes the highest.

In doing this, the algorithm performs a crowded-sorting of the routes of $Z_l$ and chooses the routes with the highest crowding distance values. The crowding distance $d_i$ of route $\mathbf{r}_i$ (also denoted as $\mathbf{i}$) is a measure of the objective space around $\mathbf{i}$ which is not occupied by any other route in the considered front $Z_l$. Here, the quantity of $d_i$ is calculated by

estimating the cuboid's perimeter formed by using the nearest neighbors in the objective space as the vertices, see Figure 4.2b. To ensure the selection of the front's extreme members, their crowding distance is set to infinity.

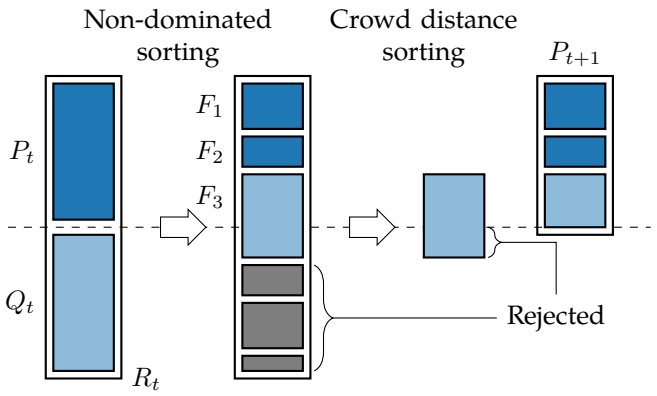
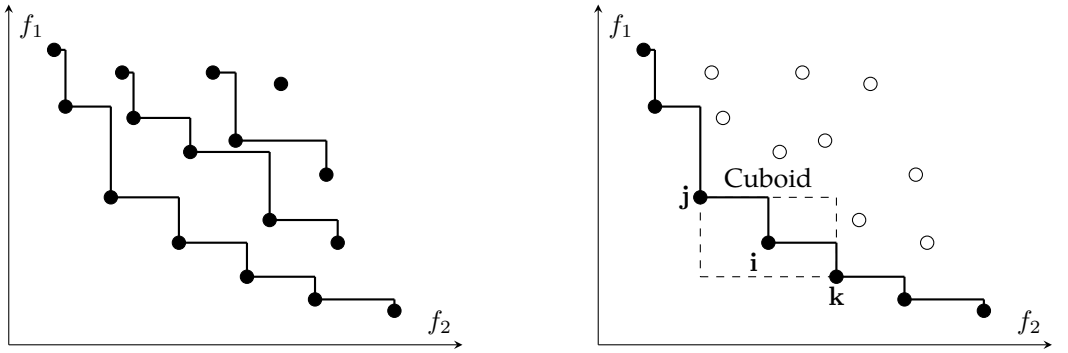



**Figure 4.1.** Schematic of the NSGA-II procedure.
The combination of the current population $P_t$ and its offspring $Q_t$ is non-dominated sorted into fronts $Z_1$, $Z_2$, etc. Then, the solutions of the last front which cannot be accommodated fully are sorted based on their crowding distance. The solutions with the largest crowding distance are chosen for the new population $P_{t+1}$.




(**a**) Four fronts

(**b**) Crowding distance calculation of route **i**

**Figure 4.2.** The concept of fronts and crowding distance used in NSGA-II.
Each dot represents a route in the objective space.

### 4.1.2 SPEA2

SPEA2 (Zitzler et al., 2001) is an improved version of the Strength Pareto Evolutionary Algorithm (SPEA) introduced by Zitzler and Thiele (1999). Besides the regular population $P_t$ of size $N$, it keeps an external archive $\overline{P}_t$ of size $\overline{N}$, at generation $t$. Each individual $\mathbf{i}$ in $\overline{P}_t$ and $P_t$ is assigned a fitness value $Z(\mathbf{i})$. The calculation of $Z(\mathbf{i})$ involves multiple steps. First, each solution $\mathbf{i} \in P_t \cup \overline{P}_t$ is assigned a strength value, representing the number of solutions it dominates, given by

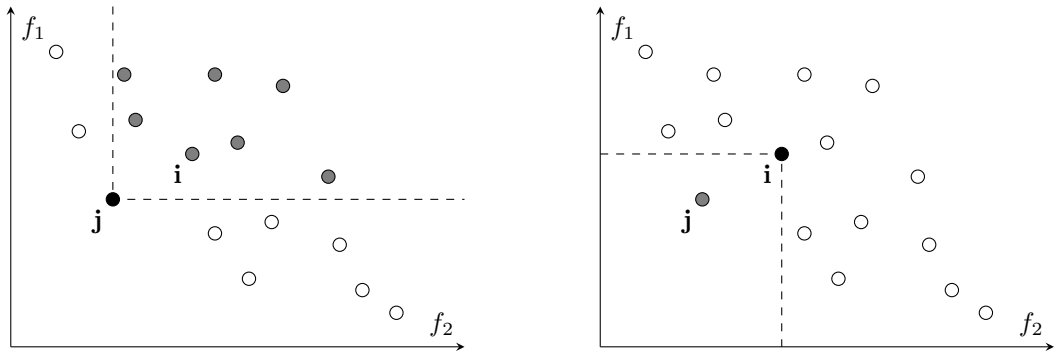
$$S(\mathbf{i}) = |\{\mathbf{j} \in P_t \cup \overline{P}_t : \mathbf{i} \succ \mathbf{j}\}|, \tag{4.1}$$

where $|\cdot|$ denotes the cardinality of a set, and $\succ$ corresponds to the Pareto dominance relation.

Then, based on the strength values, the raw fitness $R(\mathbf{i})$ of a route **i** is calculated as

$$R(\mathbf{i}) = \sum_{\mathbf{j} \in P_t \cup \overline{P}_t \,:\, \mathbf{j} \succ \mathbf{i}} S(\mathbf{j}). \tag{4.2}$$

That is, the raw fitness is determined by the strength of its dominators in both the archive and population. Let us consider the example in Figure 4.3. Route **j** dominates seven routes, as shown in Figure 4.3a. Its strength is, therefore, equal to 7. Route **i** is dominated by **j** and by **j** only. Thus, its raw fitness $R(\mathbf{i}) = 7$.




(**a**) Strength, $S(\mathbf{j}) = 7$ (**b**) Raw fitness, $R(\mathbf{i}) = S(\mathbf{j}) = 7$

**Figure 4.3.** The concept of strength and raw fitness used in SPEA2.
Each dot represents a route in the objective space.

Like the crowding distance of NSGA-II, SPEA2 uses a density measure to ensure diversity in the selected individuals for the next population. To calculate the density $D(\mathbf{i})$ of a route $i$, the Euclidean distance to each other route in the objective space is calculated. The list of distances relative to route **i** is then sorted in ascending order. The density $D(\mathbf{i})$ takes the inverse of the $k$-th distance of the list, denoted by $\sigma_i^k$. A typical setting for $k$ is the square route of the sample size, i.e., $k = \lfloor \sqrt{N + \overline{N}} \rfloor$. Afterward, the density $D(\mathbf{i})$ of route **i** is defined by

$$D(\mathbf{i}) = \frac{1}{\sigma_i^k + 2}. \tag{4.3}$$

To ensure $0 < D(\mathbf{i}) < 1$, 2 is added in the denominator. Finally, the fitness value $F(\mathbf{i})$ of route **i** is calculated as the sum of its raw fitness and strength

$$F(\mathbf{i}) = R(\mathbf{i}) + D(\mathbf{i}). \tag{4.4}$$

After assigning fitness values to each individual in $P_t$ and $\overline{P}_t$, all non-dominated routes are copied to $\overline{P}_{t+1}$, i.e., those who have a fitness lower than one, so that

$$\overline{P}_{t+1} = \{\mathbf{i} \in P_t \cup \overline{P}_t : F(i) < 1\}. \tag{4.5}$$

If $|\overline{P}_{t+1}| = \overline{N}$, the construction of the archive is completed. In case the archive is too small ($|\overline{P}_{t+1}| < \overline{N}$), the archive is filled with the best individuals from the previous archive and population. Else, if the archive is too large ($|\overline{P}_{t+1}| > \overline{N}$), individuals are iteratively removed

from $\overline{P}_{t+1}$ by a truncation procedure until $|\overline{P}_{t+1}| = \overline{N}$. In each iteration, the individual $\mathbf{i}$ with the smallest distance to another individual is, denoted by $\mathbf{i} \leq_d \mathbf{j}$ for $\mathbf{i}, \mathbf{j} \in \overline{P}_{t+1}$, where

$$\begin{aligned} \mathbf{i} \leq_d \mathbf{j} \iff & \forall 0 < k < |\overline{P}_{t+1}| : \sigma_i^k = \sigma_j^k \quad \vee \\ & \exists 0 < k < |\overline{P}_{t+1}| : [(\forall 0 < l < k : \sigma_i^l = \sigma_j^l) \wedge \sigma_i^k < \sigma_j^k], \end{aligned} \tag{4.6}$$

where $\sigma_i^k$ denotes the distance of individual $\mathbf{i}$ to its $k$-th nearest neighbor $\overline{P}_{t+1}$.

The next step represents the mating selection phase, where the algorithm performs binary tournaments on the new archive $\overline{P}_{t+1}$ to fill the mating pool with $N$ routes. Finally, it applies recombination and mutation operators to the mating pool to create the new population $P_{t+1}$.

### 4.1.3 M-PAES

As a third MOEA, we select Multi-Pareto Archived Evolution Strategy (M-PAES), a memetic algorithm presented by Knowles and Corne (2000). Memetic algorithms are a special case of hybrid MOEAs that incorporate local search methods to reduce the likelihood of premature convergence. The local search method in M-PAES is based on the Pareto Archived Evolution Strategy (PAES) algorithm (Knowles & Corne, 1999). PAES is a local search algorithm unique in its use of a form of Pareto ranking for selection. It performs well on a range of multi-objective problems, and it is suitable for the hybridization of MOEAs. Therefore, M-PAES uses a population of solutions and recombines the local optima found by applying PAES independently on the population members.

Similar to SPEA2, M-PAES uses archives to store solutions that serve a dual purpose. Firstly, it memorizes the solutions found during the algorithm execution, and, secondly, it uses the solutions in the archives to compare new candidate solutions by using Pareto dominance. M-PAES requires two archives to perform these two tasks, as each local search phase needs to be partially independent of the global search. Thus, we maintain a global archive $G$ of a finite set of non-dominated solutions, and a local archive $H$, used as a comparison set in the local search phase.

The algorithm initializes with a population $P_0$ of $N$ routes and updates the global archive $G$ with the non-dominated individuals in $P_0$. At generation $t$, the local search procedure, shown in Figure 4.4, is applied to each individual independently in population $P_t$. It starts with an empty local archive $H$ and fills it with the candidate solution $\mathbf{i}$ and its non-dominated solutions in $G$, so that

$$H = \mathbf{i} \cup \{\mathbf{j} \in G : \mathbf{i} \preceq \mathbf{j}\}. \tag{4.7}$$

The procedure continues by mutating the candidate solution $\mathbf{i}$ to obtain the neighborhood solution $\mathbf{i}'$, and estimating its quality by comparison to the local archive $H$. The new neighborhood solution is rejected if it is dominated by the current solution and accepted if it dominates the current solution. If the two solutions are mutually non-dominated, the TEST procedure compares the two solutions with a local archive of potentially Pareto optimal so-

lutions and accepts the one that resides in the least crowded region of the objective space. This acceptance rule is an additional dispersion mechanism. We use the crowding-distance measure from the NSGA-II algorithm to measure the density of the objective space around a solution.

Each time $\mathbf{i}'$ is dominated by the current solution $\mathbf{i}$, the variable $fails$, initially zero, is incremented. If the neighborhood solution is accepted as the new solution, $fails$ is reset to zero. Thus, $fails$ counts the number of rejected solutions between improving moves. If this number exceeds the maximum number of fails $loc_{fails}$ or if the number of moves exceeds the maximum number of moves $loc_{fails}$, the local search terminates.

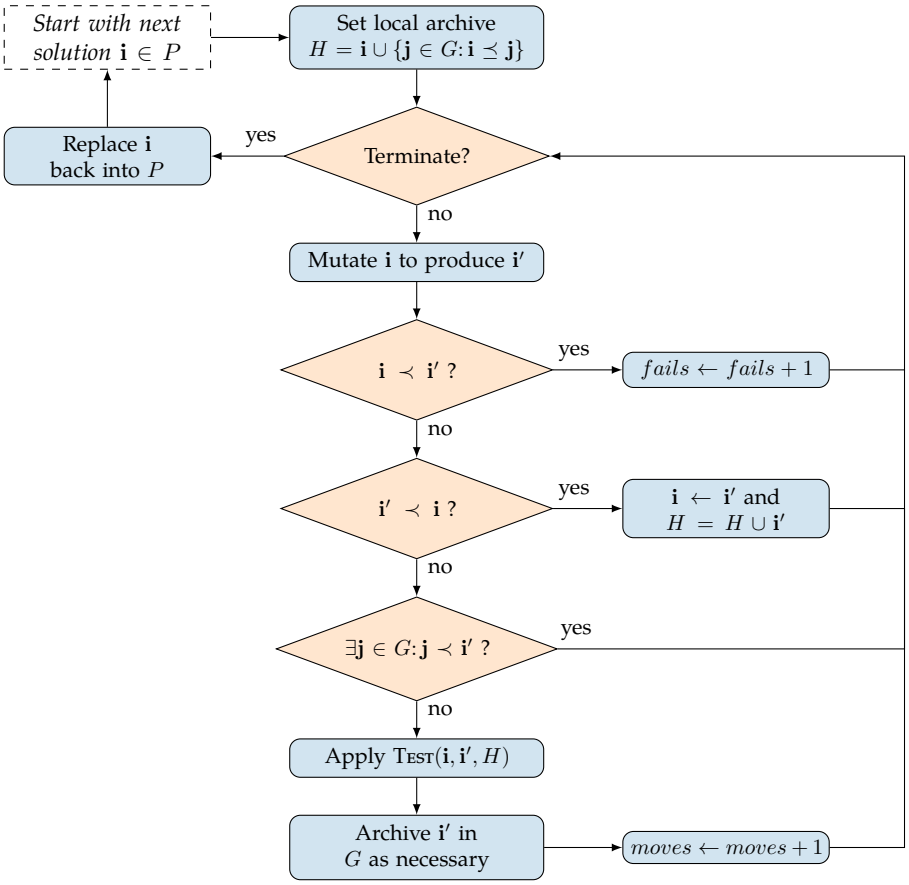



**Figure 4.4.** Diagram of the M-PAES local search procedure.

After improving each member of the $P_t$, M-PAES continues with the recombination phase. It initializes with an empty new population (i.e., $P_{t+1} = \emptyset$), which is iteratively filled to size $N$ by the following procedure. For recombination, the procedure randomly selects two parents from the union of the post-local search population and the global archive. The resultant child is accepted if it is non-dominated by the entire global archive, and it resides in a less crowded region than at least one of its parents. If it dominates any member of $G$, it is accepted as well. However, a child dominated by any member of $G$, or residing in more crowded regions than both parents, is rejected. In this case, two new parents are selected again, and recombination is applied once more. The procedure is repeated until either a child is accepted or a maximum number of recombinations $rec_{trials}$ is exceeded.

In the latter case, a solution is selected by performing a binary tournament on the global archive and added to the next population $P_{t+1}$.

## 4.2 Multi-objective performance metrics

The previous section describes three MOEAs, which we test according to their performance on solving the MOSWR problem using specialized Pareto performance metrics. Zitzler et al. (2003) give an extensive analysis and review of various performance measures (indicators), such as the ones described below. Two types of metrics are used in multi-objective optimization literature: unary and binary metrics.

Unary metrics receive as parameter only one set of objective solutions $Z$ that approximates the Pareto front $Z^*$ to be evaluated. Each unary metric $I$ is a function $I : Z \rightarrow \mathbb{R}$ mapping an objective set to a real value after considering a combination of (1) cardinality, (2) accuracy, and (3) diversity metrics.

The cardinality of $Z$ refers to the number of solutions that exist in $Z$. The accuracy of $Z$ refers to the convergence of $Z$ to the theoretical Pareto front. Lastly, the diversity of an objective set gives an indication of the distribution and the range of values covered by $Z$. As unary measures for accuracy are designed to compare an objective set to the Pareto front, it requires information on this front. Because we do not have this information for the MOSWR problem, unary metrics are less suitable to assess the performance of the selected MOEAs.

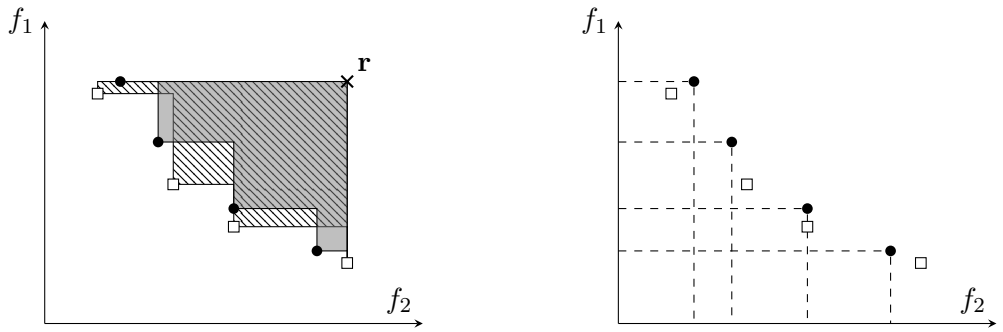
Binary metrics do not require any information on the Pareto front. These metrics consider mainly the relationship between two objective sets, $Z$ and $Z'$, in terms of dominance to give an idea of which one is better. Riquelme et al. (2015) present a review and analysis of numerous Pareto metrics in publications of Evolutionary Multi-Criterion Optimization conferences and discuss the most cited ones.

We select one binary metric, the $\mathcal{C}$ metric, from the literature for measuring the cardinality and dominance relations of a pair of Pareto approximation set. Additionally, we select the hypervolume or $\mathcal{S}$ metric, which is a unary metric, and apply some modifications so that it is suitable for comparing multiple Pareto approximation sets. Figure 4.5 illustrates the metrics for sample solution sets.

### 4.2.1 Hypervolume metrics

The hypervolume indicator, or $\mathcal{S}$ metric, is the most popular unary performance metric in recent literature (Riquelme et al., 2015). It provides a quantitative measure of convergence to the true Pareto front and diversity in a combined sense. This hypervolume indicator can be defined as the hypervolume of the $m$-dimensional objective space that is dominated by the set $Z$ and is bounded by a reference point $\mathbf{r} \in R^m$, such that for all $\mathbf{z} \in Z$, $\mathbf{z} \prec \mathbf{r}$. The hypervolume indicator of a Pareto approximation set $Z$ with for $m = 2$ is given by

$$HV(Z, \mathbf{r}) = \bigcup_{\mathbf{z} \in Z} [z_1, r_1] \times [z_2, r_2], \tag{4.8}$$




(**a**) Hypervolumes of $Z$ (hatched) and $Z'$ (gray), bounded by their mutual reference point $\mathbf{r}$.

(**b**) $\mathcal{C}$ metric. Two of four solutions in $Z'$ are (weakly) dominated by a solution from $Z$; thus, $\mathcal{C}(Z, Z') = 2/4$.

**Figure 4.5.** Illustration of performance measures for a two-objective minimization problem. White squares and black dots represent the solutions from approximation sets $Z$ and $Z'$ in objective space, respectively.

where $[z_1, r_1] \times [z_2, r_2]$ is the rectangle consisting of all points that are dominated by $\mathbf{z}$, but not dominated by the reference point.

If the Pareto front $Z^*$ is known, the hypervolume ratio is defined by

$$HVR(Z, Z^*, \mathbf{r}) = \frac{HV(Z, \mathbf{r})}{HV(Z^*, \mathbf{r})}, \tag{4.9}$$

where the mutual reference point $\mathbf{r}$ can be found by constructing a vector of worst objective function values in $Z^*$.

This ratio maps the pair of $m$-dimensional fronts to the interval $(0, 1]$, as the Pareto approximation front's hypervolume is always positive and less or equal to the hypervolume of the true Pareto front. A hypervolume ratio $HVR(Z, Z^*, \mathbf{r}) = 1$ indicates that the Pareto approximation set dominates an equal area in the objective space, which means that the fronts are identical.

We modify the hypervolume ratio for our use case by calculating the ratio of the hypervolume of a Pareto approximation set $HV(Z)$ to hypervolume of its union with the reference set $HV(Z \cup Z')$. For two Pareto approximation sets, we define the binary hypervolume ratio $HVR_2(Z, R)$ by

$$HVR_2(Z, Z', \mathbf{r}) = \frac{HV(Z, \mathbf{r})}{HV(Z \cup Z', \mathbf{r})}, \tag{4.10}$$

where the reference point $\mathbf{r}$ can be found by constructing a vector of worst objective function values in $Z \cup Z'$. Figure 4.5a shows two overlapping hypervolumes of a pair of Pareto approximation sets. The cross indicates the reference point $\mathbf{r}$.

By representing $Z'$ as the union of $n$ reference sets, i.e., $Z' = \bigcup_{i=1}^{n} Z'_i$, we can define a more generic $n$-ary hypervolume ratio by

$$HVR_n(Z, \bigcup_{i=1}^{n} Z'_i, \mathbf{r}) = \frac{HV(Z, \mathbf{r})}{HV(Z \cup \bigcup_{i=1}^{n} Z'_i, \mathbf{r})}. \tag{4.11}$$

Again, $\mathbf{r}$ denotes the mutual reference point of the union of all Pareto fronts. We find $\mathbf{r}$ by constructing a vector of worst objective function values in $Z \cup Z'$, where $Z' = \bigcup_{i=1}^{n} Z'_i$.

The $m$-ary hypervolume ratio gives the ratio of the area dominated by $Z$ with respect to the total area dominated by $Z \cup Z'$. Hence, $HVR_n(Z, \bigcup_{i=1}^{n} Z'_i, \mathbf{r}) = 1$ indicates that the $Z$ covers the entire area that is dominated by the union of all considered Pareto approximation sets $Z \cup \bigcup_{i=1}^{n} Z'_i$. In other words, $Z$ is equal to or has better performance than the union of all reference sets $\bigcup_{i=1}^{n} Z'_i$. Contrarily, if the same ratio converges to 0, it indicates that the hypervolume of $Z$ is negligible compared to the reference sets.

For the performance measurement of the three selected MOEAs in this study, we use the ternary hypervolume ratio $HVR_3$ to get a global indication of the best Pareto approximation set. We use the binary hypervolume ratio $HVR_2$ to make a mutual comparison of an approximation set to either of the two resulting approximation sets. As a result, we compute three ternary hypervolume ratios and six binary hypervolume ratios to measure the performance of three Pareto approximation sets.

The computational complexity of calculated a hypervolume is $\mathcal{O}(n^{m+1})$, where $n$ is the number of points in a Pareto (approximation) set and $m$ denotes the number of objectives. Hence, in the case of $m = 3$, the computation time increase with the number of points to the third power. To keep this large computation time at a minimum, we reuse the calculated hypervolumes where possible. For example, for three Pareto approximation $A, B, C$, both binary hypervolume ratios $HVR_3(A, (A \cup B), \mathbf{r})$ and $HVR_3(B, (A \cup B), \mathbf{r})$ require the hypervolume $HV(A \cup B, \mathbf{r})$.

The hypervolume ratio is non-cardinal. That is, it does not compare a pair of Pareto approximation sets according to their cardinality. Therefore, the following section describes a cardinal measure we select to assess the performance of the tested algorithms based on cardinality and dominance relations.

### 4.2.2 $\mathcal{C}$ metric

The $\mathcal{C}$ metric, or Two Sets Coverage ratios, is a binary performance metric proposed by Zitzler and Thiele (1998). Let $Z, Z' \subseteq X$ be two Pareto approximation sets, with $X$ representing the objective space. $\mathcal{C}$ is a cardinality measure and maps the ordered pair $(Z, Z')$ to the interval $[0, 1]$ by capturing the proportion of points in a Pareto front approximation $Z'$ that are dominated by at least one point in the Pareto set approximation $Z$. Zitzler and Thiele define the $\mathcal{C}$ metric as

$$\mathcal{C}(Z, Z') = \frac{|\{\mathbf{z}' \in Z' : (\exists \mathbf{z} \in Z : \mathbf{z} \preceq \mathbf{z}')\}|}{|Z|}. \tag{4.12}$$

The value $\mathcal{C}(Z, Z') = 1$ means that all the elements of $Z'$ are (weakly) dominated by the elements of $Z$. The opposite, $\mathcal{C}(Z, Z') = 0$, indicates that none of the solutions in $Z'$ is weakly dominated by any element in $Z$. Both orderings always need to be considered, as $\mathcal{C}(Z, Z')$ is not necessarily equal to $1\mathcal{C}(Z', Z)$.

Knowles and Corne (2002) note that the non-symmetric nature of $\mathcal{C}$ complicates the analysis of the dominance relations between a pair of sets. The $\mathcal{C}$ metric provides only comparable results if $\mathcal{C}(Z, Z') = 1$ and $\mathcal{C}(Z', Z) < 1$. For other values, $\mathcal{C}$ may not give an ordered comparison of the Pareto approximation set. That is, for three Pareto approximation $A, B, C \subseteq X$, $\mathcal{C}$ might consider $B$ better than $A$, $C$ better than $B$, but $A$ better than $C$. Therefore, we can only draw relevant conclusions if the $\mathcal{C}$ metrics provide decisive results. However, for two evenly-distributed sets of the same cardinality, the metric gives comparable results and an intuitive notion of quality.

The pros of the $\mathcal{C}$ metric are its low computational time compared to the hypervolume ratio, and it is scale and reference point independent. Furthermore, no knowledge of the true Pareto front $Z^*$ is required to assess the dominance relations of a pair of Pareto approximation sets.

## 4.3 Initialization

Section 4.1 described three MOEAs that we test on the Multi-Objective Ship Weather Routing problem using the performance metrics described in the previous section. An MOEA initializes with a set of initial routes. If this set contains feasible routes and approximates the Pareto set, the MOEA will converge quickly. Hence, we present an initialization procedure that finds such a set given a departure and destination. Optimizing for each objective taken separately, it finds initial paths on a Geodesic Discrete Global Grid (GDGG) with finer resolutions at coastlines using the A* algorithm (Hart et al., 1968). Additionally, it calculates alternative routes avoiding water channels if passing any in an obtained initial path. The following section gives a basic definition of a GDGG and describes the construction of the one we use in this study.

### 4.3.1 Geodesic Discrete Global Grid

A directed shortest path algorithm tries to find initial paths on a network of nodes and arcs within the navigable area in the initialization procedure. We use a GDGG to construct this graph, which is a class of Discrete Global Grids (DGGs). A DGG is a space partitioning consisting of a set of regions that reference the Earth's surface (Sahr et al., 2003). Each region is represented by a single point, typically its centroid. A region-point combination is called a "cell". Commonly used DGGs are square-cell ones based on the geodetic coordinate system; for example, WGS84 (NIMA, 2000) a standard DGG system for use in cartography. Such systems have numerous practical advantages but also limitations. Either square DGGs do not have equal-area cell regions or become increasingly distorted in area and shape spacing as one moves north and south from the equator. Also, square grids do not have uniform adjacency; that is, each square grid cell has four neighbors with which it shares an edge and four neighbors with which it shares only a vertex. Many researchers have proposed alternatives, one of which is the Geodesic DGG, constructed by discretizing a sphere based on multi-resolution partitions of polyhedra. A complete survey of systems using a GDGG is conducted by Sahr et al. (2003).

In this study, we model the Earth with a GDGG by projecting each face of a polyhedron onto the surface of the Earth's sphere. The edges of the polyhedron become great circle segments that form the mesh that partitions the sphere. Polyhedra with smaller faces reduce the distortion introduced when transforming to its corresponding spherical surface (White et al., 1998). Therefore, the sphere approximation is based on a recursive subdivision of the faces of a regular solid that results in a variable resolution of the underlying area. Large homogeneous regions, e.g., oceans, yield low-resolution partitions, whereas busy areas, such as coastlines, cause finer resolutions to be created.

We choose the hexagon as a basis for the GDGG. Among the three regular polygons that tile the plane (i.e., triangles, squares, and hexagons), hexagons are the most compact. They represent a spherical surface with the smallest average error, and, unlike square and triangle grids, hexagon tiles do have uniform adjacency (Sahr et al., 2003). Moreover, Yap (2002) shows that the branching factor of both a hexagon and square grid is 3 for a directed shortest path algorithm and that one doesn't need to search as deep on a hexagon graph. To illustrate this, the shortest path for both the hexagon and square grid is shown in Figure 4.6.

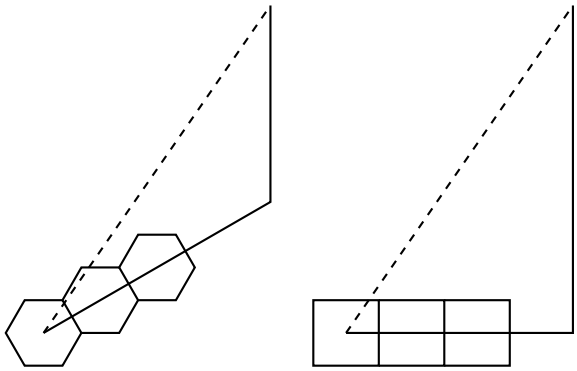


**Figure 4.6.** Optimal paths on different grids

Given the same distance, a square grid searches with depth $D$ while a hexagon grid will search with depth $0.81D$ (Yap, 2002). It follows that a directed shortest path algorithm searches through $\mathcal{O}(3^D)$ tiles on a square grid and searches $\mathcal{O}(3^{0.81D}) \approx \mathcal{O}(2.42^D)$ tiles on a hexagon grid. Hence, a hex grid is exponentially faster than a tile grid for a directed shortest path algorithm, such as A*.

To find the shortest path on a hexagon mesh, we define only the hexagon centroids and the arcs connecting the neighboring hexagons. This yields a triangle grid, which is the dual of a hexagon grid, shown in Figure 4.7.

The spherical versions of the five Platonic solids represent the only ways in which a sphere can be partitioned into cells, as each solid consist of congruent (identical in shape and size) regular (all angles equal and all sides equal) polygon faces, with the same number of polygons meeting at each vertex. These are the (1) tetrahedron with four triangles as faces, (2) the hexahedron (or cube) with six squares, (3) the octahedron with eight triangles, (4) the dodecahedron with twelve pentagons, and (5) the icosahedron with twenty

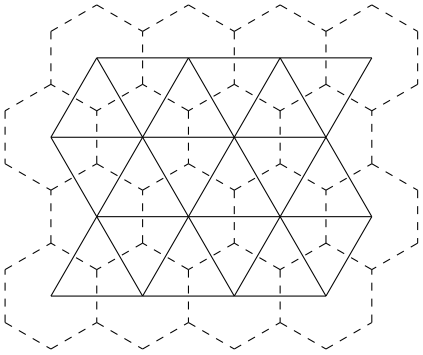


**Figure 4.7.** Hexagon grid and its dual, triangle grid.

triangles.

A finer approximation to the sphere can be obtained by subdividing the faces of the polyhedra recursively until the desired resolution is reached. We choose the geodesic partitioning of the icosahedron (a geodesic icosahedron) for our approximation to the Earth's sphere, as it has the most faces among the Platonic solids and each face is a triangle. This enables us to use its vertices and edges as nodes and arcs for the triangle grid. Figure 4.8 shows the icosahedron, a level 1 subdivision, and its dual with their vertices projected on a sphere.

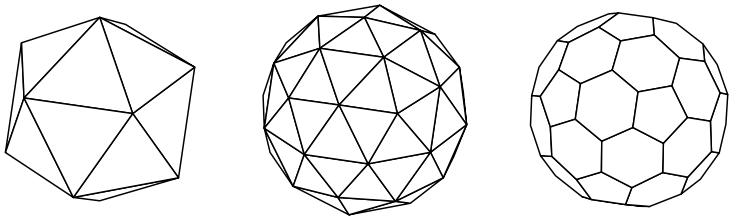


**Figure 4.8.** Geodesic icosahedron - subdivided once and the corresponding dual.

The web blog from Kahler (2009) provides a method for recursively subdividing the icosahedron, which we describe below. The icosahedron has twelve vertices $\mathbf{v}_i$, $i \in [12]$, represented by the corners of three orthogonal golden rectangles, i.e., rectangles whose edge lengths are in the golden ratio $1 : \phi$, with $\phi = \frac{1+\sqrt{5}}{2}$. Hence, for an icosahedron centered at the origin with an edge length of 2 we describe its vertices in a Cartesian coordinate system by circular permutations of $(0, \pm 1, \pm\phi)$, giving

$$\begin{array}{llll} \mathbf{v}_1 & \coloneqq (-1, & \phi, 0), & \mathbf{v}_5 \coloneqq (0, -1, \quad \phi), \quad \mathbf{v}_9 \coloneqq (\quad \phi, 0, -1), \\ \mathbf{v}_2 & \coloneqq (\quad 1, & \phi, 0), & \mathbf{v}_6 \coloneqq (0, \quad 1, \quad \phi), \quad \mathbf{v}_{10} \coloneqq (\quad \phi, 0, \quad 1), \\ \mathbf{v}_3 & \coloneqq (-1, -\phi, 0), & \mathbf{v}_7 \coloneqq (0, -1, -\phi), & \mathbf{v}_{11} \coloneqq (-\phi, 0, -1), \\ \mathbf{v}_4 & \coloneqq (\quad 1, -\phi, 0), & \mathbf{v}_8 \coloneqq (0, \quad 1, -\phi), & \mathbf{v}_{12} \coloneqq (-\phi, 0, \quad 1). \end{array} \tag{4.13}$$

Next, we define the twenty triangle faces $T_j$, $j \in [20]$, with the vertices ordered in counter-

clockwise direction, as

$$\begin{array}{llll} T_1 \coloneqq (\mathbf{v}_1, \mathbf{v}_{12}, \mathbf{v}_6\ ), & T_6\ \coloneqq (\mathbf{v}_2\ , \mathbf{v}_6\ , \mathbf{v}_{10}), & T_{11} \coloneqq (\mathbf{v}_4, \mathbf{v}_{10}, \mathbf{v}_5\ ), & T_{16} \coloneqq (\mathbf{v}_5\ , \mathbf{v}_{10}, \mathbf{v}_6\ ), \\ T_2 \coloneqq (\mathbf{v}_1, \mathbf{v}_6\ , \mathbf{v}_2\ ), & T_7\ \coloneqq (\mathbf{v}_6\ , \mathbf{v}_{12}, \mathbf{v}_5\ ), & T_{12} \coloneqq (\mathbf{v}_4, \mathbf{v}_5\ , \mathbf{v}_3\ ), & T_{17} \coloneqq (\mathbf{v}_3\ , \mathbf{v}_5\ , \mathbf{v}_{12}), \\ T_3 \coloneqq (\mathbf{v}_1, \mathbf{v}_2\ , \mathbf{v}_8\ ), & T_8\ \coloneqq (\mathbf{v}_{12}, \mathbf{v}_{11}, \mathbf{v}_3\ ), & T_{13} \coloneqq (\mathbf{v}_4, \mathbf{v}_3\ , \mathbf{v}_7\ ), & T_{18} \coloneqq (\mathbf{v}_7\ , \mathbf{v}_3\ , \mathbf{v}_{11}), \\ T_4 \coloneqq (\mathbf{v}_1, \mathbf{v}_8\ , \mathbf{v}_{11}), & T_9\ \coloneqq (\mathbf{v}_{11}, \mathbf{v}_8\ , \mathbf{v}_7\ ), & T_{14} \coloneqq (\mathbf{v}_4, \mathbf{v}_7\ , \mathbf{v}_9\ ), & T_{19} \coloneqq (\mathbf{v}_9\ , \mathbf{v}_7\ , \mathbf{v}_8\ ), \\ T_5 \coloneqq (\mathbf{v}_1, \mathbf{v}_{11}, \mathbf{v}_{12}), & T_{10} \coloneqq (\mathbf{v}_8\ , \mathbf{v}_2\ , \mathbf{v}_9\ ), & T_{15} \coloneqq (\mathbf{v}_4, \mathbf{v}_9\ , \mathbf{v}_{10}), & T_{20} \coloneqq (\mathbf{v}_{10}, \mathbf{v}_9\ , \mathbf{v}_2\ ). \end{array} \tag{4.14}$$

A triangle can be divided into $L^2$, $L \in \mathbb{N}^+$, smaller equilateral triangles by breaking each edge into $L$ pieces and connecting the break points with lines parallel to the triangle edges. $L$ is the so-called recursion level of a geodesic polyhedron. We rewrite the vertices of a triangle to $(\mathbf{v}_1', \mathbf{v}_2', \mathbf{v}_3')$, while keeping the counter-clockwise order, e.g., $T_1 : (\mathbf{v}_1, \mathbf{v}_{12}, \mathbf{v}_6) \rightarrow (\mathbf{v}_1', \mathbf{v}_2', \mathbf{v}_3')$. For $L = 1$, we find the three breakpoints $(\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3)$ by obtaining the midpoints of the triangle edges

$$\begin{aligned} \mathbf{w}_1 &= \frac{\mathbf{v}_1' + \mathbf{v}_2'}{|\mathbf{v}_1' + \mathbf{v}_2'|}, \\ \mathbf{w}_2 &= \frac{\mathbf{v}_2' + \mathbf{v}_3'}{|\mathbf{v}_2' + \mathbf{v}_3'|}, \\ \mathbf{w}_3 &= \frac{\mathbf{v}_3' + \mathbf{v}_1'}{|\mathbf{v}_3' + \mathbf{v}_1'|}, \end{aligned} \tag{4.15}$$

and the four new triangles are given by

$$\begin{aligned} T_1^1 &= (\mathbf{v}_1', \mathbf{w}_3, \mathbf{w}_2), \\ T_2^1 &= (\mathbf{v}_2', \mathbf{w}_1, \mathbf{w}_3), \\ T_3^1 &= (\mathbf{v}_3', \mathbf{w}_2, \mathbf{w}_1), \\ T_4^1 &= (\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3), \end{aligned} \tag{4.16}$$

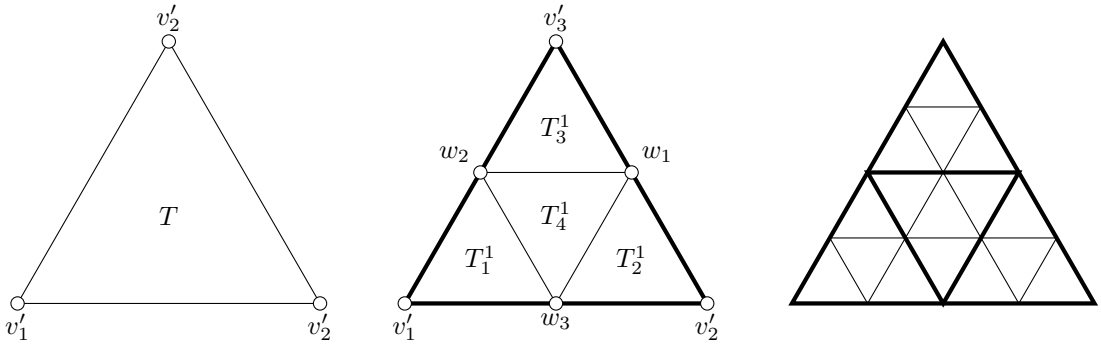
where the superscript denotes the recursion level $L = 1$. Each newly obtained vertex $\mathbf{w}$ is normalized to ensure that it lies on the unit sphere, i.e.,

$$\mathbf{w}' = \frac{\mathbf{w}}{|\mathbf{w}|}. \tag{4.17}$$

The recursion can be repeated to any desired level $L$. Figure 4.9 shows three recursion levels defined on a single triangle face.

If the edge length of a regular icosahedron is $c$, the radius of its circumscribed sphere is

$$\begin{aligned} r &= \frac{c}{2}\sqrt{\phi\sqrt{5}} \\ &= \frac{c}{2}\sqrt{\frac{1+\sqrt{5}}{2}\sqrt{5}} \\ &\approx 0.951{,}056{,}516 \cdot c. \end{aligned} \tag{4.18}$$




**(a)** $L = 0$ **(b)** $L = 1$ **(c)** $L = 2$

**Figure 4.9.** Three recursion levels $L$ of a single triangle face.

Thus, the edge length of an icosahedron with a radius equal to the Earth's mean radius ($\approx$ 3440.0695 nmi) is approximately 3617.10 nmi. To obtain the length of the edge projected on the Earth's sphere, we transform from Cartesian to the geodetic coordinate system. The conversion to longitude is

$$\lambda = \arctan2\,\frac{y}{x}, \tag{4.19}$$

and the conversion to latitude is

$$\phi = \arctan2(z, \sqrt{x^2 + y^2}), \tag{4.20}$$

where the function $\arctan2(y, x)$ returns the angle in the Euclidean space, in degrees, between the positive $x$-axis and the ray to the point $(x, y) \neq (0, 0)$. This transformation enables great circle distance calculations using the longitude and latitude of the vertices of an edge and the haversine formula described in Section 3.3.

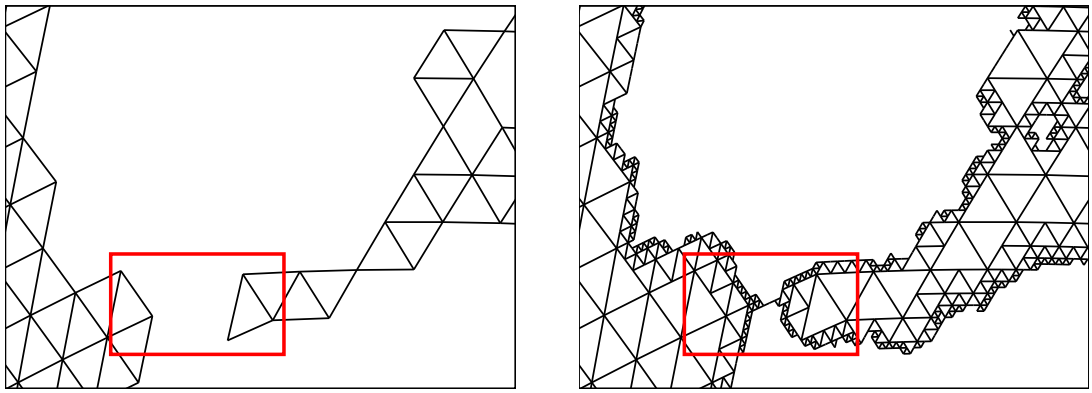
For a recursion level of $L$, the length of the icosahedron edges decreases with $2^L$. The corresponding great circle distance decrease with approximately the same rate and approximates the chord length as the recursion level increase, as shown in Table 4.2.

**Table 4.2.** Chord length $c$ and arc length $s$ of subdivided icosahedrons.
Each icosahedron is projected on the Earth's surface (with mean radius 3440 nmi) and is subdivided with $L$ recursions. All lengths are in nautical miles.

| L | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| c | 3617.10 | 1808.55 | 904.28 | 452.14 | 226.07 | 113.03 | 56.52 | 28.26 | 14.13 | 7.06 | 3.53 |
| s | 3808.67 | 1830.06 | 906.90 | 452.46 | 226.11 | 113.04 | 56.52 | 28.26 | 14.13 | 7.06 | 3.53 |

We construct a GDGG with a fine resolution at coastlines while maintaining a more crude resolution at large homogeneous areas, e.g., oceans. A search on such a graph can find feasible paths near coastlines and passing narrow areas (e.g., water channels and islands), while the graph's size remains minimal. To construct a GDGG with recursion level $L$ and variable recursion level $L_v$, we partition the icosahedron to level $L$, ensuring its vertices lie on a sphere with a radius equal to the Earth's mean radius. After projecting the coastlines defined in the GSHHG database (Wessel & Smith, 1996) on this sphere, we sub-

divide the two adjacent triangle faces of each edge that intersects a coastline until a desired variable recursion level $L_v$ is reached. To ensure only feasible arcs exist, every edge that intersects any polygon defined by the coastlines is removed.



(**a**) $L = 6, L_v = 0$ (**b**) $L = 6, L_v = 3$

**Figure 4.10.** A snippet of a variable density Geodesic Discrete Global Grid.
Recursion level and variable recursion level are denoted by $L$ and $L_v$, respectively. The red box highlights the region near the Strait of Gibraltar.

Figure 4.10 shows two graphs representing the region around the Strait of Gibraltar. For both graphs, we set the recursion level $L$ is to 6, and the variable recursion level $L_v$ for the first and second graphs to 0 and 3, respectively. As a result, a feasible route from the Atlantic Ocean to the Mediterranean Sea passing the Strait of Gibraltar exists in Figure 4.10b, while there does not exist such a route in Figure 4.10a.

Despite setting a high variable recursion level, a water channel (e.g., the Panama Canal) may still exist that is not represented correctly in the graph due to its complex or narrow shape. If this is the case, we create an artificial arc connecting the components representing the water masses that the channel connects, so that it is represented by a single arc. The newly created vertices of this artificial arc are connected to the $k$ nearest neighboring vertices in the graph (a sufficient setting for $k$ is 3). A silver lining to this cumbersome process is that manually defining water channels enable the incorporation of additional passage costs and the possibility of finding alternative routes in the optimization. We will discuss this in Section 4.3.2.

We performed several tests to determine the values of the recursion and variable recursion levels, $L$, and $L_v$, respectively. These tests showed that $L = 4$ and $L_v = 6$ are well-performing values which provide a high resolution at shorelines, such that no isolated components exist. With recursion level $L = 4$, the graph has a coarse resolution at large open seas. Consequently, this reduces the number of waypoints contained in an initial route so that fewer waypoints need to be removed from the initial routes to obtain a good solution.

The GDGG based on an icosahedron partitioning that approximates the Earth's surface, is presented in three dimensions in Figure 4.11. This graph is constructed using the "intermediate" resolution of the GSHHG database, avoiding the Antarctic circle, a recursion level of $L = 3$, and a variable recursion level of $L_v = 4$.

**Figure 4.11.** Variable density Geodesic Discrete Global Grid.
Recursion and variable recursion levels are $L = 3$ and $L_v = 4$, respectively.

### 4.3.2 Initial paths

The initialization procedure applies the A* algorithm to the GDGG to find two paths between the origin and the destination. One minimizing distance, and another minimizing distance with penalties for sailing in high-cost areas. To find a minimal distance route, we set the weight of the arcs equal to their corresponding great circle distance. For the initial route avoiding high-cost areas, we construct a second graph for which we increase the arcs' weight intersecting these areas by a penalty factor $p \in [1, +\infty)$.

The A* algorithm was designed by Hart et al. (1968) to solve for the shortest path between an origin and a destination on a network of nodes and arcs. This algorithm performs a directed search on a graph using a distance heuristic, which calculates the lower bound distance from any node to the destination. In this case, we use the great circle distance as such bound, calculated using the haversine formula, described in Section 3.3. For both paths (minimal distance and minimal cost), the great circle is always a lower bound, as it represents the shortest distance between two points on a sphere. For a description of the A* algorithm for solving the shortest path problem, the reader should refer to (Hart et al., 1968).

In the previous section, we mentioned that we could manually define water channels in the GDGG. By doing this, the initialization procedure can find alternative routes avoiding these connections if the shortest path crosses any. In this study, we include the Panama Canal and the Suez Canal. If the end-user wishes to consider these canals, the initialization procedure first applies the A* algorithm to the complete GDGG, including the Panama Canal and Suez Canal. If the algorithm finds any shortest path crossing $n$ predefined canals, it tries to find $2^n - 1$ alternative paths, such that all other possible permutations of the set of crossed canals are considered. Suppose any shortest path crosses both the Panama Canal

and the Suez Canal. In this case, the procedure tries to find alternative paths on three different graphs, including/excluding a crossed canal, as illustrated in Table 4.3. If an alternative path equals previously calculated shortest or alternative paths, it is discarded.

**Table 4.3.** Alternatives for a shortest path crossing the Panama Canal and the Suez Canal.

| | Panama Canal | Suez Canal |
|---|---|---|
| Shortest path | included | included |
| Alternative path 1 | included | excluded |
| Alternative path 2 | excluded | included |
| Alternative path 3 | excluded | excluded |

Because an alternative path is entirely different from other found paths, an optimization is performed for each path individually. Given the computation time of a route simulation, it is convenient to discard any path that is unlikely to lead to a satisfactory route. The obtained paths give a rough approximation of the geographic path of the final route. This enables comparison of the shortest path with the alternative paths with respect to their distance, which allows discarding an alternative if it is significantly longer than the original. The end-user may set a threshold for the relative distance increase for discarding an alternative path.

An MOEA requires a set of initial solutions of the same size as the population size $N$. The initialization procedure creates two routes from each found global path: one minimizing distance and another minimizing cost. A minimal time route is created by translating the arcs and nodes constituting the minimal distance path to legs and waypoints, respectively.

Furthermore, we set the maximum allowed nominal ship speed at each leg of the minimal time initial route. We create a second initial route with minimal cost based on the minimal cost path and select the most fuel-efficient nominal ship speed $V^*$ for each leg. This is the speed with the least fuel consumption per unit distance, i.e.,

$$V^* = \underset{V}{\operatorname{argmin}} \frac{fc(V)}{V}, \tag{4.21}$$

where $fc(V)$ is the hourly fuel consumption given a nominal ship speed $V$ in knots. We should note that setting different speeds for the initial routes enables a varying speed profile if the routes are recombined in a later stage. Therefore, the initial routes' speed is set to a single constant speed if a constant speed profile is desired in the optimization.

The procedure continues by adding these two routes to an empty set of initial routes. Subsequently, it generates $N - 2$ mutants from the minimal time and cost routes to fill the remaining slots. More specifically, $N/2 - 1$ mutants are created from each of the two routes. A mutant is generated by applying a random sequence of mutations to one of the routes using the mutation operators discussed in Section 4.4.2.

To select the number of mutations, we draw a random integer $k$ such that $k_{min} \leq k \leq k_{max}$. Setting high values for both $k_{min}$ and $k_{max}$ yield a diverse set of initial routes, while

low values allow mutants in the neighborhood of the minimal time and cost routes.

According to prespecified relative weights, the procedure selects a mutation operator and assigns it to the $i$-th position of the mutation sequence, with $i \in [k]$. The relative weights for the *insert*, *move*, and *speed* operators are set to 1 and the *delete* operator is set to 3. Meaning that the procedure selects the delete operator with a 50% probability so that mutants are likely to contain fewer waypoints than the routes found on the graph. Because the initialization procedure takes the great circle distance between two waypoints, a route with few waypoints is generally shorter than a routing containing many.

## 4.4 Operators

In evolutionary algorithms, new individuals are created by recombining parent individuals and possibly mutating the resulting offspring. This population variation is performed through a combination of genetic operators: recombination and mutation. Section 4.4.1 describes the single-point crossover operators used to recombine a pair of "parent" individuals. Section 4.4.2 proceeds with an introduction of mutation operators specialized for the MOSWR problem. Before presenting the genetic operators, we give a general outline of the genetic variation process which contains three stages: (1) selection, (2) recombination, and (3) mutation.

For each MOEA considered in this study, the selection procedure of each is different. NSGA-II and SPEA2 select a pair of parents for crossover with a probability $pb_{cr}$. In contrast, the M-PAES algorithm performs a crossover if and only if the resultant child is accepted according to several criteria. Similarly, NSGA-II and SPEA2 mutate a child solution with a mutation probability $pb_{mut}$, while M-PAES performs mutations in the local search phase only. These characteristic recombination and mutation procedures are described previously in Sections 4.1.1, 4.1.2, and 4.1.3.

When a pair of parents is selected for recombination, a child solution is generated by performing a single-point crossover. This type of crossover operator is described in the following subsection. In Section 4.4.2, we present four specialized mutation moves: insertion, translation, deletion, and speed change. If an individual is selected for mutation, the mutation operator performs $k_{moves}$ moves on the individual, where $k_{moves}$ is a random integer drawn from the interval $\{1, \ldots, n_{moves}\}$. The maximum mutation moves per selected individual $n_{moves}$ is a parameter of the algorithm.

It is worth tuning parameters such as the mutation probability $pb_{mut}$, crossover probability $pb_{cr}$, and the maximum moves $n_{moves}$, to find settings that ensure (fast) convergence to the Pareto front. For example, a very small mutation probability may lead to genetic drift, whereas setting the probability too high may turn the search into a random search. Similarly, a recombination rate that is too high may lead to premature convergence of the genetic algorithm as it decreases the diversity in the population. Section 5.1 describes the tuning of these and other parameters for the MOEAs.

### 4.4.1 Recombination

In evolutionary computation, crossover, or recombination, is a genetic operator used to combine the genetic information of two parents to generate new offspring. It is the main operation to generate new routes from an existing population stochastically so that the population is gradually improved over the generations. The traditional single-point crossover introduced by Holland (1992) swaps the parents' genes, starting at a randomly selected point from the solution vector. Extended versions are the $k$-point crossover, with $k$ a positive integer, and uniform crossover. However, these methods are typically used in binary coded algorithms and less suitable for the MOSWR problem. The latter is because a crossover performs poorly if the crossover points, i.e., waypoints, lie far from each other.

Therefore, we use the single-point crossover to recombine routes in the MOSWR problem. Let us recall that a route leg $r_i$ is defined by a start waypoint $p_i$, an end waypoint $p_{i+1}$, and a constant calm water speed $V_i$. This single-point crossover cuts parent routes in between two route legs and swaps the right parts. For a route with $n$ legs, each waypoint $p_i$, with $2 \leq i \leq n$, is a potential crossover point. To increase the likelihood of well-performing offspring, the crossover operator ensures feasibility and minimizes the distance between two crossover points.

For the recombination of parents $\mathbf{r}_A$ and $\mathbf{r}_B$ containing $n_A$ and $n_B$ waypoints, respectively, the operators starts by selecting any waypoint $p_i^A$ from $\mathbf{r}_A$ as crossover point, with $2 \leq i \leq n_A$. Next, the operator calculates the distance of $p_i^A$ to every other waypoint $p_j^B$ in $\mathbf{r}_B$, with $2 \leq j \leq n_B$. The operator selects the $k$ waypoints in $\mathbf{r}_B$ with smallest distance to consider for crossover with $p_i^A$, where $\kappa = \min(\kappa_{max}, n_B - 2)$ and $\kappa_{max}$ is a parameter of the algorithm. The operator then attempts to iteratively perform a crossover at crossover points $p_i^A$ and one of $k$ selected crossover points from $\mathbf{r}_B$, starting with the nearest points.

If a crossover trial is feasible, the crossover is executed, creating two children, $\mathbf{r}_C$ and $\mathbf{r}_D$, from the parents. First, a crossover exchanges the route legs to the right of $p_i^A$ and $p_j^B$ the two parents. Then, it updates the end waypoints of the preceding route legs $r_{i^-}^A$ [1] and $r_{j^-}^B$ to the start waypoints of route legs $r_i^A$ and $r_j^B$, respectively. This crossover yields two children routes, written as

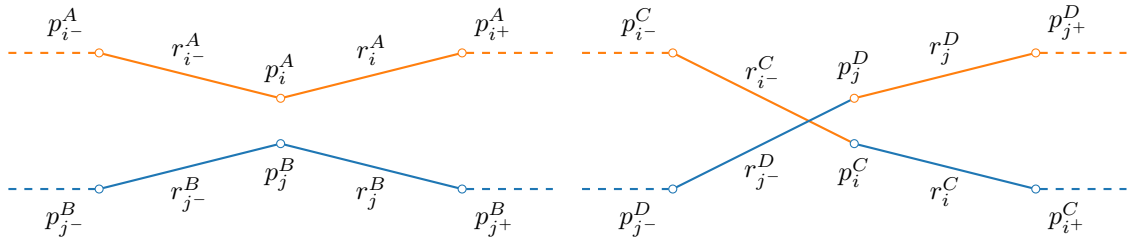
$$\begin{aligned} r_{i^-}^C &= (p_{i^-}^A, p_j^B, V_{i^-}^A), \\ r_{j^-}^D &= (p_{j^-}^B, p_i^A, V_{j^-}^B). \end{aligned} \tag{4.22}$$

Figure 4.12 shows a schematic of this single-point crossover.

If no feasible crossover exists, a new route leg from parent $\mathbf{r}_A$ is considered for crossover. This process is repeated until a crossover is performed or if there does not exist a route leg in $\mathbf{r}_A$ for which a feasible crossover exists.

A crossover is infeasible if at least one of the newly created route legs $r_{i^-}^A$ and $r_{j^-}^B$ at the crossover points is infeasible. A route leg is infeasible if it intersects an impassable area or if

[1] $i^- := i - 1$ and $i^+ := i + 1$




(**a**) Before crossover (**b**) After crossover

**Figure 4.12.** A single-point crossover.
Parent routes $\mathbf{r}_A$ and $\mathbf{r}_B$ create children $\mathbf{r}_C$ and $\mathbf{r}_D$.

its start and end waypoints are the same. The latter may occur if two parents have waypoints with equal location other than the start and destination. Therefore, a convenient setting for the maximum number of inspected nearest waypoints is $\kappa_{max} > 1$, so that neighboring waypoints are considered for a crossover if the the closest waypoint in $\mathbf{r}_B$ to closest waypoint from $\mathbf{r}_B$ to point $p_i^A$ has equal location. We choose $\kappa_{max} = 3$, so that up to three nearest waypoints are considered for crossover.

### 4.4.2 Mutation

Mutation is a genetic operation used to preserve genetic diversity from one generation to the next. It is used in an attempt to avoid local minima by diversifying a population, which prevents slowing or even stopping convergence to the global optimum. A mutation operator changes one or more values in an individual from its initial state and may even change the individual entirely.

Since a route contains real values (i.e., waypoint locations and nominal ship speeds) and has a restrictive sequence of waypoints, we use real-value-based mutations. We present four types of mutation moves specialized for the MOSWR problem: (1) insert, (2) translate, (3) delete, and (4) speed. The first three moves alter a route's path by inserting, translating, or deleting one or more waypoints, while the last operator changes the speed over one or more route legs.

We test two different mutation methods for the insertion and translation operators: uniform and Gaussian mutations. Several mutation methods exist for real value encoded evolutionary algorithms, of which these are suitable for application to the MOSWR problem. The uniform mutation replaces the value from a chosen element of the individual with a uniform random value selected between upper and lower bounds. The Gaussian mutation adds a random value from a Gaussian distribution to the value of an individual's chosen element. In contrast to the uniform mutation, values mutated by the Gaussian variant are more biased to the mean of the replaced value and may take more extreme values at the tails of the distribution.

*Insert*

The insert operator alters a route's path by inserting one or more waypoints between randomly selected legs. It samples the number of route legs to be selected using the exponential distribution. That is, the operator samples a variable $x_{\text{exp}} \sim \exp(\frac{1}{\beta})$, where $\beta$ is the scale parameter.

We choose $\beta = 10/n$, so that the probability of selecting a single route leg for insertion is high and decreases exponentially with a linear increase in route legs. The probability density function is then expressed by

$$f(x;\beta) = \begin{cases} \frac{1}{\beta} e^{-x/\beta} & x \geq 0, \\ 0 & x < 0. \end{cases} \tag{4.23}$$

Next, the speed operator rounds $x_{\text{exp}}$ towards the nearest integer $k$. If $k$ is not in the range $1, 2, \ldots, n$, the operator samples a new $x_{\text{exp}}$ from $\exp(\frac{1}{\beta})$. If $k$ lies within the desired range, the operator selects $k$ legs from route $\mathbf{r}$ to be considered for insertion, without resampling. By selecting route legs weighted according to their lengths, the inserted waypoints are more or less uniformly distributed along the routes.

Suppose the insert operator inserts a single waypoint $p'$ between the start and end waypoint of a randomly selected leg $r_i$, with $2 \leq i \leq n$, of a route $\mathbf{r}$ containing $n$ legs. It splits a chosen leg into two legs, such that the first leg starts at $p_i$ and ends at $p'$ and the other leg starts and ends at $p'$ and $p_{i^+}$, respectively. Hence, the resulting mutant $\mathbf{r}'$ contains $n+1$ route legs. Finally, the nominal ship speed at each of the two newly created legs is set to the original leg's speed $r_i$. More formally, a leg of mutant route $\mathbf{r}'$ obtained by mutating $\mathbf{r}$ is defined as

$$r'_j = \begin{cases} (p_j, p_{j^+}, V_j) & \forall j = 1, \ldots, i-1, \\ (p_j, p', V_j) & \text{for } j = i, \\ (p', p_j, V_{j^-}) & \text{for } j = i+1, \\ (p_{j^-}, p_j, V_{j^-}) & \forall j = i+2, \ldots, n+1. \end{cases} \tag{4.24}$$

Figure 4.13 shows a schematic of an insertion.


(**a**) Before insertion (**b**) After insertion ($j = i$)

**Figure 4.13.** An insert mutation of route $\mathbf{r}$, creating mutant $\mathbf{r}'$.

Two values define the location of a waypoint, i.e., the latitude $\lambda$ and longitude $\varphi$. We set the location of a newly inserted waypoint using either Gaussian or uniform mutation. The uniform mutation uses a prespecified bounded region from which it selects a uniformly random point. For simplicity, we assume that waypoint locations are projected onto a two-dimensional Euclidean space, i.e., we define $(x, y) \coloneqq (\varphi, \lambda)$.

We test two different shapes that define this region: (1) the rhombus and (2) the ellipse. Both shapes have a major and minor axis with lengths of $2a$ and $2b$, respectively. We choose the endpoints of the major axis such that they intersect the start and end waypoints $p_i$ and $p_{i^+}$ of the selected route leg for insertion. Hence, the length between the points $l_i$ is equal to the major axis length $2a$ The length of the minor axis $2b$ of both shapes is obtained from the relation

$$b = c_{\text{insert}} * a, \tag{4.25}$$

where $c_{\text{insert}} \in \mathbb{R}^+$ is a user-defined scale factor.

*Sampling within an rhombus*

To sample uniformly distributed points inside a rhombus that lies between two waypoints, we first define an $x$-axis-aligned rhombus $\rho$ centered at the origin, with major and minor axis lengths $2a$ and $2b$, respectively. We sample a uniformly distributed point $\mathbf{x}_\rho$ inside $\rho$ using

$$\mathbf{x}_\rho = U_1\mathbf{v}_1 + U_2\mathbf{v}_2 - \begin{bmatrix} a \\ 0 \end{bmatrix}, \tag{4.26}$$

where $\mathbf{v}_1 = \begin{bmatrix} a \\ b \end{bmatrix}$, the top vertex intersecting the minor axis, $\mathbf{v}_2 = \begin{bmatrix} a \\ -b \end{bmatrix}$, the bottom vertex intersecting the minor axis. In addition, $U_1$ and $U_2$ are two independent uniform random variables, i.e., $U_1 \sim \mathcal{U}(0,1)$ and $U_2 \sim \mathcal{U}(0,1)$.

Next, we project $\rho \rightarrow \rho'$, such that the endpoints of the major axis of $\rho'$ intersect with waypoints $p_i$ and $p_{i^+}$. For this, we compute the rotation matrix $R$, given by

$$R = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{bmatrix}, \tag{4.27}$$

where $\theta_i$ is the angle in radians between the positive $x$-axis and the line through points $p_i$ and $p_{i^+}$. The angle $\theta_i$ is given by

$$\theta_i = \arctan2(y_{i^+} - y_i, x_{i^+} - x_i). \tag{4.28}$$

Finally, using Equations 4.26 to 4.28 we express a uniformly distributed point $\mathbf{x}_{\rho'}$ inside rhombus $\rho'$ by the equation

$$\mathbf{x}_{\rho'} = R\mathbf{x}_\rho + \mathbf{m}, \tag{4.29}$$

where $\mathbf{m} = \frac{1}{2}\begin{bmatrix} x_i + x_{i^+} \\ y_i + y_{i^+} \end{bmatrix}$, the center of $p_i$ and $p_{i^+}$.

*Sampling within an ellipse*

For the case of sampling a uniformly distributed point within an ellipse, we propose a similar approach. First, we generate a uniformly random point $\mathbf{x}_\epsilon$ in an ellipse $\epsilon$ centered at the origin and aligned with the $x$-axis. Its major and minor axes have lengths $2a$ and $2b$,

respectively. Then, $\mathbf{x}_\epsilon$ is expressed by

$$\mathbf{x}_\epsilon = \sqrt{U_1} \begin{bmatrix} \cos U_2 \\ \sin U_2 \end{bmatrix} \odot \begin{bmatrix} \cos a \\ \sin b \end{bmatrix}, \tag{4.30}$$

where $\odot$ denotes an element-wise product and $U_1$ and $U_2$ are i.i.d. variables from $\mathcal{U}(0, 1)$. Finally, we transform $\epsilon \to \epsilon'$ using Equations 4.28 to 4.30, so that we get random point $\mathbf{x}_{\epsilon'}$ inside the ellipse $\epsilon'$, such that

$$\mathbf{x}_{\epsilon'} = R\mathbf{x}_\epsilon + \mathbf{m}, \tag{4.31}$$

where $\mathbf{m}$ is the center of $p_i$ and $p_{i+}$.

Figure 4.14 shows both shapes for $c_{\text{insert}} = 0.5$, and a sample of 1000 uniformly generated points. It also shows that points sampled in a rhombus are more biased towards the center of a route leg than the points drawn inside an ellipse with equal width.

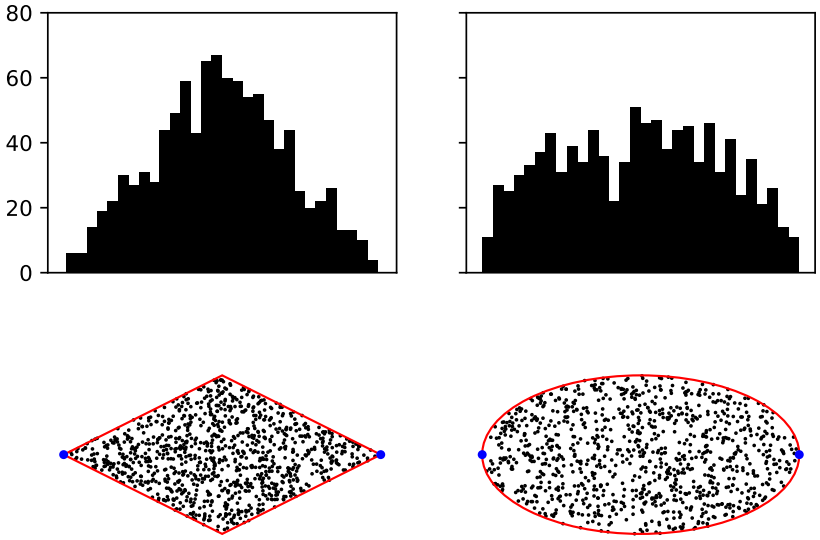



**Figure 4.14.** Distribution of 1000 uniformly sampled points within two shapes.
The rhombus (left) and an ellipse (right) are defined by two blue waypoints, depicting the major axis, and the width ratio $wr = 0.5$, such that minor axis' length is half the major axis's length.

The Gaussian mutation generates a new waypoint from a bivariate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. It then transforms the sampled point, such that the point is drawn within the ellipse $\epsilon'$ with 95% confidence. To do this, we first choose a $\boldsymbol{\mu} = [0, 0]$ and $\boldsymbol{\Sigma} = \sigma^2\mathbf{I}$. Consequently, the confidence interval (CI) is represented by a circle centered at the origin. The problem is to find $\sigma^2$, such that the 95% CI is represented by the unit circle. Then, we transform the distribution such that the 95% CI is projected on the ellipse $\epsilon'$, located between the waypoints $p_i$ and $p_{i+}$.

In general, the equation of an $x$-axis aligned ellipse with a major axis of length $2a$ and a

minor axis of length $2b$, centered at the origin, is defined by

$$\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 = 1. \tag{4.32}$$

We choose the length of the axes as the standard deviation, i.e., $a = b = \sigma$, so that the equation of the confidence ellipse becomes

$$\left(\frac{x}{\sigma}\right)^2 + \left(\frac{y}{\sigma}\right)^2 = s, \tag{4.33}$$

where $s$ defines the scale of the ellipse. Solving Equation 4.33 for $\sigma^2$, such that it lies on the unit circle gives

$$\frac{1}{s\sigma^2}(x^2 + y^2) = (x^2 + y^2), \tag{4.34}$$

yielding a variance $\sigma^2$ of $s^{-1}$.

As the Gaussian mutation samples a point according to the bivariate Gaussian with zero covariance, both $x$ and $y$ are normally distributed too. Therefore, the left-hand side of Equation 4.33 represents the sum of squares of independent normally distributed points. The sum of squared Gaussian points is distributed according to a chi-square distribution with two degrees of freedom $\chi^2_2$. Using the probability table of $\chi^2_2$, we find

$$P(s < 5.991) = 1 - 0.05 = 0.95. \tag{4.35}$$

Thus, we choose $s = 5.991$ to obtain a 95% CI represented by a unit circle. This yields a variance $\sigma^2$ of $\frac{1}{5.991} \approx 0.1669$.

Next, we transform the sampled point $\mathbf{x}_\nu$ such that the 95% CI is projected on ellipse $\epsilon'$. Using Equations 4.27 and 4.28, we express the transformed random point by

$$\mathbf{x}_{\nu'} = R\Big(\mathbf{x}_\nu \odot \begin{bmatrix} a \\ b \end{bmatrix}\Big) + \mathbf{m}. \tag{4.36}$$

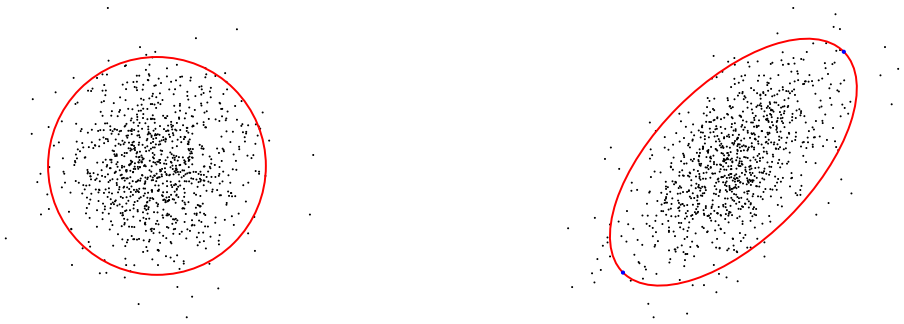
Figure 4.15 illustrates 1000 points uniformly distributed according to $\mathcal{N}(\mathbf{0}, 0.1669\mathbf{I})$ and its 95% CI. It also shows the same points after transformation, such that 95% CI is an ellipse with major axis endpoints at waypoints $p_i$ and $p_{i+}$ and minor axis length $2b = 0.5 * 2a = a$ (i.e., the minor axis scale factor $c_{\text{insert}}$ is 0.5).

To ensure that a new waypoint $p'$ is located on the Earth's surface, i.e.,

$$p' = \{(\lambda', \varphi') \colon -90 \leq \lambda' \leq 90, -180 \leq \varphi' \leq 180\},$$

we define the location of $p'$ by the following

$$p' = \begin{bmatrix} \lambda' \\ \varphi' \end{bmatrix} = \begin{bmatrix} \min\left(\max(y, -90), 90\right) \\ x - (\lfloor x/180 \rfloor \mod 360) \end{bmatrix}, \tag{4.37}$$



(**a**) $\mathcal{N}(\mathbf{0}, 0.1669\mathbf{I})$ (**b**) Transformed

**Figure 4.15.** 1000 random sampled points and the 95% confidence interval (CI).
The set of points is transformed (right), such that the CI represents an ellipse with major axis endpoints intersecting the waypoints in blue, and minor axis scale factor $c_{\text{insert}} = 0.5$.

where $x$ and $y$ are coordinates in two-dimensional Euclidean space defined by the random variable $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$, which we obtain from one of the three sampling methods described above.

*Translation*

The translation operator changes a route's path by moving $k$ waypoints, where $k$ is a random variable $x_{\text{exp}}$ rounded to the nearest integer, with $x_{\text{exp}} \sim \exp(\frac{1}{\beta})$. A route consists of $n+1$ waypoints; therefore, at most $n-1$ waypoints can be translated, as the origin and destination are fixed. Thus, we choose the value for the scale parameter $\beta = \frac{10}{(n-2)}$.

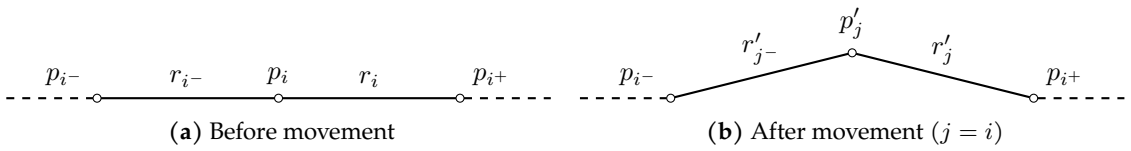
Routes with sharp angles are difficult to realize in practices, as the turning radius of a ship is limited. To obtain a smooth ship trajectory, the translation operator selects $k$ waypoints weighted according to their angle, with $1 \leq k \leq n-1$. The angle of a waypoint is the difference in ship course in adjacent route legs. The optimization algorithm already calculates and memorizes the course between consecutive waypoints; thus, the calculation of waypoint angles has a negligible effect on the computational performance.

Suppose, the translation operator chooses to translate a single waypoint $p_i$ in route $\mathbf{r}$, with $2 \leq i \leq n$. Moving waypoint $p_i \to p'_j$ requires updating its corresponding route legs $r_{i^-}$ and $r_i$. A route leg of mutant $\mathbf{r}'$ obtained by applying the translation operator to $\mathbf{r}$ is defined as

$$r'_j = \begin{cases} (p_j, p_{j^+}, V_j) & \forall j = 1, \ldots, i-2, \\ (p_j, p'_j, V_j) & \text{for } j = i-1, \\ (p'_j, p_{j^+}, V_j) & \text{for } j = i, \\ (p_j, p_{j^+}, V_j) & \forall j = i+1, \ldots, n. \end{cases} \tag{4.38}$$

Figure 4.16 shows a schematic of a translation $p_i \to p'_i$.

The translation operator uses a uniform or Gaussian mutation to pick the location of $p'_i$. For the uniform mutation, this location is sampled uniformly within a circle $c$ with




(**a**) Before movement (**b**) After movement ($j = i$)

**Figure 4.16.** A translation operator of route **r**, creating mutant **r**′.

radius $r_{\text{move}}$ centered at the location of $p_i$. The Gaussian variant picks a location according to a bivariate normal distribution, which has a 95% CI projected on the circle $c$. Updating the location of $p_i$ follows a similar approach as the insert operator's approach. Like the insert operator, the translation operator ensures that a new waypoint lies within the region defined by the Earth's surface using Equation 4.37.

### *Delete*

Next, we propose a delete operator that removes one or more waypoints from a route. The operator randomly selects $k$ waypoints to be deleted from a route, without resampling, with $1 \leq k \leq n-1$. Similar to the translation operator, $k$ is obtained by rounding the random variable $x_{\text{exp}}$ sampled from $\exp(3, 10/(n-1))$.

Suppose the delete operator chooses to delete a single waypoint $p_i$ from a route **r**, with $2 \leq i \leq n$. After removing $p_i$, its adjacent route legs $r_{i^-}$ and $r_i$ merge into one new route leg $r'_{j^-}$. In more formal writing, we define the route legs of mutant **r**′ obtained by mutating **r** as

$$r'_j = \begin{cases} (p_j, p_{j+1}, V_j) & \forall j = 1, \ldots, i-1, \\ (p_j, p_{j+2}, V_j) & \text{for } j = i, \\ (p_{j+1}, p_{j+2}, V_{j+1}) & \forall j = i+1, \ldots, n-1. \end{cases} \tag{4.39}$$

A schematic of this deletion operator is given by Figure 4.17.


(**a**) Before deletion (**b**) After deletion ($j = i$)

**Figure 4.17.** A delete mutation of route **r**, creating mutant **r**′.

### *Speed*

The speed operator is the only operator capable of changing the boat speed. It selects one or more route legs from **r** to change the corresponding boat speeds. That is, it selects the set of route legs $R = r_i : i = U_2, \ldots, U_2 + U_1 - 1$, where $U_1 \sim \mathcal{U}\{1, n\}$ and $U_2 \sim \mathcal{U}\{1, n-k\}$. In other words, $U_2$ consecutive route legs are selected for mutation starting from the $U_1$-th route leg, such that both variables are independently distributed discrete uniform variables. Next, a boat speed $V'$ is randomly selected from the set of discrete boat speeds $\mathbf{V}$. We ensure that the speed operator changes the boat speed of a leg $r_i$ in $R$, by testing whether the average boat speed over all legs in $R$, is equal to $\mathbf{V}$. If this holds, the operator repeats

the steps described above to select a new boat speed $V'$. Otherwise, a mutant is created by updating the boat speed of each leg $r_i$ in $R$ to $V'$.

*Gaussian versus uniform mutation*

This section earlier presented the concept of Gaussian and uniform mutation and applied it to the move operators 'Insert' and 'Translate'. We used a parameter tuning method which performed multiple test to assess the effect on the algorithm's convergence for each mutation method.

The uniform mutation showed to be the better method for finding successful waypoints insertions and waypoint translations. This has likely two causes:

1. Compared to the uniform mutation, the Gaussian variant has a smaller bias from the mean, i.e., the center of the route leg for the 'Insert' move and in the case of 'Translate', the location of the waypoint selected for translation. As a result, the Gaussian method more frequently inserts a near the center of a route leg and translates a waypoint near its original location.

2. We can increase the bias can be increased by increasing the variance or enlarging both the 'width ratio' (for 'Insert') and the 'radius' (for 'Translate'). However, a side-effect is that the number of outliers increase, resulting in too aggressive mutations which are may be infeasible or result in a worse solution value.

The uniform mutation method does not have these drawbacks and is more comprehensible than the other method. For these reasons, we choose to use the uniform mutation method in the remainder of this study.

## 4.5 Line segment intersection

Section 3.2 defines the sailing region as a closed set of longitude and latitude pairs. Within the sailing region, we defined impassable and high-cost areas. To test whether a route leg intersects with areas, e.g., impassable or high-cost areas, we introduce a fast method for finding intersection points of a line segment with a polygon.

First, Section 4.5.1 adopts a method for a line segment intersection test, presented by Antonio (1992), and rewrites it to the generalized case of an intersection test of a line segment and polygon in two-dimensional space. We apply the intersection test of straight line segments to the problem of determining whether an intersection exists of a route leg and polygon in the sailing region. The proposed ship routing algorithm performs this test thousands of times to evaluate ship routes. Section 4.5.2 describes a spatial indexing method to reduced the number of calculations. We propose a modification to this method in Section 4.5.3, to improves its computational performance. Finally, Section 4.5.4 concludes with computational results from a test case of line segment intersections.

### 4.5.1 Line intersection

The problem is to determine whether an intersection exists for a given line segment and polygon in two-dimensional space. Suppose we have a straight line segment $l_{12}$ defined by two endpoints $p_1$ and $p_2$ and a polygon $\omega$ defined by $|\omega|$ edges. If there exists an intersection of line segment $l$ with any edge in polygon $\omega$, then $l_{12}$ intersects $\omega$. Therefore, we have to determine for each edge in polygon $\omega$, whether it intersects with line segment $l_{12}$. The following describes a fast line segment intersection test presented by Antonio (1992).

Let us consider an edge in polygon $\omega$ as a straight line segment, denoted by $l_{34}$, which is defined by the vertices $p_3$ and $p_4$. Figure 4.18 shows line segments $l_{12}$ and $l_{34}$, and their intersection point $p^*$.

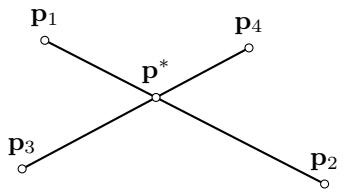



**Figure 4.18.** Two intersecting line segments $l_{12}$ and $l_{34}$.
Line segments $l_{12}$ and $l_{34}$ defined by endpoints $\mathbf{p}_1$, $\mathbf{p}_2$ and $\mathbf{p}_3$, $\mathbf{p}_4$, respectively, intersect at point $\mathbf{p}^*$.

For convenience, we represent each point as a vector, i.e., $\mathbf{p}_1 = (x_1, y_1)$, etc. Point $p$ anywhere on the line segment $l_{12}$ can be represented parametrically by a linear combination of $\mathbf{p}_1$ and $\mathbf{p}_2$

$$\mathbf{p} = \alpha\mathbf{p}_1 + (1-\alpha)\mathbf{p}_2, \tag{4.40}$$

$$\mathbf{p} = \mathbf{p}_1 + \alpha(\mathbf{p}_2 - \mathbf{p}_1), \tag{4.41}$$

where $\alpha$ is in the interval $[0, 1]$.

With the linear equations for $l_{12}$ and $l_{34}$ we locate the intersection point $\mathbf{p}^*$ by solving the following linear system of equations for $\alpha$ and $\beta$.

$$\mathbf{p}^* = \mathbf{p}_1 + \alpha(\mathbf{p}_2 - \mathbf{p}_1), \tag{4.42}$$

$$\mathbf{p}^* = \mathbf{p}_3 + \beta(\mathbf{p}_4 - \mathbf{p}_3). \tag{4.43}$$

Subtracting the equations yields

$$0 = (\mathbf{p}_1 - \mathbf{p}_3) + \alpha(\mathbf{p}_2 - \mathbf{p}_1) + \beta(\mathbf{p}_3 - \mathbf{p}_4). \tag{4.44}$$

For readability, we write the intermediate values as

$$\mathbf{A} = \mathbf{p}_2 - \mathbf{p}_1,$$

$$\mathbf{B} = \mathbf{p}_3 - \mathbf{p}_4,$$

$$\mathbf{C} = \mathbf{p}_1 - \mathbf{p}_3.$$

Then, the solutions for Equation 4.44 are expressed by the equations

$$a = \frac{ByCx - BxCy}{AyBx - AxBy}, \tag{4.45}$$

$$b = \frac{AxCy - AyCx}{AyBx - AxBy}, \tag{4.46}$$

where $a, b \in \mathbb{R}$. If the resulting values of $a$ and $b$ are both in the interval $[0, 1]$, the line segments $l_{12}$ and $l_{34}$ intersect.

Noting that the denominators of Equations 4.45 and 4.46 are the same; evaluating both expressions requires at most nine additions and six multiplications. Since $a$ and $b$ only need to be tested for the interval $[0, 1]$, the actual values of $a$ and $b$ are not required. Therefore, we avoid the division operators in the above equations by applying a division-avoiding test, given in Algorithm 1 (Antonio, 1992).

**Algorithm 1** Division-avoidance test

```
if denominator > 0 then
    if numerator < 0 or numerator > denominator then
        segments do not intersect
    else if numerator > 0 or numerator < denominator then
        segments do not intersect
```

At this point, we show how the above line intersection test can be applied to the MOSWR problem to determine whether route legs intersect certain areas within the sailing region. A route $\mathbf{r}$ contains $n$ route legs, each route leg's geodesic is defined by the great circle track connecting the endpoints of the route leg $r_i$, $i \in [n]$. Each area in the sailing region is represented by a polygon defined by a series of vertices connected by straight edges. The problem is to test whether there exists an intersection of at least one route leg $r_i \in \mathbf{r}$ with a polygon $\omega$. For this, we need to test whether the geodesic of $r_i$ intersects any edge of $\omega$.

Suppose we have a polygon $\omega$ with $|\omega|$ straight edges and a route leg $r$ with endpoints $p_1$ and $p_2$. Then, let us split the geodesic of $r$ into $n_{\text{seg}}$ smaller segments, such that each segment has equal geodesic length $g$. Also, each segment $s_{j,j+1}$, $j \in [n_{\text{seg}}]$, connects endpoints $q_{j-1}$ and $q_j$, with $p_1 = q_1$ and $p_2 = q_{n_{\text{seg}}+1}$. By choosing an infinitely small value for $g$, yielding an infinitely large integer for $n_{\text{seg}}$, we can approximate line segment $s_{j,j+1}$ with a straight line segment $l_{j,j+1}$ connecting the same endpoints by Theorem 1.

Now that we have a set of straight line segments associated with the geodesic of route leg $r$, we can apply the previously described method to the problem of finding an intersection of leg $r$ with polygon $\omega$. Hence, for each straight line segment $l_{j,j+1}$, $j \in [n_{\text{seg}}]$, we test whether it intersects at least one edge in $\omega$. If such an intersection exists, route leg $r$ intersects polygon $\omega$.

In the worst case scenario, the number of line segment intersection calculations required

to test whether route $r$ intersects with a polygon $\omega \in \Omega$ is

$$\prod_{i}^{n} n_{\text{seg}}^{i} n_{\omega}, \tag{4.47}$$

where $n_\omega$ is the number of edges in polygon $\omega$ and $n_{\text{seg}}^{i}$ denotes the number of line segments used to approximate the geodesic of route leg $r_i$, $i \in [n]$.

Despite that the line segment intersection test of Antonio is fast, the test becomes computationally expensive for the MOSWR problem. Generally, a huge amount of calculations is required for a single route $\mathbf{r}$, considering many line segments in $\mathbf{r}$ tested for a large set of polygons. With the introduction of R-tree spatial indexing, we present an improvement of the computational efficiency of intersection tests applied to the MOSWR problem.

### 4.5.2 R-tree spatial indexing

R-tree indexing techniques are common for spatial data management and is initially proposed by Guttman (1984). An R-tree is a hierarchical data structure designed to execute intersection queries efficiently. It stores a collection of rectangles that can change over time through insertions and deletions. First, we provide an overview of the R-tree data structure. Secondly, we introduce the packing algorithm used in this study to construct an R-tree. Finally, we improve the computational performance querying a geometric region in an R-tree by subdividing the input polygons.

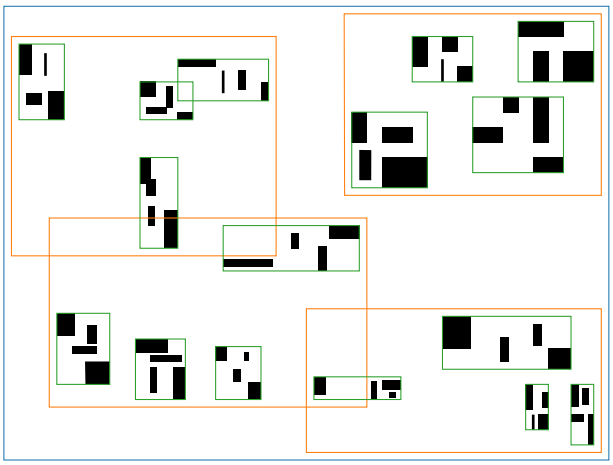


**Figure 4.19.** A sample R-tree with three levels.
Input geometric objects are represented by their minimum bounding rectangle in solid black. The leaf nodes enclosing four nearby rectangles are shown in green. The nodes in the next higher level and the root node are drawn orange and blue, respectively.

Figure 4.19 illustrates a sample R-tree with three levels and a node capacity of four rectangles. Arbitrary geometric objects, such as polygons, are stored in the tree by representing each object by its minimum bounding rectangle (MBR), i.e., the smallest upright rectangle enclosing an object. The R-tree then groups nearby objects and represents them with their MBR in the next higher level, i.e., the leaf level. It aggregates rectangles at yet higher levels and represents them by their MBRs, iteratively, until each MBR is nested into one node at

the root level.

When performing a search, the R-tree takes a query region $Q$. All rectangles that intersect $Q$ must be retrieved and examined. This retrieval is accomplished using a recursive procedure that starts at the root level and may follow several paths down the tree. A node is processed by inspecting each MBR that intersects with $Q$. If the node is not at the leaf level, the subtrees corresponding to the retrieved MBRs are searched recursively. Otherwise, the node is a leaf node, and the geometric objects of the MBRs intersecting $Q$ are returned.

A so-called packing algorithm constructs an R-tree from geometric objects. The basic idea is that it "tiles" the data space so that $r$ rectangles are ordered in $\lceil r/m \rceil$ groups of $m$ rectangles, where each group is to be placed in the same leaf level node. The algorithm then loads the $\lceil r/m \rceil$ groups into MBRs and recursively packs these bounding boxes into next-level nodes, until the root node is created.

The packing algorithm we use in this study is the Sort-Tile-Recursive (STR) algorithm from Leutenegger et al. (1997) with a node capacity of $m = 10$. Considering a data space ordered along two dimensions, the STR uses $\sqrt{r/m}$ slices to split the data space into vertical slices. Each slice contains enough rectangles to pack approximately $\sqrt{r/m}$ nodes. The algorithm packs each slice into nodes containing a maximum of $m$ rectangles. As a result, the two-dimensional data space contains $\lceil r/m \rceil$ nodes representing the MBR of the objects it encloses. The same algorithm packs the resulting nodes into subtrees until $r \leq m$, i.e., the resulting rectangles fit into a single node.

### 4.5.3 Polygon subdivision

If the number of geometric objects is large, R-trees usually provide performance advantages over intersection testing for all objects. For example, in this study, the data representing the land obstacles include 32,832 polygons, some of which are large polygons. Performing an intersection test for each edge of each polygon is a computationally expensive task. With R-trees, the number of these tests can be greatly reduced.

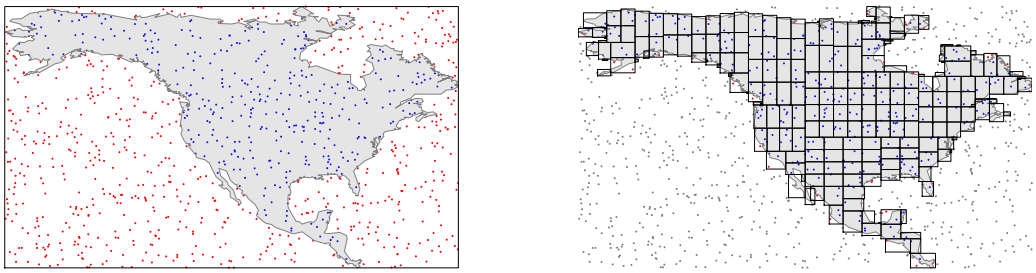
However, R-trees may provide little performance improvement if the queried region, e.g., the MBR of a line segment, is significantly smaller than the MBR of a complex polygon stored in the tree. In general, the leaf node storing a complex polygon consisting of many edges has large coverage and contains considerable "dead space", i.e., empty area. The coverage is the entire area that covers all related rectangles. Minimal coverage reduces the amount of dead space covered by the nodes of the R-tree. Correspondingly, searching a leaf node containing one or more complex polygons is less likely to return an intersected polygon than searching a leaf node containing one or more simple polygons, i.e., polygons with few edges. Hence, subdividing a complex polygon into smaller simple polygons improves search efficiency.

To illustrate this, we generate 1000 random points uniformly distributed within the MBR of a large complex polygon, as shown in Figure 4.20. We want to test which of these points

are contained by the actual polygon. All points in Figure 4.20a lie within the MBR of the polygon. So every point has to be tested against the actual polygon, even if we use an R-tree. 64.6 % of the covered area of the MBR is dead space; thus, approximately 646 points lie outside the polygon.

To reduce the number of expensive tests, we divided the polygon into smaller polygons using Algorithm 3, in Appendix E. This algorithm recursively divides a polygon into two polygons until the MBR of each polygon has a desired size $A_{max}$, measured in squared degrees of longitude and latitude.

Subdividing the same polygon, using $A_{max} = 5$, we obtain 328 smaller polygons and their MBRs, as shown in Figure 4.20b. 11.2 % of the combined areas for all MBRs is dead space. As a result, considering the same set of randomly generated points, only 4.1 % lies within an MBR of a subdivided polygon but is not contained by the polygon. Hence, the number of expensive calculations is reduced by 52 % by dividing this polygon into 328 smaller polygons.



(**a**) A large, complex polygon and its MBR (**b**) MBRs after subdivision

**Figure 4.20.** Illustration of the potential of subdividing a large, complex polygon. 1000 points are uniformly random sampled within a complex polygon's minimum bounding rectangle (MBR). Points that intersect the polygon are marked blue. Every other point is marked red if it intersects any MBR, and marked gray, otherwise. Using an R-tree, an expensive intersection test is required for red and blue points only.

A subdivided (child) polygon usually consists of fewer edges than its parent. As a result, the average number of edge intersections per inspected polygon reduces with the subdivision of large complex polygons. Hence, this subdivision may further improve computational performance.

To illustrate this, suppose we subdivide a parent polygon $\omega$ into a set of smaller polygons $\Omega_c$, and we have a query region $Q$ representing the MBR of a query object. Let us make the following two assumptions:

1. A child polygon $\omega_c \in \Omega_c$ contains fewer edges than its parent polygon $\omega$, i.e., $|\omega_c| < |\omega|$.

2. The query region $Q$ intersects polygon $\omega_c \in \Omega_c$ exclusively.

Following the first assumption, an intersection test of a line segment with $\omega_c$ requires fewer intersection calculations than for the same test with $\omega$. In general, this assumption holds

for large complex polygons.

The second assumption implies that testing whether $Q$ intersects $\omega$ boils down to testing whether $Q$ and $\omega_c$ intersect. This assumption is reasonable if the size of $Q$ is significantly smaller than the MBR size of each polygon in $\Omega_c$. Such a query region may be the MBR of a line segment representing a segment of a route leg, which we choose very small. Following the assumptions mentioned above, the required computations decrease with rate $|\omega_c|/|\omega|$ (i.e., the ratio between the number of edges in $\omega_c$ and $\omega$).

### 4.5.4 Computational performance

To demonstrate the query performance of R-trees, we perform many line intersection tests on a large set of polygons. We use the 'intermediate' resolution data from the GSHHG database (Wessel & Smith, 1996) which include 32,832 polygons, containing a total of 340,369 edges. Uniformly distributed and orientated in the same space region, we generate 1000 random line segments, as shown in Figure 4.23a.

We test the performance of three approaches to determine whether there exists an intersection of a line segment with any polygon in a set of polygons.

1. The "naive approach" inspects every polygon and its edges until it finds such an intersection, using the method described in Section 4.5.1.

2. The second approach, "R-tree", stores each polygon in an R-tree, taking a query for a line segment. It returns the polygons whose MBR intersects the query region. Next, for each edge constituting the polygons corresponding to these MBRs, it performs the same intersection test used in the naive approach.

3. The "R-tree$^+$" approach is similar to R-tree but differentiates itself by subdividing large polygons into smaller polygons with a specified maximum size $A_{max}$ before packing the tree.

The threshold $A_{max}$ introduces a trade-off in computation time caused by line segment intersection tests and R-tree queries. For a high value for $A_{max}$, the number of polygons is relatively small, but large complex polygons exist. This results in many line segment intersections and a smaller R-tree, whereas a small value yields a large number of polygons with a small average size. Which, in turn, results in a large R-tree and a small amount of line segment intersection tests. We tested a range of threshold values on this set of polygons and found that a setting of $A_{max} = 3^2$ provides good computational performance, as shown in Figure 4.21. With this threshold value, each polygon has an MBR size of at most $3^2$ squared degrees, measured in longitude and latitude coordinates.

Figure 4.22 shows the distribution of the number of edges and the size per polygon, before and after subdividing large polygons. The majority is smaller than the threshold value; hence, these polygons are not subdivided. Sixteen large polygons are split into 2635 smaller polygons, causing an 8.0 % increase in the total number of polygons. As shown

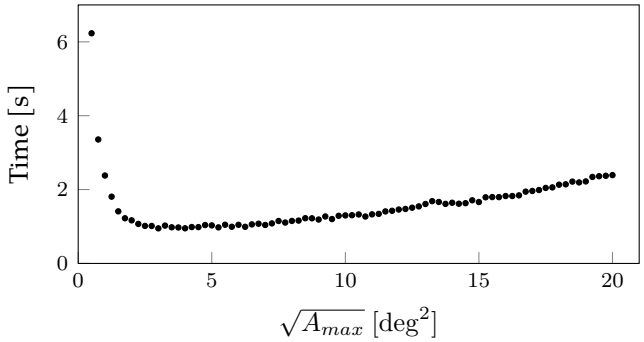



**Figure 4.21.** The computation time of 1000 R-tree queries for $\sqrt{A_{max}} \in \{0.5, 0.75, ..., 20\}$. Each query corresponds to a random sampled line segments performed on a set of polygons. The set of polygons are preprocessed such that each polygon is split until its largest dimension is smaller than $A_{max}$.

in the figure, these large polygons contain the largest number of edges, and most child polygons have a size just below the threshold value.

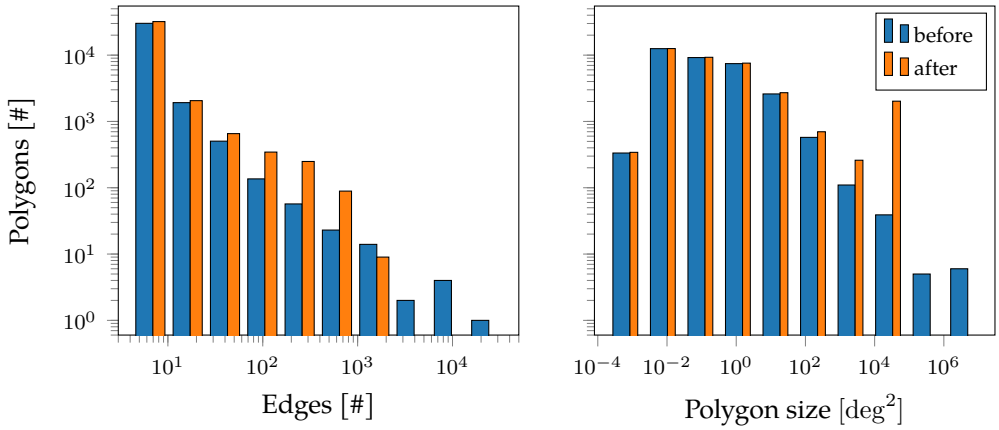



**Figure 4.22.** Distribution of polygon properties, before and after subdivision. For a maximum polygon size of $A_{max} = 25$ in squared longitude and latitude degrees.

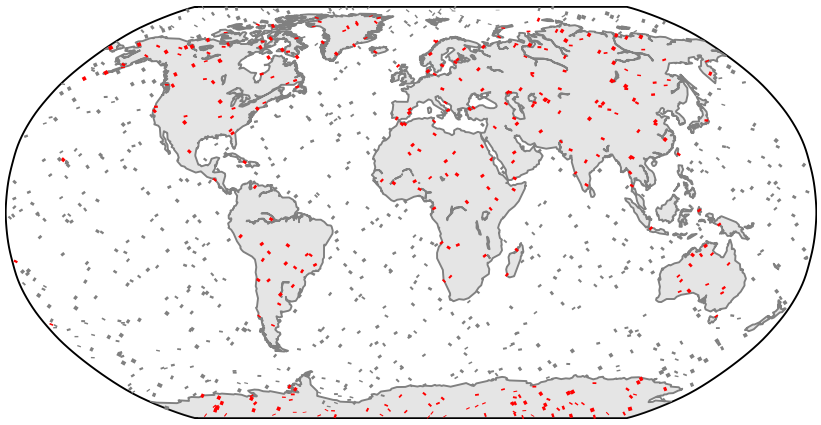
Figure 4.23b shows results on each approach's computational performance, measured in terms of the number of inspected polygons, the number of edges per inspected polygon, and computation time. A polygon is inspected if the queried line segment is tested for intersection with the polygon. Thus, in the case of an R-tree approach, a polygon is inspected if and only if the queried line segment intersects the polygon's MBR.

The naive approach performs roughly 26,000 (35,000) times more polygon inspections than the R-tree (R-tree$^+$) approach. This significant difference is because this approach inspects every polygon for each line segment, whereas an R-tree query only inspects polygons whose MBR intersects the query region. As a result, the R-tree approach's computation time is approximately 12 times shorter, and 590 times for R-tree$^+$.

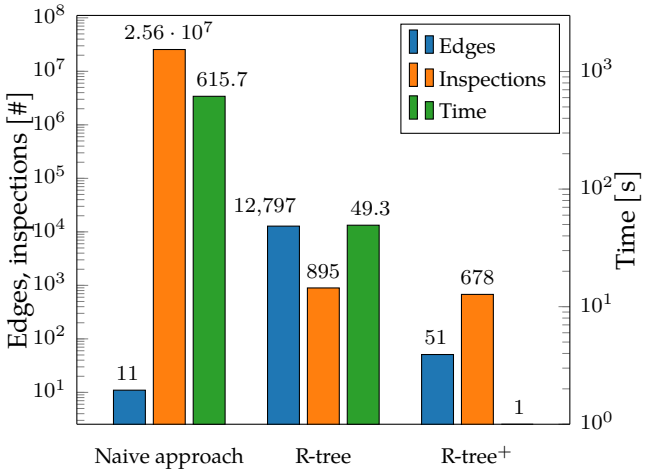
Although the decrease in computation time of the R-tree approach compared to the naive approach is significant, it is not proportional with respect to the decrease in the number of tests. This is explained by the significantly higher average number of edges per inspected polygon in the R-tree approach. That is, the R-tree approach performs, on average, 1163

times more expensive line intersection tests per inspected polygon compared to the naive approach.

On average, an inspected polygon contains many edges because a queried line segment is likely to intersect the MBR of a large complex polygon, as these MBRs cover a large area. The R-tree$^+$ decreases a large complex polygon's number of edges by subdivision. This leads to a 49 times shorter computation time compared to the standard R-tree approach.



(**a**) Illustration of tested line segment intersections on a large sample of polygons.
1000 line segments are uniformly distributed and orientated in a space containing a large set of polygons. Red segments intersect a polygon and gray ones do not.




(**b**) Computational results for three approaches.
The average number of edges per inspected polygon (blue), the number of polygon inspections (orange), and the computation time (green) per approach. NB: the green bar of R-tree$^+$ is not visible due to its small size.

**Figure 4.23.** Performance assessment of three approaches for testing line segment - polygon intersections.

### *Time complexity*

An R-tree has the same depth in every branch, and objects are stored only in the leaf nodes. If we perform a search on an R-tree storing $n$ objects and having a node capacity of $m$, it always traverses $\mathcal{O}(\log n/\log m)$ nodes and all $m$ entries in each traversed node. Thus, the average case time complexity of a query on an R-tree is $\mathcal{O}(m \log n/\log m)$. We can validate this by comparing the naive approach to the R-tree$^+$ approach by applying the former approach

on the same set of subdivided polygons created in the latter one.

The computational results, provided in Table 4.4, show that the difference in time complexity is proportional to the difference in computation time. More specifically,

$$\frac{\mathcal{O}\left(m \log n / \log m\right)}{\mathcal{O}\left(n\right)} \hat{=} 1.3\text{e}{-}03 \sim 1.6\text{e}{-}03 = \frac{T_{\text{R-tree}^+}}{T_{\text{Naive approach}}}, \tag{4.48}$$

where $T_{\text{Naive approach}}$ and $T_{\text{R-tree}^+}$ are the computation times of the naive approach and the R-tree$^+$ approach, respectively, as listed in the table below.

**Table 4.4.** Computational results and time complexity of the naive and R-tree$^+$ approach. The number of leaf nodes equals the number of input polygons, i.e., $n = 35451$.

| | Time [s] | Tests [#] | Node capacity $m$ | Time complexity |
|---|---|---|---|---|
| Naive approach | 615.7 | 2.6e+07 | - | $\mathcal{O}\left(n\right)$ |
| R-tree$^+$ | 1.0 | 6.2e+02 | 10 | $\mathcal{O}\left(m \frac{\log n}{\log m}\right)$ |

## 4.6 Fitness evaluation

This section describes different elements of the fitness evaluation of a solution to the MOSWR problem. Section 3.2 formulates a route as a sequence of route legs between the start and end waypoints, with each route leg comprising a start and end waypoint and a constant engine speed over the leg. The fitness values of a route are represented by a vector containing the total travel time in days and the total fuel cost in user-specified cost units. The total travel time (fuel cost) is calculated as the sum of the travel time (fuel cost) of all route legs. Before calculating these fitness values, a route is tested for feasibility.

### 4.6.1 Feasibility

To determine the feasibility and cost parameters of a route leg, Section 4.5 proposed a modified spatial indexing method that tests for route leg intersections with impassable or high-cost areas. This method takes a straight line segment and returns the intersected areas, if any. For a straight line segment with small length $l_{fb}$, it is a good approximation of the curved line segment. We choose $l_{fb} = 15$ nautical miles for the case studies presented in 5. If the segment intersects an impassable area, its fitness values are set very large, and the actual computation of travel time and fuel cost is not performed. We choose very large fitness values such that feasible solutions always strictly dominate infeasible ones.

### 4.6.2 Travel time and fuel cost

The presented approach approximates the curved route leg with a sequence of straight line segments, defined by $k$ equal distant evaluation points between the leg's start and end waypoint. Similarly as described in Section 4.6.1, the length of each segment is defined by the parameter $l_{eval}$. We choose $l_{eval} = 8$ nautical miles for the case studies presented in 5.

The actual ship speed is calculated using Equation 3.28, given the constant engine speed, the time at the route leg start, and the environmental conditions at the $k$ evaluation points at the time of the ship's passage. The derivation of these variables is described in Section 3.4.

Using the intersection method described in Section 4.5, we determine whether a straight line segment connecting two route evaluation points intersects a high-cost area. For example, if a route segment intersects an ECA, an additional cost factor is incurred on the segment's fuel cost. Likewise, an intersection with a shallow water area yields an optimization penalty for both fitness values. We note here that these penalties do not affect the final objective values of a route and including penalties is optional.

The calculation of additional costs due to high-cost areas is performed using Equations 3.9 and 3.10. Penalties are computed in a similar manner: for each segment connecting two route evaluation points that intersect penalty area (e.g., shallow water area), its fitness values are multiplied with a penalty factor $f_p$, with $f_p \in \mathbb{R}^+$. For the case studies, we choose $f_p = 2$, such that the fitness values of route segment crossing shallow water have two times higher travel time and fuel cost compared to a similar segment not intersecting such area.

### 4.6.3 Speed improvements

Multiple values are evaluated or retrieved for each segment connecting two evaluation points, such as

- the segment length,
- intersection with an impassable area (feasibility),
- intersections with high-cost areas (cost evaluation),
- the eastward $S_E$ and northward $S_N$ ocean current speed components at the segment start at time $t$, and
- the wind speed $BN$ and direction $TWD$ at the segment start at time $t$,

where $t$ is the time at the route evaluation point. Many of these values need to be recalculated at a later stage as routes and child routes are to be re-evaluated many times in an MOEA. Therefore, the presented solution approach calculates or retrieves each value once and uses memoization (i.e., storing intermediate results) to reduce the number of expensive computations.

To obtain an additional reduction in the computation time, we can distribute the function that evaluates a population of solutions over multiple computer processors. By evaluating solutions in parallel, the computation time can be significantly reduced, depending on the number of used processors. Since each process acts independently from other processes, memoization needs to be performed in an overarching process for the multiprocessing to be effective. In other words, process A needs to communicate its intermediate results with

process B to reduce the computation time using multiprocessing effectively. In the present Python model, this is not yet implemented but recommended for future development.

## 4.7 Termination criterion

An MOEA is a metaheuristic without a clear termination criterion. We discuss two main approaches: (1) a priori termination methods and (2) a posteriori methods. In most cases, the algorithm terminates after a number of generations or a maximum amount of computation time. These a priori methods do not use any information on the convergence to the Pareto front at the time of termination. The required number of generations may be problem instance dependent and the determination of this number is computationally expensive for complex multi-objective optimization problems.

A posteriori methods in single-objective are typically based on the improvement of the best individuals in subsequent generations. However, the examination of convergence to the Pareto front is often impossible in multi-objective problems if no gradient information is available. Therefore, the design of termination criteria has become an essential aspect in evolutionary multi-objective optimization.

A common approach is to measure the progression of the current population by combining performance indicators and statistic tools, e.g., Martí et al. (2016) use a mutual domination rate as a performance indicator with a simplified Kalman filter to estimate the state of the evolutionary process. In this study, we use a termination criterion based on the generational distance indicator, as suggested by Collette and Siarry (2004). To detect situations where no further progress will be made, we combine the generational distance indicator with its variance over several generations.

The generational distance indicator $GD$ (Van Veldhuizen, 1999) is designed as a unary Pareto performance metric to measure the average distance from the approximated front $Z$ to the Pareto front $Z^*$. The indicator is given by

$$GD(Z) = \frac{1}{|Z|} \sum_{\mathbf{z} \in Z} \min_{\mathbf{z}^* \in Z_D^*} \|\mathbf{z} - \mathbf{z}^*\|^2 , \tag{4.49}$$

where $|Z|$ is the number of objective vectors in the approximated front $Z$ and $Z_D^*$ is a discrete representation of the Pareto front $Z^*$.

Collette and Siarry (2004) suggested that the generational distance can be used as a progress measure for a termination criterion in multi-objective optimization. A slight variation of the generational distance $GD(Z_t, Z_{t^+})$ between two successive generations could mean a convergence towards the Pareto front:

- If $GD(Z_t, Z_{t^+}) \gg 0$, then the population from generation $t^+$ is considerably different than its predecessor.
- The case $GD(Z_t, Z_{t^+}) = 0$ implies that there is no improvement from generations $t$ to

$t^+$. That is, for each of the two generations, its non-dominated routes lie on the same frontier as the non-dominated routes from the other generation.

Each MOEA considered in this study already evaluates the required objective values for the $GD$ for their own purpose. Thus, it is reasonable to use this indicator as a measure for performance improvement between two algorithms. The variance of the last $n_{stop}$ generational distances is computed. If the variance drops below a certain threshold $\sigma^2_{stop}$, the algorithm is terminated. Section 5.1 lists the values for $n_{stop}$ and $\sigma^2_{stop}$ used in the case study.

# 5 Computational results

The previous chapter described a method for solving the MOSWR problem. The method uses an MOEA as a framework to find a set of Pareto optimum routes. We selected three different MOEAs, which we test on its performance regarding the quality of the approximated Pareto front and the algorithms' computational complexity for this problem. We do this with specific performance metrics designed for multi-objective optimization methods. Each selected MOEA initializes with a set of initial routes, found on a hierarchical graph. It then seeks to find Pareto optimum routes with recombination and mutation techniques specialized for the MOSWR problem. The feasibility of a route trajectory is ensured using a fast line intersection method using R-tree spatial indexing.

This chapter reports the computational results of various case studies. The parameter settings used for the case studies are listed in Section 5.1. Furthermore, the section describes the tuning of the parameters using a sequential parameter tuning method. Next, Section 5.2 gives an overview of the data required for the route evaluation. This includes historical satellite altimetry-derived ocean current data, weather forecasts from the Global Forecast System (GFS), and shorelines provided by the GSHHG database.

Our solution approach is tested on different scenarios containing a variety of environmental conditions and conceptual case studies involving high-cost areas. To test the effect of environmental conditions on the routing, we perform two case studies that either consider ocean currents or weather. For the case study involving ocean current routing, we constructed three experiments that consider either static or time-varying ocean currents. Section 5.3 reports our findings and compares the results of the third problem instance with results obtained from an exact approach presented in the literature. For the case studies involving weather routing, we constructed three problem instances that represent a variety of weather conditions. Numerical results are presented in Section 5.4. Next, Section 5.5 provides several conceptual cases to demonstrate routing capabilities regarding high-cost areas, such as avoiding shallow waters and routing through ECAs. Finally, Section 5.6 reports our findings on the performance assessment of three selected MOEAs. Three problem instances are considered with different types of environmental conditions.

We implemented the MOEAs in a Python-based model using Distributed Evolutionary Algorithms in Python (DEAP) (Fortin et al., 2012), a Python framework for evolutionary algorithms. All simulations were executed using the NSGA-II algorithm unless stated otherwise. The numerical experiments are executed on a standard computer with an Intel Core i7-8650U 2.11 GHz processor and 16GB of RAM. Every instance has been run five times, and we report the average and standard deviations of these runs if applicable.

## 5.1 Algorithm parameters

The selected MOEAs include several parameters affecting the quality of the solutions found. For SPEA2 and M-PAES, we used identical population and archive sizes, i.e., $N = \overline{N}$, as suggested by the developers of the algorithms. Consequently, the same number of solutions are returned by both SPEA2 and M-PAES. In M-PAES, three more parameters must be set. These are the number of crossover trials $cr_{trials}$, the maximum number of local moves $l_{opt}$, and the maximum number of consecutive failing local moves $l_{fails}$. Values that give good general performance were found to be $l_{opt} = 5$, $l_{fails} = 3$, and $cr_{trials} = 5$. The parameter settings for the MOEAs used in the case studies are listed in Table 5.1.

The parameter values for which an interval is given are obtained by sequential optimization with the random forest minimizer presented by Hutter et al. (2011). The interval represents the optimization space for the parameter values. A tree-based regression model is used to model the expensive to evaluate algorithm. The model is improved by sequentially evaluating the function at the next best point in the optimization space. Thereby finding the minimum of the function with as few evaluations as possible.

The total number of evaluations, $n_{\text{calls}}$, are performed by first evaluating $n_{\text{points}}$ initial points from the optimization space are evaluated, followed by $n_{\text{calls}} - n_{\text{points}}$ evaluations guided by the surrogate model. The first $n_{\text{points}}$ points are uniformly selected from the optimization space $S_q$ of each parameter $q$ considered for optimization. For the number of evaluations and initial points, we choose $n_{\text{calls}} = 200$ and $n_{\text{points}} = 10$. The parameters are tuned for the NSGA-II algorithm using a training set of 5 instances covering different departure times, locations, and environmental effects. Since the M-PAES and SPEA2 are only used in the comparative study of the MOEAs, their populations' sizes are set manually, which we describe in Section 5.6.

## 5.2 Data description

This section gives a description of the data used as input to the presented routing algorithm. First, Section 5.2.1 lists the ship characteristics provided by an industrial partner. Then, Section 5.2.2 presents the environmental data, i.e., global wind forecasts and altimetry-derived ocean current information. The navigation area is defined using the shorelines described in Section 5.2.3. Finally, Section 5.2.4 explains the interpolation of the gridded environmental data to obtain a datum at any given location in the navigation area.

### 5.2.1 Ship characteristics

Ship characteristics for the calculation of ship speed loss due to weather and route parameters are listed in Table 5.2. The fuel price is set to 300 USD/t to represent the current market situation.[1]. The ECA multiplier $f_{\text{eca}}$ represents the multiplication factor of the fuel price

[1]Three-month average fuel prices in the period of May 1 to July 31, 2020, are retrieved from https://shipandbunker.com, accessed on August 13, 2020.

**Table 5.1.** Parameters for the MOEAs
U and G denote uniform and Gaussian, respectively, and F and E denote Feasibility and Evaluation, respectively.

| Parameter | Symbol | Value | Unit | Optimization space | Section |
|---|---|---|---|---|---|
| MOEA | | | | | |
| Population size | $N$ | 336 | - | $\{25k, \ldots, 200k : k = 4\}$ | 4.1 |
| M-PAES | | | | | |
| Max. local fails | $l_{fails}$ | 3 | - | | 4.1.3 |
| Max. local moves | $l_{opt}$ | 5 | - | | 4.1.3 |
| Max. crossover trials | $cr_{trials}$ | 5 | - | | 4.1.3 |
| NSGA-II and SPEA2 | | | | | |
| Mutation probability | $pb_{mut}$ | 0.28 | - | $(0.2, 0.7)$ | 4.4 |
| Crossover probability | $pb_{cr}$ | 0.81 | - | $(0.5, 1.0)$ | 4.4 |
| Genetic operators | | | | | |
| Insert width ratio - U | $wr$ | 4.22 | - | $(0.1, 10)$ | 4.4.2 |
| - G | | 6.35 | - | | |
| Move radius - U | $ri$ | 1.35 | - | $(0.1, 10)$ | 4.4.2 |
| - G | | 2.38 | - | | |
| Max. moves | $n_{moves}$ | 5 | - | $\{2, \ldots, 20\}$ | 4.4.2 |
| Geodesic Discrete Global Grid | | | | | |
| Recursion level | $L$ | 4 | - | | 4.3.1 |
| Variable recursion level | $L_v$ | 6 | - | | 4.3.1 |
| Termination | | | | | |
| Min. variance | $\sigma^2_{stop}$ | 1.64e−5 | - | $(1\text{e}{-7}, 1\text{e}{-4})$ | 4.7 |
| Sample size | $n_{stop}$ | 33 | - | $\{10, \ldots, 50\}$ | 4.7 |
| Fitness evaluation | | | | | |
| Seg. length - Feasibility | $l_{fb}$ | 15 | nm | $(1\text{e}{-7}, 1\text{e}{-4})$ | 4.6.1 |
| Seg. length - Evaluation | $l_{eval}$ | 10 | nm | $\{10, \ldots, 50\}$ | 4.6.2 |

consumed in ECA zones with respect to the standard fuel price. For the case study testing the cost-effect of ECA zones on ship routes, we adopt the ratio between the two fuel prices as in (Fagerholt et al., 2015). That is, fuel is assumed to be 55.93 % more expensive than standard fuel.

Table 5.3 lists the relations between engine power, nominal ship speed, and fuel consumption for the vessel considered in the case studies. These relations are obtained from an industry partner. The most fuel-efficient engine power in terms of fuel consumption per distance unit is the one corresponding to the lowest engine speed, i.e., the lowest nominal ship speed. Alternatively, one could choose to assume the fuel consumption rate as a function of the nominal ship speed. According to surveys of Psaraftis and Kontovas (2013), a frequently used nonlinear function for the fuel consumption rate is

$$FC(v) = A + Bv^3, \tag{5.1}$$

where $A, B \geq 0$ are parameters and $v$ is the nominal ship speed. Other ship characteristics, i.e., draft, length, displacement, and block coefficient, are used to calculate speed reduction due to wind.

**Table 5.2.** Characteristics of the ship considered in the case study and route parameters.

| Parameter type | Symbol | Value | Unit |
|---|---|---|---|
| Ship | | | |
| Length | $L_{pp}$ | 152.9 | m |
| Water displacement | $\nabla$ | 27,150 | $m^3$ |
| Block coefficient | $C_b$ | 0.80 | - |
| Route | | | |
| Fuel price | $c_f$ | 300.00 | USD/t |
| ECA multiplier | $f_{\text{eca}}$ | 1.5593 | - |

**Table 5.3.** Fuel table of the ship considered in the case study.
Engine settings directly relate to fuel consumption rates and nominal ship speed.

| Engines | Power [%] | Fuel [$t\,d^{-1}$] | Speed [kn] |
|---|---|---|---|
| 2 | 100 | 39.00 | 15.20 |
| 2 | 95 | 36.80 | 15.00 |
| 2 | 90 | 34.40 | 14.80 |
| 2 | 85 | 32.00 | 14.50 |
| 2 | 80 | 30.30 | 14.30 |
| 2 | 75 | 28.90 | 14.10 |
| 1 | 100 | 20.10 | 10.80 |
| 1 | 95 | 18.90 | 10.60 |
| 1 | 90 | 17.60 | 10.30 |
| 1 | 85 | 16.40 | 9.90 |
| 1 | 80 | 15.50 | 9.50 |
| 1 | 75 | 14.30 | 8.80 |

### 5.2.2 Environmental conditions

Table 5.4 gives a data summary of the environmental conditions data, including wind and ocean currents. During algorithm's execution, it loads the data into memory to achieve faster data access. If a data array is too large to fit into memory, the data array is split into many smaller arrays using Dask arrays (Dask Development Team, 2016). This enables computations on arrays with sizes exceeding the memory size using all computer processors. The remainder of this subsection gives a more detailed description of the wind and ocean current data separately.

**Table 5.4.** Environmental data summary

| | Wind | Ocean current |
|---|---|---|
| Source | (NCEP, 2020) | (GlobCurrent project, 2015) |
| Values | northward, eastward velocity components | |
| Unit | $\mathrm{m\,s^{-1}}$ | |
| Time period | 2007 to 2020 | 1993 to 2016 |
| Grid size | $0.5° \times 0.5°$ | $0.25° \times 0.25°$ |
| Time step | 6 h | 3 h |
| Data format | GRIB2 | netCDF |

*Wind*

The Global Forecast System (GFS) is a weather forecast model from the NCEP (2020). The NCEP is a part of the National Oceanic and Atmospheric Administration (NOAA) branch of the Department of Commerce. It covers the entire globe for the period of January 2007 to May 2020 at the time of writing. Multiple atmospheric and land-soil variables are available through this dataset, of which the wind direction and speed are of interest in this study. GFS also provides forecast datasets that are run four times daily at 00z, 06z, 12z, and 18z out to 192 hours with a 0.5° horizontal resolution and a 3-hour temporal resolution. Horizontal resolution drops to 1 degree for forecasts between one and two weeks.

For each latitude, longitude, and timestamp, we are provided the eastward velocity $u$ and northward velocity $v$ in meters per second. Using these values, we obtain the wind speed $BN$ on the scale of Beaufort and the true wind direction $TWD$, i.e., the direction from which the wind blows measured as the clockwise angle from the north line in degrees.

The distribution format used by GFS is GRIB2, a standardized data format commonly used in meteorology to store historical and forecast weather data. GRIB files are a collection of self-contained records of two-dimensional data. The individual records stand alone as meaningful data with no references to other records or an overall schema. This enables us to append GRIB records to create concatenated datasets of the desired time period.

*Ocean currents*

Many local sea currents exist and are affected by local and remote factors, including highly variable winds, surface and internal waves, tides, mixed layer depth, and buoyancy fluxes.

Considering these local currents in the ship routing model is a complicated task that we choose to leave out of scope. Instead, we use satellite altimetry-derived currents data produced by GlobCurrent project (2015). McCord et al. (1999) show that strategic routing using altimetry-derived currents is feasible and can be beneficial for a commercial shipping company.

Ocean currents are approximated with the sum of geostrophic currents and Ekman currents at the surface. The former is due to the horizontal pressure gradient force, balanced by the Coriolis force, while the latter result from the balance between friction induced by wind stress and Coriolis forces. A more detailed description of these ocean currents and their computation is provided in (GlobCurrent project, 2015).

Each data point contains an eastward and northward ocean current velocity component in meters per second. These values are converted to knots (i.e., $\mathrm{nmi\,h^{-1}}$) for the optimization. A different dataset than the one decribed in Table 5.4 is available for the Mediterranean Sea. It has a spatial resolution of 1/8 degree and a temporal resolution of three hours.

### 5.2.3 Navigation area

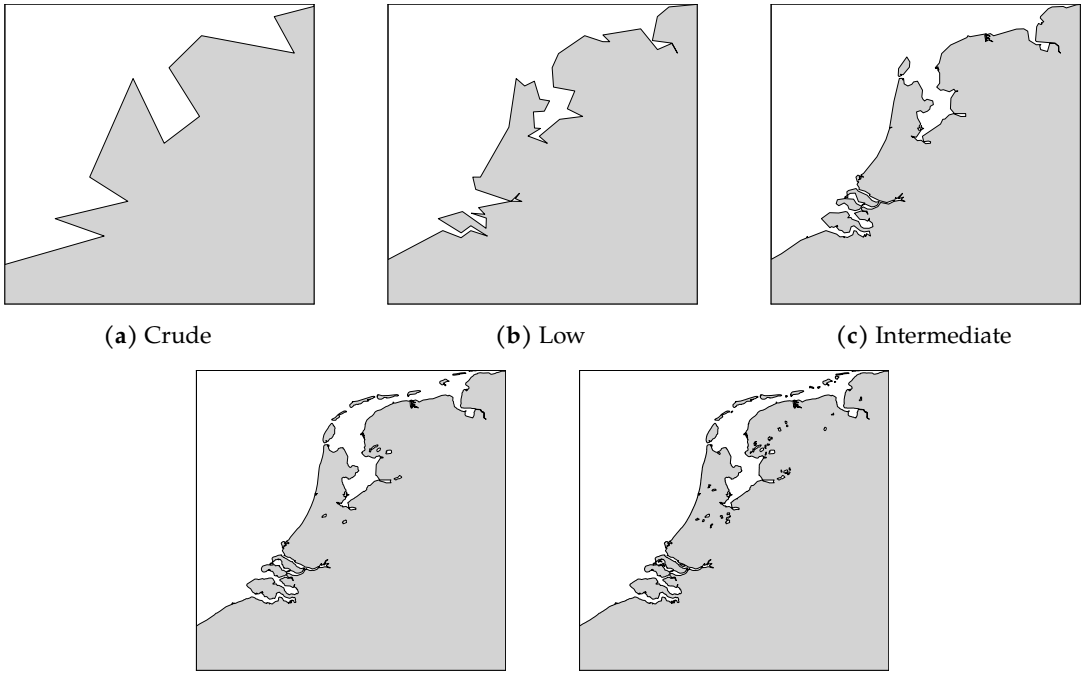
Section 3.2 formulates the sailing region as a spherical surface projected on the Earth ellipsoid. A location in the sailing region is defined by its latitude and longitude coordinates. To define the navigable area, the area in the sailing region excluding impassable areas, we use the GSHHG database produced by the NOAA (Wessel & Smith, 1996). GSHHG is a hierarchical database that provides shorelines, lakes, political borders, and rivers. In the case of the MOSWR problem, we are interested in the shorelines. The geography data come in five resolutions:

1. Full resolution: Original (full) data resolution.
2. High resolution: Approximately 80 % reduction in size and quality.
3. Intermediate resolution: Another 80 % reduction.
4. Low resolution: Another 80 % reduction.
5. Crude resolution: Another 80 % reduction.

Figure 5.1 illustrates these five resolution levels for the coast of the Netherlands as a sample region. Unless stated otherwise, we use the intermediate resolution in this study. The intermediate solution data include 32,832 polygons. The distributions of polygon sizes and edges are described in Section 4.5.4.

*Shallow waters*

In general, a ship cannot sail up close to a shoreline due to ship draft restrictions. Therefore, during optimization, an optional penalty is introduced for segments of a route leg that intersect shallow waters. These shallow areas are based on global bathymetry (ocean depth) data at a $0.5° \times 0.5°$ resolution provided by Becker et al. (2009). Kelso and Patterson (2018)



(**a**) Crude (**b**) Low (**c**) Intermediate

(**d**) High (**e**) Full

**Figure 5.1.** Five resolution levels of the GSHHG database (Wessel & Smith, 1996). The coast of the Netherlands is shown as example region.

provide ESRI shapefiles containing nested polygons that represent the bathymetry contours at 200 m water depth. Typically, the draft of a ship varies from ten to twenty meters; thus, water depth of 200 m is disproportionate to any ship's draft. However, we have limited access to data containing bathymetry contours that represent the maximum allowed depth of the ship in consideration. Nevertheless, we use this data to test the capabilities of the routing algorithm considering ocean depth.

*Emission Control Areas*

ECAs are considered as high-cost areas in the case study. The coordinates of ECAs designated under MARPOL Annex VI are given by the IMO. Using these coordinates, we defined ECAs with polygons and stored these in ESRI shapefiles.

### 5.2.4 Interpolation of gridded data

In the ship route optimization, it is necessary to obtain environmental conditions at any given time $t$ and ship's position using the data described above. Numerical interpolations over space are used to calculate the above-mentioned environmental data in the routing algorithm. As numerous interpolations have to be carried out in the simulation, we adopt a simple linear interpolation to minimize computation time.

The linear interpolation is performed by the following formula:

$$Q = Q_{31} + \frac{(Q_{32} - Q_{31})(\lambda - \lambda_1)}{\lambda_2 - \lambda_1}, \tag{5.2}$$

where

$$Q_{31} = Q_{11} + \frac{(Q_{21} - Q_{11})(\varphi - \varphi_1)}{\varphi_2 - \varphi_1}, \tag{5.3}$$

$$Q_{32} = Q_{12} + \frac{(Q_{22} - Q_{12})(\varphi - \varphi_1)}{\varphi_2 - \varphi_1}, \tag{5.4}$$

$Q$ are the environmental conditions at the ship's position $(\lambda, \varphi)$, and $Q_{ij}$ are the environmental data at the grid points $(\lambda_i, \varphi_j)$, for $1 \leq i, j \leq 2$, such that $\lambda_1 \leq \lambda \leq \lambda_2$ and $\varphi_1 \leq \varphi \leq \varphi_2$.

## 5.3 Ocean current routing

This section demonstrates the ocean current routing capabilities of the presented algorithm. It provides numerical results on two experiments: (1) time-varying ocean currents and (2) an exact approach comparison. Section 5.3.1 presents the first experiment: a case study in the Gulf Stream region involving time-varying ocean currents. Besides the effect of ocean currents on the ship routing, the case study assesses the effect of a constant versus a variable engine speed profile. Secondly, Section 5.3.2 presents the results of the second experiment. We validate and compare the results with numerical results from Tanaka and Kobayashi (2019).

Within the regions considered in the latter two experiments, western boundary currents (WBCs) exist, i.e., the Gulf Stream and the Kuroshio Current. There is a strong commercial shipping activity in this region (Kiln, 2012). Also, Chang et al. (2015) describe these regions as potential energy-saving areas for tactical ship routing. Thus, finding energy-saving routes may be of interest to shipping companies.

### 5.3.1 Time-varying ocean currents

The Gulf Stream is a WBC near the east coast of North America. It originates in the Gulf of Mexico and follows the eastern coastlines of the United States before crossing the Atlantic Ocean as the North Atlantic Current. The average velocity of the Gulf Stream is 3.5 kn, which decreases to approximately 0.8 kn as it widens to the north. For this experiment, we consider the bounded region near the east coast of the United States, where the upper and lower west (east) corners are denoted by W1 and W2 (E1 and E2), respectively. The corners are defined by the coordinates

**E1:** (38° 0′ N, 50° 0′ W),

**E2:** (46° 0′ N, 55° 0′ W),

**W1:** (34° 19′ N, 72° 40′ W), and

**W2:** (36° 40′ N, 73° 20′ W).

We select the region corners to serve as the route endpoints. Four eastbound (westbound) origin-destination pairs are formed by connecting each of the two western (eastern)

corners to each of the two eastern (western) corners. For each origin-destination pair, we perform route simulations with ship departure dates 25/11/2014, 00z, and 4/5/2015, 00z. We represent a voyage by its origin-destination pair and month of departure (e.g., "W1E2 - May").

Besides the effect of time-varying ocean current on the route trajectory, we test whether there is a potential in varying engine speed. For this, we perform simulations with a constant speed profile (C) and simulations for which the engine speed may vary along the route (V). Two constant speed settings are tested: (1) the maximum engine speed $U_{max}$, and (2) the fuel-efficient engine speed $U_{eff}$, i.e., the minimum engine speed. Each route simulation is compared to a reference route (R) neglecting ocean currents to quantify the ocean current optimization's potential. A reference route (R) is simulated excluding the effect of ocean currents but evaluated including these currents.

As a result, the algorithm performs 32 different route simulations comprising of eight origin-destination pairs. That is, two simulations that either include or exclude ocean currents, and three different speed profiles. Since each simulation is run five times, a total of 160 route simulations are performed for this experiment.

The remainder of this section discusses the computational results of the experiment described above. First, we discuss the results of the ocean current effects on the routing. Then, we analyze the potential of enabling a varying ship speed profile in the route optimization.

*Ocean current effect*

In the considered sailing region, the Gulf stream flows in the east-northeast direction. Hence, the routing algorithm finds, presumably, eastbound routes that follow the Gulf Stream, taking advantage of the strong current. On the other hand, the ocean current routing for westbound routes should minimize the adverse effect of opposite ocean currents by determining whether to avoid or cross the Gulf Stream at a weak point. In other words, ships following eastbound routes would likely ride the current if positive speed effects are prescribed to be better than the shortest path, and ship heading west likely travel in regions of low current activity.

Figure 5.2 shows a selection of eastbound and westbound routes. The corresponding Pareto fronts and each member's average speed change due to ocean current are shown by Figure 5.3. For both scenarios, the routing algorithm performs different routing decisions:

(a) The eastbound routes shown in Figure 5.2a have the same origin but different destinations. The routes in both Pareto approximation sets follow the same path in the early stage of the voyage to utilize the strong current of the Gulf Stream. Consequently, the average speed increase due to ocean currents is higher than the reference routes, shown in Figures 5.3a and 5.3b. This speed increase results in less costly, faster routes, or a combination of these two.

(b) In contrast to the eastbound routes, the westbound routes shown in Figure 5.2b have

different origins but the same destination. The algorithm found routes that avoid ocean currents but keep sailing distance at a minimum. As a result, current-optimized routes have less average speed decrease due to ocean currents than the reference routes, as shown in Figures 5.3c and 5.3d. The Pareto fronts of current-optimized route simulations are therefore shifted to the lower-left corner with respect to the reference simulations' fronts.

Two general paths are found for both westbound voyages. For the first voyage, E1W2, the first path passes the current field along the north side. Consequently, this path crosses the Gulf Stream two times at a weak point, resulting in a shorter distance but a decrease in ship speed. The second path passes the current underneath and is only taken by routes with a high speed profile. A possible reason is that currents in this area are only favorable for certain time periods, such that ships with lower engine speed would encounter adverse currents. For the second voyage, E1W2, routes either cross the Gulf Stream near the destination, or avoid the current field by possibly increasing the sailing distance.

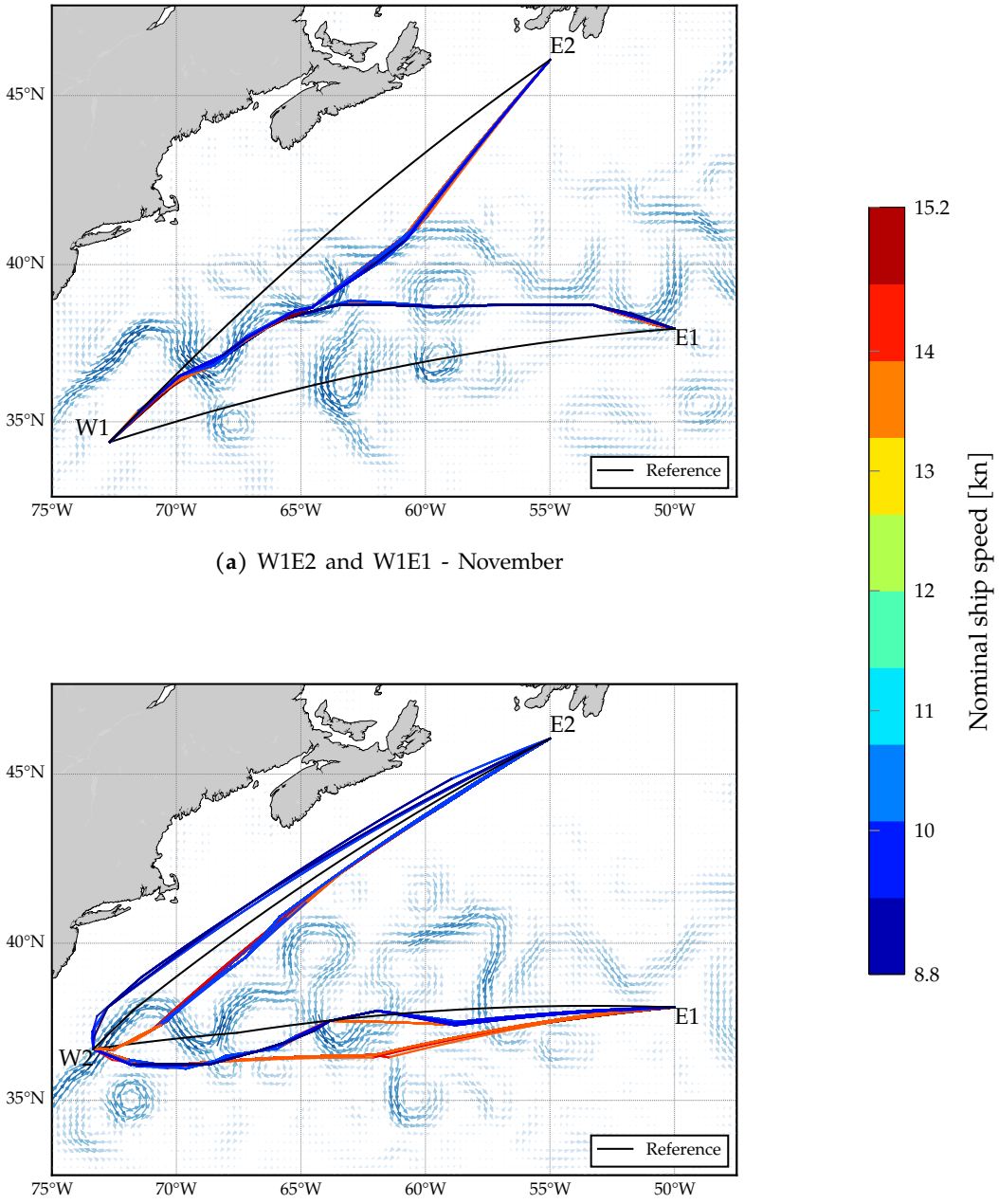
For each Pareto approximation set shown in Figure 5.3, we notice a change in the average speed increase as the travel time increases. Since the average speed change is similar for the voyages with the same departure date, this is likely caused by a variation of ocean currents in the considered time period.

Tables 5.5 and 5.6 list the five-run averaged extreme objective values of the Pareto fronts corresponding to simulations with departure in November 2014 and May 2015, respectively. Furthermore, the tables provides computational results for constant engine speed simulations. On average over all routes, 1.5 % and 2.8 % savings are achieved in travel time and fuel cost, respectively. The least-time routes save on average 2.0 % travel time and the least-fuel routes save on average 4.1 %. The largest fuel saving for a variable speed setting are found in W1E1 - May, i.e., the least-fuel cost route optimized for ocean current is 7.8 % (1.8 kUSD) less than its reference route.

*Engine speed effect*

The Pareto sets corresponding to route simulations with constant speed profiles contain a single point. This is due to a linear relation between fuel consumption and travel time, since the fuel consumption rate is fixed for a constant engine speed. Looking at the Pareto fronts, we see that the points corresponding to the simulation with constant speed profile lie at the Pareto approximation sets' extreme ends for variable speed. More specifically, the objective value of a simulation with maximum engine speed $U_{max}$ is approximately equal to the objective value of the simulation's minimum time route with variable engine speed. This is confirmed by the percentage differences shown in Tables 5.5 and 5.6.

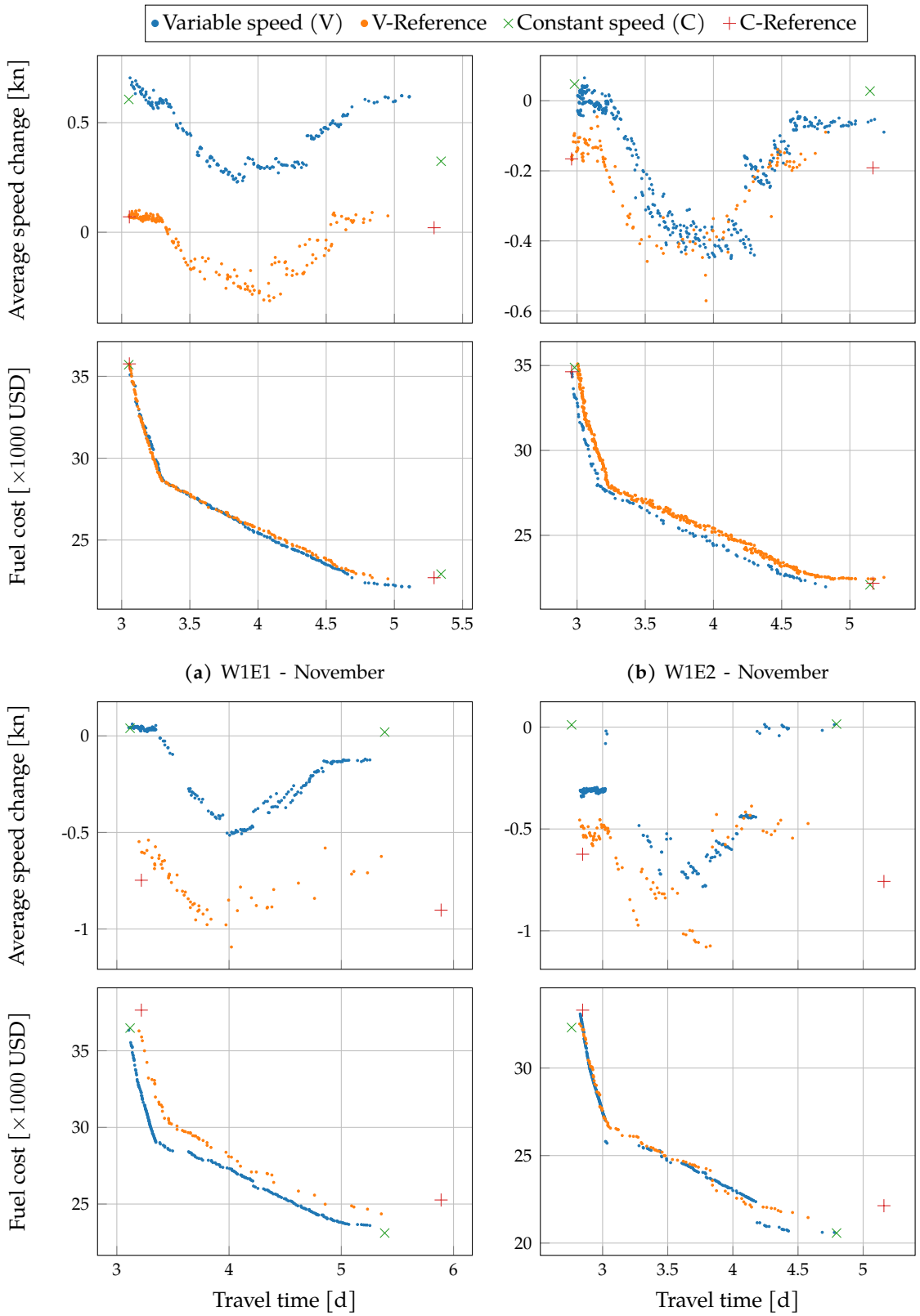
The results show that a least-time route with variable engine speed has approximately equal objective values to the route with maximum engine speed $U_{max}$. That is, the savings in travel time and fuel cost are within 0.0 % – 0.1 %, averaged over all simulations. This is




(**a**) W1E2 and W1E1 - November

(**b**) E2W2 and E1W2 - May

**Figure 5.2.** Pareto-optimal routes for a selection of voyages in the Gulf Stream region. The color of a route leg indicates the nominal ship speed at the leg and directly relates to the engine speed. Black routes represent reference routes neglecting the effect of ocean currents during optimization. The ocean current velocity and direction at the time of departure is shown by the vector field. Problem instances are described in the text. NB: Many routes roughly follow the same path. Consequently, colored legs represent the route leg speeds of the top-layered routes, i.e., routes with low fuel cost.




(**a**) W1E1 - November

(**b**) W1E2 - November

(**c**) E1W2 - May

(**d**) E2W2 - May

**Figure 5.3.** Computational results for a selection of four route simulations.
The average speed change due to ocean current (top) and the objective values (bottom) for each member of the Pareto set. Results for both current-optimized and reference routes are shown, as well as two constant (C) and one variable (V) speed settings.

**Table 5.5.** Objective values of extreme routes for the 'November' simulations.
Extreme routes are the least-time and least-fuel routes of the Pareto front. Relative difference ($\Delta$) are listed with respect to the reference route neglecting ocean currents and to routes with constant engine speed. The least-fuel and least-fuel routes are compared to the fuel-efficient $U_{eff}$ and maximum engine speed $U_{max}$, respectively. Values are obtained by averaging five runs.

| | | | w.r.t reference | | w.r.t constant speed | |
|---|---|---|---|---|---|---|
| Objective | T [d] | C [kUSD] | $\Delta$T[1] [%] | $\Delta$C[1] [%] | $\Delta$T[1] [%] | $\Delta$C[1] [%] |
| E1W1 | | | | | | |
| Fuel | 5.3 | 23.0 | −2.1 | −2.0 | −1.1 | 0.1 |
| Time | 3.1 | 36.2 | −0.8 | −0.7 | −0.1 | −0.1 |
| E1W2 | | | | | | |
| Fuel | 5.3 | 23.8 | −1.7 | −4.5 | −4.7 | −0.3 |
| Time | 3.2 | 37.0 | −2.1 | −1.1 | 0.4 | 0.4 |
| E2W1 | | | | | | |
| Fuel | 5.2 | 22.5 | 2.7 | −1.7 | −4.0 | −2.6 |
| Time | 3.0 | 35.0 | −1.1 | −0.2 | −1.2 | −1.2 |
| E2W2 | | | | | | |
| Fuel | 4.9 | 21.4 | −0.5 | −5.7 | −1.1 | 0.1 |
| Time | 2.8 | 33.2 | −2.4 | −2.4 | 0.1 | 0.1 |
| W1E1 | | | | | | |
| Fuel | 4.9 | 20.9 | −2.5 | −7.8 | 0.0 | 0.0 |
| Time | 3.0 | 34.6 | −2.8 | −2.8 | 0.0 | 0.0 |
| W1E2 | | | | | | |
| Fuel | 4.5 | 19.4 | −5.6 | −5.6 | −0.4 | −0.4 |
| Time | 2.8 | 32.2 | −2.8 | −2.9 | −0.1 | −0.1 |
| W2E1 | | | | | | |
| Fuel | 4.7 | 20.1 | −1.1 | −6.3 | −0.5 | −0.1 |
| Time | 2.9 | 33.4 | −3.1 | −3.1 | −0.1 | −0.1 |
| W2E2 | | | | | | |
| Fuel | 4.3 | 18.6 | 1.0 | −0.3 | −0.4 | −0.1 |
| Time | 2.6 | 30.3 | −1.3 | 1.9 | 0.0 | 0.0 |
| Average savings | | | | | | |
| Fuel | | | −1.2 | −4.2 | −1.5 | −0.4 |
| Time | | | −2.1 | −1.4 | −0.1 | −0.1 |
| Overall | | | −1.7 | −2.8 | −0.8 | −0.3 |

[1] $\Delta = \frac{\text{value} - \text{reference value}}{\text{reference value}} \times 100\%$

**Table 5.6.** Objective values of extreme routes for the 'May' simulations.
Extreme routes are the least-time and least-fuel routes of the Pareto front. Relative differences ($\Delta$) are listed with respect to the reference route neglecting ocean currents and to routes with constant engine speed. The least-fuel and least-fuel routes are compared to the fuel-efficient $U_{eff}$ and maximum engine speed $U_{max}$, respectively. Values are obtained by averaging five runs.

| | | | w.r.t reference | | w.r.t constant speed | |
|---|---|---|---|---|---|---|
| Objective | T [d] | C [kUSD] | $\Delta$T[1] [%] | $\Delta$C[1] [%] | $\Delta$T[1] [%] | $\Delta$C[1] [%] |
| E1W1 | | | | | | |
| Fuel | 5.2 | 22.5 | 0.2 | −3.5 | 0.0 | 0.3 |
| Time | 3.1 | 35.9 | −1.6 | −1.5 | 0.1 | 0.1 |
| E1W2 | | | | | | |
| Fuel | 5.2 | 22.6 | −0.9 | −5.7 | −0.3 | 0.1 |
| Time | 3.1 | 35.9 | −2.4 | −2.3 | 0.2 | 0.2 |
| E2W1 | | | | | | |
| Fuel | 4.9 | 21.1 | 1.7 | −5.5 | 0.2 | 0.2 |
| Time | 2.9 | 33.8 | −3.2 | −0.8 | 0.1 | 0.1 |
| E2W2 | | | | | | |
| Fuel | 4.7 | 20.5 | −1.0 | −5.2 | −0.3 | 0.1 |
| Time | 2.8 | 32.4 | −2.2 | −2.1 | 0.1 | 0.1 |
| W1E1 | | | | | | |
| Fuel | 5.1 | 22.0 | −1.7 | −2.7 | −2.9 | −2.4 |
| Time | 3.0 | 35.3 | −1.7 | −1.7 | −0.3 | −0.3 |
| W1E2 | | | | | | |
| Fuel | 4.7 | 20.4 | −4.8 | −4.7 | −2.3 | −2.2 |
| Time | 2.8 | 33.0 | −2.6 | −2.6 | −0.1 | −0.1 |
| W2E1 | | | | | | |
| Fuel | 4.9 | 21.5 | −5.2 | −3.6 | −16.8 | −15.4 |
| Time | 3.0 | 34.8 | −2.0 | −0.9 | 0.1 | 0.1 |
| W2E2 | | | | | | |
| Fuel | 4.5 | 19.5 | 4.2 | −0.4 | −0.3 | 0.0 |
| Time | 2.7 | 31.2 | 0.0 | 0.0 | 0.0 | 0.0 |
| Average savings | | | | | | |
| Fuel | | | −0.9 | −3.9 | −2.8 | −2.4 |
| Time | | | −1.9 | −1.5 | 0.0 | 0.0 |
| Overall | | | −1.4 | −2.7 | −1.4 | −1.2 |

[1] $\Delta = \frac{\text{value}-\text{reference value}}{\text{reference value}} \times 100\%$

due to the least time routes selecting the highest attainable speed for each leg, resulting in a constant (maximum) speed at each leg. Hence, for the least time routes, a varying speed profile seems to have no potential.

On the other hand, the difference in the least-fuel route with variable engine speed versus fuel-efficient speed routes is larger. The highest savings are achieved for routes with longer travel times, e.g., the travel time and fuel cost for E2W1 - November are reduced by 4.0 % and 2.6 %, respectively. And the average savings in travel time (fuel cost) is 2.15 % (1.4 %) for 'November'. This means that there is a potential benefit in varying the engine speed for minimization of the fuel consumption considering ocean currents. The fuel consumption increases marginally with the increase in ship speed at a low engine speed; thus, varying the engine speed increases the flexibility of route decisions. For instance, slightly increasing the ship speed to 'catch' a favorable current may lead to less total fuel consumption.

Overall, our findings show that savings are highest for routes with longer travel times, as ocean current has a larger contribution to the actual ship speed in this case and more routing decisions can be considered. Furthermore, savings with respect to a constant speed settings are marginal for the least-time routes and slightly better for least-fuel routes. Although, a varying engine speed increases the routing possibilities significantly and might provide more significant improvements for other, more balanced routes in the Pareto front. Overall, our findings show that ship routing considering strong time-varying strong ocean currents has a strong potential for minimizing both fuel cost and travel time but savings due to a variable engine speed are only significant for low speeds.

### 5.3.2 Exact approach comparison

This numerical experiment involves ocean current routing within the region of the Kuroshio Current (KC). The KC is a WBC in the East China Sea, which flows from the east of Taiwan along the south coast of Japan into the North Pacific. The average velocity of the KC increases from about 2 kn along the east coast of Taiwan to 2.7 kn near Japan. As mentioned earlier in Section 2.2, Tanaka and Kobayashi propose a routing algorithm based on dynamic programming and test their algorithm on the same region. By simulating an identical problem instance, we compare the numerical results found by Tanaka and Kobayashi (2019) with our results provided below.

The sailing region is the rectangular region defined by the pair of origin-destination locations

**K:** (26° 0′ N, 123° 0′ W), near Keelung, Taiwan, and

**T:** (34° 0′ N, 139° 0′ E), near the coast of Tokyo, Japan.

As Tanaka and Kobayashi assumed the ocean currents to be static, no departure date is specified. To match the characteristics of the ship in the case study of Tanaka and Kobayashi,

the fuel consumption rate $FC$ in tonnes per day has a cubic relation with the nominal ship speed, expressed as

$$FC(V) = 5.466 \times 10^{-4} V^3. \tag{5.5}$$

where $V$ is the nominal speed in knots and may take any value from the discrete set

$$\{8, 8.1, 8.2, \ldots, 14\}.$$

Similar to the previous experiment, each current-optimized route simulation is compared to a reference route neglecting ocean currents.

Tanaka and Kobayashi provided the identical ocean current dataset, comprising the current directions and velocities at various points in the sailing region for the period 1985 to 1988, accumulated by the JODC (2015). By averaging the ocean current velocities over time and linearly interpolating missing values in the sailing region, the data is processed in a similar way as Tanaka and Kobayashi did. Also, we use the 'low' resolution data from the GSHHG database, as in (Tanaka & Kobayashi, 2019). Figure 5.4 shows the resulting ocean current data projected in the considered region.

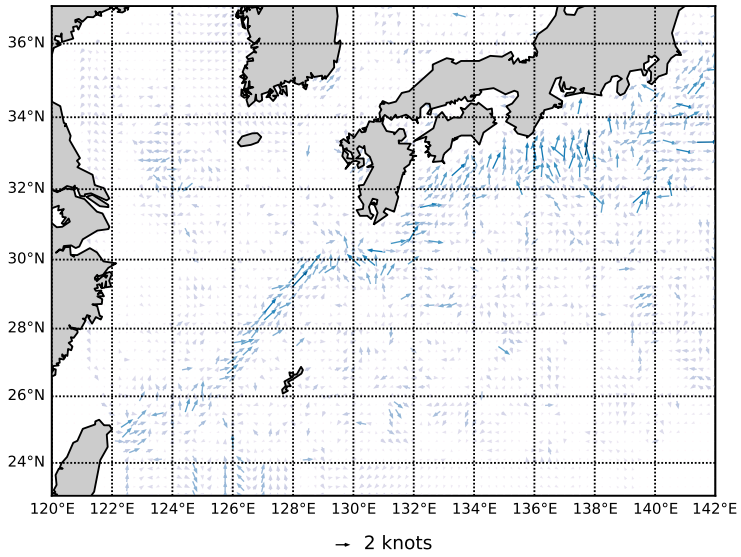
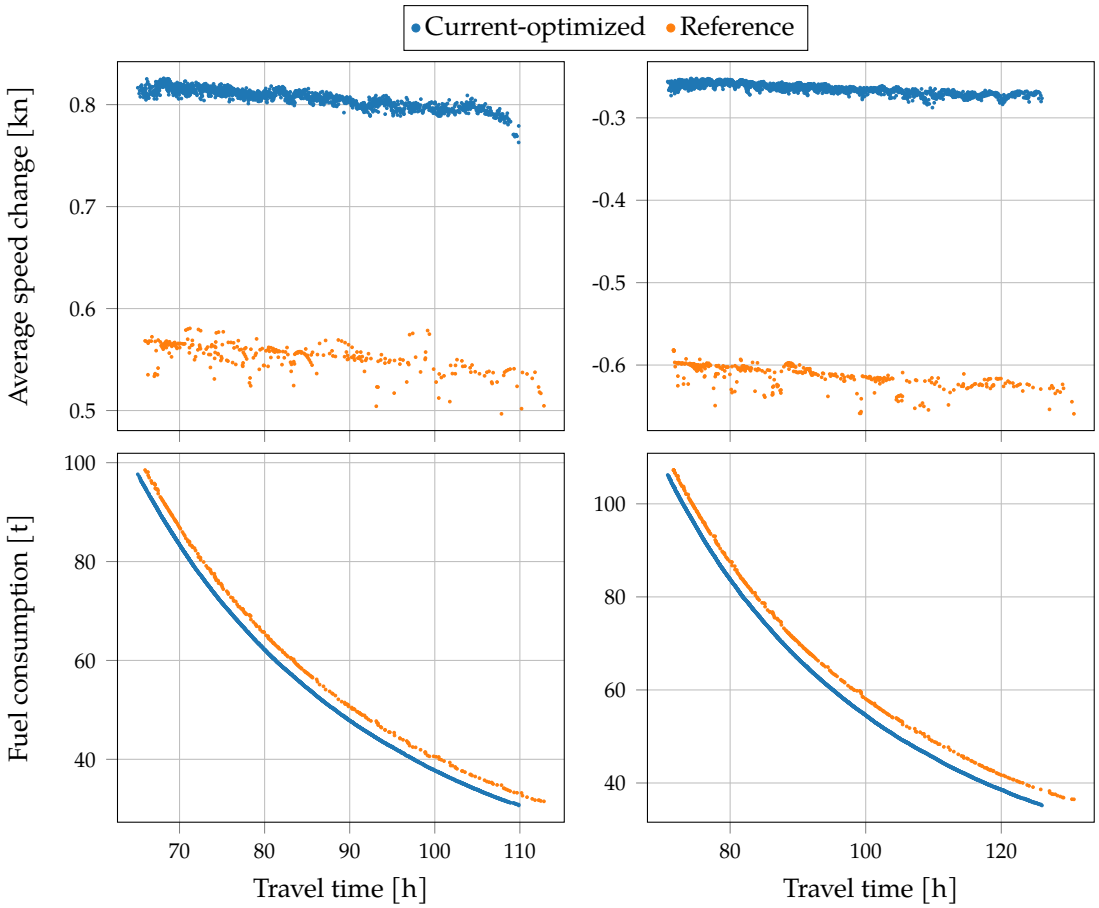



**Figure 5.4.** Bin-averaged ocean current data in the Kuroshio Current region (JODC, 2015).

*Results*

For both the eastbound and westbound voyages, Figure 5.5 shows the corresponding Pareto fronts, and Figure 5.6 visualizes the associated routes. Expectedly, the reference routes follow the great circle route, i.e., the route with the shortest distance. The current-optimized routes follow the current to take advantage of the favorable current direction. As the direction and velocity of the ocean currents are constant over time, the algorithm finds few optimal paths so that the variation in engine speed mainly causes the difference in Pareto optimality. As Equation 5.5 defines a cubic relation between fuel consumption and ship speed, the travel time and fuel consumption have an exponential relation. The shape of the




**(a)** Keelung - Tokyo (KT)

**(b)** Tokyo - Keelung (TK)

**Figure 5.5.** Computational results for routes KT and TK.
The average speed change due to ocean current (top) and the objective values (bottom) for each member of the Pareto set. Each scenario contains a reference set of routes neglecting the effect of ocean currents during optimization.

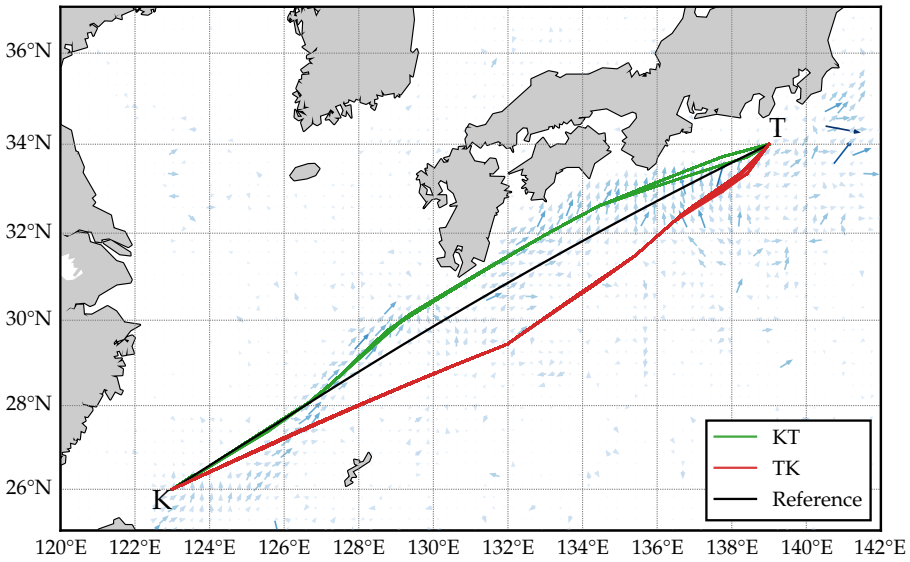



**Figure 5.6.** Pareto optimum routes for Keelung - Tokyo (KT) and Tokyo - Keelung (TK). The black route represents the reference route neglecting the effect of ocean currents during optimization. The ocean current velocity and direction within the sailing region is represented by the vector field.

Pareto fronts confirms this exponential relationship. For both voyages, the Pareto front corresponding to the current-optimized route simulations dominates the Pareto front of the reference simulation entirely.

The route trajectories of Keelung-Tokyo (KT) and Tokyo-Keelung (TK) are clearly distinct. KT routes take advantage of the KC by following the current, whereas TK routes head south to avoid the current. Figure 5.5 confirms this by showing the speed increase due to ocean currents for each route. For all KT (TK) routes, the average speed increase (reduction) is highest (lowest) for the current-optimized routes compared to the reference routes. We notice even smaller speed reductions for some routes of TK in the travel time interval of 102 h – 125 h. This jump in speed decrease is presumably due to a decision to take a longer path, passing an area with calmer water. Consequently, the travel time increases while the fuel consumption decreases.

### *Ocean current routing algorithm comparison*

The ship routing problem studied by Tanaka and Kobayashi (2019) is to minimize the fuel consumption of a voyage with arrival time constraints. In this problem, the effect of ocean currents is taken into account, and the ship speed is constrained within a range of $V_{min}$ and $V_{max}$, such that a route may attain a varying ship speed profile. As Tanaka and Kobayashi represent the sailing region as a grid, computational results are provided for different resolutions. By sequential relaxation of the arrival time constraint, the algorithm finds optimum ship routes on the grid. If the algorithm terminates before certifying the optimality, the best-known feasible solution is returned. For both the eastbound (KT) and westbound (TK) voyage, Tanaka and Kobayashi performed simulations with three grid resolutions and five arrival times.

We compare the worst-case and best-case scenario of the case study by Tanaka and Kobayashi with our algorithm's computational results. That is, in the worst-case scenario a low grid resolution of $40 \times 40$ nodes is used, while the best-case scenario has a high grid resolution of $160 \times 160$ nodes. We compare all arrival time constraints with matching routes in our algorithm's returned Pareto set for both cases. Table 5.7 provides a comparison of the results from both case studies.

**Table 5.7.** Comparison of results from the presented algorithm and the algorithm presented in (Tanaka & Kobayashi, 2019).
The relative difference ($\Delta$) is calculated with respect the best- and worst-case values from Tanaka and Kobayashi. ATC denotes the arrival time constraint in hours.

| | Fuel Consumption [t] | | | | $\Delta^1$ [%] | |
|---|---|---|---|---|---|---|
| | Tanaka and Kobayashi | | Presented algorithm | | | |
| ATC [h] | Best-case | Worst-case | Mean | Std. | Best-case | Worst-case |
| Keelung - Tokyo | | | | | | |
| 110 | 29.7 | 30.2 | 31.0 | (0.1) | 4.5 | 2.8 |
| 105 | 31.1 | 32.3 | 34.4 | (0.0) | 10.5 | 6.4 |
| 100 | 35.0 | 36.3 | 38.3 | (0.4) | 9.4 | 5.5 |
| 95 | 39.6 | 40.9 | 43.0 | (0.4) | 8.5 | 5.0 |
| 90 | 45.1 | 46.4 | 48.4 | (0.5) | 7.4 | 4.4 |
| Tokyo - Keelung | | | | | | |
| 130 | 33.9 | 35.1 | 35.1 | (0.4) | 3.6 | 0.0 |
| 125 | 33.9 | 35.3 | 35.7 | (0.1) | 5.3 | 1.1 |
| 120 | 34.5 | 38.2 | 38.3 | (0.2) | 11.1 | 0.3 |
| 115 | 37.6 | 41.5 | 41.5 | (0.0) | 10.2 | −0.1 |
| 110 | 41.2 | 45.2 | 45.2 | (0.4) | 9.7 | 0.0 |

[1] $\Delta = \frac{\text{value}-\text{reference value}}{\text{reference value}} \times 100\%$

In general, the exact approach by Tanaka and Kobayashi finds routes with less fuel consumption with similar travel time. For the eastbound voyage, KT, our algorithm's fuel consumption is higher for both the best- and worst-case scenarios for every arrival time constraint. Comparing our results to the worst-case scenario from Tanaka and Kobayashi, we notice a 3.8 % – 6.4 % higher fuel consumption for the eastbound voyages. This is 10.5 % for the best-case scenario. For the westbound voyage, TK, these differences in objective values are smaller but still reach 11.1 % for the best-case scenario.

Tanaka and Kobayashi compute the relative gap between the obtained lower and upper bound of a solution's objective value. They use this relative gap as an indication for the optimality gap. For routes with a high ATC, Tanaka and Kobayashi find objective values with the smallest gap. That is, the gap lies typically between 0.0 % – 5 % for routes with loose time constraints and increase up to 34.7 % for low ATC values, i.e., more restrictive time constraints. This means that the algorithm of Tanaka and Kobayashi finds near-optimal solutions on the grid for routes with loose arrival time constraints. Since our approach is not limited to a grid, it finds similar objective values for routes with high values for the ATC easily. The small difference in objective values can be interpreted using the routes shown in Figure5.6 and the routes presented in (Tanaka & Kobayashi, 2019).

KT routes follow the KC by deviating north from the great circle route to take advantage of the strong current at Japan's south coast. In contrast, TK routes try to avoid the current as it navigates in the opposed direction by deviating south from the great circle route. Compared to the routes found by Tanaka and Kobayashi, the paths of our TK routes are roughly similar, but the KT routes take an entirely different path. That is, both routes presented in (Tanaka & Kobayashi, 2019) follow a path south from the great circle, implying that both routes avoid ocean currents. This difference in route trajectories explains the difference in objective values between both voyages. That is, the difference in objective values for the voyage TK is smaller than for KT, as shown in Table 5.7.

To conclude; our algorithm finds good solutions for routes with less fuel consumption, but slightly worse solutions for routes with less travel time. However, routes found by Tanaka and Kobayashi are less smooth, meaning that routes found by our approach represent a more realistic scenario.

Regarding the computation time, the algorithm presented by Tanaka and Kobayashi is terminated after roughly one hour of computation time for the majority of the simulations. In contrast, our algorithm converges in less than three minutes for all runs. Additionally, the results presented in Table 5.7 can be obtained in a single run for each voyage, whereas the algorithm presented by Tanaka and Kobayashi needs to perform a simulation for each arrival time constraint. That is, their algorithm's required computation time for finding the 20 presented solutions is 45.6 min. Therefore, our algorithm is a better alternative for ocean current routing if one desires low computation time, requires several route alternatives that are Pareto-optimal, or seeks to find realistic ship routes.

## 5.4 Weather routing

This section provides numerical experiments on three problem instances to assess the weather routing capabilities of the proposed algorithm. Within these problem instances, we include coverage of routes in various weather conditions at locations around the world. The weather conditions range from a maximum Beaufort number (BFT) of 6 ("strong breeze") to 12 ("hurricane-force"). The weather routing problem instances are the following:

**KT-SF:** Keelung, Taiwan, to San Francisco, USA, departing on 28 May 2011, 00z,

**EC-NYC:** English Channel, to New York, USA, departing on 25 January 2011, 15z, and

**PLM-HAV:** Plymouth, England, to Havana, Cuba, departing on 24 September 2013, 12z.

Section 5.4.1 presents the results for KT-SF. In this instance, a storm with maximum wind speeds up to 12 BFT moves from the east coast of Taiwan northeast to Japan's south coast. The shortest route passes the storm at the north side, which may lead to adverse speed effects. To avoid the storm, a ship could speed up and minimize the time near the storm or detour north of Japan to sail in calm waters. In the remaining part of the voyage, the shortest

route moves through large areas in the Pacific, with wind speeds up to 8 BFT before arriving at San Francisco.

The results of EC-NYC are presented in Section 5.4.2. In EC-NYC, routes encounter rough sea conditions at the North Atlantic. Shao et al. (2012) consider a similar problem instance. They formulated a ship weather routing problem with similar objectives as the MOSWR problem and enables a variation in ship speed during the route optimization. They also use Kwon's (2008) method for the calculation of ship speed loss due to wind. Since different ship characteristics and weather data is used, we only compare the trajectories of the ship routes visually.

Section 5.4.3 discusses the results of instance PLM-HAV. Equivalently to EC-NYC, this instance considers a voyage within the North Atlantic. A similar problem instance is considered by Szapczyska (2015), who also applies an MOEA to simultaneously optimize multiple objectives. We provide a brief comparison of the trajectories of the routes found in (Szapczyska, 2015) and the routes returned by our algorithm.

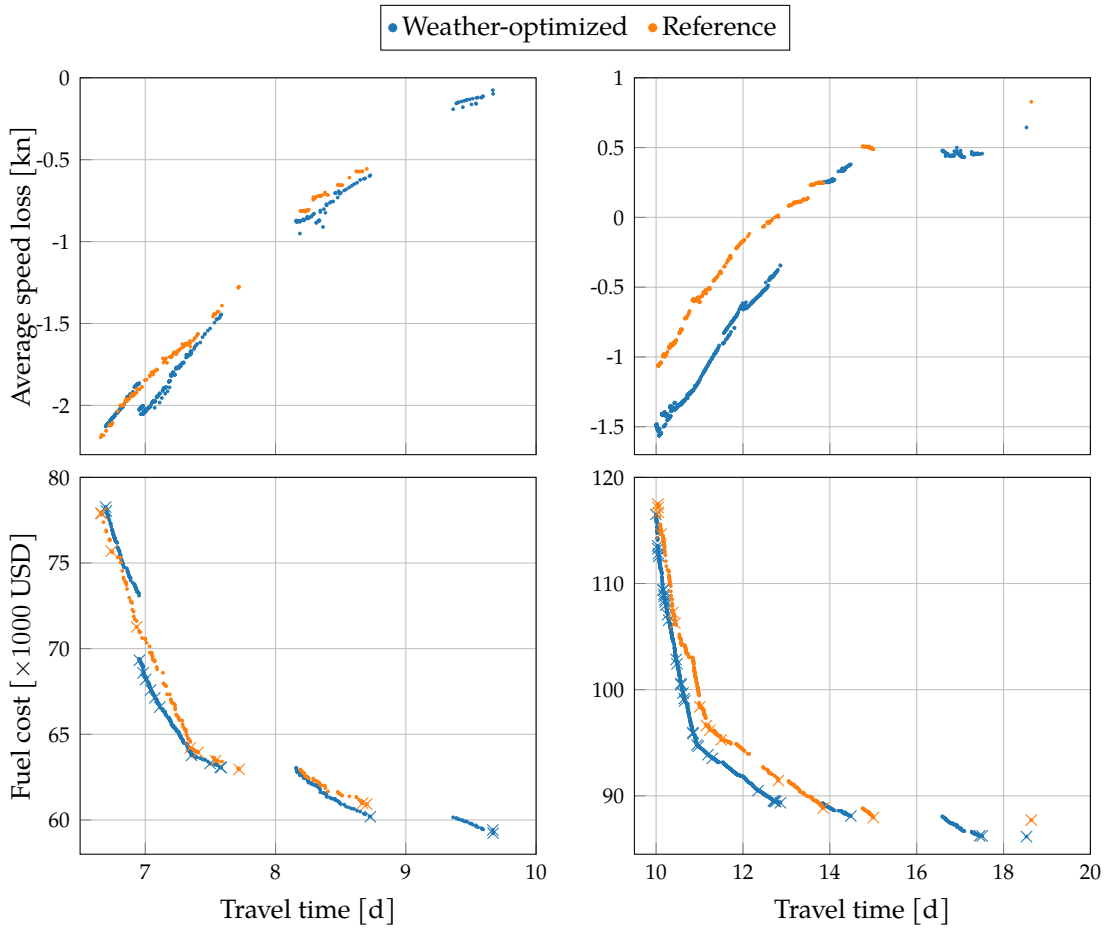
For each instance, a reference route is calculated, ignoring the effects of weather. Furthermore, the effect of a varying speed profile is determined by performing two additional simulations with either a maximum or minimum constant speed profile. The extreme objective values of the varying and constant speed simulations are discussed in Section 5.4.4.

Solution routes are visualized in Figure 5.8. Weather-optimized routes are colored according to each leg's nominal ship speed, and the reference routes are colored black. The map is shaded based on the wind speed at the ship's location over the entire travel time period. It roughly visualizes the wind speed at the location of a ship that follows the route with average travel time. In order to do this, the rectangular sailing region between the start and end location is subdivided into $\lceil \overline{T}/6 \rceil$ vertical weather slices, where $\overline{T}$ is the average travel time of the routes in the obtained Pareto front. Each slice corresponds to a 6-hour time period of weather data. Since we choose the average travel time to visualize the weather on the map, a ship following routes with short or long travel times may encounter different weather effects than the weather shown on the map. The shown routes are returned by a single route simulation.

### 5.4.1 Keelung to San Francisco

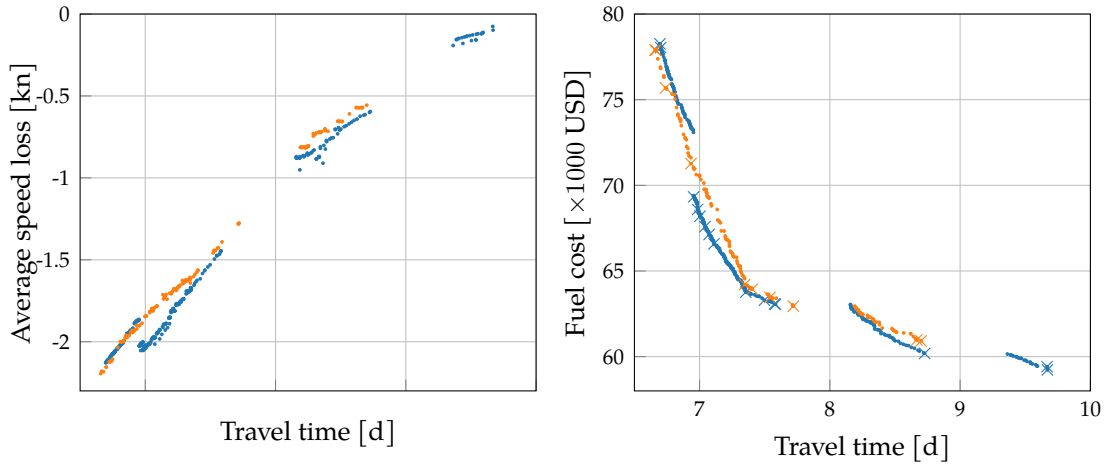
The voyage in KEE-SF encounters three rough weather areas. One near the south coast of Japan with winds up to 12 BFT, a smaller area with lighter winds halfway the voyage, and a third large area with high winds near the destination.

The Pareto set corresponding to a single route simulation with different settings is presented in Figure 5.7c. Figure 5.8a shows the routes corresponding to the convex hull of the Pareto front. We choose the convex hull routes so that shown routes are more distinctive, as the convex hull contains a diverse set of routes, i.e., it typically includes all major routing decisions made by the algorithm. The routes are projected on a stereographic projection,



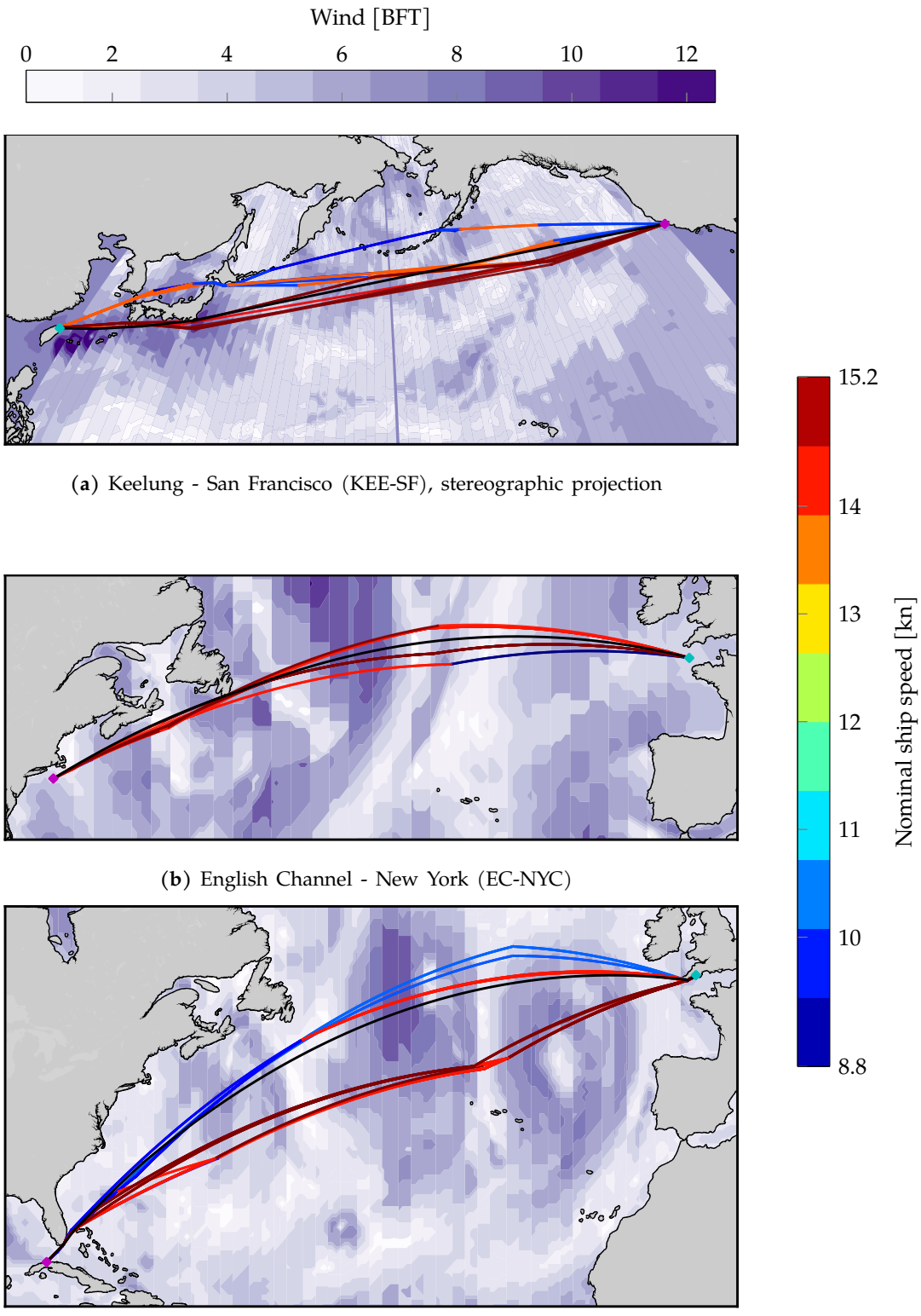

(a) English Channel - New York (EC-NYC)

(b) Plymouth - Havana (PLH-HAV)




(c) Keelung - San Francisco (KEE-SF)

**Figure 5.7.** Computational results for three weather problem instances.
The average involuntary speed change due to weather (top) and the objective values (bottom) for each member of the Pareto front. Results for reference routes are shown in orange. Crosses represent the vertices of the front's convex hull.




(**a**) Keelung - San Francisco (KEE-SF), stereographic projection

(**b**) English Channel - New York (EC-NYC)

(**c**) Plymouth - Havana (PLH-HAV)

**Figure 5.8.** Visualization of weather-optimized routes.
Shown routes represent the convex hull vertices of each problem instance's Pareto front. Weather-optimized routes are colored according to the nominal ship speed at each route leg. Reference routes, neglecting weather effects, are colored black. Each vertical slice corresponds to a six-hour time period of the weather forecast. The number of slices between origin (cyan diamond) and destination (magenta diamond) is the number of 6-hour time periods constituting the average travel time of the solution routes. Thus, the first and last slice correspond to the first and last six hours of the average travel time, respectively.

which is conformal. That means that it preserves angles but does not preserve areas. As a result, the great circle routes are projected as a straight line.

The majority of the routes in the Pareto front corresponding to the weather simulation follow the great circle route, shown in black. Some routes slightly deviate southwards from the great circle route to avoid bad weather near the east coast of Japan. The algorithm found two alternative routes that avoid the bad weather near the origin by navigating west from Japan. As a result, these route options encounter calmer weather areas, and, consequentially, the average speed loss is minimized. This is confirmed by the average speed loss shown Figure 5.7c.

The Pareto set of weather simulation entirely dominates the reference set. The shape of the front corresponding to the former set has some irregularities at the lower right end. Several gaps and corners exist due to a large variety in the Pareto set. Large detours north of Japan yield longer travel times than most reference routes but considerably reduce the fuel cost.

### 5.4.2 English Channel to New York

For the voyage English Channel - New York (EC-NY), a route considering weather effects should avoid the rough weather area developing in the North Atlantic. The heavy weather area is large at the departure date but weakens after a few days, enabling closer navigation to this area. As the bad weather area partly overlaps the great circle route between the English Channel and New York, it may be favorable to slow down before passing the area so that the storm has decreased in strength during the passage.

The routes visualized in Figure 5.8b show the routes with variable engine speed from the convex hull of the Pareto front depicted in the bottom diagram of Figure 5.7a. The algorithm finds two major route trajectories for this problem instance. The first takes a path at the north side of the great circle route. This option contains mainly routes with a high-speed profile. The other decision involves navigating the ship south from the great circle route. The first legs of the routes corresponding to the latter decision have low ship speed and accelerate to the upper end of the speed profile at a later stage.

In the Pareto fronts, we notice a gap in the travel time interval 7.7 d – 8.2 d. As this gap is apparent in both the weather and reference simulation, it is likely caused by the bad weather area in the neighborhood of the sailing region. The gap in the front corresponding to the weather route simulation is larger than the reference front gap.

The corner at the Pareto front indicates the point at which a decision is made between either using two engines or only one. We see a significant reduction in fuel cost for routes with similar travel time in the reference set for the routes on which two engines are used. However, the weather routes lie closer to the reference front for longer travel times but still dominates the reference front entirely.

The average speed loss due to weather effects is largest for the routes with short travel

time. The difference in Pareto fronts is highest for the routes with the largest gap in average speed loss. The weather-optimized routes from this problem instance successfully avoid bad weather and, consequently, avoid large speed reductions.

The routes obtained by the algorithm presented by Shao et al. (2012) deviate south from the great circle route but lie closer to the great circle route than the southern routes found by our algorithm. This is likely due to a difference in ship characteristics and weather data. The weather data used by Shao et al. depicts a large bad weather region in the southern area of the great circle route, whereas this bad weather is less pronounced in the weather data used in this case study.

### 5.4.3 Plymouth to Havana

The third problem instance considers a voyage between Plymouth and Havana (PLM-HAV). The Pareto set corresponding to a single route simulation, with different settings is presented in the bottom diagram of Figure 5.7b. Figure 5.8c shows the routes corresponding to the convex hull of the Pareto front.

Three major routing decisions are made. The first is represented by the blue routes that have low nominal ship speed, i.e., long travel time. They avoid the bad weather area in the North Atlantic by navigating North. We note here that the rough weather shown in orange shifted slightly to the south when the ship approaches this area. A second route option is to take more southern path and cross a small area with strong winds at an early stage, but avoid the larger rough weather area described earlier.

A third major route decision is performed when the routes pass the south side of the heavy weather area. Two alternative routes are considered, of which the first takes the great circle route to the destination while crossing an area with wind speeds of up to 7 BFT. The other alternative avoids the rough weather and chooses to change course at a later stage. The latter route approaches the north coast of the Bahamas, whereas the other routes move near the coast of Florida.

The Pareto front corresponding to the weather simulation dominates the reference Pareto front for a large part. This is because the reference route navigates straight through heavy weather, resulting in a large speed loss. However, the reference set performs better at the extreme end at the lower right corner of the objective space. Reference routes with low fuel costs have significantly shorter travel time than weather-optimized routes and even have an average speed increase of approximately 0.5 kn. This is likely due to the strong winds dying down and an acceleration due to tailwind. The weather routes deviating northwards from the great circle route do not take advantage of this change in weather.

The Pareto set returned by the algorithm presented by Szapczyska (2015) contains similar routes that move southwards from the great circle route and move back towards the great circle route after passing the storm. However, Szapczyska did not find Pareto optimum routes that deviate north of the great circle route. This may be due to a different

setting of ship characteristics, as all north routes found by our algorithm have an equal speed profile.

### 5.4.4 Variable and constant engine speed

Table 5.8 shows the five-run averaged objective values corresponding to the least-time and least-fuel route (i.e., the extreme routes) for each problem instance. Additionally, the table shows the percentage difference with respect to the reference route neglecting ocean currents and difference with respect to a route with constant engine speed. A constant speed profile with the maximum (minimum) ship speed would intuitively yield a route with minimum travel time (fuel cost). Therefore, we select these two constant speed profiles for comparison with the minimum travel time and minimum fuel cost routes (the extreme routes) in the Pareto front obtained by a simulation with variable ship speed.

**Table 5.8.** Least-time and least-fuel route objective values for weather-optimized routes. Least-fuel and least-time routes represent the extreme solutions of the Pareto front and are compared to their reference routes, and corresponding constant speed route. That is, least-fuel vs. fuel-efficient speed $U_{eff}$ and least-time vs. the maximum speed $U_{max}$.

| | | | w.r.t reference | | w.r.t constant speed | |
|---|---|---|---|---|---|---|
| Objective | $\Delta$T[1] [d] | C [kUSD] | $\Delta$T[1] [%] | $\Delta$C[1] [%] | $\Delta$T[1] [%] | $\Delta$C[1] [%] |
| EC-NYC | | | | | | |
| Fuel | 9.2 | 59.7 | 2.6 | −1.8 | −51.6 | −26.7 |
| Time | 6.7 | 77.8 | 0.0 | 0.0 | 0.2 | 0.1 |
| PLY-HAV | | | | | | |
| Fuel | 18.2 | 85.9 | 4.1 | −1.8 | −8.7 | 0.7 |
| Time | 10.1 | 117.5 | 0.1 | 0.0 | −0.1 | −0.3 |
| KEE-SF | | | | | | |
| Fuel | 21.8 | 120.0 | −3.2 | −4.3 | −49.1 | −34.8 |
| Time | 14.4 | 165.1 | −0.3 | −2.1 | 0.6 | −1.2 |

[1] $\Delta = \frac{\text{value}-\text{reference value}}{\text{reference value}} \times 100\%$

For all three simulations, the travel time of the least-time route found by the simulations with variable speed is not significantly different from the travel time of the maximum engine speed $U_{max}$ simulations. That is, for problem instances EC-NYC, PLH-HAV, and KEE-SF, the travel time savings of the variable simulation are −0.2 %, 0.1 %, and −0.6 %, respectively.

The speed profiles of both route simulations are the same. Thus, the slight increase in travel time for two of three problem instances may be caused by the size difference of the Pareto fronts. That is, the Pareto front corresponding to a constant speed simulation contains only a single member, as fuel cost and travel time are directly related in constant speed optimization. Consequently, a constant speed simulation boils down to a single-objective optimization, which is less computationally expensive.

On the other hand, the minimum fuel cost routes corresponding to variable speed simulations are generally better than routes with a constant speed profile. For the least-fuel route of PLH-HAV, a variable engine speed profile yields a 0.7 % fuel cost increase, but a signifi-

cant travel time reduction of 8.7 %. However, the fuel savings for EC-NYC and KEE-SF are much higher, i.e., 26.7 % and 34.8 %, respectively. These large savings are likely caused by a large speed decrease due to heavy weather. Due to the wind direction relative to the ship's heading, the involuntary speed reduction is significantly higher for the problem instances EC-NYC and KEE-SF compared to PLH-HAV. Namely, for the $U_{eff}$ routes in EC-NYC and KEE-SF the ship speed is reduced by 14.4 % and 30.9 %, respectively, whereas this is 5.7 % for PLH-HAV. A large involuntary speed reduction causes a significantly longer travel time and, consequently, an increase in fuel cost.

Overall, the results for the instances presented above show that there is a significant savings potential when including weather conditions into the optimization of ship routes. The algorithm typically avoids areas with wind speeds greater than 6 kn – 7 kn. Minimal time routes sometimes choose to cross heavy weather with full speed or take a small detour. The greatest reductions in fuel cost are achieved for the routes with low ship speeds. Table 5.8 confirms this, showing fuel savings of up to 4.3 %. This has likely two causes:

1. Reference routes with low ship speeds stay for a longer period in rough weather, yielding a larger negative impact compared to routes with higher ship speeds. On the other hand, routes with low ship speeds can take advantage of weather effects that slightly increase the ship speed for a longer time period. However, ship speed loss has a more pronounced effect than ship speed increase.

2. A longer travel time creates more routing possibilities, such as avoiding rough weather areas or slowing down to wait for calmer weather.

## 5.5 High-cost areas

This section presents the algorithm capabilities of routing through high-cost areas, while *excluding* any environmental conditions. Two different types of high-cost areas are considered: (1) ECAs, in which the fuel cost rate is higher compared to regions outside ECAs, and (2) shallow waters that impose ship draft limitations. These high-cost areas differ in their characteristics as ECAs affect the final objective values, whereas shallow waters do not. Each route segment that intersects an area with shallow water is penalized during optimization, but the final travel time and fuel cost of a route are not affected by sailing through these areas. The remainder of this section presents the findings of the case studies.

### 5.5.1 Emission Control Areas

To determine the effect of ECAs on the presented algorithm's route decisions, we create a problem instance based on problem instances used by Fagerholt et al. (2015), who researched the economics of ship routing through ECAs. In the problem instance, we define a voyage with departure at Florø, Norway, and Santander, Spain, as the destination. We choose an ECA multiplier $f_{\text{eca}} = 1.5933$, to represent the fuel price differential used by Fagerholt et al. In other words, the fuel price is 477.99 USD/t within ECAs and 300.00 USD/t

outside. The results are compared to results from a reference scenario, excluding the ECA price differential, i.e., $f_{\text{eca},R} = 1$. Figure 5.9 shows the Pareto fronts of both scenarios and a visualization of their routes.

Figure 5.9c shows that the routes excluding ECAs follow the shortest path. Because the routes follow approximately the same path, only the top layered routes are visible, i.e., the routes with a low-speed profile (blue). The main difference between the routes in the Pareto set of the reference simulation is the engine speed profile. The fastest route attains the maximum engine speed at each leg, while the most economical route chooses the fuel-efficient engine speed. The latter speed is, in this case, the minimum engine speed.

The routes considering ECAs follow three different paths, as shown in Figure 5.9d. These major paths are characterized by the following:

**R1:** the routes following the shortest path. Expectedly, these routes have the least travel time.

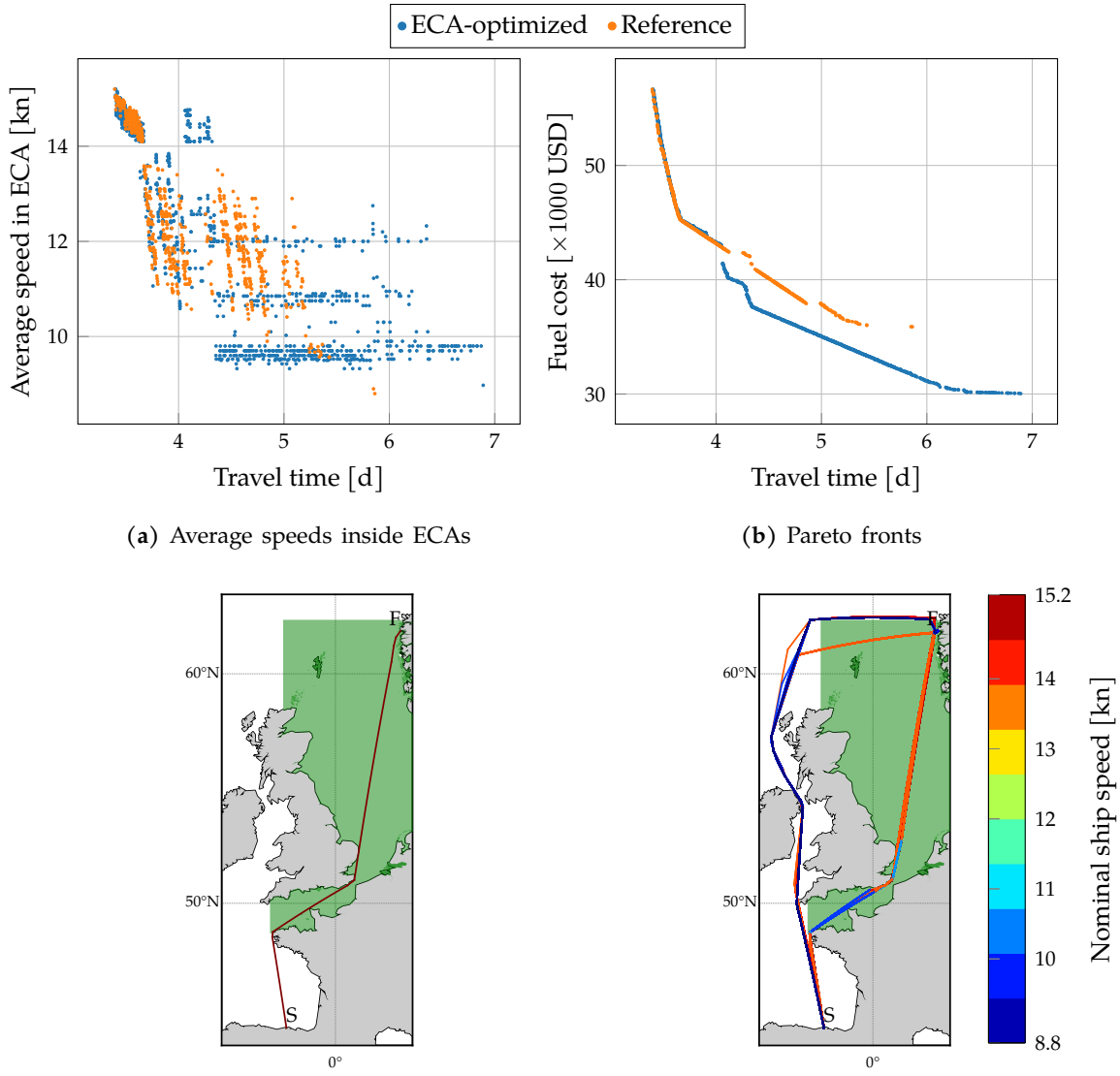
**R2:** the routes that partly avoid the ECA by cutting the corner and joining the other economical routes when exiting the ECA.

**R3:** the most economical routes. These routes try to avoid the ECA entirely by heading up north following the ECA border before going south between the landmasses of the United Kingdom.

Figure 5.9b shows Pareto fronts of both simulations. The Pareto front of the reference simulation has one distinct corner, while the front corresponding to the routes considering ECAs have multiple corners. Each corner represents either a decision in the number of engines used or a decision to follow a different route trajectory. Both fronts have a corner at (3.7 d, 45.3 kUSD) due to switching in the number of engines used. For routes with travel times less than 4.04 days, both Pareto fronts are approximately similar. These routes represent the routes that follow the shortest paths (R1). The routes with travel times in the interval $[4.05, 4.28]$ correspond to R2. The next corner is at point (4.28 d, 39.37 kUSD), representing the decision to follow either R2 or R3.

Figure 5.9a shows the average speed inside the ECA, the so-called ECA speed. Expectedly, this ECA speed is highest for both ECA-optimized and reference routes with short travel times. As the travel time increases, there seem to be roughly three selected ECA speeds for the ECA-optimized, while the reference routes do not have distinctive values for ECA speeds. Specifically for routes with low fuel consumption, the ECA speed is generally lowest for ECA-optimized routes. Thus, to attain the same travel time as the reference route, an ECA-optimized route slows down within ECAs and accelerates outside these areas if fuel consumption is minimized.

Fagerholt et al. studied this same problem instance. They selected three route options from Florø to Santander. One taking the shortest path, another taking the most economical path, and a third with balanced travel time and fuel cost. The trajectories R1 and R3 are




(**a**) Average speeds inside ECAs

(**b**) Pareto fronts

(**c**) Routes excluding ECAs

(**d**) Routes including ECAs

**Figure 5.9.** Results for ECA-optimized routes.
Results are compared to reference routes excluding additional costs due to ECAs. Departure and destination are Florø, Norway, (F) and Santander, Spain, (S), respectively. The ECA in consideration is marked green. NB: Many routes roughly follow the same path. Consequently, colored legs represent the route leg speeds of the top-layered routes, i.e., routes with low fuel cost.

**Table 5.9.** Comparison of lengths of similar routes found by Fagerholt et al. (2015). Besides the total route length, the length inside and outside the ECA is given.

| | ECA | | Non-ECA | | Total | |
|---|---|---|---|---|---|---|
| Route option | [nmi] | $\Delta^1$ [%] | [nmi] | $\Delta^1$ [%] | [nm] | $\Delta^1$ [%] |
| R1 | 936 | 1.0 | 304 | −16.4 | 1240 | −4.1 |
| R2 | 316 | 3.0 | 1076 | 5.3 | 1392 | 4.7 |
| R3 | 41 | 20.9 | 1414 | −0.8 | 1455 | 0.3 |

[1] $\Delta = \frac{\text{value}-\text{reference value}}{\text{reference value}} \times 100\%$

similar to the fastest and most economical routes from Fagerholt et al., respectively. The balanced route from the considered article is similar to R2 but differs at the first route as it passes the Shetland Islands at the south, while R2 passes the islands north.

Table 5.9 lists comparable results from this case study and the study from Fagerholt et al. For all three route options, the presented algorithm finds routes within 5 % difference in length. The route length inside the ECA is approximately similar for R1 and R2, but significantly longer for R3. The latter is because Fagerholt et al. manually select the shortest route to the ECA border, whereas our approach finds an approximation of the shortest route. Together with a small distance (41 nmi), this explains the large difference of 20 %. Furthermore, the route length of R1 outside the ECA is much shorter (by 16.4 %) than the similar route option found by Fagerholt et al. As our approach finds routes near shorelines, the route lengths are reduced by 'cutting corners' at, for example, the west point of Normandy. Again, the Non-ECA route length for R1 is shortest; hence, relative differences are expectedly higher.

From the result of this problem instance, we conclude that the presented approach is able to find different route alternatives that have similar characteristics to route manually selected by hand. Most noteworthy is that the approach is able to find route option R2, which has a balance between longer travel time and additional fuel cost by partly avoiding the ECA.

### 5.5.2 Shallow waters

The presented algorithm may take additional high-cost areas in which penalties are incurred but do not affect the final objective values. This demonstration provides experimental results for routes that avoid such areas. We choose areas with less than 200 m of water depth, representing restricted areas due to ship draft limitations. We note here that restricting routes to a 200 m water depth does not provide realistic routes. However, the purpose of this demonstration is to show the effect of taken into account areas with water depth limitations. In addition, it aims to demonstrate the capability of the routing algorithm to find alternative routes if an initial route passes a major canal, e.g., the Panama Canal.

For this instance, we select the port of Veracruz, Mexico, and Concepcion, Chile, as origin and destination, respectively. Figure 5.10 shows four routes. Each route represents a

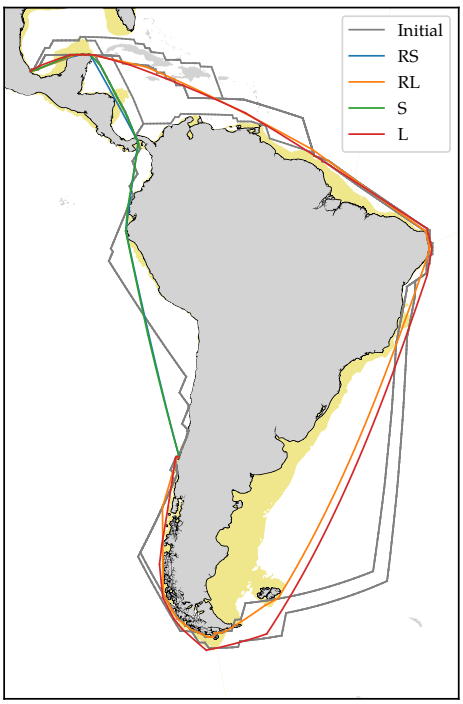



**Figure 5.10.** Illustration of routing through major shipping canals and avoiding shallow waters.
S and L denote short and long route options (i.e., pass and avoid Panama Canal). R denotes a route disregarding shallow areas. Initial paths found on the graph are shown in gray.

combination of two decisions. A first decision is made to avoid shallow waters or ignore these areas. The latter option is denoted by R. The second decision is to pass the Panama canal, i.e., take the shortest route (S) or detour to avoid the canal fees, i.e., the long route (L). Shallow waters with less than 200 m of water depth are highlighted yellow in Figure 5.10.

The routes penalized for intersecting shallow water areas presented in the figure avoid these areas in most cases. However, near the origin, both routes L and S intersect the first encountered shallow water area. This is probably due to the additional costs of avoiding this area, rather than the algorithm not considering a detour around the area. That is because, apart from the shortest path and the path avoiding ECAs, the initialization procedure finds routes on the geodesic graph, avoiding shallow areas where possible. As a result, different routes avoiding and passing shallow areas are adopted at an early stage of the execution.

## 5.6 Multi-objective evolutionary algorithm performance

This section provides the performance results of the solution approach tested using three different MOEAs separately. The performance is measured according to the computation time per route simulation, and the quality of the approximated Pareto front is measured with performance metrics described in Section 4.2. These Pareto performance metrics are denoted by

$\mathcal{C}$: the coverage ratio,

$HVR_2$**:** the binary hypervolume ratio, and

$HVR_3$**:** the ternary hypervolume ratio.

Three problem instances are tested for which 50 route simulations are performed per tested MOEA:

**Current:** Salvador, Brazil to Paramaribo, Suriname with departure date and time 25 November 2014, 0z, considering ocean currents.

**Weather:** Plymouth, England, to Havana, Cuba with departure date and time 24 September 2013, 12z, considering weather.

**ECA:** Rotterdam, Netherlands, to Houston, USA, considering ECAs but neglecting any environmental condition effects.

We use a population size of 100 and run each simulation until 21,000 route evaluations are performed. Since the population size is small, it is not likely that the true Pareto front will be found. Nevertheless, it will give a clear indication of the convergence speed and the distribution of the approximated Pareto front in the early stage of the algorithm execution.

To provide a fair comparison between the three MOEAs, first, Section 5.6.1 discusses the minimization of the number of excess evaluations for each simulation. Next, Section 5.6.2 discusses the computational results on the Pareto performance for each problem instance. Finally, Section 5.6.3 compares the computational times for each MOEA set-up and gives a general note on the computation time of the presented approach.

### 5.6.1 Overshoot of number of evaluations

Considering the expected difference in the number of (excess) evaluations between each tested algorithm, we derive the expected overshoot in the number of evaluations for each tested MOEA. This is not a standard procedure implemented by studies on MOEAs, but may be necessary to provide a fair comparison if the overshoots are large. The overshoot $\mathcal{E}$ is defined as the number of evaluations exceeding the threshold of 21,000 solution evaluations.

SPEA2 creates a new population filled with newly created individuals in every generation. Thus, for an evaluation limit of 21,000 and population size 100, a total of 210 generations is created. Since the entire population is evaluated once per generation, SPEA2 has zero overshoot, i.e.,

$$E[\mathcal{E}_{\text{SPEA2}}] = 0. \tag{5.6}$$

For NSGA-II, the expectation of the number of evaluations is dependent on the crossover probability $pb_{cr}$ and mutation probability $pb_{mut}$. The expected number of evaluations per generation $E[evals_{\text{NSGA-II}}]$ for the parameter settings as described in Section 5.1 can be ex-

pressed as

$$E[evals_{\text{NSGA-II}}] = \left(1 - \frac{1}{2}(1 - pb_{cr})(1 - pb_{mut})\right) N, \tag{5.7}$$

$$= \left(1 - \frac{1}{2}(1 - 0.81)(1 - 0.198)\right) 100 = 93.16, \tag{5.8}$$

where $N$ denotes the population size, $pb_{cr}$ the crossover probability, and $pb_{mut}$ the mutation probability. Then, the expected overshoot for NSGA-II for an evaluation limit of 21,000 is given by

$$E[\mathcal{E}_{\text{NSGA-II}}] = 21,000 \mod 93.16 = 39. \tag{5.9}$$

For M-PAES, the number of evaluations per generation has a different order of magnitude, as its mutation and recombination procedures differ significantly from the procedures of the other MOEAs. The number of evaluations per generation is equal to the number of moves per generation. Moves are performed in the PAES and recombination procedures:

- in the PAES procedure, the worst-case number of moves is expressed by $l_{opt}(l_{fails} - 1)$, where $l_{opt}$ and $l_{fails}$ are parameters described in Section 4.1.3, and
- in the recombination procedure, a maximum of $cr_{trails}N$ moves is performed to fill the new population.

As a result, the worst-case number of evaluations per generation $WC[evals_{\text{M-PAES}}]$ for the parameter settings as described in Section 5.1 is given by

$$WC[evals_{\text{M-PAES}}] = (l_{opt}(l_{fails} - 1) + cr_{trials})\, N, \tag{5.10}$$

$$= ((5(3 - 1) + 5)\, 100 = 1500, \tag{5.11}$$

where $N$ denotes the population size. Several experiments showed that the number of evaluations per generation is approximately the worst-case number divided by two. Hence, by choosing an evaluation limit of 21,000, the expected overshoot for M-PAES is approximately zero, i.e.,

$$E[\mathcal{E}_{\text{M-PAES}}] \approx 21,000 \mod \frac{1500}{2} = 0. \tag{5.12}$$

The average number of excess evaluations, i.e., overshoot, is listed in Table 5.10 for each problem instance and tested MOEA. Still, M-PAES has a relatively large overshoot of roughly 1.5 %. As a result, M-PAES may attain better Pareto fronts, compared to SPEA2 and NSGA-II, as well as an increase in computation time. For the interpretation of the computational results, we must take this into consideration.

**Table 5.10.** Average overshoot for each problem instance and tested MOEA.

| | Current | Weather | ECA | Average |
|---|---|---|---|---|
| M-PAES | 333.4 | 295.2 | 312.7 | 313.8 |
| NSGA-II | 42.3 | 39.6 | 44.2 | 42.0 |
| SPEA2 | 0.0 | 0.0 | 0.0 | 0.0 |

### 5.6.2 Pareto performance

For each problem instance, Figure 5.11 shows box plots of the computational results in terms of Pareto performance metrics and computation time. For the $\mathcal{C}$ metric, results are shown using violin plots, because $\mathcal{C}$ typically returns values near 0 or 1. The binary performance metrics, $\mathcal{C}$ and $HVR_2$, take an approximation and reference Pareto set pair, i.e., $(Z, Z')$. All six possible combinations are presented, as a different order of input sets may yield different scores, i.e., $\mathcal{I}(Z, Z') \neq \mathcal{I}(Z', Z)$. In the remainder of this section, we denote an arbitrary Pareto approximation set obtained by M-PAES, NSGA-II, and SPEA2 by $M$, $N$, and $S$, respectively.

*Coverage ratio*

The $\mathcal{C}$ metric gives an indication of the cardinality and dominance relations for two Pareto sets. A high score for the $\mathcal{C}$ metric indicates that many solutions in $Z'$ are (weakly) dominated by at least one solution of $Z$.

There is a significant difference in the first two instances, Current and Weather, compared to the third instance, ECA. The scores in the first two problem instances tend to take a value near the extreme values of the interval $(0, 1)$, while ECA instance scores are distributed over a larger part of the interval. As earlier discussed in Section 4.2, any pair of $\mathcal{C}$ metric scores for a pair $(Z, Z')$ in which neither $\mathcal{C}(Z, Z') = 1$ nor $\mathcal{C}(Z', Z) = 1$, indicates that the two sets might be incomparable due to cyclic dominance relations. Therefore, the $\mathcal{C}$ might provide unambiguous measures for the performance of the algorithms on the ECA instance. However, for all cases, the score for $S$ never takes a value near 1, indicating worse performance than the other two algorithms regarding this metric.

For the first two problem instances, there is a wide spread in the $\mathcal{C}$ scores for all pairs, indicating that there is no algorithm that structurally (weakly) dominates the other. This is explained by the fact that $\mathcal{C}$ metric takes a value near 1 (0) if $Z$ mostly weakly dominates (is dominated by) $Z'$.

Regarding this metric, M-PAES seems to perform better than NSGA-II, as $\mathcal{C}(M, N) > 0.5$ and $\mathcal{C}(N, M) < 0.5$ for all instances. On average, $M$ contains more solutions that weakly dominate $N$, and vice versa. For the comparisons regarding SPEA2, $\mathcal{C}$ seems to provide worse performance compared to the other algorithms. Although, it is true that for some cases, the median of SPEA2's coverage ratio is large, such that $\mathcal{C}(S, Z') > \mathcal{C}(Z', S)$, for $M, N \in Z'$, the score for SPEA2 tends to be 0 rather than take a value near 1. Especially on the ECA instance, SPEA2 performs worse than every other algorithm, as mentioned earlier. This may be caused by a difference in the cardinality of $S$, compared to the cardinalities of both $M$ and $N$, such that $|S| < |M|$ and $|S| < |N|$.

*Hypervolume ratios*

The ternary ($HVR_3$) and binary ($HVR_2$) hypervolume ratios indicate the proportion of the hypervolume in the objective space covered by the union of all input sets. The median of

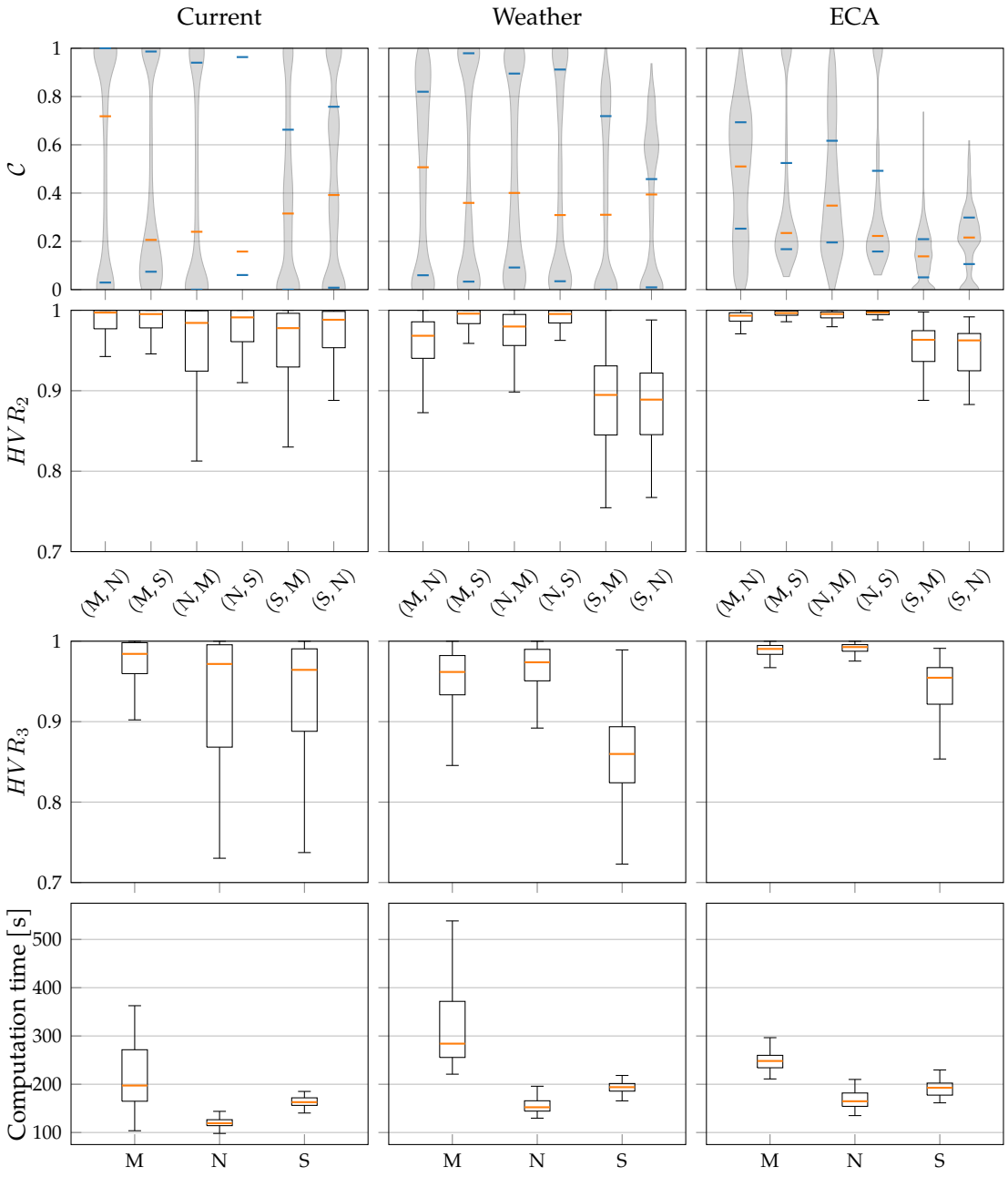



**Figure 5.11.** Computational results for three tested MOEAs.
M, N, and S denote M-PAES, NSGA-II, and SPEA2, respectively. For each algorithm, 50 simulations are performed per problem instance (Current, Weather, and ECA). Orange horizontal lines depict the median, and black horizontal lines mark the first $(Q1)$ and third $(Q3)$ quartiles. For the box plots, the lower whisker is at the lowest value above $Q1 - 1.5(Q3 - Q1)$, and the upper whisker at the highest value below $Q3 + 1.5(Q3 - Q1)$.

$HVR_3$ corresponding to the ocean current simulations is highest for the M-PAES algorithm, with NSGA-II taking a close second place. However, for the Weather and ECA instance, the NSGA-II has a slightly better median score for $HVR_3$ than M-PAES. This means that the ternary hypervolume ratios of both algorithms are of comparable magnitude in many cases. For all instances, SPEA2 has by far the lowest median but still has a score of more than 0.85 in most cases.

The binary hypervolume ratios provide similar results and give a more detailed comparison between a pair of algorithms. The results for $HVR_2$ confirm the large difference between SPEA2 and both M-PAES and NSGA-II Furthermore, M-PAES performs better in the Weather instance compared to NSGA-II but performs slightly worse in the remaining instances. Both ratios $HV_2(M, N)$ and $HV_2(M, N)$ have large overlapping interquartile ranges (IQRs), indicating that neither M-PAES nor NSGA-II structurally covers a larger hypervolume than the other.

Overall, NSGA-II yields the largest hypervolume ratio on average for both the ternary as well as the binary metric. And, both M-PAES and NSGA-II outperform SPEA2 regarding hypervolumes. In other words, the Pareto set of NSGA-II dominates the largest area in the objective space, and M-PAES is a close second.

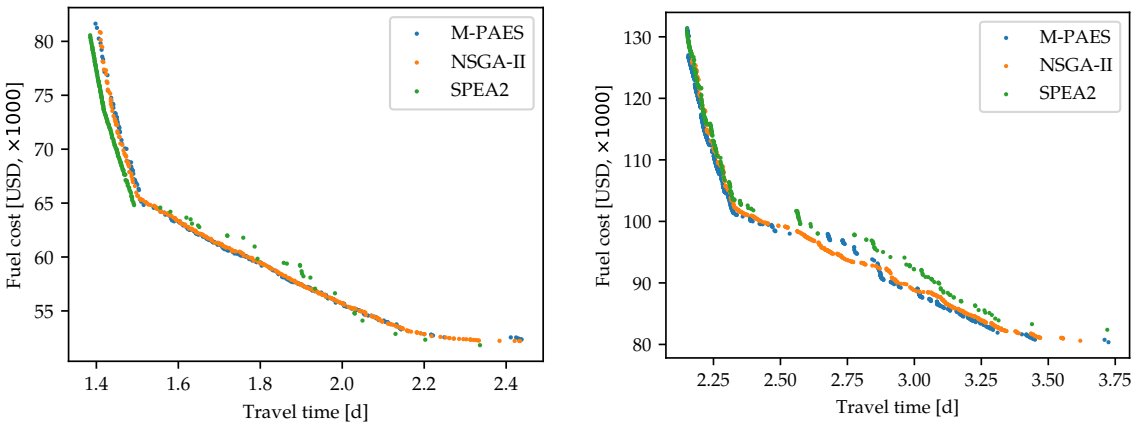
The results from all three problem instances indicate that SPEA2 has slightly worse performance than the other two algorithms. M-PAES and NSGA-II generated similar results, and no clear winner can be appointed regarding the performance metrics.

However, it should be noted that the number of evaluations might be different for the three algorithms, as described earlier in this section. M-PAES performs approximately 21,314 evaluations, while SPEA2 evaluates exactly 21,000 solutions during the simulation. The difference in the performance of SPEA2 compared to NSGA-II and M-PAES could be partially explained by the difference in the total number of evaluations. It is, however, a complicated task to quantify this effect.

Figure 5.12 shows Pareto front comparisons for the ocean current and weather routing simulations that are, to some extent, representative for the numerous simulations. The fronts corresponding to SPEA2 have slightly lower cardinality than the fronts of both M-PAES and NSGA-II. This difference in cardinality contributes to the difference in the $\mathcal{C}$ scores. Furthermore, the Pareto sets of NSGA-II and M-PAES do not clearly dominate one another. On the weather route simulation, NSGA-II performs slightly better at longer travel times, while M-PAES finds a wider spread in extreme points.

### 5.6.3 Computational performance

For all three problem instances, NSGA-II has the shortest computation time, and M-PAES the highest. The computation times of NSGA-II, SPEA2, and M-PAES, are 146.9 s, 182.9 s, and 264.8 s, respectively, on average for all three instances. Hence, for this set-up, NSGA-II is fastest, and SPEA2 (M-PAES) has 24.5 % (80 %) longer computation time on average.




(**a**) Current (**b**) Weather

**Figure 5.12.** Example Pareto fronts for three multi-objective evolutionary algorithms. Problem instances Current and Weather are described in the text.

Although there is a small difference in the total number of evaluations performed by each algorithm (see Table 5.10), it is presumably not the cause of the difference in computation time. That is, the difference in the number of evaluations is at most 1.5 %, whereas the difference in computation times is much higher. Thus, we conclude NSGA-II is the fastest algorithm, despite a small difference in the number of evaluations.

We summarize our findings from the performance assessment of the three MOEAs with the following:

- M-PAES performs best regarding the $\mathcal{C}$ metric; NSGA-II takes a close second place.
- For both hypervolume metrics, NSGA-II yields the highest average score. M-PAES finds slightly lower values, and SPEA2 performs significantly worse.
- NSGA-II yields the shortest computation time of 146.9 s on average. SPEA2 and M-PAES have, on average, 24.5 % and 80 % longer computation times, respectively.

NSGA-II has the shortest computation time and performs approximately equally well to M-PAES concerning the performance metrics. Thus, NSGA-II is best suited for solving the MOSWR problem, compared to M-PAES and SPEA2.

# 6 Conclusions and recommendations for further work

Following upon our findings presented in the previous chapter, this chapter provides conclusions and recommendations for further research. First, Section 6.1 concludes our work and answers the research questions previously stated in Chapter 1. Then, recommendations for further research are provided in Section 6.2.

## 6.1 Conclusions

This thesis aims to answer the following research question:

> *Can we develop a solution approach for the Multi-Objective Ship Weather Route problem which considers different routing aspects and has better performance than existing ship routing methods?*

To answer this question, first, a formal definition of the MOSWR problem is given and, secondly, an evolutionary approach to solving this problem is presented. We mathematically formulated the MOSWR problem such that it is generic for a variety of objectives and a selection of environmental conditions, such as weather and ocean currents. A generic formulation provides a flexible framework for ship routing to various ship types and environmental conditions. The objectives considered are a ship route's travel time, and total fuel cost, depicting opposing criteria, as improving one deteriorates the other. This suggests an optimization that is able to identify Pareto optimum routes. The presented solution approach uses an evolutionary framework to identify a set of Pareto optimum routes in a single run. It handles multiple objectives separately, as aggregating opposing objectives into a single objective function becomes difficult to comprehend. Besides multi-objective optimization, the presented approach has the following main features:

**Versatile initialization procedure** Find a diverse initial population of routes on a variable density graph. The procedure provides the flexibility of a graph search approach, while the final solutions are not subjected to a discrete graph. That is, it can obtain a variety of initial routes by manually (dis)regarding certain waterways or special areas and, subsequently, feeding the initial routes to the continuous evolutionary stage.

**Continuous optimization** Find a set of smooth routes in continuous space that are not restricted to a set of predefined paths or an arbitrary grid.

**Variable engine speed profile** Enable a variable engine speed during along the route, allowing for, e.g., slow-steaming within high-cost areas, or avoiding adverse environmental conditions.

**Ocean current** Consider the effect of ocean current on the ship's speed and direction using historical ocean current information.

**Weather** Include the effect of weather on the ship speed using a semi-empirical method (Kwon, 2008) and weather information based on either historical data or weather forecasts.

**Emission Control Areas** Take into account additional costs in high-cost areas, such as Emission Control Areas.

**Shallow waters** Find routes that avoid shallow waters.

**Shipping canals** Find routes passing or avoiding major shipping canals, e.g., the Panama and Suez Canal.

**Spatial indexing** Significantly reduce computation time by applying a fast line intersection method based on R-tree spatial indexing.

We formulated five sub questions to provide a substantiated answer to the main research question. In the following, we answer each sub-question separately.

*SQ1. What is the effect of ocean currents on the set of solutions to the MOSWR problem?*

The value of considering ocean current effects in ship navigation is widely acknowledged in ship routing literature. McCord et al. (1999) estimated that the commercial shipping industry could save about 1 % in annual fuel costs by ocean current routing. In regions with strong western boundary currents (WBCs), the potential fuel savings are even higher.

For the Gulf Stream region, a case study considering time-varying ocean currents shows that the presented approach finds fuel savings of up to 7.8 % and travel time reductions of up to 3.2 % for ships with a nominal speed of 15.2 knots. This includes most tankers, bulk carriers, and freighters.

In a second case study considering the Kuroshio Current, our approach is tested against the exact approach presented by Tanaka and Kobayashi (2019). It turns out that our approach finds good solutions for routes with less fuel consumption but slightly worse solutions for routes with less travel time. This was to be expected as our metaheuristic approach was not guaranteed to outperform an exact approach. However, the presented approach finds smoother routes and proves to be more versatile for different navigation areas and constraints. Moreover, the total time required to find a set of Pareto optimal routes is much shorter, making it more suitable for practical applications.

Other WBCs such as the Brazil, Agulhas, and East Australian currents are similar to the Gulf Stream and Kuroshio in strength and dynamic activity (Chang et al., 2015). According to accumulated ship tracking data, almost all global ship routes pass one or more of these WBCs (Kiln, 2012). Therefore, the potential benefit of considering ocean currents in ship routing is apparent.

*SQ2. What is the effect of weather on the set of solutions to the MOSWR problem?*

Expectedly, in calm weather, the shortest route is the optimum one. In a rough weather situation, the routing decision becomes more complex as the shortest route is no longer the best one. The optimum becomes a trade-off between (1) additional fuel costs due to ship speed loss and (2) fuel savings by circumnavigation. In general, we find that areas with wind speeds greater than 6 kn – 7 kn are avoided.

For the rough weather scenarios considered in this study, it turns out that applying optimization yields a fuel cost reduction of up to 4.3 % or a 3.2 % shorter travel time. The greatest reductions are achieved for the routes with low fuel costs.

*SQ3. What is the effect of Emission Control Areas on the set of solutions to the MOSWR problem?*

Stricter controls have been imposed within ECAs to minimize airborne emissions from ships. To comply with these regulations, many ship operators choose to switch to Ultra-Low Sulfur Fuel Oil (ULSFO) when sailing inside ECAs. ULSFO is more expensive than fuel types with higher sulfur content, which can be used outside ECAs. The presented approach handles ECAs in two ways:

1. It finds routes with longer distances that avoid ECAs, reducing the consumption of expensive fuel.
2. Another result is that ships sail at lower speeds within and higher speeds outside ECAs so that less of the more expensive fuel is used and longer travel time is compensated for outside ECAs.

These effects depend on the price spread of the two fuel types and become stronger if the difference increases. The presented approach finds approximately similar routes as presented in (Fagerholt et al., 2015), who manually constructed different route options for testing the effect of ECAs. A comparison with these routes shows that our approach finds plausible results for ECA-optimized routes.

*SQ4. What is the effect of a variable engine speed profile in combination with ocean current, weather effects, or Emission Control Areas on the set of solutions to the MOSWR problem?*

In general, varying the engine speed may significantly reduce fuel costs if one does not seek to find the route with the least time. To minimize travel time, a simulation with the maximum engine speed setting is sufficient. Setting a constant engine speed imposes an additional constraint to the MOSWR problem, leading to faster convergence and providing a similar least-time route returned by the simulation with a variable engine speed profile. Although a constant engine speed setting only returns a single solution in the Pareto front, because in this case, the fuel consumption rate is constant.

Regarding more balanced routes with lower fuel costs and longer travel times, a variable

engine speed setting outperforms constant speed. The presented solution approach varies the engine speed in three scenarios:

1. Inside ECAs, the engine speed is reduced to save fuel and increased outside to compensate for lost travel time.
2. In the case of time-varying ocean currents, slightly increasing the ship speed to catch a favorable current may result in better objectives.
3. Most significant is the effect of enabling variable engine speed for weather optimization: for routes with low fuel costs, the savings in both objectives can be very large.

*SQ5. What is the performance of a selection of multi-objective evolutionary algorithms on solving the MOSWR problem considering different environmental conditions?*

Concerning the demands of the MOSWR problem, it is expected that the objective functions show humps and hollows when environmental constraints govern optimization results. Therefore, the solution method should be able to overcome local optima. With the help of MOEAs, optimization criteria can easily be extended or disregarded, making it a flexible approach. The stochastic nature of the evolutionary algorithms serves well for the simultaneous search of several Pareto optimal solutions over the entire solution space. These favorable characteristics make an MOEA well suited as a framework for our solution approach.

From the literature, we selected three MOEAs to be tested according to their performance in solving the MOSWR problem: (1) M-PAES (Knowles & Corne, 2000), (2) NSGA-II (Deb et al., 2002), and (3) SPEA2 (Zitzler et al., 2001). The performance is measured with metrics designed for multi-objective optimization problems and according to the computation time.

On three different routing problem instances, M-PAES and NSGA-II were able to maintain a better spread of solutions compared to SPEA2 and converge better in the obtained Pareto front. The leading cause is likely due to a better diversity preserving mechanism used in both M-PAES and NSGA-II, as the $\mathcal{C}$ metrics provided structurally lower scores for the Pareto fronts obtained by SPEA2 More importantly, NSGA-II shows to have the shortest computation time among the three MOEAs studied here. While M-PAES finds a Pareto front as good as the one obtained by NSGA-II, its computation time is roughly 50 % longer.

With the properties of an effective diversification mechanism, a fast nondominated sorting procedure, and the least required parameter settings among the MOEAs tested, NSGA-II is best suited to be used in our evolutionary approach to solving the MOSWR problem.

### 6.1.1 Discussion on the computational performance

A frequent requirement for the development of an optimization algorithm is the reduction of computational effort. For evolutionary algorithms, the computational effort within one generation depends on the population size, the number of free variables, and the number of objectives. The free variables of a route represent the route evaluation points for deter-

mining environmental effects and area intersections. The route evaluation points must be tight enough to recognize the pattern of environmental conditions.

During an optimization, tens of thousands of routes are investigated. And each route containing hundreds or thousands of route evaluation points, depending on the route length and whether environmental conditions are considered. Thus, a fast method to evaluate a route's fitness is necessary. For this purpose, the route evaluation is based on the following:

- A fast semi-empirical method to obtain ship speed reduction due to weather (Kwon, 2008).
- A simple to use a polynomial equation to determine the actual ship speed given the ocean current velocity and nominal ship speed.
- Fast access of large environmental data arrays avoiding excess memory using parallelized computing (Dask Development Team, 2016).
- Memoization of expensive calculations to avoid numerous recomputations.
- An improved R-tree spatial indexing technique to test for route segment intersections with impassable or high-cost areas. Compared to conventional line intersection methods, this method provides an additional speed improvement by subdividing large complex geometries.

In the present setup, the whole optimization converges typically within 3 min – 5 min, depending on the route length. The most time-consuming element is the route intersection test followed closely by the determination of dominance relationships used for route selection and updating the Pareto front. There is a strong potential to accelerate the computation time, which we discuss in Section 6.2.

Nevertheless, the reported computation time makes our approach suitable for on-board application and tactical voyage planning in an early stage because it returns multiple route alternatives in a single run. Since the number of returned solutions is large in general, an external method is required to select a single optimal solution according to the decision maker's preferences.

In general, we conclude that the presented approach proves to be a valuable tool for ship routing in the presence of ocean currents, weather, Emission Control Areas, and certain constraints on the navigation area. It provides a set of Pareto optimal routes and offers fully customizable support for multiple optimization criteria, ship characteristics, and both static and dynamic constraints. Compared to other multi-objective approaches to ship weather routing, our approach finds routes with similar objective values that show to be more realistic and are generally found within shorter computation time.

All of the above make the presented approach the best choice for a practical solution method to the Multi-Objective Ship Weather Routing problem. Since it considers several aspects of ship navigation and finds a set of Pareto optimal routes within a reasonable time,

this routing algorithm is a useful tool for tramp trade shipping companies interested in optimizing ocean voyages according to their preferences.

## 6.2 Recommendations for further work

1. A more sophisticated ship performance model is required to calculate the ship responses to different sea conditions. The ship performance can be investigated in three ways: (1) numerical simulation, (2) towing tank experiments, and (3) data analysis obtained from ship trials.

2. The current approach assumes perfect information on environmental conditions. To account for inaccuracies in weather forecasts, one could choose to use ensemble forecasts instead. With ensemble weather forecasts, the robustness of a ship route can be minimized as proposed by Hinnenthal and Clauss (2010).

3. Future research should validate whether the altimetry-derived, historical ocean currents are a good representation of actual, future ocean currents.

4. To navigate near shallow waters, the ocean bathymetry data should be processed such that it defines the maximum allowable water depth for a specific ship.

5. Safety limitations on rolling, slamming, the amount of green water (deck wetness), and other hazards in adverse weather conditions should be considered in future studies. Furthermore, the risk of piracy attacks inside piracy zones can be easily included in the current approach as an additional optimization criterion or constraint.

6. Our evolutionary approach allows for parallelization of route evaluation, which may significantly reduce the overall computation time. For this to work, memoization (i.e., caching of intermediate results) must be performed in an overarching process, such that parallel processes do not perform repetitive computations.

7. As an alternative approach to the GDGG used in the initialization procedure, a visibility graph could return initial routes containing fewer waypoints. As a result, no excess waypoints need to be removed in the early stage of the algorithm to obtain better routes.

   A visibility graph is a graph of intervisible locations, typically for a set of geometries (i.e., impassable areas) in the Euclidean plane. In this case, the visibility graph's arcs should be modified to represent great circle paths on the Earth's surface.

8. Two additional move operators for the route mutation should be tested: (1) merge waypoints and (2) split waypoints. The first operator selects two consecutive waypoints for deletion and inserts a waypoint inside an arbitrary area defined by the two selected waypoints. The split operator is the counter variant, which selects a waypoint for deletion and inserts two waypoints inside an arbitrary area around the selected waypoint. These operators might improve the algorithm's local search and possibly

the convergence to the true Pareto front.

# References


*Admiralty manual of navigation* (Vol. 1). (1987). The Stationery Office.

Antonio, F. (1992). Faster line intersection. In D. Kirk (Ed.), *Graphics gems III* (*IBM version*) (pp. 199–202). Kaufmann, Morgan.

Becker, J. J., Sandwell, D. T., Smith, W. H. F., Braud, J., Binder, B., Depner, J., Fabre, D., Factor, J., Ingalls, S., Kim, S.-H., Ladner, R., Marks, K., Nelson, S., Pharaoh, A., Trimmer, R., Von Rosenberg, J., Wallace, G., & Weatherall, P. (2009). Global bathymetry and elevation data at 30 arc seconds resolution: SRTM30_PLUS. *Marine Geodesy*, *32*(4), 355–371.

Bellman, R. (1952). On the theory of dynamic programming. *Proceedings of the National Academy of Sciences*, *38*(8), 716–719.

Bijlsma, S. J. (1975). *On minimal-time ship routing* (Doctoral dissertation). Delft University of Technology. Delft, The Netherlands.

Bleick, W. E., & Faulkner, F. D. (1965). Minimal-time ship routing. *Journal of Applied Meteorology*, *4*(2), 217–221.

Bosman, P. A., & Thierens, D. (2003). The balance between proximity and diversity in multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, *7*(2), 174–188.

Bowditch, N. (1802). *The American practical navigator*. National Image; Mapping Agency.

Chang, Y.-C., Tseng, R.-S., Chen, G.-Y., Chu, P. C., & Shen, Y.-T. (2013). Ship Routing Utilizing Strong Ocean Currents. *Journal of Navigation*, *66*(6), 825–835.

Chang, Y.-C., Tseng, R.-S., Chu, P. C., & Shao, H.-J. (2015). Global energy-saving map of strong ocean currents. *Journal of Navigation*, *69*(1), 75–92.

Chen, H. (2013). *Voyage optimization versus weather routing* (tech. rep.). Jeppesen Marine Inc. a Boeing Company.

Chu, P. C., Miller, S. E., & Hansen, J. A. (2015). Fuel-saving ship route using the navys ensemble meteorological and oceanic forecasts. *The Journal of Defense Modeling and Simulation*, *12*(1), 41–56.

Collette, Y., & Siarry, P. (2004). *Multiobjective optimization*. Springer.

Dask Development Team. (2016). *Dask: Library for dynamic task scheduling*. Retrieved October 13, 2020, from https://dask.org/

Deb, K. (2000). An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, *186*(2-4), 311–338.

Deb, K., A, P., Agarwal, S., & T, M. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197.

Deb, K., & Goel, T. (2001). A hybrid multi-objective evolutionary approach to engineering shape design. In E. Zitzler, L. Thiele, K. Deb, C. A. Coello Coello, & D. Corne (Eds.), *Evolutionary multi-criterion optimization* (pp. 385–399). Springer.

Fagerholt, K., Gausel, N. T., Rakke, J. G., & Psaraftis, H. N. (2015). Maritime routing and speed optimization with emission control areas. *Transportation Research Part C: Emerging Technologies*, *52*, 57–73.

Fagerholt, K., Laporte, G., & Norstad, I. (2010). Reducing fuel emissions by optimizing speed on shipping routes. *Journal of the Operational Research Society*, *61*(3), 523–529.

Clarkson Research Services Limited. (2015). *The tramp shipping market* (tech. rep.). London, United Kingdom.

Japan Oceanographic Data Center. (2015). *J-DOSS: JODC data on-line service system*. Retrieved October 13, 2020, from http://www.jodc.go.jp/service.htm

National Centers for Environmental Prediction. (2020). *Global forecast system*. Retrieved October 13, 2020, from https://www.ncdc.noaa.gov/data-access/model-data/model-datasets/global-forcast-system-gfs

National Imagery and Mapping Agency. (2000). *World geodetic system 1984* (3rd ed., tech. rep. TR8350.2). Department of Defense. Bethesda, MD.

United Nations Conference on Trade and Development. (2019). *Review of maritime transport 2019* (tech. rep.). United Nations. Geneva, Switzerland.

World Trade Organization. (2019). *World trade statistical review 2019* (tech. rep.). Geneva, Switzerland.

Fonseca, C. M., & Fleming, P. J. (1998). Multiobjective optimization and multiple constraint handling with evolutionary algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, *28*(1), 26–37.

Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A., Parizeau, M., & Gagné, C. (2012). DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, *13*, 2171–2175.

GlobCurrent project. (2015). *The combined geostrophy + Ekman currents*. Retrieved October 13, 2020, from http://products.cersat.fr/details/?id=CLS-L4-CUReul_hs-ALT_SUM-v01.0

Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. *SIGMOD Record*, *14*(2), 47–57.

Hagiwara, H. (1989). *Weather routing of (sail-assisted) motor vessels* (Doctoral dissertation). Delft University of Technology. Delft, The Netherlands.

Hagiwara, H., & Spaans, J. A. (1987). Practical weather routing of sail-assisted motor vessels. *Journal of Navigation*, *40*(1), 96–119.

Haltiner, G., Hamilton, H., & 'Arnason, G. (1962). Minimal-time ship routing. *Journal of Applied Meteorology*, *1*(1), 1–7.

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, *4*(2), 100–107.

Hinnenthal, J., & Clauss, G. (2010). Robust Pareto-optimum routing of ships utilising deterministic and ensemble weather forecasts. *Ships and Offshore Structures*, *5*(2), 105–114.

Holland, J. H. (1992). *Adaptation in natural and artificial systems*. The MIT Press.

Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. *International Conference on Learning and Intelligent Optimization*, 507–523.

Inman, J. (1849). *Navigation and nautical astronomy, for the use of British seamen* (4th ed.). Francis & John Rivington.

James, R. W. (1957). *Application of wave forecasts to marine navigation* (Doctoral dissertation). Washington, DC, U.S. Naval Oceanographic Office.

Jung, J.-S., & Rhyu, K. S. (1999). A study on the optimal navigation route decision using A* algorithm. *Journal of the Korea Society of Computer and Information (in Korean)*, *4*(1), 38–46.

Kahler, A. (2009). *Creating an icosphere mesh in code*. Retrieved October 13, 2020, from http://blog.andreaskahler.com/2009/06/creating-icosphere-mesh-in-code.html

Kelso, N. V., & Patterson, T. (2018). *Global bathymetry* [v4.1.0]. Natural Earth. Retrieved October 13, 2020, from https://www.naturalearthdata.com/downloads/10m-physical-vectors/10m-bathymetry/

Kiln. (2012). *Ship map*. UCL Energy Institute. https://www.shipmap.org/

Klompstra, M. B., Olsder, G. J., & van Brunschot, P. K. (1992). The isopone method in optimal control. *Dynamics and Control*, *2*(3), 281–301.

Knowles, J. D., & Corne, D. W. (1999). The Pareto archived evolution strategy: A new baseline algorithm for Pareto multiobjective optimisation. *Proceedings of the 1999 Congress on Congress on Evolutionary Computation*, *1*, 98–105.

Knowles, J. D., & Corne, D. W. (2000). M-PAES: A memetic algorithm for multiobjective optimization. *Proceedings of the 2000 Congress on Evolutionary Computation, CEC 2000*, *1*, 325–332.

Knowles, J. D., & Corne, D. W. (2002). On metrics for comparing nondominated sets. *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, *1*, 711–716.

Konak, A., Coit, D. W., & Smith, A. E. (2006). Multi-objective optimization using genetic algorithms: A tutorial. *Reliability Engineering & System Safety*, *91*(9), 992–1007.

Kwon, Y. J. (2008). Speed loss due to added resistance in wind and waves. *The Naval Architect - RINA*, 14–16.

Lara, A., Sanchez, G., Coello Coello, C. A., & Schutze, O. (2010). HCS: A new local search strategy for memetic multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, *14*(1), 112–132.

Leutenegger, S. T., Lopez, M. A., & Edgington, J. (1997). STR: A simple and efficient algorithm for R-tree packing. *Proceedings 13th International Conference on Data Engineering*, 497–506.

Li, X., Wang, H., & Wu, Q. (2017). Multi-objective optimization in ship weather routing. *2017 Constructive Nonsmooth Analysis and Related Topics*.

Lo, H. K., & McCord, M. R. (1998). Adaptive ship routing through stochastic ocean currents: General formulations and empirical results. *Transportation Research Part A: Policy and Practice*, *32*(7), 547–561.

Marie, S., & Courteille, E. (2009). Multi-objective optimization of motor vessel route. *The International Journal on Marine Navigation and Safety of Sea Transportation*, *3*(2), 411–418.

*International Convention for the Prevention of Pollution from Ships* (Vol. Annex VI). (2005).

Martí, L., García, J., Berlanga, A., & Molina, J. M. (2016). A stopping criterion for multi-objective optimization evolutionary algorithms. *Information Sciences*, *367–368*, 700–718.

McCord, M. R., Lee, Y.-K., & Lo, H. K. (1999). Ship routing through altimetry-derived ocean currents. *Transportation Science*, *33*(1), 49–67.

Montes, A. A. (2005). *Network shortest path application for optimum track ship routing* (Doctoral dissertation). Naval Postgraduate School. Montery, CA.

Ngatchou, P., Zarei, A., & El-Sharkawi, A. (2005). Pareto multi objective optimization. *Proceedings of the 13th International Conference on, Intelligent Systems Application to Power Systems*, 84–91.

Notteboom, T. E. (2006). The time factor in liner shipping services. *Maritime Economics & Logistics*, *8*(1), 19–39.

Padhy, C. P. (2008). Application of wave model for weather routing of ships in the North Indian Ocean. *Natural Hazards*, *44*(3), 373–385.

Psaraftis, H. N., & Kontovas, C. A. (2013). Speed models for energy-efficient maritime transportation: A taxonomy and survey. *Transportation Research Part C: Emerging Technologies*, *26*, 331–351.

Riquelme, N., Von Lücken, C., & Baran, B. (2015). Performance metrics in multi-objective optimization. *2015 Latin American Computing Conference (CLEI)*, 1–11.

Roh, M.-I. (2013). Determination of an economical shipping route considering the effects of sea state for lower fuel consumption. *International Journal of Naval Architecture and Ocean Engineering*, *5*(2), 246–262.

Rudolph, G. (1994). Convergence analysis of canonical genetic algorithms. *IEEE Transactions on Neural Networks*, *5*(1), 96–101.

Sahr, K., White, D., & Kimerling, A. J. (2003). Geodesic discrete global grid systems. *Cartography and Geographic Information Science*, *30*(2), 121–134.

Sen, D., & Padhy, C. P. (2015). An approach for development of a ship routing algorithm for application in the North Indian Ocean region. *Applied Ocean Research*, *50*, 173–191.

Shao, W., Zhou, P., & Thong, S. K. (2012). Development of a novel forward dynamic programming method for weather routing. *Journal of Marine Science and Technology (Japan)*, *17*(2), 239–251.

Srinivas, N., & Deb, K. (1994). Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, *2*(3), 221–248.

Szapczyska, J. (2007). Multiobjective approach to weather routing. *TransNav, the International Journal on Marine Navigation and Safety of Sea Transportation*, *1*(3), 273–278.

Szapczyska, J. (2015). Multi-objective weather routing with customised criteria and constraints. *Journal of Navigation*, *68*(2), 338–354.

Tanaka, M., & Kobayashi, K. (2019). A route generation algorithm for an optimal fuel routing problem between two single ports. *International Transactions in Operational Research*, *26*(2), 529–550.

van Veldhuizen, D. A. (1999). *Multiobjective evolutionary algorithms: Classifications, analyses, and new innovations* (Doctoral dissertation). Air Force Institute of Technology. Dayton, OH.

de Wit, C. (1990). Proposal for low cost ocean weather routeing. *Journal of Navigation*, *43*(3), 428–439.

Veneti, A., Konstantopoulos, C., & Pantziou, G. (2018). Evolutionary computation for the ship routing problem. In C. Konstantopoulos & G. Pantziou (Eds.), *Modeling, computing and data handling methodologies for maritime transportation* (pp. 95–115). Springer.

Vincenty, T. (1975). Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations. *Survey Review*, *23*(176), 88–93.

Wessel, P., & Smith, W. H. F. (1996). Global self-consistent, hierarchical, high-resolution shoreline database. *Journal of Geophysical Research: Solid Earth*, *101*(B4), 8741–8743.

White, D., Kimerling, A. J., Sahr, K., & Song, L. (1998). Comparing area and shape distortion on polyhedral-based recursive partitions of the sphere. *International Journal of Geographical Information Science*, *12*(8), 805–827.

Yap, P. (2002). Grid-based path-finding. In R. Cohen & B. Spencer (Eds.), *Advances in artificial intelligence* (pp. 44–55). Springer.

Zaccone, R., Ottaviani, E., Figari, M., & Altosole, M. (2018). Ship voyage optimization for safe and energy-efficient navigation: A dynamic programming approach. *Ocean Engineering*, *153*, 215–224.

Zitzler, E., Deb, K., & Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation*, *8*(2), 173–195.

Zitzler, E., Laumanns, M., & Thiele, L. (2001). SPEA2: Improving the strength Pareto evolutionary algorithm. *TIK-report*, *103*.

Zitzler, E., & Thiele, L. (1998). Multiobjective optimization using evolutionary algorithms — a comparative case study. In A. E. Eiben, T. Bäck, M. Schoenauer, & H.-P. Schwefel (Eds.), *Parallel problem solving from nature* (pp. 292–301). Springer.

Zitzler, E., & Thiele, L. (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, *3*(4), 257–271.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., & da Fonseca, V. G. (2003). Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, *7*(2), 117–132.

Zoppoli, R. (1972). Minimum-time routing as an n-stage decision process. *Journal of Applied Meteorology*, *11*(3), 429–435.

Zwillinger, D. (Ed.). (1995). CRC standard mathematical tables and formulae. CRC Press.

# Appendices

# A Vincenty's inverse method

Vincenty's inverse method is an iterative method used in geodesy to calculate the distance between two points on the surface of a spheroid, developed by Vincenty (1975). It is based on the assumption that the figure of the Earth is an oblate spheroid; hence, is more accurate than methods that assume a spherical Earth.

Given the coordinates of the two points $(\phi_1, \lambda_1)$ and $(\phi_2, \lambda_2)$, the inverse problem finds the length of the geodesic $s$ and azimuth between the given points. Algorithm 2 describes the part of Vincenty's inverse method that calculates the length of the geodesic $s$.

We define the following notation:

| | |
|---|---|
| $a$ | major semi-axis of the ellipsoid (3443.918,467 nmi in WGS-84 (National Imagery and Mapping Agency, 2000)). |
| $f = \frac{a-b}{a}$ | flattening of the ellipsoid (1/298.257,223,563 in WGS-84 (National Imagery and Mapping Agency, 2000)). |
| $b$ | minor semi-axis of the ellipsoid. |
| $\phi_1, \phi_2$ | latitude of the points, positive north of the equator. |
| $\lambda_1, \lambda_2$ | longitude of the points, positive east. |
| $L = \lambda_2 - \lambda_1$ | difference in longitude. |
| $s$ | length of the geodesic. |
| $\alpha$ | azimuth of the geodesic at the equator. |
| $\lambda$ | difference in longitude on an auxiliary sphere. |
| $\sigma$ | angular distance on the sphere between the two points. |
| $\sigma_1$ | angular distance on the sphere from the equator to the first point. |
| $\sigma_m$ | angular distance on the sphere from the equator to the midpoint of the line. |

**Algorithm 2** Vincenty's inverse problem

$U_i = (1 - f) \tan \phi_i$, for $i = 1, 2$

$\lambda = L$ ▷ first approximation

**while** the change in $\lambda$ is not negligible **do**

$\sin^2 \sigma = (\cos U_2 \sin \lambda)^2 + (\cos U_1 \sin U_2 - \sin U_1 \cos U_2 \cos \lambda)^2$

$\cos \sigma = \sin U_1 \sin U_2 + \cos U_1 \cos U_2 \cos \lambda$

$\tan \sigma = \sin \sigma / \cos \sigma$

$\sin \alpha = \cos U_1 \cos U_2 \sin \lambda / \sin \sigma$

$\cos 2\sigma_m = \cos \sigma - 2 \sin U_1 \sin U_2 / \cos^2 \alpha$

$C = \frac{f}{16} \cos^2 \alpha [4 + f(4 - 3 \cos^2 \alpha)]$

$L = \lambda - (1 - C) f \sin \alpha \{\sigma + C \sin \sigma [\cos 2\sigma_m + C \cos \sigma (-1 + 2 \cos^2 2\sigma_m)]\}$

$u^2 = \frac{\cos^2 \alpha (a^2 - b^2)}{b^2}$

$A = 1 + \frac{u^2}{256} [64 + u^2 (-12 + 5u^2)]$

$B = \frac{u^2}{512} [128 + u^2 (-64 + 37u^2)]$

$\Delta\sigma = B \sin \sigma [\cos 2\sigma_m + \frac{1}{4} B \cos \sigma (-1 + 2 \cos^2 2\sigma_m)]$

$s = bA(\sigma - \Delta\sigma)$

# B Ship speed reduction coefficients

**Table B.1.** Speed direction reduction coefficient $C_\beta$.
$TWA$ is the true wind angle in degrees with respect to the ship's bow (Kwon, 2008).

| Weather direction | $TWA$ | $C_\beta$ |
|---|---|---|
| Head sea (irregular waves) and wind | 0° to 30° | 1 |
| Bow sea (irregular waves) and wind | 30° to 60° | $0.85 - 0.015(BN - 4)^2$ |
| Beam sea (irregular waves) and wind | 60° to 150° | $0.45 - 0.030(BN - 6)^2$ |
| Following sea (irregular waves) and wind | 150° to 180° | $0.20 - 0.015(BN - 8)^2$ |

**Table B.2.** Speed reduction coefficient $C_U$ (Kwon, 2008).

| Block coefficient $C_b$ | Ship loading conditions | $C_U$ |
|---|---|---|
| 0.55 | normal | $1.7 - 1.4F_n - 7.4F_n^2$ |
| 0.60 | normal | $2.2 - 2.5F_n - 9.7F_n^2$ |
| 0.65 | normal | $2.6 - 3.7F_n - 11.6F_n^2$ |
| 0.70 | normal | $3.1 - 5.3F_n - 12.4F_n^2$ |
| 0.75 | loaded or normal | $2.4 - 10.6F_n - 9.5F_n^2$ |
| 0.80 | loaded or normal | $2.6 - 13.1F_n - 15.1F_n^2$ |
| 0.85 | loaded or normal | $3.1 - 18.7F_n + 28.0F_n^2$ |
| 0.75 | ballast | $2.6 - 12.5F_n - 13.5F_n^2$ |
| 0.80 | ballast | $3.0 - 16.3F_n - 21.6F_n^2$ |
| 0.85 | ballast | $3.4 - 20.9F_n + 31.8F_n^2$ |

**Table B.3.** Hull form coefficient $C_{Form}$ (Kwon, 2008).

| Type of ship | $C_{Form}$ |
|---|---|
| All ships (except container ships) in loaded conditions | $0.5BN + BN^{6.5}/(2.7\nabla^{\frac{2}{3}})$ |
| All ships (except container ships) in ballast conditions | $0.7BN + BN^{6.5}/(2.7\nabla^{\frac{2}{3}})$ |
| Container ships in normal loading conditions | $0.5BN + BN^{6.5}/(22.0\nabla^{\frac{2}{3}})$ |

# C Step-by-step solving for speed over ground

Using $\sin(\arccos x) = \sqrt{1-x^2}$, we can rewrite Equation 3.24 as

$$V_a \sin\beta = V\sqrt{1 - \Big(\frac{V_a \cos\beta - S_N}{V}\Big)^2} + S_E. \tag{C.1}$$

Through subtracting $S_E$ and squaring the result, we can express the above equation as the quadratic equation

$$(\cos^2\beta + \sin^2\beta)V_a^2 - 2(S_E \sin\beta + S_N \cos\beta)V_a + S_E^2 + S_N^2 - V^2 = 0. \tag{C.2}$$

Using $\cos^2 x + \sin^2 x = 1$, we find the equation

$$V_a^2 - 2(S_E \sin\beta + S_N \cos\beta)V_a + S_E^2 + S_N^2 - V^2 = 0. \tag{C.3}$$

Solving the above equation for $V_a$ gives two solutions

$$V_a'' = \frac{2(S_E \sin\beta + S_N \cos\beta) \pm \sqrt{\Big(-2(S_E \sin\beta + S_N \cos\beta)\Big)^2 - 4(S_E^2 + S_N^2 - V^2)}}{2}, \tag{C.4}$$

which we can rewrite to

$$\begin{aligned} V_a'' =& S_E \sin\beta + S_N \cos\beta \\ &\pm \sqrt{V^2 - \Big(S_E^2(1-\sin^2\beta) - 2S_E S_N \sin\beta\cos\beta + S_N^2(1-\cos^2\beta)\Big)}, \end{aligned} \tag{C.5}$$

again, using $\cos^2 x + \sin^2 x = 1$, we find

$$V_a'' = S_E \sin\beta + S_N \cos\beta \pm \sqrt{V^2 - (S_E \cos\beta - S_N \sin\beta)^2}. \tag{C.6}$$

Finally, by taking the maximum of $V_a''$ we find the value for the actual ship speed

$$V_a = S_E \sin\beta + S_N \cos\beta + \sqrt{V^2 - (S_E \cos\beta - S_N \sin\beta)^2}, \tag{C.7}$$

such that there exists a real solution for the square root $\sqrt{V^2 - (S_E \cos\beta - S_N \sin\beta)^2}$.

# D Special case of fundamental theorem of calculus

**Theorem 1.** *Let*

$$F(t) = \begin{cases} x(t) \\ y(t) \end{cases} \quad , t \in [a, b] \in \mathbb{R}, \tag{D.1}$$

*be a smooth curve with continuous first derivative. We split $F$ into several small parts $f_i$, $i = 1, 2, \ldots$. Let the arc length of $f_i$ be $l_i$ and the length of the straight line connecting the two endpoints of $F_i$ be $d_i$. Then, $l_i = d_i + o(d_i)$, where $\lim_{d_i \to 0} 0(d_i) = 0$. In other words, $d_i$ converges to $s_i$ when it becomes infinitely small.*

*Proof.* Let the part of the curve with length $l_i$ start from origin, then we have

$$l_i = \int_0^h \sqrt{\{x'(t)\}^2 + \{y'(t)\}^2} dt, \tag{D.2}$$

and

$$d_i = \sqrt{\{x(h)\}^2 + \{y(h)\}^2}, \tag{D.3}$$

and we need to show that $\lim_{h \to 0} l_i/d_i = 1$. Assuming that one of the derivatives $x'(0)$, $y'(0)$ is non-zero, via the fundamental theorem of calculus we can see that

$$\frac{l_i}{h} \to \sqrt{\{x'(0)\}^2 + \{y'(0)\}^2}, \tag{D.4}$$

and by definition of derivative $d_i/h$ also tends to the same value. ■

# E Subdividing large polygons for faster computation

A method for subdividing large polygons into smaller polygons to a desired size is proposed described in Algorithm 3. First, the polygon is split into two parts across it's shortest dimensions. Ether left to right, or top to bottom. For each part, if it is larger than a certain threshold, it is split again in a similar manner. This is process repeated until all parts have the desired size.

**Algorithm 3** Split polygons recursively

**function** SPLIT POLYGONS($\omega, \epsilon, rec_{max}, i = 0$)
  $box =$ Minimum bounding rectangle of $\omega$
  **if** $box_{area} \leq \epsilon$ or $i = rec_{max}$ **then** ▷ $box$ has desired size, or max recursions reached
    **return** result = [ $\omega$ ]
  **else if** $box_{height} \geq box_{width}$ **then** ▷ Split $box$ left to right
    $a =$ top half of $box$
    $b =$ bottom half of $box$
  **else** ▷ Split $box$ top to bottom
    $a =$ left half of $box$
    $b =$ right half of $box$
  $result = [\,]$
  **for** $d \in [a, b]$ **do** ▷ Split $\omega$ at edges of $a$ and $b$
    $c =$ INTERSECTION($\omega, d$)
    **for** $e \in c$ **do** ▷ Split obtained polygons recursively
      $result = result \cup$ SPLIT POLYGONS($e, \epsilon, rec_{max}, i + 1$)
  **return** $result$

# F Route visualization web-application

For the purpose of visualizing the least-time and least-fuel route given a set of input arguments, we developed a web-application in collaboration with ORTEC, which provides services in optimization software and analytics solutions. The web-application comprises three components:

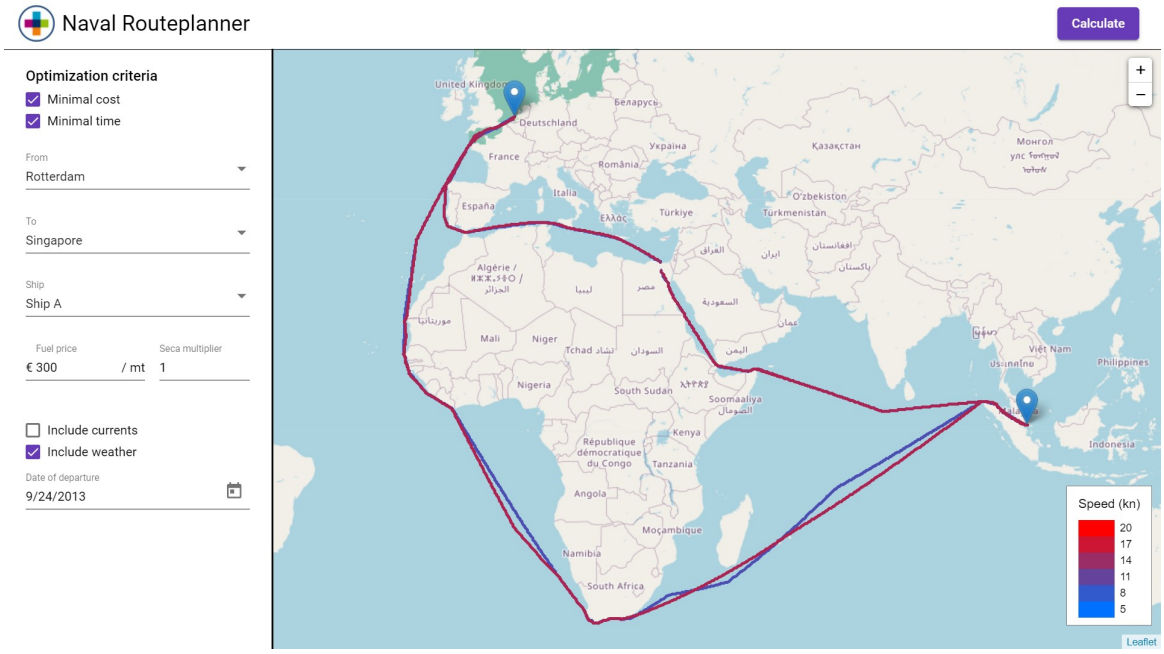
1. The core-component is the routing algorithm presented in this research. For this setup, it takes the following input arguments:

   - Which optimization criteria to include, i.e., minimal cost and/or minimal time.
   - Start and destination coordinates in longitude and latitude degrees.
   - Vessel name.
   - Fuel price per metric tonne outside ECAs and the multiplication factor for fuel price inside ECAs.
   - Whether to include or exclude ocean currents and weather in the optimization.

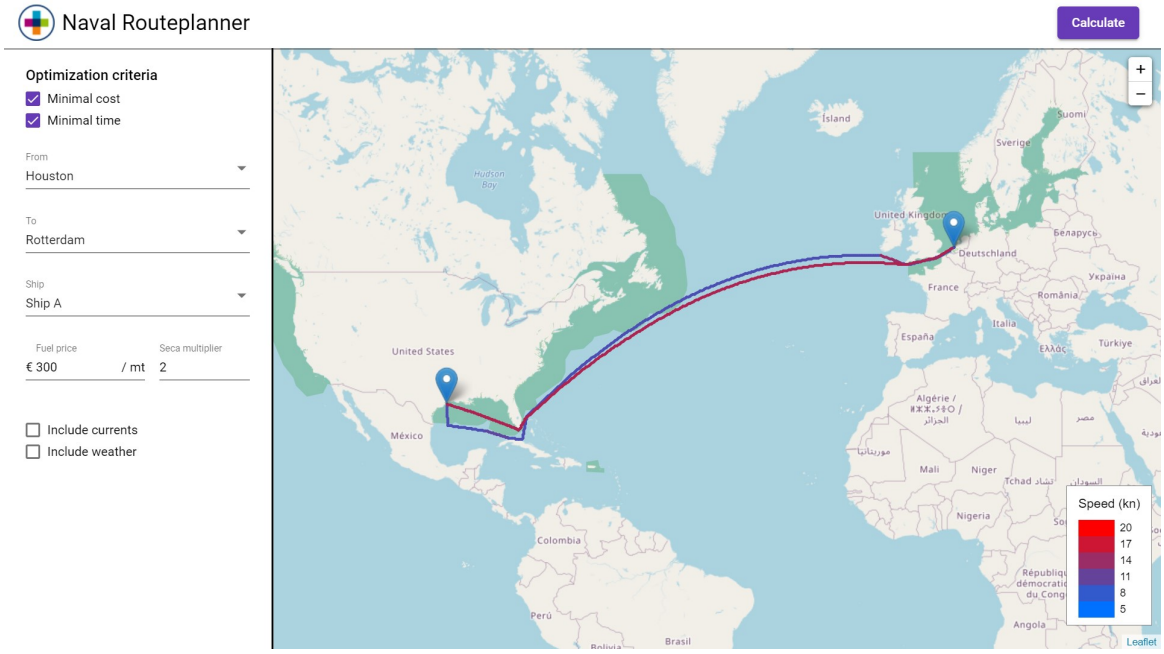
   Given these input arguments, the algorithm returns the least-time and least-cost solutions defined by a series of waypoints and nominal ship speeds. Additionally, included in the output are the distance and optimization criterion of a route.

2. A web user interface (UI) developed in Angular, a platform for building mobile and desktop web applications. Figure F.1 shows screenshots of the UI. The left pane enables selecting input arguments for the route optimization. The map shows the returned routes colored according to the nominal ship speed at each route leg. In general, the least-time route is colored red, i.e., attain the highest speed, and least-fuel route is colored blue as it selects the lowest ship speed for each route leg. ECAs are shown in green.

3. An application programming interface (API) that defines interactions between the routing algorithm and the UI.




(**a**) Sample routes passing and avoiding the Suez Canal, including weather.




(**b**) Sample routes crossing the Atlantic Ocean.

**Figure F.1.** Screenshots of the route visualization user interface.

# G Route planner based on historic routes

This section describes the route planner earlier proposed by ORTEC and shows its drawbacks compared to the presented approach. Based on historic voyages, a set of nodes and arcs are defined throughout the world. The construction of the graph is outlined as follows:

1. Historical voyages are split into smaller arcs, resulting in multiple sets of similar arcs representing a route segment.
2. From a set of arcs with equal endpoints, the arc with median length is selected. As ar results, for every route segment, an arc is selected that gives a good representation of the sailing duration of the route segment and that is less sensitive to outliers.
3. The set of nodes contains
   (a) a set of historically visited ports,
   (b) other ports are connected to the network by adding the nearest waypoint to these ports as nodes to the network, and
   (c) a manually defined set of route congestion points referred to as corridors. These corridors are formulated as a geographical line between two points, $\mathbf{c}_A$ and $\mathbf{c}_B$, such that the ship route crosses the straight line between $\mathbf{c}_A$ and $\mathbf{c}_B$ if the corridor is selected in the route.
4. For all the historic voyages, a start port, an end port, and several subsequent corridors (if any) it surpassed, is derived.
5. Based on this, a set of arcs is derived, which is checked to be fully connected.

   In addition, for each arc is checked whether it surpasses an ECA in which only Ultra-Low Sulfur Fuel Oil (ULSFO) is allowed and whether it is only used during a specific season. Seasonality is taken into account in the route planning by restricting the routes that are not historically sailed due to seasonal constraints in a specific time period. This means that in some cases, the route in the summer period may be different from the winter season.
6. At this point, we have a complete network of nodes and arcs. It can be used to find routes between any two ports in the world using Dijkstras shortest path algorithm.

Figure G.1 shows the network of nodes and arcs constructed using the procedure described above. Five sample shortest routes are highlighted in red. This approach provides a fast estimate of the route's length and fuel cost, given a start and destination. For voyages that have been sailed frequently, this approach provides realistic results. However, for

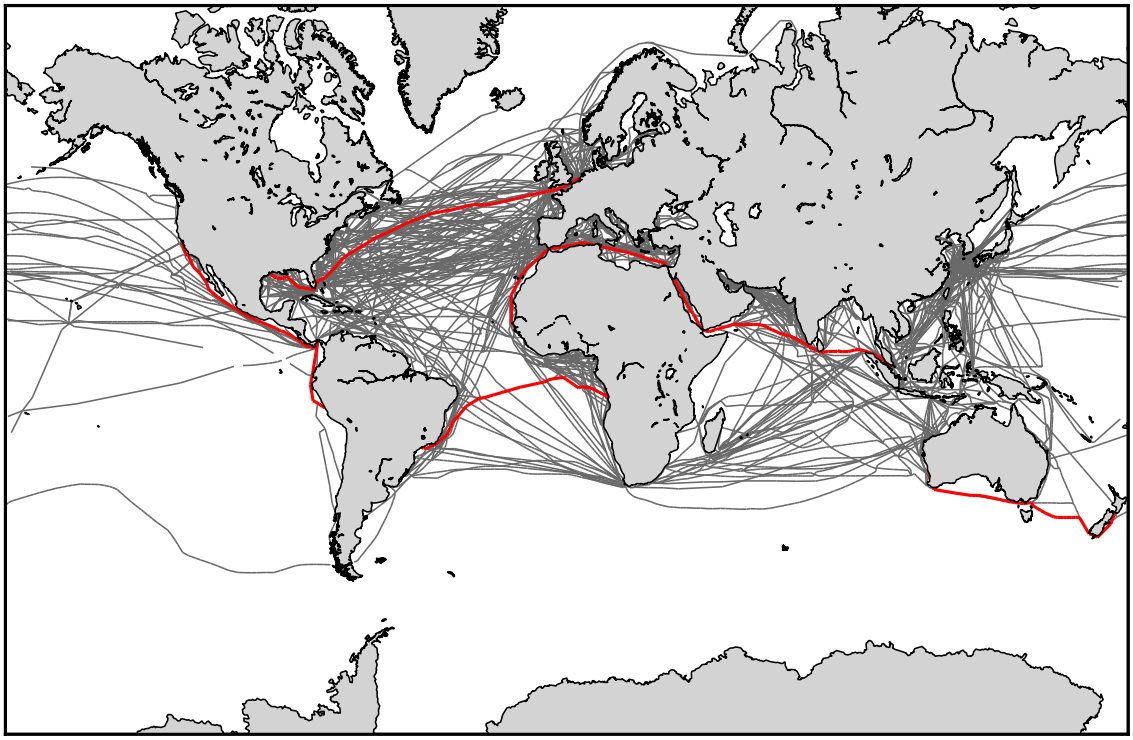


**Figure G.1.** Network graph based on historic voyages projected on the world map. Five sample shortest routes are highlighted in red.

voyages that have been sailed less frequently or not at all, this approach does not provide satisfactory results. For instance, near the west coast of South America, the routing options are poor. Moreover, this approach does not consider environmental conditions such as weather and ocean currents, nor does it allow a variable engine speed setting.