Maritime Transportation Optimization using Evolutionary Algorithms in the era of Big Data and Internet of Things

Fatemeh Cheraghchi

Thesis submitted in partial fulfillment of the

requirements for the Doctorate of Philosophy degree in

Computer Science

School of Electrical Engineering and Computer Science

Faculty of Engineering

University of Ottawa

Abstract

With maritime industry carrying out nearly 90% of the volume of global trade, the algorithms and solutions to provide Quality of Services (QOS) [18] in maritime transportation are of great importance to both academia and the industry. This research investigates an optimization problem using evolutionary algorithms and big data analytics to address an important challenge in maritime disruption management, and illustrates how it can be engaged with information technologies and Internet of Things (IoT). Accordingly, in this thesis, we design, develop and evaluate methods to improve decision support systems (DSSs) in maritime supply chain management.

We pursue three research goals in this thesis. First, the Vessel Schedule Recovery Problem (VSRP) is reformulated and a bi-objective optimization approach is proposed. We employ bi-objective evolutionary algorithms (MOEAs) to solve optimization problems. An optimal Pareto front provides a valuable trade-off between two objectives (minimizing delay and minimizing financial loss) for a stakeholder in the freight ship company. We evaluate the problem in three domains, namely scalability analysis, vessel steaming

policies, and voyage distance analysis, and statistically validate their performance significance. According to the experiments, the problem complexity varies in different scenarios, while Non-dominated Sorting Genetic Algorithm II (NSGAII) performs better than other MOEAs in all scenarios.

In the second work, a new data-driven VSRP is proposed, which benefits from the available Automatic Identification System (AIS) data [2]. In the new formulation, the trajectory between the port calls is divided and encoded into adjacent geohashed [133] regions. In each geohash, the historical speed profiles are extracted from AIS data. This results in a large-scale optimization problem called Granular Speed-based Vessel Schedule Recovery Problem (G-S-VSRP) with three objectives (i.e., minimizing loss, delay, and maximizing compliance) where the compliance objective maximizes the compliance of optimized speeds with the historical data. Assuming that the historical speed profiles are reliable to trust for actual operational speeds based on other ships' experience, maximizing the compliance of optimized speeds with these historical data offers some degree of avoiding risks. Three MOEAs tackled the problem and provided the stakeholder with a Pareto front which reflects the trade-off among the three objectives. Geohash granularity and dimensionality reduction techniques were evaluated and discussed for the model.

G-S-VSRP is a large-scale optimization problem and suffers from the curse of dimensionality (i.e. problems are difficult to solve due to exponential growth in the size of the multi-dimensional solution space [94]), however, due to a special characteristic of the problem instance, a large number of function evaluations in MOEAs can still find a good set of solutions.

Finally, when the compliance objective in G-S-VSRP is changed to minimization, the regular MOEAs perform poorly due to the curse of dimensionality. We focus on improving the performance of the large-scale G-S-VSRP through a novel Distributed Multiobjective Cooperative Coevolution Algorithm (DMOCCA). The proposed DMOCCA improves the quality of performance metrics compared to the regular MOEAs (i.e. NSGAII, NSGAIII, and GDE3). Additionally, the DMOCCA results in speedup when running on a cluster.

Acknowledgement

Completion of this doctoral dissertation was possible with the support of several people. I would like to express my sincere gratitude to all of them. I would like to express my special appreciation and thanks to my supervisor Professor Bijan Raahemi and co-supervisor Professor Emil Petriu. I would like to thank you for encouraging my research and for allowing me to grow as a research scientist. Your advice on both research as well as on my career have been priceless. Moreover, I am extremely grateful to my other research advisors Dr. Ibrahim Abualhaol, Dr. Rafael Falcon, and Dr. Rami Abielmona at Larus Technologies Inc. Their patience, motivation, enthusiasm, and immense knowledge passed any expectation level. It was a fantastic collaboration that any Ph.D. student should wish to experience with the industry.

A very special thanks goes out to my friends Arina, Mana, and Mehran for their continued support and encouragement I received throughout the research work. I also extend my gratitude to Shahin that her endless support and help to take care of my baby, while I was working on my thesis,

was priceless. With a special mention to Waeal, Mohammad Hossein, and other fellow labmates at KDD lab for being always there with their unfailing support and assistance. I am also grateful to Diman whose effort made this Ph.D. journey possible.

I cannot find any word to express how grateful I am to my beloved husband Arash who never stopped believing in me and never got frustrated during this long journey. I also should thank my little baby boy Alborz for his patience who accompanied me every day since I was carrying him. I am also grateful to my family: my mother, my late father who I wish was here to celebrate my graduation together, my brother Hossein and sisters Mahboub and Zahra for all of the sacrifices they have made on my behalf which led me to be the one I am today.

I acknowledge the financial support of the Ontario Centres of Excellence (OCE) and the National Sciences and Engineering Research Council of Canada (NSERC) for the project entitled "Big Data Analytics for the Maritime Internet of Things" (NSERC CRD 499024-16), and NSERC Discovery Grant Nbr RGPIN/341811-2012. Infrastructure support was provided by the Southern Ontario Smart Computing Innovation Platform (SOSCIP) and the Telfer School of Management at the University of Ottawa.

Contents

List of Figures

List of Tables

Acronyms

ACO Ant Colony Optimization.

AI Artificial Intelligence.

AIS Automatic Identification System.

AOI Area of Interest.

BDA Big Data Analytics.

CC Cooperative Coevolution.

CCGA Cooperative Coevolution Genetic Algorithm.

CPS Cyber-Physical Systems.

DE Differential Evolution.

DMOCCA Distributed Multiobjective Cooperative Coevolution Algorithm.

DSS Decision Support System.

EA Evolutionary Algorithm.

EPI Epsilon Indicator.

ET Elapsed Time.

G-S-VSRP Granular Speed-based Vessel Schedule Recovery Problem.

GA Genetic Algorithm.

GD Generational Distance.

GDE3 Generalized Differential Evolution 3.

GFS Google's File System.

GSM Global Speed Mining.

HDFS Hadoop Distributed File System.

HV Hypervolume.

ICT Information and Communication Technology.

IGD Inverted Generational Distance.

IIoT Industrial Internet of Things.

IoS Internet of Services.

IoT Internet of Things.

KPI Key Performance Indicators.

LS Local Search.

LSGO Large-Scale Global Optimization.

MH Metaheuristic.

MI Machine Intelligence.

MILP Mixed Integer Linear Programming.

MOCCGA Multiobjective Cooperative Coevolution Genetic Algorithm.

MOEA Multiobjective Evolutionary Algorithm.

MOGA Multiobjective Genetic Algorithm.

MOO Multiobjective Optimization.

MOOP Multiobjective Optimization Problem.

MPE Max Pareto front Error.

NFE Number of Function Evaluations.

NPGA Niched Pareto Genetic Algorithm.

NSGAII Non-dominated Sorting Genetic Algorithm II.

OCC Operational Control Center.

OCE Ontario Centres of Excellence.

PESA-II Pareto Envelope-based Selection Algorithm II.

PM Polynomial Mutation.

PMOEA Parallel Multiobjective Evolutionary Algorithm.

POI Period of Interest.

Pop.size Population Size.

PSO Particle Swarm Optimization.

QOS Quality of Services.

RT Running Time.

S-VSRP Speed-based Vessel Schedule Recovery Problem.

SBX Simulated Binary Crossover.

SNDP Service Network Design Problem.

SOSCIP Southern Ontario Smart Computing Innovation Platform.

SPC Spacing.

SPEA2 Strength Pareto Evolutionary Algorithm 2.

TSP Traveling Salesman Problem.

VSRP Vessel Schedule Recovery Problem.

Chapter 1

Introduction

The aim of this chapter is to introduce the reader to a number of challenges in the maritime supply chain industry and the integration of this industry with the emerging information and communication technologies. We unveil the potential benefits of exploiting Big Data Analytics (BDA) within various optimization problems that exist in the field. To achieve this goal, we boost metaheuristic optimization methods with valuable knowledge extracted from the data.

The motivation and the overall milestones of this research are presented next. Furthermore, the list of contributions is presented to support the scientific novelty of this work. This chapter concludes with publications and

the outline of the thesis structure.

1.1 Motivation

Supply chain management is filled with complex challenges. Overcoming these challenges requires systems to be automated, intelligent, and optimized. For this sake, any effort which improves the quality of the system, delivers a great value. As collecting and accessing data become easier every day, it is vital to link data from related fields and to leverage them to gain insight to make better decisions. Particularly, in the maritime supply chain, which is the area of our interest, decision-makers can take advantage of data to tackle complicated challenges such as those outlined below:

(a) **Routing management**: with the help of IoT, acquiring information about the real-time status of sailing objects is now possible. We can estimate or predict their future or near-future behaviour. It helps us provide a reliable routing system. For example, real-time access to the latest updates regarding the events such as weather conditions, marine traffics, port congestion, bunker fuel status and oil price rate can improve the routing system significantly [136], [137], [140].

(b) **Risk management**: by real-time monitoring the ship sensors, those sensors violating thresholds are detected. In an intelligent system, a model (off-line or on-line based on the data source) can learn the normal and risky situations of the sensors. An alerting system can be included to inform the staff about potential risks [134, 135].

(c) **Vessel scheduling**: a realistic and real-time scheduling can be provided for pick-ups and deliveries. An optimal vessel scheduling solution is the one that helps to ensure that every decision is the best for the company, its customers, and other related organizations. When planning a voyage, it has to be adjusted for real-time demand fluctuations, changes in the weather or unexpected delays. This can save the company significant amount of time, trouble and money [138, 139, 140, 124].

(d) **Disruption management**: when disruption happens, following the planned schedule would cause financial loss and will impact the reputation of the maritime organization. In order to minimize the negative disruption impact, having recovery plans is vital. The goal of this study is to address this challenge through proposing optimization approaches for the vessel schedule recovery problem [10], [41], [66], [17] [38].

The large-scale planning problems that maritime logistics companies encounter at the strategic, tactical, and operational levels are usually treated separately due to complexity and practical considerations [9]. At the strategic level, the company decides about the layout of the network (e.g. the fleet size, and what customers to serve). At the tactical level, the fleet deployment is determined and each service is scheduled, while physical transportation network is handled at the operational level which includes operations and services such as loading and unloading the containers, berthing of the vessels in ports, and disruption management due to events such as bad weather or port delay.

Operational data can lead to better predictions of what will happen in the future and carriers are constantly receiving sensor data from vessels that can help predict such as disruptions or required maintenance. Similarly, data received from terminals can be used to predict delays and help vessels adjust sailing speed to save fuel. These sensor data as well as other sources of information such as AIS data can potentially generate high volumes of data which can help maritime industry to benefit from big data technologies. The usefulness of these technologies might be limited by the cost and quality of on-board sensors and data acquisition systems, satellite connectivity, data

ownership and technical obstacles related to collecting and using big data effectively [60].

Difficulties (and potential solutions) regarding big data management can be categorized into three groups [60]:

(a) **Volume and storage**: while traditional storage systems struggle with storing and providing access to big volumes of data, distributed storages such as HDFS [146], S3 [142] and Cassandra [143] solve this problem by providing scalable databases. Raw data will be transformed to other forms to best serve the purposes. The extracted data will be fed to any big data frameworks or other analytic tools.

(b) **Data quality**: data might be unreliable. There might be no consistent relationship between data values and observed phenomena. Unreliability can be context specific or temporal, where the former is when some additional and possibly unavailable information is needed to interpret data such as the exact location of sensors, and the former is when the input data has an irregular frequency sampling. Additionally, unreliability might be non-proportional which is when the input data is not proportional to the phenomenon observed such as a lag between the real events and the report. Faulty data can also contain information, e.g. on

sensor reliability.

(c) **Data analytics**: data analytics convert raw data to useful information. A model can be built from data and such a model can predict future events or one can discover new patterns in the data based on statistical techniques (e.g. correlation) or Machine Intelligence (MI). The extracted information can then be used in the decision making process.

By taking into account the challenges discussed earlier and available AIS data in maritime, we want to address an important topic in disruption management in this domain which is VSRP. To use AIS data in this problem, new challenges emerge. Benefit from data comes at the expense of redefining the problem and redesigning the existing solutions. In VSRP, redefining the problem led to a large-scale optimization problem. Distributed technologies which have had great success in recent years are employed in our proposed method to improve the search quality by breaking the large problem into subproblems running in parallel.

1.2 Objectives and Methodology

In the previous section, we addressed the core issues that formulate our overall research goal: the challenges related to disruption management in VSRP and redefining the problem using historical AIS data.

Even though disruption management has been studied for quite a while, there is not much literature in the applications of big data in the maritime industry. Most of the works in this area are just plain theory or introducing the opportunities of using data in the field. This work is centered around disruption management in VSRP leveraging AIS data to improve currently available solutions. We explain their challenges and present solutions for them. We formulate the problems, and due to their dynamic behaviour, we offer metaheuristic methods as the suitable solutions. Solutions are tested on a number of case studies. The experiments are designed to identify the limitations. We are not trying to find a generalizable solution, rather, the intention is to lay the foundation for further research endeavors from academia and industry to fine tune it to serve real-world use cases.

For a better understanding of this study, the overall goal is split into three phases. Assuming that speedup is a potential recovery strategy in VSRP,

we investigate the possibility of defining VSRP as a multiobjective problem and tackle it with metaheuristic approaches. Then, a big-data-enabled formulation of the problem is proposed. At the end, we take a closer look at existing limitations and make suggestions to minimize the impact of these limitations. One particular limitation is the low performance of MOEAs in the experiments when redefining VSRP with big data due to the curse of dimensionality where the size of the search space grows exponentially. We investigate distributed approaches along with cooperative coevolution methods [106, 122] to tackle this problem. Among the large-scale optimization approaches (refer to the section 5.1.2), cooperative coevolution was chosen. It is a divide-and-conquer approach and more intuitive with regards to our problem which is composed subcomponents of geohashes between every two ports. Comparing the results of this approach with other approaches is not in the scope of this thesis.

This should be mentioned that however the optimization problems defined in this thesis are specifically in maritime domain, but the proposed methods can be used in other applications such as aerial transportation system.

1.2.1 Phase I: Speed-based Vessel Schedule Recovery Problem (S-VSRP)

Maritime transportation is the major link between international trades. Since there is not much difference in the services that freight ship companies offer, the main competition among them is cost-based. Moreover, with 90% of international trades done through the sea, the cost of shipping services has a significant impact on the global economy. Therefore, efficient and optimized services become an important factor to reduce the costs and present reliable services. However, liner shipping operations are vulnerable to many uncertain factors which prevent the services from being executed seamlessly. In fact, disruptions occur often in a global liner shipping network. According to [69] approximately 70-80% of vessel round-trips experience delays in at least one port. When a disruption happens in a liner shipping network, it causes delay and consequently, financial and credential losses. To mitigate these negative effects, a recovery plan should help put the vessels and cargoes back on schedule.

In disruption management, VSRP copes with unexpected disruptions which often require real-time decisions. VSRP has been studied in the litera-

ture as a single objective problem or as a weighted sum of many objectives. In Chapter 3, we formulate the VSRP as a Multiobjective Optimization (MOO) Speed-based VSRP (S-VSRP) and propose metaheuristic methods to find the optimal Pareto set. The solutions in this set provide a trade-off between delay and loss objectives for the stakeholder. To the best of our knowledge, no Pareto-based MOO formulation for the S-VSRP is publicly available in the literature.

1.2.2 Phase II: AIS Big-Data-Enabled Speed-based Vessel Schedule Recovery Problem

With growing technology in the liner industry, ships are equipped with a wide range of sensors. Ships are also equipped with the AIS technology that sends out the vessel's movement and other relevant voyage data to a marine vessel traffic operator. AIS is a vessel tracking system that automatically provides updates on the ship's static and dynamic status [2].

Even though in recent years there have been many attempts in the literature to use IoT and big data to improve the maritime services and logistics, it is still in its early stages. Most of the articles are focused on explaining the challenges and opportunities. Here, we are aiming to leverage AIS data to

redefine S-VSRP. A three-objective optimization algorithm G-S-VSRP takes into account the statistical speed limits extracted from AIS data. It is not realistic that a ship travels the whole trip with a fixed speed value (i.e. the assumption in the first phase). The route between the port calls is encoded into the geohash regions and the extracted speed profiles are assigned to each geohash. Chapter 4 is devoted to this problem.

1.2.3 Phase III: Distributed Multiobjective Cooperative Coevolution Algorithm

The G-S-VSRP is a large-scale optimization problem such that regular MOEAs need a big number of function evaluations to find solutions. Usually, the regular MOEAs perform poorly with large-scale optimization problems due to the curse of dimensionality. However, the special characteristic of the compliance objective (i.e. a measurement to show how much the optimized speed values comply with the historical navigational patterns) in the G-S-VSRP directs the search towards a relatively small search space. There is a small range of speed values (i.e. the average of differences between $\overline{s_i}$ s and s_i^{max} s is 1.15 knots in our problem instance) in which the compliance with historical patterns of ship movements is maximized. On that account, even though it is

a large-scale problem, the search space is not very wide, however. Therefore, large number of function evaluations can find reasonably good solutions. But, this is not always the case, because the problem instance is sampled from the AIS data (i.e. the speed statistics in geohashes only come from five months of January to May 2015 due to the limited capacity of our computational resources at the time) and if we add more data, the problem instance changes and it may no longer have the same characteristics of a small range of speed values for the compliance. Also, the stakeholder may not be willing to comply with the historical speed patterns but is interested to be aware of the objections against these patterns. Both of these suggested a change to the compliance objective from maximization to minimization. Through this change, we face a large-scale optimization problem with a huge search space. To deal with the curse of dimensionality in the G-S-VSRP, we propose a distributed cooperative coevolution algorithm in Chapter 5. The cooperative coevolution is a divide-and-conquer approach which splits the large variable vector into sub-vectors and cooperatively searches the subspace in each sub-vector.

1.3 Contributions

The scientific novelty of this thesis comes in the form of the following contributions:

1.3.1 Multiobjective Vessel Schedule Speed Recovery Problem

To the best of our knowledge, this is the first time the VSRP has been solved as a MOO problem with speedup as the recovery strategy. This gives the decision-makers the ability to trade-off between the two objectives which are defined as delay and financial loss. We consider three scenarios to evaluate the optimization problem for the scalability, different recovery speed policy, and different voyage length on synthetic instances.

In the first scenario, an experiment has been designed to run the problem on instances with a different number of port calls (the number of ports to be visited in the vessel trip). The detailed design of the experiment is explained in Chapter 3. Six multiobjective evolutionary algorithms are evaluated with different metrics to measure the performance of their Pareto-fronts.

The second scenario chooses the recovery speed range from two policies:

one from slow steaming where the speed range is 18 to 20 knots, and the other one covers sailing with normal steaming which contains the range of 20-25 knots. All the scenarios are using the same set of parameter values that have been obtained through a tuning process. The sensitivity of the algorithms is evaluated in two speed steaming.

The third scenario compares the sensitivity of the optimization problem for long-distance traveling voyages versus short-distance traveling voyages. In the three scenarios, the significance level of difference between algorithms is validated by a set of non-parametric statistical tests. These research experiments led to the publication of a journal paper [100].

1.3.2 Big-Data-Enabled Multiobjective Vessel Schedule Speed Recovery

The number of researches analyzing and taking actions for the collected data in maritime operations and logistics is rare. Recently, this industry has started to catch up with other transportation systems in using information and communication technology. With available AIS data, S-VSRP problem is extended by using extracted historical speed data.

It is very unlikely that a ship sails with a fixed speed the entire route

between two port calls. This brings us to analyze the historical data to see what speed limits have been reported at different regions. A Global Speed Mining (GSM) engine has been designed to compute the statistics of the reported speeds for the encoded regions throughout the world. The route between the port calls is divided into the sub-regions encoded by the geohash system. The speed information for each geohash is obtained from GSM. This makes the S-VSRP problem a large-scale optimization problem where the number of speed variables in the search space has grown to large numbers.

We defined a three-objective-optimization problem for this big-data-enabled S-VSRP. The first objective minimizes the delay and the second objective minimizes the financial loss similar to the previous MOO formulation of S-VSRP. The last objective measures how much the speed variables comply with the historical data, and we call it compliance and we set it to be maximized. A Pareto-front solution set provides a trade-off between these objectives for the stakeholder.

Moreover, since this speed granularity impacts the convergence time of the algorithms, a merging algorithm clusters the adjacent geohashes which have similar speed profiles. The quality of solutions from merged geohashes

is compared with the original granular speeds. Lastly, the impact of precision of geohash on the optimization problem is investigated. The proposed speed granularity in S-VSRP (G-S-VSRP) led to a conference paper [101].

1.3.3 Distributed Multiobjective Cooperative Coevolution Algorithm

To address the high-dimensional optimization in G-S-VSRP a novel DMOCCA is designed. Cooperative coevolution is a divide-and-conquer approach. It splits the large variable vector into sub-vectors and searches the subspace in each sub-vector while cooperating among them. In DMOCCA, grouping the geohashes between every two ports into a subcomponent forms the splitting strategy. DMOCCA performs well when the regular MOEAs stop improving their performance. The solutions found by a MOEA are added to two archives (i.e. one keeps the early MOEA's solutions, the other one, beside those solutions, stores new solutions found in the next steps of the algorithm). Then, DMOCCA, in an iterative process, starts improving those solutions by a parallel search on the subspaces corresponding to the subcomponents. DMOCCA showed its ability in improving the performance when no time constraint is considered. Also, DMOCCA proved that it can speedup

when more computational resources (i.e. worker nodes in the cluster) are available. Chapter 5 is dedicated to the detailed design of DMOCCA and the experiment which examines the performance of DMOCCA vs. regular MOEAs.

Through this thesis, the reported results are the average of 30 experiments. Occasionally, the number of experiments is less than 30 (due to the experiments becoming computationally expensive). In most cases, we group the experiments to achieve 30 runs in total. Also, a predefined upper limit for the number of function evaluations provides an appropriate stopping condition [39, 144]. We follow the same criterion for the stopping point in the EAs.

1.4 Publications

The following publications directly reflect the contributions of this research which are also parts of Chapters 3 and 4.

1. "**Cheraghchi F**, Abualhaol I, Falcon R, Abielmona R, Raahemi B, Petriu E. Modeling the speed-based vessel schedule recovery problem using evolutionary multiobjective optimization. Information Sciences.

2018 Jun 1;448:53-74." [100]

2. "**Cheraghchi F**, Abualhaol I, Falcon R, Abielmona R, Raahemi B, Petriu E. Big-data-enabled modelling and optimization of granular speed-based vessel schedule recovery problem. In Big Data IEEE International Conference on 2017 Dec 11 (pp. 1786-1794). IEEE." [101]

3. "Petriu E.M., Abielmona R, Falcon R, Palenychka R, Abualhaol I, **Cheraghchi F**, Teske A, Primeau N, Panchapakesan A, BIG DATA ANALYTICS FOR THE MARITIME INTERNET OF THINGS. Artificial Intelligence for Insights into Regulations, October 19, 2018" [130]

The following papers are also related to my research on big data analytics:

4. "**Cheraghchi F**, Iranzad A, Raahemi B. Subspace selection in high-dimensional big data using genetic algorithm in Apache Spark. In Proceedings of the Second International Conference on Internet of things and Cloud Computing 2017 Mar 22 (p. 54). ACM." [83].

5. "Esmaeilpour A, Bigdeli E, **Cheraghchi F**, Raahemi B, Far BH. Distributed Gaussian Mixture Model Summarization Using the MapReduce Framework. InCanadian Conference on Artificial Intelligence 2016 May 31 (pp. 323-335). Springer, Cham." [29].

6. "Mohammadi MM, Raahemi B, **Cheraghchi F**, Obidallah W, Bigdeli E. Big data analytics using hadoop. InProceedings of 24th Annual International Conference on Computer Science and Software Engineering 2014 Nov 3 (pp. 323-325). IBM Corp." [7].

1.5 Thesis Outline

The remainder of the thesis is structured as follows: Chapter 2 starts with the technical background for the formulations and experiments (i.e., evolutionary algorithms and metaheuristic methods in solving multiobjective optimization problems). We will present previously published relevant research studies (related works) in Chapters 3 and 4.

Chapter 3 is devoted to the metaheuristic multiobjective optimization for a speed-based vessel schedule recovery problem and the detailed discussion on the experiments.

Chapter 4 addresses the previous problem from a data-driven aspect, namely, big-data-enabled modelling and optimization of granular speed-based vessel schedule recovery problem (G-S-VSRP). A comprehensive discussion elaborates on the experimented scenarios.

Chapter 5 addresses the limitations of regular MOEAs in large-scale optimization which happens in the modified G-S-VSRP with three minimization objectives. A novel distributed cooperative coevolution method is introduced in this chapter, and its performances are verified using a set of experiments.

Chapter 6 concludes this research work. The contributions, limitations and future works are summarized in this chapter.

1.6 Summary

In this chapter, we introduce the reader with the challenges in the maritime industry when information and communication technologies can unveil some potential benefits in various problems such as routing management, vessel scheduling, and disruption management. The main motivation of this work is to link AIS data available in the maritime field and to leverage them to gain insight to make better decisions. Among the various challenges in this industry, we study VSRP in disruption management following our motivation in discovering the potential benefits of data. The overall three milestones of this research (i.e. defining the VSRP as a bi-objective optimization problem, redefining this problem using AIS data to build a big-data-enabled model,

and proposing a distributed optimization method to deal with this model) are presented in this chapter. It is followed by the list of contributions to support the scientific novelty of this work. Publications and the outline of the thesis structure are presented in this chapter as well.

Chapter 2

Background Review

In this chapter the reader will be presented with a brief yet descriptive set of concepts in maritime supply chain, followed by the techniques and methodologies employed to tackle the problems introduced in this research. The related works corresponding to the contributions of the thesis are surveyed in Chapters 3, 4, and 5 where each contribution is explained in detail. They include literature surveys in VSRP, big data applications in maritime industry, large-scale optimization, and distributed optimization.

2.1 Maritime Supply Chain Management

Supply chain management encompasses the planning and management of all activities involved in sourcing and preparation, conversion, and all strategic, tactical and operational management issues. It also includes coordination and collaboration with other partners, which can be suppliers, intermediaries, third-party service providers, and customers. In essence, supply chain management integrates supply and demand management within and across companies [8].

The majority of global trade is carried by sea (nearly 90% by volume) and it is the backbone of almost every international supply chain [46, 26]. Maritime has a major impact on the global economy. Standardization of containerization has been a key concept in achiving this success for maritime transportation. It has contributed significantly to the global supply chain and reduced transport costs [43] by providing easy and efficient handling a container from origin to destination with different transportation systems (vessel, train, truck) without the need to reorganize the content within.

Quality of liner shipping services is greatly impacted by dynamic operations and uncertain activities associated with long geographical distances in

container shipping. The social and environmental impacts of shipping also add some new challenges to the shipping performance and operations. The latter has been reflected in research on optimizing the scheduled speed for the vessel to reduce the greenhouse emissions [31] and it has also been a major driver behind the slow steaming policy [14].

2.2 Maritime Disruption Management

According to the authors in [53], nearly 65.5% of unreliability in the ships' schedule is caused by port congestions. The dynamics of weather condition is another factor on unreliability. Also, other occasional events such as port closure due to high wind or flooding, terminal unavailability due to the failure of some machines may disrupt the schedule. All these lead to the fact that approximately $70 - 80\%$ of the vessels experience delays in at least one port during each round-trip [53].

Disruption events plus regular uncertainties constitute two types of uncertainties in liner shipping operations. Variable port productivity or unexpected waiting time for port access are examples of the regular uncertainties that occur frequently. Regular uncertainties can be managed by modeling the

degree of uncertainty probabilistically and adding some certain time buffers to lead to a robust network design [71, 72], while disruptions events are rare and usually unpredictable uncertainties.

The importance of disruption management in the supply chain is that the functionality of supply chain is dramatically influenced by a major disruption. Finding resilient and robust strategies in the face of major disruptions improves the vulnerability of the supply chain. Identifying different types of risks, estimating the likelihood of occurring, evaluating the potential loss and identifying the strategies to reduce the risks are some ways to mitigate the impact of disruptions [66], [79].

A more detailed literature review on VSRP can be found in the related Chapters 3 and 4.

2.3 Maritime Internet of Things (mIoT) and Big Data

The Maritime industry has slowly started embracing the huge potential of the Industrial Internet of Things (IIoT). Ships have been equipped with sensors to collect data. Sensors, computers, industrial controls and equipments on

ships have generated tons of data. However, a small percentage of existing devices are designed to connect to the internet and are able to share the data with the cloud. In the current technology in IoT, the intention is to connect this operational data with the IT technology to be sent in real-time to all related parties. These sensors monitor everything from the ship's speed to the temperature of its cargo, allowing for an optimized shipping ecosystem as authors in [76] tried to introduce as an integrated BD-IIOT framework.

An example of the need for a real-time IoT equipped system in maritime industry is cargo tracking. A statement from Ericsson [34] highlights the importance of having a clear idea of what state cargo is in for the duration of any voyage: "keeping track of so much cargo is incredibly challenging, and there is much to be gained from connecting containers wirelessly, monitoring them and making real-time information about their whereabouts and environmental conditions easily available via an integrated dashboard."

Big data is all about data and analysing large amounts of data to support use cases such as predictive maintenance and capacity planning. While, IoT is about data, devices, and connectivity. It is not necessarily big data by nature, but low latency, low duration, and high volume machine-generated data coming from a wide variety of sensors to support real-time use cases.

IoT analytic solutions require BDA but must also include: *streaming data management* with the ability to ingest, aggregate (e.g., mean, median, mode) and compress real-time data coming off a wide variety of sensor devices "at the edge" of the network, and *Edge Analytics* that automatically analyses real-time sensor data and renders real-time decisions (actions) at the edge of the network that optimizes operational performance or flags unusual performance/behaviours for immediate investigation (security breaches, fraud detection).

2.4 Optimization Techniques

In this section, we briefly review the optimization techniques we employed in the thesis related to the proposed algorithms and the experiments.

2.4.1 Nature-inspired Metaheuristic Optimization

Metaheuristics, unlike exact methods, allow tackling large-size problem instances by providing satisfactory solutions in a reasonable time. It does not guarantee to find global optimal solutions or even bounded solutions. Their use in many applications shows their efficiency and effectiveness to solve large

and complex problems [49]. They refer to developments of heuristic concepts that go beyond Local Search (LS) procedures as a means of seeking a global optimum, typically by analogy with the natural (physical or biological) processes.

Nature-inspired Evolutionary Algorithm (EA) are very well-known techniques in metaheuristic optimization. They are a subset of generic population-based evolutionary computation in Artificial Intelligence (AI). An EA uses mechanisms inspired by biological evolution, such as reproduction, mutation, recombination, and selection [49]. Two popular approaches used in the experiments are briefly explained in this section.

Genetic Algorithms (GA)

Genetic Algorithms (GAs) are among the most popular metaheuristic techniques [35]. The algorithm begins by creating a random initial population. Subsequently, it creates a sequence of new populations. At each step, the algorithm uses the individuals in the current generation to create the next population. To create the new population, the algorithm scores each member with so called fitness scores. The algorithm then selects members, called parents, based on their fitness score. Children are produced from these par-

ents either by making random changes to a single parent (mutation) or by combining the vector entries of a pair of parents (crossover). The current population is replaced with the children to form the next generation. Some of the individuals in the current population that have the best fitness are chosen as elite. These elite individuals are directly passed to the next population. The algorithm stops when one of the stopping criteria is met.

Differential Evolution (DE)

DE, another popular population-based metaheuristic, starts with a population of randomly initialized solution vectors [65]. However, instead of using variation operators with predetermined probability distributions, DE uses the difference vector of two randomly chosen members to modify an existing solution in the population. The difference vector is weighted with a user-defined parameter $F > 0$ and to a third (and different) randomly chosen vector as shown below:

$$\mathrm{v}_{i,t+1} = \mathrm{x}_{r_1,t} + F.(\mathrm{x}_{r_2,t} - \mathrm{x}_{r_3,t}) \tag{2.1}$$

At each generation t, it is ensured that r_1, r_2 and r_3 are different from each other and also from i. The resultant vector v_i is called a mutant vector of x_i.

F remains constant with generations. In a variation of crossover in DE, each component of offspring is randomly chosen from one of the parents which are x_i and its corresponding mutant vector v_i. The recombinant is called a trial vector.

2.4.2 Multiobjective Optimization

A Multiobjective Optimization Problem (MOOP) deals with more than one objective function. In most practical decision-making problems, multiple objectives or multiple criteria are evident. Because of a lack of suitable solution methodologies, a MOOP has been mostly cast and solved as a single-objective optimization problem in the past. An ideal multiobjective optimization procedure finds multiple trade-off optimal solutions with a wide range of values for objectives and chooses one of the obtained solutions using higher-level information. In a more quantitative manner, we explain three approaches to MOOP.

1) Weighted-Sum Approach: the weighted-sum method is the most widely used classical technique in MOOP. It scalarizes a set of objectives

into a single objective using a set of known weights for each objective [24].

$$\begin{aligned} \text{Minimize} \quad & F(\mathrm{x}) = \sum_{m=1}^{M} w_m f_m(\mathrm{x}) \\ \text{subject to} \quad & g_j(\mathrm{x}) \geqslant 0 \quad j = 1, 2, \ldots, J \\ & h_k(\mathrm{x}) = 0 \quad k = 1, 2, \ldots, K \\ & x_i^{(L)} \leqslant x_i \leqslant x_i^U \quad i == 1, 2, \ldots, n. \end{aligned} \tag{2.2}$$

Here, w_m is the weight of mth objective and weights are choosen such that $\sum_{m=1}^{M} w_m = 1$. A sample two-objective problem is shown in Figure 2.1. One problem which makes it difficult to apply weighted-sum reliably on any problem is that Pareto-optimal solutions which lie on the non-convex portion of the Pareto-optimal front cannot be found [36].

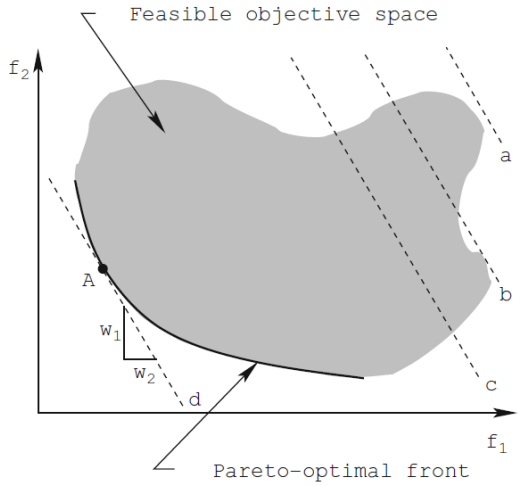

Figure 2.1: Weighted-sum approach on a convex Pareto-optimal front [36].

2) ε-Constraint Method: this method is a solution for the problem of having non-convex objective space [33]. It is a reformulation of MOOP by keeping one of the objectives and restricting the rest of the objectives within user-specified values. One of the difficulties of this method is that the solution for the problem introduced in Eq. 2.3 largely depends on the chosen ε vector. As the number of objectives increases, finding ε vector gets harder (more information needed from the user).

$$\begin{aligned} \text{Minimize} \quad & f_\mu(\mathrm{x}) \\ \text{subject to} \quad & f_m(\mathrm{x}) \leqslant \varepsilon_m \\ m = 1, 2, \ldots, M \text{ and } & M \neq \mu \\ g_j(\mathrm{x}) \geqslant 0 \quad & j = 1, 2, \ldots, J \\ h_k(\mathrm{x}) = 0 \quad & k = 1, 2, \ldots, K \\ x_i^{(L)} \leqslant x_i \leqslant x_i^U \quad & i == 1, 2, \ldots, n. \end{aligned} \tag{2.3}$$

3) Non-dominated Solutions and Pareto-Optimal Solutions: a solution x_1 is said to dominate the other solution x_2 if solution x_1 is not worse than x_2 in all objectives and if solution x_1 is strictly better than x_2 in, at least, one objective. If either of these conditions is violated, the solution x_1

does not dominate the solution x_2.

Since the concept of domination allows a way to compare solutions with multiple objectives, most multiobjective optimization methods use this domination concept to search for non-dominated solutions. A non-dominated set of solutions P, called *Pareto-optimal set*, are those that are not dominated by any member of the solution set P in entire search space [36]. Multiobjective Evolutionary Algorithm (MOEA) can be easily used to find Pareto-optimal set by using the domination definition. The set of Pareto optimal outcomes is often called the *Pareto front*, Pareto frontier, or Pareto boundary. We use the term Pareto front in this thesis.

In the following, we describe few state-of-art MOEA algorithms which we use in our experiments. The aim is to show an overview of how these MOEAs work.

Elitist Non-dominated Sorting GA (NSGA-II): NSGA is an extension of the Genetic Algorithm for multiobjective function optimization. Three main features of NSGA-II are keeping the elitism, using an explicit diversity preserving mechanism and exploiting non-dominated solutions [19].

NSGA-II after population initialization creates the offspring population

using the parent population (i.e., initial population) and the typical generic operators. The parent population and the new offspring population are combined and then sorted into a hierarchy of sub-populations based on the ordering of Pareto dominance.

A new population is created by adding the best non-dominated front and continues with solutions of the second front, followed by the third front and so on until the population size is filled. A crowding sort is performed on each front and calculates the average distance between members of each front.

New offspring population is created from the former population where the selection operators incorporate both rank (order of dominated precedence of the front to which the solution belongs) and distance rank within the front. Crossover and mutation operators are applied as well. This procedure continues until a stopping condition is met.

NSGA-III: This algorithm is a Pareto-based method designed to work well with problems with many-objective optimization problems (three or more). It is an extension of the NSGA-II algorithm with non-dominated sorting mechanism. The main difference between the two algorithms is that NSGA-III uses a set of reference points to maintain the diversity of the Pareto

front during the search. The number of reference points is based on the population size. This results in a very even distribution of Pareto front across the objective space, even when the number of objectives is large [20].

Strength Pareto Evolutionary Algorithm (SPEA-II): This is another Pareto-based optimization method. It uses an evolutionary process with procedures for genetic recombination and mutation to explore the search space. A fitness assignment scheme is used, which takes for each individual into account how many individuals it dominates and is dominated by (strength). Also, a nearest neighbor density estimation technique is incorporated which allows a more precise guidance of the search process. It needs to compute the distance matrix containing the pair-wise distances between solutions in the objective space and the density-based strength evaluation of the solutions. An archive of the non-dominated set is maintained separate from the population of candidate solutions used in the evolutionary process, providing a form of elitism. Its archive truncation methods guarantees the preservation of boundary solutions [81].

Generalized Differential Evolution Algorithm (GDE3): Differential Evolution is an instance of an EA. It is related to EAs such as the Genetic

Algorithm. The DE algorithm involves maintaining a population of candidate solutions via iterations of recombination, evaluation, and selection. The recombination approach involves the creation of new candidate solution components based on the weighted difference between two randomly selected population members added to a third population member. This self-organizes the sampling of the problem space, bounding it to known areas of interest. GDE3 which is an extension of DE is designed for any M number of objectives and K number of constraints without introducing any extra control parameters to the original DE. In the case of unconstrained single-objective optimization problems, GDE3 falls back to the original DE. The selection mechanism in GDE3 is based on the domination rule. When infeasible vectors are presented, the trial vector is selected if it weakly dominates the old vector in constraint violation space, otherwise the old vector is selected. When there are feasible and infeasible vectors, the feasible vector is selected. If both vectors are feasible, then the trial is selected if it weakly dominates the old vector in the objective function space. If the old vector dominates the trial vector, then the old vector is selected. If neither vector dominates each other in the objective function space, then both vectors are selected for the next generation. If the size of the generation is more than the popula-

tion size, then it decreased back to the original size using a similar selection approach used in NSGA-II. Vectors in the population are sorted based on non-dominance and crowdedness and according to these measurements, the population is pruned. [39].

Pareto Envelope-based Selection Algorithm (PESA2): The main attraction of PESA is the integration of selection and diversity management into one technique (selection and archiving are based on the diversity maintaining crowding measure). The main improvement of PESA is the use of region-based selection. Instead of selecting individuals, hyperboxes in the objective space are selected. The hyperboxes represent a set of possible solutions. By selecting a hyperbox, a random individual from that hyperbox is selected. This was shown to "ensure a better spread of development along the Pareto frontier than individual based selection." When there is a non-uniform distribution of individuals, containing large clusters, tournament selection is more likely to choose individuals from those large clusters (because there are so many of them). Region-based selection, however, will treat the large clusters and the isolated individuals as equal groups, with equal probability of participating in each tournament [16].

Random Search Algorithm: We only refer to this algorithm as a benchmark algorithm to evaluate how well the other algorithms perform. There is no searching and breeding mechanisms in random search, except in each iteration a new random population is generated.

2.4.3 Geohash Encoding System

Geohash is a public domain geocoding system, which encodes a geographic location into a short combination of letters and digits. It is a hierarchical spatial data structure which subdivides space into buckets of grid shape, which is one of the many applications of generally space-filling curves. Arbitrary precision is provided by geohash system where there is the capability of gradually removing characters from the end of the code to reduce its size (and gradually lose precision). As a consequence of the gradual precision degradation, nearby places will often (but not always) present similar prefixes. The longer a shared prefix is, the closer the two places are [145].

2.4.4 Distributed Metaheuristic Optimization

Usually, the main idea behind the metaheuristic algorithms is not sequential. Population-based methods maintain and improve multiple candidates

solutions simultaneously. Authors in [75] introduced a taxonomy for parallel and distributed metaheuristic algorithms based on control, data and memory. Figure 2.2 shows this taxonomy with three axes. The strongest model (fully distributed) to solve very large problem is the one with the distributed control, data decomposition, and distributed memory.

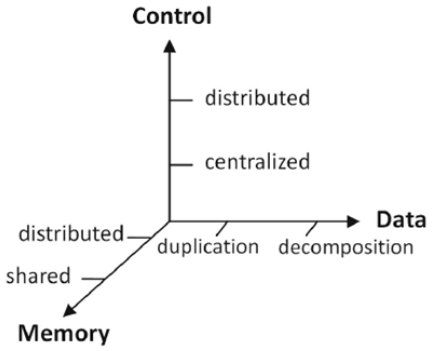

Figure 2.2: Taxonomy of parallel and distributed metaheuristic algorithms based on control, data and memory.

In the centralized control (master-slave) master process manages the population and sends the individuals to be evaluated in slaves. The master process iteratively collects the results and applies some global operations. In this design, the master node could be a bottleneck due to its limited capacity or failure. While distributed control can guarantee the robustness of computation even when some computing nodes fail. This is due to the independent design of computing units.

Data parallelism is the execution of the same operation or instruction on multiple large data subsets at the same time. Depending on the problem, data can be partitioned or if the whole data has to be seen by the algorithm data duplication is the other option. But, duplication is applicable for the small size of data. Large data, that needs to be seen all at once by the algorithm, need a special design for efficiency.

Finally, shared memory is a way of parallelism. However, data sharing is more convenient among multiple computation nodes but handling multiple memory access makes it difficult. Distributed memory is scalable and larger problem can be addressed, however, data exchange between nodes makes the communication the main obstacle.

2.5 Big Data Analytic Frameworks

Today, the massive amount of data (referred as **V**olume) generated at a fast pace (referred as **V**elocity) are coming in different types such as numbers, texts, images, videos, and etc. in forms of structured, semi-structured and unstructured formats (referred as **V**ariety). Besides, there is always some degree of uncertainty on the quality of data (referred as **V**eracity). These

4Vs are the main characteristics of big data [82]. The process of extracting hidden knowledge from such large volumes of raw big data introduces new challenges. Extracting valuable knowledge from this chaos highlights, like never before, the importance of machine learning and data mining. Unfortunately, traditional data analysis methods and platforms need significant and in cases fundamental changes to be able to host technologies fitted for analyzing big data. Novel solutions based on distributed platforms are needed to analyze such volume of data.

Moreover, there are problems that do not necessarily have large volume of data but huge computations. Distributed platforms can be employed for such applications due to the parallelism feature of these platforms and the computations can be run faster. These applications can be designed in a way to provide scalable solutions which means more computation resources can provide faster computations. Our last Chapter 5, focuses on designing an application of this type. It is a distributed multiobjective cooperative coevolution which addresses large-scale G-S-VSRP problem introduced in Chapter 4.

Two important and state-of-the-art frameworks for BDA are briefly introduced in the following.

2.5.1 Apache Hadoop

Hadoop is an open source framework to store and process large quantities of data. Nowadays, the Hadoop framework is widely used in nearly all data processing industries [146]. Hadoop is made up of several modules:

1) Hadoop Distributed File System (HDFS) module It is a file system used by Hadoop cluster. The file system provides redundancy to ensure system availability. It derives from Google's File System (GFS).

2) Hadoop YARN It is a resource management system which is responsible for all the computing resource scheduling and assigning based on user's application.

3) Hadoop MapReduce It is the heart of Apache Hadoop. It is a parallel programming paradigm based on the MapReduce structure. MapReduce refers to two separate and distinct tasks that Hadoop programs perform. The first is the map job, which takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). The reduce job takes the output tuple it to produce a smaller set of tuples.

2.5.2 Apache Spark

Spark is another open source computing framework specializing in data analytics. It is built on top of Hadoop HDFS system but not tied to the MapReduce platform. Spark offers high-speed computing based on HDFS system and developers claim to be 100 times faster for some applications with calculation in memory and 10 times faster calculation in the disk [63].

Spark loads the necessary data into the cluster memory based on user's application and applies calculations directly in memory making it faster than the traditional Hadoop-HDFS approach. Unlike Hadoop which requires loading the necessary data from the HDFS into memory each time it is referenced, Spark keeps the data in memory in between uses as shown in Figure 2.3. Because of this mechanism, Spark is very suitable for algorithms that iterate on the data.

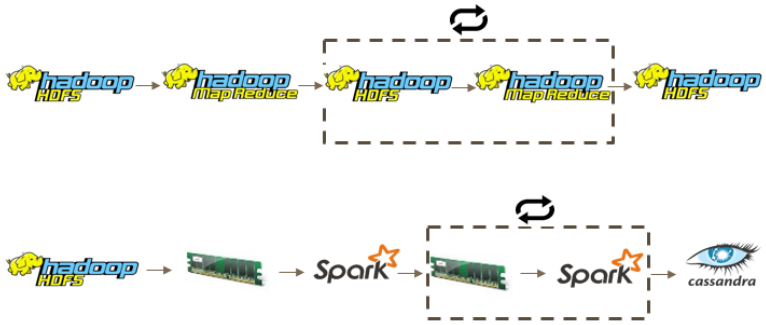

Figure 2.3: Spark computing is up to 100 times faster on memory and up to 10 times on disk compared to HDFS-MapReduce in Hadoop [132].

2.6 Summary

In this chapter, a set of concepts in maritime supply chain management related to this research are reviewed. Also, the techniques and methodologies employed to tackle our optimization problems are introduced which include the description of optimization techniques, multiobjective optimization, and a brief yet descriptive introduction on detail of MOEA algorithms used in this thesis. It is followed by an introduction of the state-of-arts distributed frameworks (i.e. Apache Hadoop and Apache Spark). The related works corresponding to the contributions of the thesis are surveyed in Chapters 3, 4, and 5 where each contribution is explained in detail. They include literature surveys in VSRP, big data applications in maritime industry, large-scale optimization, and distributed optimization.

Chapter 3

Speed-based Vessel Schedule Recovery Problem (S-VSRP): a Multiobjective Optimization Approach

Marine shipping embraces 90% of global trade and has a significant impact on the global economy [46, 26]. Because of limited differentiation of services in this industry, the main competition indicator among its companies is cost-based [74]. Also, there is overcapacity in transport services due to the impact

of the economic recession and financial crisis of 2008, which led to a decline in the current trade demand [41]. Moreover, the freight rates have extremely fluctuated in recent years. Considering all these challenges, to be able to sustain competitiveness, freight ship companies are under pressure to keep prices low and ensure a high quality of service. However, the quality of shipping services is influenced by dynamic operations and uncertain activities. Therefore, presenting robust and reliable services becomes a vital factor for freight ship companies.

The ability to quickly detect a potentially disruptive event and proactively mitigate its effects could indeed lead to a competitive advantage over industrial peers. Disruptive events such as labor strikes or port closure due to hurricanes or flooding are rare and usually unpredictable uncertainties, as opposed to regular uncertainties such as variable port productivity, or unexpected wait time in port access, which happen more frequently and their degree of uncertainty can be estimated through probabilistic models [10, 44, 79]. According to [53] approximately 70-80% of vessel round-trips experience delays in, at least, one port. When disruption happens, it may cause a severe delay in the overall shipping service and hence, following the pre-scheduled plan without adapting to the new situation could result in huge financial and

reputation losses. Disruption management refers to dynamically recovering from various disruption events, which prevent the original operational plan from being seamlessly executed [79]. To mitigate the disruption's impact, the VSRP may take different strategies such as speeding up, skipping port calls, swapping the port calls, extending the working hours at the ports or their combination [10], [41]. To take any recovery action the associated cost and the amount of saved time should be known a priori to the stakeholder.

In this work, we model a speed-based VSRP (S-VSRP) as a MOO problem with two conflicting objectives: total delay and total financial loss. The MOO problem deals with optimization problems with more than one conflicting objective functions. The objectives should be optimized over a feasible set of constraints. MOO problems are typically solved by discovering a set of mutually non-dominated solutions belonging to the Pareto front. In the latter, no solution is better than the other. Improving one objective function in a non-dominated set would hinder the quality of at least another objective in the same solution. There are different techniques to identify a Pareto front reported in the literature. We used multiobjective evolutionary algorithms (MOEAs) to discover Pareto fronts for our problem.

The rest of the chapter is structured as follows: in Section 3.1, related

works are briefly reviewed. Section 3.2 formalizes the S-VSRP problem and Section 3.3 unveils the mechanism of synthetic instance generation. Section 3.4 is concerned with the algorithmic building blocks for the MOEA schemes and the empirical evaluation of our proposed methodology. Finally, Section 3.5 concludes the chapter.

3.1 Related Works

To mitigate the impact of uncertainties in liner shipping, some works have considered some level of uncertainties in the ship scheduling process. Song et. al. in [123] have considered a bi-objective optimization problem (cost and service reliability) for the vessel scheduling with the port and sea uncertainties. They defined Key Performance Indicators (KPI) in the service design problem from different stakeholders' perspectives and evaluated the impact of different speed strategies (constant and flexible speeds) on the KPI and optimal solutions. They used a Multiobjective Genetic Algorithm (MOGA) in their simulation. Authors in [124] focused on ship speed optimization, which has a direct impact on fuel consumption in the presence of uncertain service times and time windows at ports. Using a dynamic programming model,

they minimized the total fuel consumption while maintaining the schedule reliability. They also proposed a dynamic programming model for bunkering problem, which considers the effect of bunker prices on liner service schedule. Making speed decisions with considering uncertainties had a notable effect on decreasing the fuel consumption cost.

Zhao et al. in [125] proposed an optimization problem for fleet deployment using a firefly algorithm, which is based on a route risk evaluation. The main environmental factors affecting navigational safety such as wind, ocean current, and storm surge construct the risk evaluation model, which is a representation of uncertainty on the routes. In a general purpose research [126] a multi-stage recoverable robust optimization for planning under disturbances is introduced. The idea is to unify the robustness and recoverability into one framework. Robust optimization concept [127] has been employed to model the problem. Also, the authors in [128] proposed a hybrid algorithm called Tabu assisted guided local search for Service Network Design Problem (SNDP) to find the most cost-effective transportation network. Their focus was on designing an efficient algorithm to increase the speed of the guided local search.

Authors in [64] proposed a stochastic MOO approach to plan the liner

shipping service schedule and sailing speeds in the presence of port time uncertainty. The optimization model minimizes three objectives (i.e., expected cost, service reliability, and CO_2 emission). They used a MOEA (NSGAII) to approximate the optimal Pareto front. Their model schedules the number of ships and their speeds in a service route. In another work [42], the authors proposed a multiobjective model as a decision support system for vessel scheduling problem. They optimized the vessel speed to find a balance between fuel emission and service level without considering disruptive events using a multiobjective particle swarm.

Despite many works on disruption management in supply chain [66, 17, 38, 70, 4], and vessel schedule design as mentioned in previous paragraphs, the VSRP started to draw attention from the research community only recently. Brouer et al. in [10] were the first ones to model the optimal recovery action under a given disruptive scenario in container shipping. The envisioned recovery strategies include speeding up, port omission, and swapping port calls. The problem has been formulated as a mixed-integer programming model to balance the fuel consumption, and the impact on cargo flows in the shipping network. They tested the model on real scenarios and reported that the suggested solutions could reduce costs by up to 58%. Qi in [58] proposed

a recovery model inspired by the aircraft industry [67, 68, 23]. It approaches VSRP through a non-linear programming (NLP) model to minimize the cost objective based on the speedup strategy and dynamic programming to optimize the cost via the port skipping option.

Li et al. in [45] determined the optimal operational action to mitigate delay in liner shipping by formulating the VSRP with non-linear programming models and dynamic programming algorithms. The three typical disruption recovery strategies (including vessel speedup, port omission, and port swapping) were analyzed. The experiments showed that the speedup strategy is effective when experiencing small delays, but major disruptions require skipping and swapping ports to recover from the delayed schedule. Both Brouer et al. [10] and Li et al. [45] consider the schedule recovery after a disruptive event happens. Their approaches are deterministic in nature and do not consider future new delays. In addition, scaling the problem to include many port calls is not practical due to the computationally prohibitive cost of finding the optimal solution in these scenarios.

Li et al. in [44] formulated VSRP as a real-time multi-stage stochastic model by considering multiple regular uncertainties along the service route. Their dynamic programming model provides proactive actions in response

to updated forecasts of the disruptive event and for reactive actions after the disruptive event while minimizing the total expected fuel consumption cost and vessel-handling delay cost. In [52], the authors present a Markov decision model to determine the optimal recovery policy. The main idea is to reallocate buffer time within a schedule to recover from disruptions.

To emphasize our contributions to the literature, we enumerate them here. First, to the best of our knowledge, this is the first time that VSRP has been formulated as a bi-objective optimization problem. Some studies have considered total cost and service level (sometimes called service reliability which is related to the delay) at the voyage network design level (tactical level), but no research study in the VSRP literature considered delay as a measure of the service reliability objective in the problem formulation. Optimizing the total cost was the only concern in mitigating the delay in the previous works. We adopted the loss objective from [58, 45], and the second objective was adapted from [64, 42], in which they consider the probability of service reliability in their models. Second, our proposed solvers (MOEAs) are powerful search techniques that are able to produce good-quality solutions in reasonable computation times compared to classical exact methods such as integer or linear programming, branch and bound or dynamic pro-

gramming, which are often very time-consuming and do not scale well when solving real-world problems with large dimensionality and many hard constraints. None of the works in the literature evaluated their VSRP solution via different MOEA algorithms to study their behaviour. Third, we generate synthetic S-VSRP instances using a 153 major world ports and proprietary vessel routing technology and make them publicly available to the research community so as to encourage more research on this problem. Fourth, the S-VSRP problem is evaluated with different (three) scenarios in order to capture interesting findings in the algorithms' behaviour related to their ability to optimize long vessel routes, with the vessel adopting a particular steaming policy and conducting regional/transoceanic voyages. Finally, we discuss some insights about the optimizers' performance and statistically validate their significance.

3.2 S-VSRP: A Multiobjective Formulation

In the current work, we examine solely the impact of speeding up strategy in VSRP and call it S-VSRP. We modeled S-VSRP as an MOO problem. It is assumed that the delays due to any disruption events are known for the

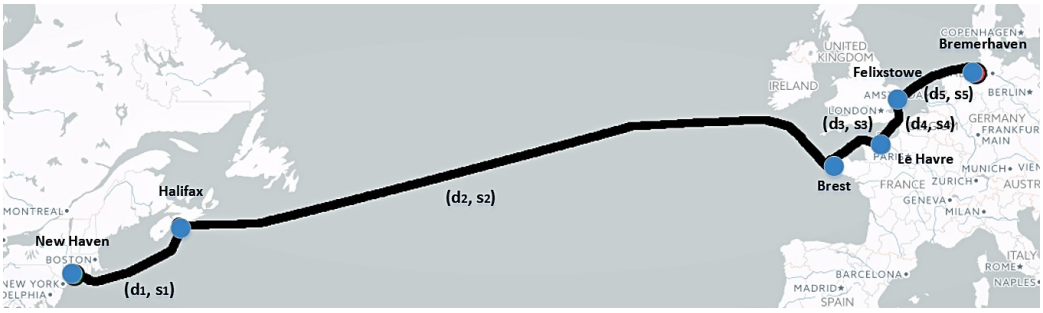

Figure 3.1: An example of a voyage with six port-calls departing at time 0 from port of New Haven (P_1) and returning back to the origin port P_7, where $P_1 = P_7$.

entire vessel route without loss of generality. Also, disruptions at sea and port calls are treated alike. According to a schedule, a vessel on a voyage with N port calls departs at time 0 from port P_1 (origin) and visits ports $\{P_2, P_3, \cdots, P_N\}$ respectively, and at the end returns back to the origin P_{N+1} ($P_{N+1} = P_1$). An example of a six port-call voyage is illustrated in Figure 3.1, where d_i and s_i represent the i-th port call's distance to next port and travel speed, respectively.

S-VSRP is defined as a bi-objective optimization problem that aims to minimize both the *total financial loss* and the *total delay*. In S-VSRP, we use the term *loss* instead of *cost* in the first objective. The reason is that there is a known cost regarding any scheduled trip, and only the extra cost in a disruption recovery plan would matter for the decision makers. When a disruption happens, a vessel can speed up to mitigate the delay at the

expense of a higher fuel cost. In other words, navigating at the maximum possible speed will maximize the fuel cost and minimize the delay. However, navigating at a very slow speed would minimize the fuel cost but does not necessarily minimize the total loss because of delay penalization at the port calls.

In contrast to other works in liner shipping disruption management [41, 45, 58], which define the VSRP as the minimization of cost objective, introducing S-VSRP as a bi-objective optimization problem has the benefit that depending on the liner shipping company's policy, a decision maker can trade-off between delay and loss. In some situations, to keep a high reputation, arriving at a port call at any expense justifies the cost for the company; on the other hand, there might be situations when speeding up is only acceptable to some extent, i.e., total loss.

The *total financial loss* and the *total delay* objectives are formulated as follows:

Minimize Total Financial Loss

$$\min_{\vec{s}}(\sum_{i=1}^{N} d_i \times (f(s_i) - f(s_{const})) + \sum_{i=1}^{N+1} c_i \times t_i^{port} \times (t_i^{RP} - t_i)) \tag{3.1}$$

subject to:

$$t_i^{RP} - t_i \geq 0$$

$$s_{min} \leq s_i \leq s_{max}$$

Minimize Total Delay Time

$$\min_{\vec{s}} \sum_{i=1}^{N+1} (t_i^{RP} - t_i) \tag{3.2}$$

Table 3.1 describes the notations used in Equations (3.1) and (3.2). The first objective minimizes the total loss of a voyage with the expense of the consumed bunker fuel due to speeding up in the first term, and the penalty from arriving late at a port call in the second term. Among many factors in fuel consumption rate such as ship type, weather conditions, and navigation speed, the latter has the greatest impact on it. We adopt the fuel consumption rate from the literature [27], which defines it as a function of speed per unit of *nautical mile* $(0.0036 \times s_i^2 - 0.1015 \times s_i + 0.8848)$. Also, the delay penalty at port i is calculated as $c_i \times t_i^{port} \times (t_i^{RP} - t_i)$ [45], where c_i is drawn from a uniform distribution with regards to port time (t_i^{port}). It is assumed

Table 3.1: S-VSRP formulation notations

Symbol	Description
N	The number of port calls. $N+1$ refers to the port of origin.
d_i	The distance in nautical miles from port i to port $i+1$.
x_i	The disruption time (in hours) occurs to arrive to port i.
s_{const}	The planned constant sailing speed (in knots) at sea.
s_i	The sailing speed (in knots) from port i to $i+1$ with a recovery plan.
s_{min}	The minimum allowed speed.
s_{max}	The maximum allowed speed.
t_i	The planned arrival time (in hours) at port i according to the schedule.
t_i^{port}	The time (in hours) a ship is anchored at port i for loading/unloading.
t_i^{RP}	The time of arrival (in hours) at port i after recovery plan.
f_i	The fuel cost per nautical mile between ports i and $i+1$.
c_i	The cost at port i.

that the delay penalty at the specific port i is linearly correlated with t_i^{port}, which indicates that the longer the loading/unloading time at a port call, the more costly the delay will be.

In the first objective, the constraint $t_i^{RP} - t_i \geq 0$ prevents the vessel from arriving earlier to port than originally scheduled. In fact, arriving earlier than scheduled to a port call is often not beneficial, since the vessel has to usually wait for the logistical services at the port call, which are serving earlier ship arrivals. It may add to the port congestion as well. Therefore, the vessel would save in bunker fuel by not rushing more than necessary. The second constraint limits the values of the speed vector to the range $[s_{min}, s_{max}]$, which is set based on the vessel type and other criteria such as traffic density, etc. In our experiments, these two bounds are set to 16 and 24 knots respectively, except in the second experiment, which examines different steaming ranges.

Also, the scheduled speed (s_{const}) is set to s_{min}.

In the Equations (3.1) and (3.2), the scheduled time of arrival at the port i (t_i) and the time of arrival after the recovery plan (t_i^{RP}) are calculated as follows:

$$t_i = \sum_{k=1}^{i-1} t_k^{port} + (d_k/s_{const}) \tag{3.3}$$

$$t_i^{RP} = \sum_{k=1}^{i-1} t_k^{port} + (d_k/s_k) \tag{3.4}$$

The *total delay* objective is the accumulation of delays throughout the voyage after enacting the proposed speed-based recovery plan. If the delay at a port call is not completely recovered, it may propagate to the next port calls.

3.3 Synthetic S-VSRP Instance Generation

The synthetic instances were generated using a list of 153 major worldwide ports with a navigational distance matrix between them, which computed through a proprietary routing algorithm developed by Larus Technologies [40]. To generate an S-VSRP instance, the user specifies the desired number

Algorithm 1: S-VSRP Instance Generation

Input: **N**: number of port calls; **cn**: continent number ($cn = 1, \ldots, 6$); $\mathbf{S_{max}}$: maximum speed; $\mathbf{S_{min}}$: minimum speed; **C**: port costs; $\mathbf{maxt_i^P}$: maximum port time;
Output: instance.csv, instance.html
1 ports = *randomMajorPortSelection*(N, cn);
2 navigationRoutes = *LarusAPI*(ports);
3 instance.html = *createHTMLFile*(ports, navigationRoutes);
4 instance.csv = *createCSVFile*(ports);
for $i = 1$ *to* N **do**
  d_i = navigationRoutes[i] ▷ distance between port i and $i+1$;
  $t_i^p = \text{Uniform}(0, maxt_i^p)$ ▷ time for loading/unloading at port i;
  $c_i = \text{Normal}(0, average(C))$ ▷ cost at port i;
  $z_i = d_i \times (S_{max} - S_{min})/S_{max} \times S_{min}$ ▷ maximum disruption that can be recovered within the speed limits;
  $x_i = \text{Uniform}(0, z_i)$ ▷ disruption to arrive at port i ;
  instance.csv.append(d_i,t_i^p, c_i, x_i, s_{min}, d_i/s_{min})
5 **return** instance.csv, instance.html

of ports (i.e., random ports) in each continent, speed bounds, maximum port time and port cost. The process to generate S-VSRP instances is described in Algorithm 1. The map for the route between selected ports is generated in line 3. In the loop, all information for traveling from one port to the next port are generated and added to a CSV file. To generate a random disruption value (x_i) in each leg the maximum disruption value that can be recovered is calculated (z_i) and used in a uniform distribution. A sample instance is shown in Table 3.2, which is associated to the instance in Figure 3.1.

We generated 230 synthetic S-VSRP instances for experimental purposes, which can be downloaded from here[1]. Each instance denotes a voyage. We have generated 10 instances for different number of port calls, which are

[1] `https://github.com/fchgithub/S-VSRP-Instances`

$\{5, 10, 15, 20, 25, 30, 35\}$ in three scenarios introduced in Section 3.4.1.

Table 3.2: An S-VSRP instance with six-port-calls.

From	To	t_i	d_i	c_i	t_i^p	x_i	s_i
New Haven	Halifax	34	544	268	9	3	16
Halifax	Brest	155	2480	276	8	18	16
Brest	Le Havre	16	256	222	6	2	16
Le Havre	Felixstowe	11	176	218	9	0	16
Felixstowe	Bremerhaven	19	304	176	9	5	16
Bremerhaven	New Haven	223	3568	283	3	48	16

3.4 Experimental Results

This section elaborates on the empirical methodology and evaluation of our proposed MOO solution for the S-VSRP problem. The experimental setup starts with defining three S-VSRP scenarios for the experiments. Then, reference set generation and parameter tuning approaches are discussed. It is followed by the performance metrics and statistical tests employed to validate the results. Comprehensive experimental evaluation is reported later in this section.

3.4.1 Experimental Setup

Synthetic Scenario Generation

We investigate three scenarios for the experimentation as presented in Table 3.3. The scenarios are governed by three various problem parameters: i) *route size*, i.e., different number of port calls in the vessel's voyage. This scenario measures the scalability of the MOO techniques when solving S-VSRP instances. ii) *vessel steaming policy*, which compares slow steaming vs. normal steaming policies. The former policy allows vessels to navigate between 18 and 20 knots, which invloves running ship engines below their capacity to save on fuel consumption, but at the expense of additional travel time, particularly over long distances. The second policy is referred to as "normal steaming" and represents the optimal cruising speed a container ship and its engine has been designed to travel at, which is between 20 to 25 knots [54, 78]. Lastly, the third scenario contrasts, iii) *transoceanic* versus *regional voyages*, i.e, comparison of intercontinental and intracontinental voyages. This scenario explores the effects of long and short distance voyages on the S-VSRP problem.

Table 3.3: Different S-VSRP instance scenarios.

Scenario	Variable	Experimental Goal
Route Size	Number of port calls in the route	Scalability analysis
Speed Limit	Speed boundaries	Comparing two vessel steaming policies
Voyage Type	Total distance between the ports	Comparing two types of voyages

3.4.2 MOEA Algorithms for S-VSRP

Various components of the MOEAs that tackle S-VSRP instances i.e., the solution encoding, genetic operators, and benchmark algorithms used in the experiments are discussed in the following subsections.

Benchmark MOEA Algorithms

We examine the performance of six MOEAs on S-VSRP. We consider NSGAII [19], which is based on Pareto dominance and a crowding distance operator from the GA category, and its successor NSGAIII [20], which follows the same idea behind NSGAII but also uses reference planes to drive the population towards good solutions in many-objective optimization problems. Next is the Strength Pareto Evolutionary Algorithm 2 (SPEA2) [81], which is a GA extension and implements a strength function as a combination of the degree to which a candidate solution is dominated (strength) and an estimation of density of the Pareto front as an assigned fitness. The fourth algorithm is Pareto Envelope-based Selection Algorithm II (PESA-II) [16],

which uses the GA mechanisms together with the selection based on Pareto envelope. It leans on a region-based approach, whereby the solution space is broken up into regions and solutions are chosen in order to keep the best and maintain a healthy spread across regions. The fifth algorithm is called Generalized Differential Evolution 3 (GDE3) [39], which is an extension of DE for global optimization with an arbitrary number of objectives and constraints and selection is crowding distance based. Lastly, the Random Search (RS) algorithm simply randomly generates new solutions uniformly throughout the search space. It is not intended as an "optimization algorithm" *per se*, but as a way to compare the performance of other MOO algorithms against RS.

Solution Encoding

A S-VSRP solution is encoded as a real-valued vector representing a set of speeds $s_i \in [s_{min}, s_{max}]$, for $i = 1, 2, ...N$, where N is the number of port calls in a voyage. Each s_i denotes the sailing speed between port i and port $i+1$ and is drawn from a uniform distribution $U(s_{min}, s_{max})$. Each solution is in the form of $(s_1, s_2, \cdots, s_N)$.

A solution is feasible when all elements in the speed vector are within the

valid range $[s_{min}, s_{max}]$. Also, none of the speed values should cause arriving at a port call earlier than scheduled. MOEA Framework [77], used in the implementation, always generates the variables in the valid range for real-valued solution vectors. Also, the values of speed variables through genetic operations (mutation and crossover) always remain within the valid range. The second constraint is handled through constraint handler in MOEA Framework.

Objective Functions

We used *total loss* in (4.1) and *total delay* in (4.2) as the objective functions in S-VSRP. As a result, each solution is evaluated in terms of these two conflicting objectives, which means that increasing the speeds will maximize the fuel cost ($) and minimize the delay (in hours), whereas reducing the speeds will result in maximum delay while minimizing extra fuel cost yet without avoiding the corresponding port penalizations.

Population Initialization

The population P contains M solutions (i.e., population size), which are $N-$dimensional vectors, where N is the number of port calls in a voyage.

Evolutionary Operators

The genetic operators have been selected based on the availability and compatibility with the MOEA Framework [77] as shown in Table 3.4.

Selection Operator *Binary tournament* [50] was chosen as the selection operator; it randomly selects two individuals from the population without replacement, then selects the best of the two. In the case that the selected solutions share the same rank, either is picked at random.

Crossover Operator *Two-Point Crossover* (2x) is our crossover operator of choice. In 2x, two crossover points are randomly selected and all decision variables between the two points are swapped between the two parents. The two offspring solutions resulting from the crossover are returned. The crossover rate is set via a tuning process. However, GDE3 uses this rate as the DE crossover rate. Also, RS does not apply any crossover operator during its search process.

Table 3.4: The generic operators' parameters

Parameters	Description	Default Value
$2x.rate$	The rate 2x is applied to produce offsprings.	1.0
$pm.rate$	The rate PM operator is applied.	$1/N$
$pm.distributionIndex$	The shape of the offspring distribution	20.0

Mutation Operator *Polynomial Mutation (PM)* is used in our experiments. The PM operator attempts to simulate the offspring distribution of binary-encoded bit-flip mutation on real-valued decision variables. The distribution index controls the shape of the offspring distribution. Larger values for the distribution index generates offspring closer to the parents. All decision variables (speed values) in a solution have an equal chance of mutation.

Stop Criterion

In S-VSRP, the freight shipping company's Operational Control Center (OCC) gathers information about disruptions and makes decisions in real-time on how to best handle them. Very quick response, in order to reduce loss and delay, becomes imperative factor among freight shipping companies. For this reason, a good schedule recovery strategy should be found within 5 seconds; controllers at Maersk have stated that up to 10 seconds is a reasonable response time for a disruption management system dealing with large S-VSRP instances [10]. Since our experiments are mostly concerned with evaluating and comparing the performance of different optimization algorithms on S-VSRP instances of varying sizes, we set the stop criterion to a fixed number

of fitness function evaluations based on the values obtained in the automated parameter tuning process. As a future work, the response time criterion will be considered.

Reference Set Generation

Pareto-based MOO approaches require the Pareto optimal set to evaluate their performance based on that. Since the Pareto optimal set is not known for any S-VSRP instance, we resort to the common practice of using the best-known approximation of the Pareto optimal set as the reference set, which consists of all Pareto optimal solutions produced by several MOEAs. To generate a reference set for each S-VSRP instance in each scenario, we run each of the MOEAs introduced in Section 3.4.2 30 times and merge all their Pareto fronts into an archive, leaving only the non-dominated solutions at the end. An example of approximation sets for a five-port-call instance is shown in Figure 3.2.

Table 3.5: Algorithms Parameter Ranges for Tuning.

Parameter	**Min Value**	**Max Value**
Number of Function Evaluations (NFE)	50,000	200,000
Population Size (Pop.size)	50	500
Two point Crossover Rate (2x.rate)	0.5	1.0
Polynomial Mutation Rate (pm.rate)	0.0	0.5
Polynomial Mutation Distribution Index (pm.distIndex)	20.0	40.0

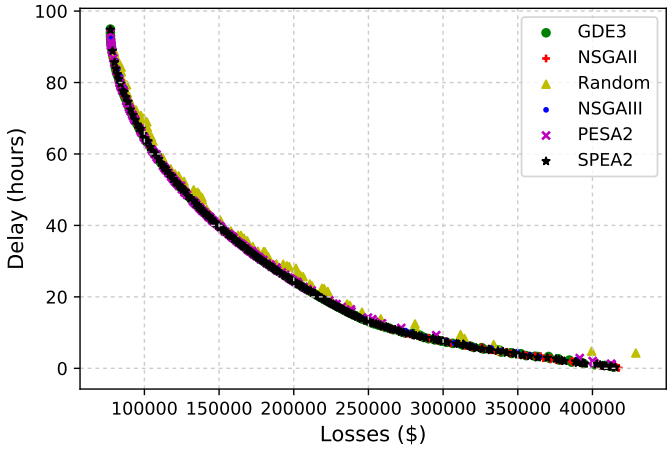

Figure 3.2: Approximation sets for a 5-port-call S-VSRP instance using six different algorithms.

Performance Metrics

Six well-known MOO performance metrics plus running time are selected in this work and outlined in Table 3.6. These metrics are indicators that judge the value of an obtained Pareto approximation set from different angles. The reported metric values are the averages, over 30 independent runs of the algorithm execution upon each synthetic S-VSRP instance type.

Algorithm Parameter Tuning

The parameters used in any optimization algorithm play a significant role in their behaviour in terms of accuracy and convergence. To determine the best parameter values and eliminate any assumption or bias, we conduct paramet-

Table 3.6: Selected Algorithm Performance Metrics

Name	Goal	Description
Hypervolume (HV)	Max	Represents the volume of the objective space dominated by solutions in the approximation set. [15]
Max Pareto Front Error (MPE)	Min	Represents the maximum distance from solutions in an approximation set to the nearest solution in the reference set.[15]
Spacing (SPC)	Min	Represents the uniformity of the Pareto approximation set.
Generational Distance (GD)	Min	Represents average distance from solutions in the approximation set to the nearest solution in the reference set.[15]
Inverted GD (IGD)	Min	Represents the average distance from solutions in the reference set to the nearest solution in the approximation set.[62]
$\epsilon-$Indicator (EPI)	Min	Measures the consistency of an approximation set. [32]
Running Time (RT)	Min	The running time (in seconds) of the algorithm.

ric tuning for each MOEA. The parameters are generated within predefined ranges as listed in Table 3.5. In our tuning stage, 243 parameter vectors (corresponding to different combinations of Number of Function Evaluations (NFE), population size, crossover, and mutation rates) were generated using the Saltelli sampling method [61].Since instance structures change with a varying number of port calls and the size of search space varies as well, we tuned instances with a different number of port calls separately. Each algorithm was run 15 times using each parameter vector over a randomly selected S-VSRP instance with a specific number of port calls. Then, the performance metric values were averaged across all the runs.

To determine the best parameter vector for each algorithm, we collected four metrics from Table 3.6, which are the Generational Distance (GD), Hy-

pervolume (HV), $\epsilon-$Indicator (EPI), and RT. The first three metrics measure the main characteristics of a MOEA implementation, which are proximity, diversity, and consistency of the approximation set, respectively [28]. Due to the importance of the response time in the field, the running time was considered as well. The average vectors of all runs were ranked from one (best) to 243 (worst) for each performance metric, then we took the weighted average of these ranks using 0.3, 0.3, 0.3, and 0.1 as weights, respectively. Assigning a lower weight to running time and equal weights to other three metrics is due to giving the highest chance to the algorithms to find solutions with the highest quality while compensating on these performances (i.e., lower the three metrics' weights and increase the running time weight) may lead to solutions in a shorter time. The vector with the highest average rank across all metrics was deemed the best set of parameters for the given S-VSRP instance.

The tuned parameter vectors for different algorithms with regards to the number of port calls are presented in Table 3.7. We used these tuned parameters in all three experimental scenarios.

It should be noted that since GDE3 requires DE-based operators for crossover and mutation, it converts the tuned operator rates to its own opera-

tor rates i.e. de.crossoverRate and de.stepSize. This means the pm.distIndex is never used by this algorithm.

Statistical Validation

To confirm whether the differences in observed performance are statistically significant, we rely on nonparametric tests as suggested by [22]. We use 5% significance level for all tests. Due to the novelty of the problem area, we do not have a control algorithm. Therefore, we must first verify the existence of statistically significant differences within the whole group of algorithms using the Friedman $N \times N$ procedure. In the case that the comparison suggests a rejection of the null hypothesis (i.e., there exists a statistically significant difference), then a set of pairwise comparisons is carried out using the following post-hoc procedures: Bergmann, Nemenyi, Holm, and Schaffer [22]. We first rely on the Bergmann procedure given its robustness and reliable adjustment of the $p-$value. If the Bergmann procedure does not reject a hypothesis we defer to the other post-hoc methods. The open source software KEEL (Knowledge Extraction based on Evolutionary Learning) was used to run the aforementioned statistical analyses [3].

Table 3.7: Tuned algorithm parameter values with different number of port calls.

Tuned Parameters: 5 port-call instances					
Algorithm	**NFE**	**Pop.Size**	**2x.rate**	**pm.rate**	**pm.distIndex**
GDE3	87646	478	0.9	0.02	N/A
RS	162646	253	0.65	0.26	38
NSGAII	176708	487	0.82	0.4	33
NSGAIII	176708	487	0.82	0.4	33
PESA2	106396	309	0.59	0.29	35
SPEA2	176708	487	0.82	0.4	23
Tuned Parameters: 10 port-call instances					
GDE3	167333	239	0.57	0.18	N/A
RS	54833	149	0.88	0.31	28
NSGAII	162646	360	0.65	0.26	38
NSGAIII	162646	360	0.65	0.26	38
PESA2	143896	56	0.74	0.26	31
SPEA2	162646	360	0.65	0.26	38
Tuned Parameters: 15 port-call instances					
GDE3	181396	394	0.86	0.13	N/A
RS	97021	473	0.77	0.35	37
NSGAII	129833	351	0.63	0.27	38
NSGAIII	148583	431	0.63	0.21	20
PESA2	148583	431	0.63	0.21	20
SPEA2	148583	431	0.63	0.21	20
Tuned Parameters: 20 port-call instances					
GDE3	167333	239	0.57	0.18	N/A
RS	50146	140	0.77	0.14	23
NSGAII	181396	394	0.99	0.1	39
NSGAIII	143896	422	0.96	0.07	20
PESA2	186083	70	0.75	0.25	21
SPEA2	143896	422	0.96	0.07	20
Tuned Parameters: 25 port-call instances					
GDE3	73583	183	0.57	0.12	N/A
RS	134521	360	0.65	0.47	22
NSGAII	186083	70	0.94	0.25	25
NSGAIII	82958	70	0.94	0.25	21
PESA2	101708	262	0.57	0.15	23
SPEA2	50146	140	0.77	0.14	23
Tuned Parameters: 30 port-call instances					
GDE3	195458	183	0.88	0.46	N/A
RS	162646	253	0.65	0.47	38
NSGAII	181396	394	0.99	0.1	39
NSGAIII	195458	206	0.88	0.12	30
PESA2	101708	262	0.57	0.15	23
SPEA2	153271	422	0.96	0.07	20
Tuned Parameters: 35 port-call instances					
GDE3	87646	135	0.9	0.01	N/A
RS	111083	295	0.69	0.0	35
NSGAII	143896	422	0.96	0.07	20
NSGAIII	82958	70	0.94	0.25	21
PESA2	101708	262	0.57	0.18	23
SPEA2	50146	140	0.89	0.14	23

3.4.3 Experiment 1: Scalability Analysis

In the first experiment, the scalability of the MOEAs is evaluated by measuring their performance with different metrics on S-VSRP instances with different sizes (number of port calls). The number of port calls determines the length of problem variables (speed vector). We run the experiment on 70 instances (10 instances for each of $\{5, 10, 15, 20, 25, 30, 35\}$-port calls). The instance generation process was described in Section 3.3.

Among the measured performance metrics listed in Table 3.6, GD, HV and EPI (described in Section 3.4.2) are illustrated and analyzed. Since they are used in the tuning process, the sensitivity of the algorithms to the tuned parameters can be better understood. The 95% confidence intervals and standard deviations of reported average results have been computed, but because the charts in this experiment become cluttered, we added them to the same GitHub repo where the synthetically generated S-VSRP instances have been published (here [2]) . All seven metrics are considered in the statistical validation of the performance differences among the algorithms. In the following plots, where RS or SPEA2 values hide the value of other algorithms, a separate plot with excluded algorithm is provided.

[2] `https://github.com/fchgithub/S-VSRP-Instances`

The HV measurements are shown in Figure 3.3. It can be seen that as we increase the number of port calls the HV values decrease. This is important to note that as the size of the decision space (speed variables) grows, the performance of each algorithm drops. In fact, searching inside a larger space with a limited number of fitness function evaluations would likely not succeed.

Not all the algorithms have an explicit mechanism that encourages the exploration of a large solution space. The RS algorithm follows no mechanism to explore the solution space (i.e., randomly creates a new set of solutions in each iteration and adds them to a non-dominated archive) and the poor performance captured in Figure 3.3 can be expected. This is true with respect to other metrics as well. RS is thus a benchmark for evaluating the performance of the remaining MOEA algorithms against randomness. Moreover, it can be seen that up to 20 ports, the HV differences remain fairly small among all the algorithms. However, for a higher number of ports NSGAII shows better performance compared to the others.

It can be noticed in Figures 3.4, 3.5 and 3.6, 3.7 that there is no significant difference among the MOEA techniques in terms of GD (measure of proximity) and EPI (measure of consistency) over the different number of port calls (the values are very small), except for the RS algorithm.

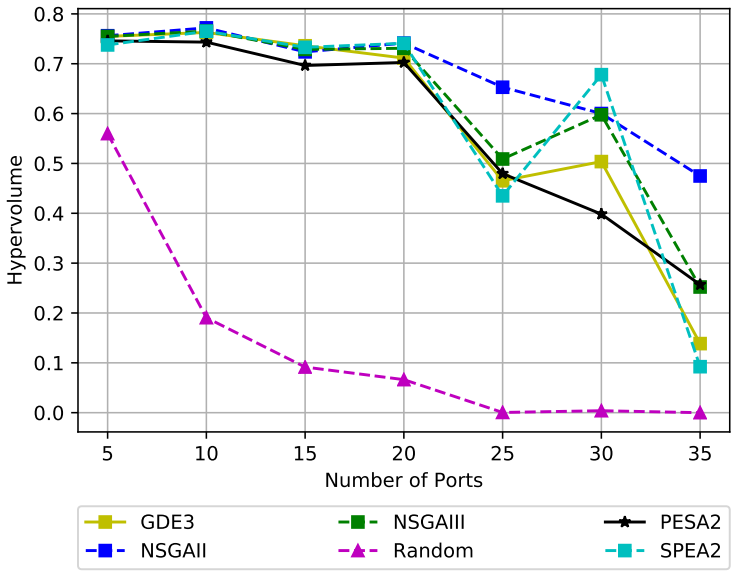

Figure 3.3: Average HV values over 30 runs on a different number of port calls.

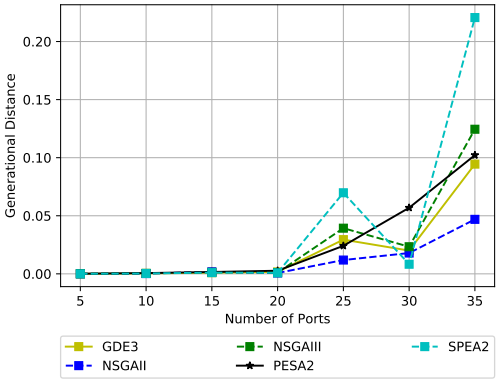

Figure 3.4: Average GD values over 30 runs on a different number of port calls without RS.

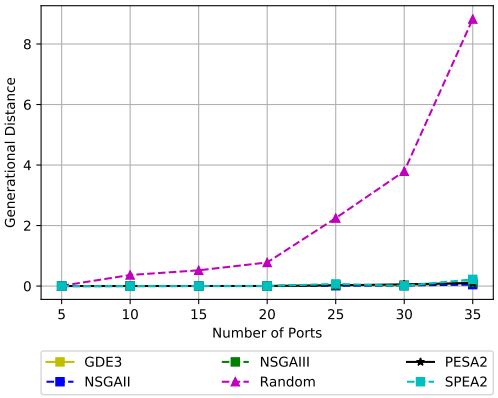

Figure 3.5: Average GD over 30 runs on a different number of port calls.

The running time of the algorithms versus the number of ports is depicted in Figures 3.8 to 3.9. Since SPEA2 poorly performs along computational time, its high values hide others in the plots. For this reason, it has been

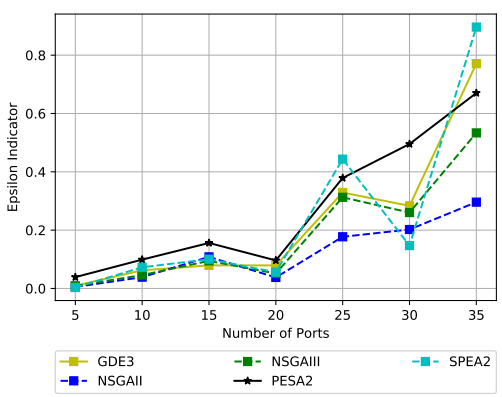

Figure 3.6: Average $\epsilon-$indicator over 30 runs on a different number of port calls without RS.

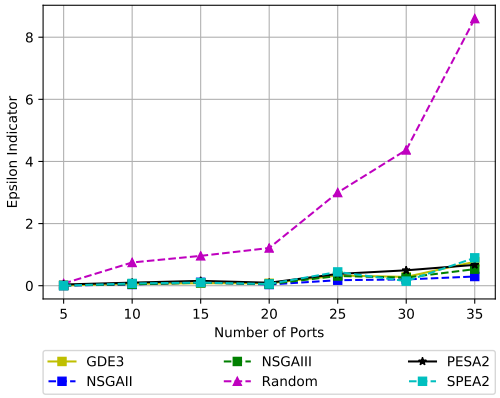

Figure 3.7: Average $\epsilon-$indicator over 30 runs on a different number of port calls.

shown in a separate Figure 3.9. In Figure 3.8, the GDE3 running time for five and 15 ports is explained by the parameter tuning values shown in Table 3.7. The running time of GDE3 algorithm gets highly affected by a large population size comparing to other algorithms.

As shown in Figure 3.9, SPEA2 which uses a density-based ranking mechanism to choose the solutions from less crowded regions takes very long time to run. This is due to its need to compute the distance matrix containing the pair-wise distances between solutions in the objective space and the density-based strength evaluation of the solutions. Except for SPEA2 and RS (i.e., low running time is due to its simple mechanism in generating not high-quality solutions), other algorithms have acceptable running times. Smaller response times can be achieved by giving more weight to running time in

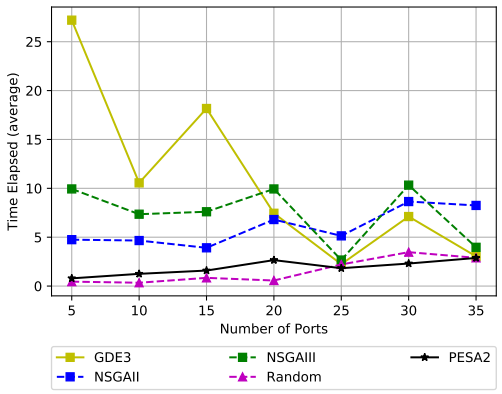

Figure 3.8: Average RT without SPEA2

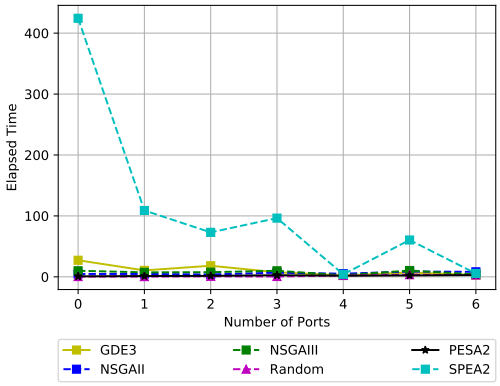

Figure 3.9: Average RT with SPEA2

parameter tuning process and compensating slightly on other performances. Table 3.8 gives an overview of the statistical analysis carried out for each of the recorded metrics. The small p-values indicate that there exist statistically significant performance differences among the group of algorithms under each metric. Table 3.9 displays those pairwise comparisons not rejected, which means none of the post-hoc procedures were able to detect a significant difference from the observed results.

Table 3.8: Experiment I - Friedman $N \times N$ Test Results

	Algorithm Rank						
Metric	**NSGAII**	**NSGAIII**	**SPEA2**	**PESA2**	**GDE3**	**RS**	p**-value**
EPI	**1.75**	2.4643	2.9643	4.4643	3.3571	6	4.7091E-11
GD	**2.0357**	3.3929	2.6429	3.75	3.1786	6	4.8926E-11
HV	**1.7857**	2.75	2.9643	4.0714	3.4464	5.9821	4.5985E-11
IGD	**1.6429**	2.6786	3.1071	4.25	3.3214	6	4.8235E-11
MPE	**1.9643**	3.0357	3.2857	2.9643	3.75	6	7.1843E-11
SPC	2.6429	3.8929	**2.3214**	3.1786	3.3571	5.6071	1.7730E-10
RT	3.8929	4.3571	6	2.1429	3.4643	**1.1429**	8.9343E-11

Regarding the post-hoc results, it is interesting to note that NSGAIII vs.

Table 3.9: Experiment I - Pairwise comparisons of algorithms not rejected by post-hoc test

Algorithms	HV	GD	IGD	SPC	MPE	EPI	RT
NSGAII vs. RS							
NSGAIII vs. RS							
RS vs. SPEA2							
GDE3 vs. RS							
NSGAII vs. PESA2				✓			
PESA2 vs. RS							✓
GDE3 vs. NSGAII		✓		✓			✓
NSGAIII vs. PESA2		✓		✓			
NSGAII vs. SPEA2	✓	✓		✓	✓	✓	
PESA2 vs. SPEA2	✓	✓	✓	✓			✓
NSGAII vs. NSGAIII	✓		✓		✓	✓	
GDE3 vs. PESA2	✓	✓	✓	✓	✓	✓	
GDE3 vs. NSGAIII	✓	✓	✓	✓	✓	✓	✓
NSGAIII vs. SPEA2	✓	✓	✓		✓	✓	
GDE3 vs. SPEA2	✓	✓	✓	✓	✓	✓	

GDE3 is not rejected by any of the tests, which means there is no significant difference between them statistically. For those metrics, which are rejected by the tests, we can refer to the Friedman ranking and pick the algorithm with the smaller rank as the one performing better. According to the Friedman test, NSGAII can be the choice of the algorithm in this experiment, due to its best ranks in the most of the metrics.

3.4.4 Experiment 2: Vessel Steaming Policies

This experiment examines the impact of changing the steaming policy on the performance of the MOEAs when solving S-VSRP instances. We consider two policies: one from the "slow steaming" policy ranging from 18 to 20 knots and the other one from the "normal steaming" policy ranging from 20 to 25

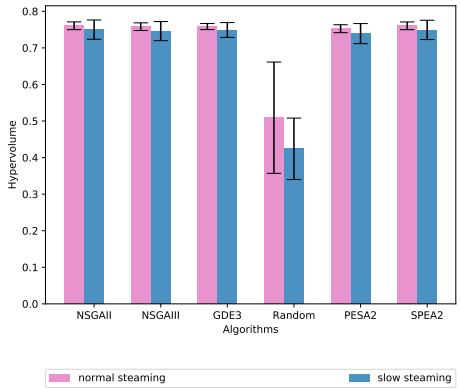

Figure 3.10: Average HV over 30 runs for the vessel steaming policies experiment.

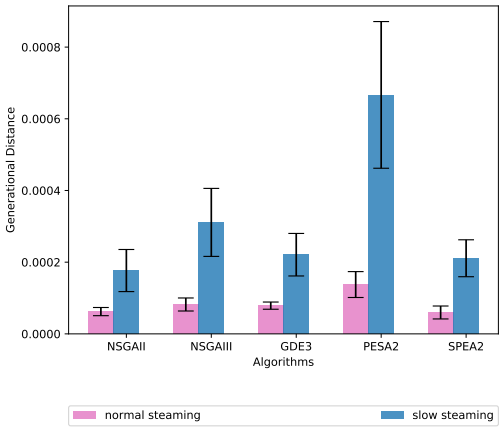

Figure 3.11: Average GD over 30 runs (without RS) for the Experiment 2.

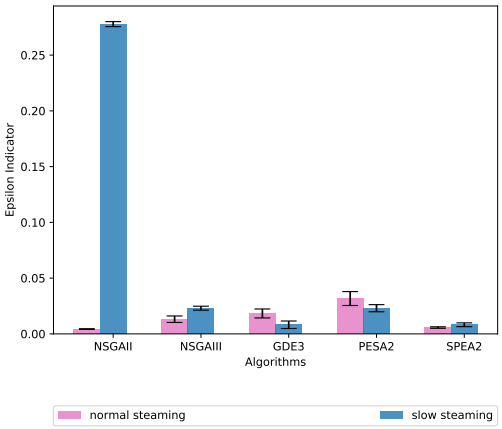

Figure 3.12: Average EPI over 30 runs (without RS) for the Experiment 2.

knots. The generation of synthetic instances is a function of speed limits in two policies such that within the speed limit a vessel will try to recover from the disruptive event (refer to z_i in Algorithm 1). To compare the policies, we show results by running the algorithms on instances with 10 port calls in this experiment. Figures 3.10, 3.11, 3.12, 3.13 and 3.14 portray the HV, GD,

EPI and RT metrics respectively against the instances from the two vessel steaming policies under consideration with 95% confidence intervals for each measurement. Standard deviation values can be found in the same GitHub repo introduced before.

Figure 3.10 reveals that all algorithms in the "normal steaming" policy perform better than "slow steaming". The GD values are reported in Figure 3.11. In this figure 3.11 and Figure 3.12, we did not include RS because its high values, which conceal other algorithm values (0.098 in normal steaming and 0.072 in slow steaming for GD, and 0.28 and 0.32 for EPI). Similar to HV metric, normal steaming can find solutions closer to the reference set because of the larger range of available speed values. Similar behaviour can be seen for $\epsilon-$indicator in Figure 3.13 as well. We can conclude that the problem in normal steaming is easier to solve compared to slow steaming. This is due to a larger boundary for speed limits in this policy, which makes it simpler to find a wider range of feasible solutions and therefore results in a better quality of approximation sets in terms of these metrics.

The running time results in Figure 3.14 unveil GDE3 and SPEA2's sensitivity to the size of the decision space. The average running time is close to 10 seconds in the scenario I in Figure 3.8, whereas there is a big difference

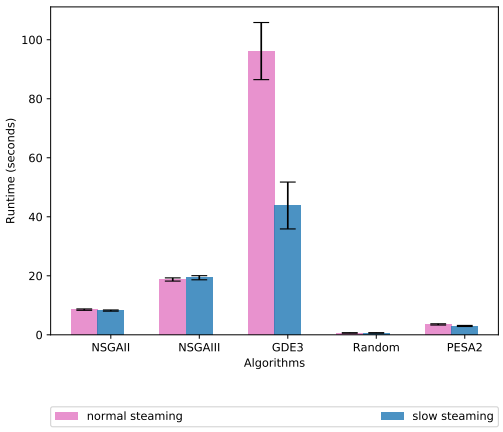

Figure 3.13: Average RT for the vessel steaming policies experiment without SPEA2.

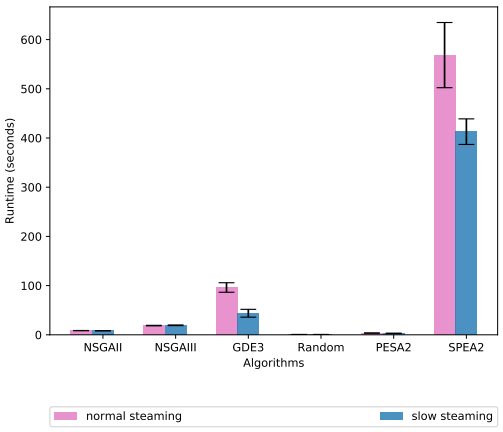

Figure 3.14: Average RT over 30 runs for the vessel steaming policies experiment.

for the same algorithms with the instances designed for scenario II.

Table 3.10: Experiment II - Friedman $N \times N$ Test Results

	Algorithm Rank						
Metric	**NSGAII**	**NSGAIII**	**SPEA2**	**PESA2**	**GDE3**	**RS**	***p*-value**
EPI	2.2917	3.25	**2.2083**	4.125	3.125	6	6.3600E-11
GD	2.875	3.5	**1.8333**	4	2.7917	6	6.6433E-11
HV	2.4583	3.5833	**2.2083**	3.875	2.875	6	4.9573E-11
IGD	2.4167	3.4167	**2.2917**	3.8333	3.0417	6	6.1480E-11
MPE	2.6667	3.6667	**2.4167**	3.5417	2.7083	6	5.3074E-11
SPC	2.4167	3.4583	**1.0833**	4.875	3.1667	6	7.1172E-11
RT	3.1667	4.1667	5.875	2.625	4.1667	**1**	8.2292E-11

Table 3.10 gives an overview of the statistical analysis carried out for each of the recorded metrics. The small p-values indicate that the results of the analysis are statistically significant for the group of algorithms under each metric. Table 3.11 lists the pairwise comparisons for which the null hypothesis was not rejected across each metric. These statistical analysis were calculated on three different port calls (i.e., 10, 15, and 20 port calls)

with 30 instances in each policy.

Table 3.11: Experiment II - Pairwise comparisons of algorithms not rejected by post-hoc test

Hypothesis	HV	GD	IGD	SPC	MPF	EPI	RT
Random vs .SPEA2							
NSGAII vs .Random							
GDE3 vs .Random							
NSGAIII vs. Random							
PESA2 vs. Random				✓			
PESA2 vs. SPEA2					✓		
NSGAII vs. PESA2	✓	✓	✓		✓		✓
NSGAIII vs.SPEA2	✓		✓		✓	✓	
NSGAII vs. NSGAIII	✓	✓	✓	✓	✓	✓	✓
GDE3 vs. PESA2	✓	✓	✓		✓	✓	
GDE3 vs. NSGAIII	✓	✓	✓	✓	✓	✓	✓
GDE3 vs. SPEA2	✓	✓	✓		✓	✓	
GDE3 vs. NSGAII	✓	✓	✓	✓	✓	✓	✓
NSGAIII vs. PESA2	✓	✓	✓		✓	✓	
NSGAII vs. SPEA2	✓	✓	✓		✓	✓	

The post-hoc test in Table 3.11 shows that the algorithms behave not significant on instances in two speed policies in many metrics. Also, SPEA2 gets the best algorithm ranks in all metrics expect for running time, which is the worst among all six algorithm. The second best algorithm option for this experiment is NSGAII, which is similar to experiment I.

3.4.5 Experiment 3: Voyage Distance Analysis

In the last experiment, the effect of transoceanic vs. regional voyages on the performance of the MOEAs is examined. Transoceanic voyages are those containing some long-distance trips between the port calls, while regional ones are those with short-distance trips. In this experiment, 30 North-America-

to-Europe S-VSRP instances are compared to 30 Europe-only instances. The former has instances planned for a trip with the total of 26,358.48 nautical miles, while the latter is concerned with a voyage inside Europe with the length of 13,264 nautical miles. Three different metrics (i.e. HV, GD, EPI) are presented in Figures 3.15, 3.16, 3.17 and 3.18, respectively against instances in two different types of voyages with 95% confidence interval for each measurement. The same as other scenarios, standard deviation values can be found in the GitHub repo.

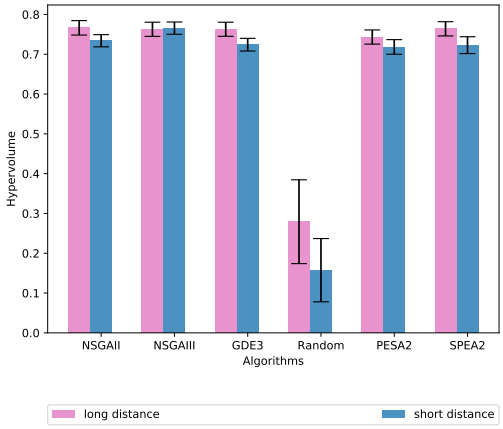

Figure 3.15: Average HV over 30 runs across all generated scenarios for the different voyage type experiment.

As illustrated in the figures, all three performance metrics show superior results on long distance voyages vs. shorter trips. This proves the important fact that finding a recovery plan is easier in long-distance trips. Longer trips are easier to optimize given the longer time horizon between two port

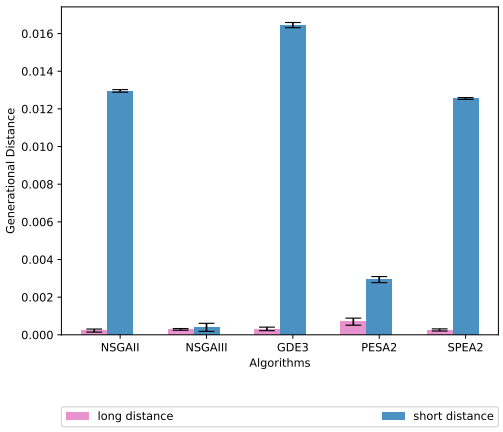

Figure 3.16: Average GD for the different distance voyages experiment without RS.

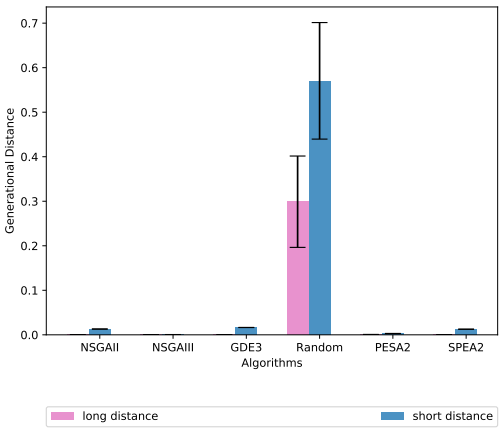

Figure 3.17: Average GD over 30 runs for the different voyage type experiment.

calls, which allows for a higher speed manoeuvrability. Also, despite the decision space stays the same (the speed range values are the same for both categories), yet the objective space expands due to the fact that both delay and loss objectives are directly a function of the distance between ports in the route. Therefore, an instance with longer port call distances can find a larger range of solutions compared to an instance with the same speed limits but shorter distances between the port calls. This provides a larger feasible objective space to compete for non-dominated solutions.

Figure 3.19 and 3.20 reveal that running time (RT) across two scenarios is affected by the distances between the port calls. Among all algorithms, as expected from the previous experiments, SPEA2 is grossly outperformed by all other algorithms under consideration. Given S-VSRP's time-sensitive

nature , SPEA2 might not be a suitable choice owing to its long running time and regardless of the quality of its solutions.

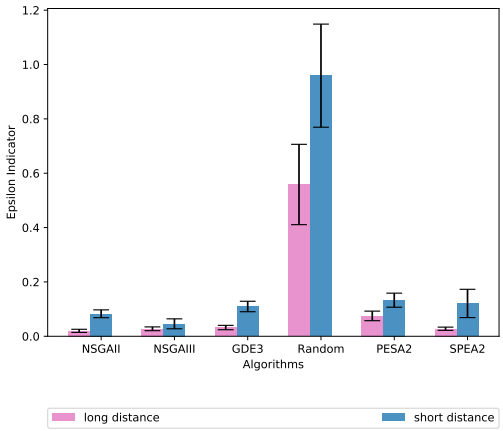

Figure 3.18: Average EPI over 30 runs for the different voyage type experiment.

The statistical analysis by N×N Friedman test carried out for each of the recorded metrics, showed in Table 3.12, rejects the null hypothesis on all metrics. Table 3.13 lists the pairwise comparisons for which the null

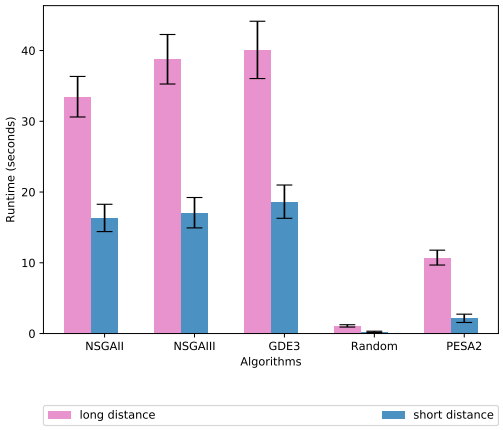

Figure 3.19: Average RT for the different voyage type experiment without SPEA2.

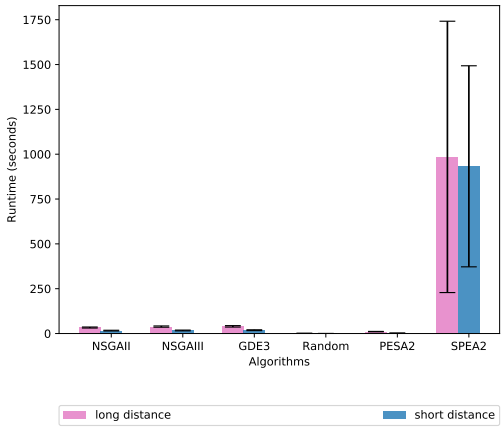

Figure 3.20: Average RT over 30 runs for the different voyage type experiment.

Table 3.12: Experiment III - Friedman $N \times N$ Test Results

	Algorithm Rank						
Metric	**NSGAII**	**NSGAIII**	**SPEA2**	**PESA2**	**GDE3**	**RS**	***p*-value**
EPI	**1.9167**	2.6667	3.0417	4.5	2.875	6	4.9924E-11
GD	2.7917	3.8333	**1.9583**	3.625	2.7917	6	5.6166E-11
HV	**2.1667**	3.4583	2.4583	4.125	2.7917	6	7.4303E-11
IGD	**2.1667**	3.2083	2.7917	4.125	2.7083	6	7.4560E-11
MPE	2.6667	3.7917	2.9167	**2.4167**	3.2083	6	6.8121E-11
SPC	2.25	3.875	**1.7083**	4.375	2.7917	6	5.6524E-11
RT	3.3333	4.5417	6	2	4.125	**1**	6.4875E-11

Table 3.13: Experiment III - Pairwise comparisons of algorithms not rejected by post-hoc test

Hypothesis	**HV**	**GD**	**IGD**	**SPC**	**MPF**	**EPI**	**RT**
Random vs .SPEA2							
NSGAII vs .Random							
GDE3 vs .Random							
NSGAIII vs. Random							
PESA2 vs. Random							✓
PESA2 vs. SPEA2			✓		✓		
NSGAII vs. PESA2		✓			✓		
NSGAIII vs.SPEA2	✓	✓	✓	✓	✓	✓	
NSGAII vs. NSGAIII	✓	✓	✓		✓	✓	✓
GDE3 vs. PESA2	✓	✓			✓		
GDE3 vs. NSGAIII	✓	✓	✓	✓	✓	✓	✓
GDE3 vs. SPEA2	✓	✓	✓	✓		✓	
GDE3 vs. NSGAII	✓	✓	✓	✓	✓	✓	✓
NSGAIII vs. PESA2	✓	✓	✓		✓		
NSGAII vs. SPEA2	✓	✓	✓	✓	✓	✓	

hypothesis was not rejected across each metric.

From the statistical analysis results, it can be seen that the most of algorithms except for RS have no significant different performances in the two categories. NSGAII can still be recommended as the most suitable algorithm for this type of experiment.

3.5 Summary

In maritime disruption management, we defined the S-VSRP as a bi-objective optimization problem that aims at optimizing the delay and loss objectives. To the best of our knowledge, this is the first multiobjective formulation of VSRP as compared to other single-objective formulations considering the cost in the literature. A Pareto-based MOO formulation gives the decision maker the ability to inspect the trade-off between two objectives. A comprehensive study on several popular MOEAs was done to find good solutions to a variety of synthetically generated S-VSRP instances. We examined the problem under three scenarios. In Scenario I, the scalability of the multiobjective optimizers under consideration was evaluated. Scenario II considered two vessel steaming policies (slow steaming and normal steaming) and proved that normal steaming is more successful in finding the solutions compared to slow steaming. In the last scenario, transoceanic voyages were compared to regional ones. The simulations revealed that delays in long-distance voyages are easier to mitigate compared to short-distance voyages. Moreover, the sensitivity of the optimizers in different scenarios to the parameter tuning was seen in the experiments. In three experiments, according to obtained

results and statistical analyses, we found NSGA-II as the winner MOEA among the six algorithms we examined.

We list some limitations of this work, which can be considered as potential future works. First, we only considered a speed-based recovery strategy, which can be extended by combining other strategies such as port swapping or port skipping, as this can increase the applicability of the work. Also, our approach can be extended to the entire fleet instead of to a single vessel. We also did not strictly meet the real-time requirements of the problem at hand. Furthermore, parameter tuning for the MOEAs was not performed per S-VSRP instance due to its costly process. However, in the future, instance-specific parameter tuning should be done as it is the most effective way to capture the particular characteristics of each S-VSRP instance.

We published an extension to this problem that deals with variable speed limits along the vessel's route [101]. Sailing at different speeds is certainly possible, especially in long voyages, due to various conditions in the route such as weather or traffic. Mining such speeds from data and solving S-VSRP with this data were the main research goals in that work. This casts the big-data-enabled S-VSRP version into the realms of data-driven and large-scale optimization problems, and we seek to solve them with state-of-the-art

techniques designed to that effect.

Chapter 4

Big-Data-Enabled Modelling and Optimization of Granular Speed-based Vessel Schedule Recovery Problem

Seaborne trade constitutes nearly 90% of the volume of global trade and is linked to almost every international supply chain, thus having a major impact on the global economy [46]. Maintaining reliable shipping services is an important challenge for liner shipping companies. In reality, there are uncer-

tainties which affect the quality of these services (i.e., regular uncertainties and disruptions). The regular uncertainties such as variable port productivity, or unexpected wait time for port access occur frequently and can be managed by probabilistically modeling the degree of uncertainty [53], while disruptions such as labor strikes or port closures due to hurricanes or floods are the rare and usually unpredictable uncertainties [10] [79]. Disruptions may cause a severe delay in shipping services which could result in significant financial and reputation losses. To manage the disruption and mitigate the delay impact, dynamically recovering from disruption events is suggested [79]. In disruption management, finding a recovery plan is called a solution for the VSRP. The solution considers four possible actions, or a combination thereof to manage the disruption's consequences [10] [41]. These actions perform recovering by speeding up, swapping the visitation order of the ports in the voyage, skipping the ports along the voyage, or accepting the delay and taking no action.

The AIS is a vessel tracking system that automatically provides updates on a vessel's movement and other relevant ship voyage data to other parties. Each ship equipped with an AIS transponder sends out a packet every few seconds. Dynamic data that is transmitted include position, speed, course,

true heading and rate of turn. Less frequent messages transmit ship static data, draught, and other voyage-related information [2].

With the ever-increasing capability of collecting abundant data from a plethora of sources, according to Ericsson [25], the maritime industry still lags behind alternative transport industries in terms of its use of information and communications technology. In an attempt to fill this gap, we model the VSRP by leveraging information from historical AIS data. We focus our efforts on a particular case of the VSRP problem that only deals with speed adjustments along the vessel route to mitigate disruptions, namely the Speed-based VSRP (S-VSRP). Existing solutions assume the vessel will sail at a constant speed throughout its voyage, which is unlikely and unrealistic. Different factors such as traffic at ports or within major world sea routes or special atmospheric conditions on different geospatial locations force vessels to adapt their speed to these situations.

We formulate the S-VSRP as a three-objective optimization problem. The first objective minimizes the delay time at arriving at the port calls. The second objective is financial loss minimization. The last objective maximizes the compliance of speed-based recovery solutions with the navigational patterns reflected in the historical AIS-based data. Approaching this problem with

metaheuristic methods is because of this fact that metaheuristic techniques are powerful search methodologies to successfully produce good-quality solutions in reasonable computation times and good enough for practical purposes comparing to classical exact methods such as integer programming or linear programming which are often extremely time-consuming when solving real-world problems specially with large dimensions and complex constrained problems.

In this chapter, we make the following contributions: (i) we model S-VSRP as a multiobjective optimization problem that includes vessel speed compliance with historical navigational patterns mined from AIS data; (ii) we mine, aggregate and granulate historical vessel speed data over a set of geohashed regions along the vessel voyage; (iii) we generate synthetic S-VSRP instances and solve this problem via three MOEAs; (iv) we discuss the trade-off among the three objectives in the discovered Pareto fronts and analyze the impact of the optimizer and information granularity (geohash precision) upon the returned solutions. To the best of our knowledge, this is the first time historical AIS data has been exploited in the literature to mitigate disruptions in vessel schedules.

From a granular computing perspective [6] [5], our G-S-VSRP implemen-

tation is likely one of the first proposed *granular optimizers* given that the underlying information granulation over the space of vessel speed bounds in the geohashed regions across the vessel's voyage will impact multiple aspects of the optimization problem such as the dimensionality of the solution vector or the type of infeasibility handling scheme used in the optimizers.

The rest of the chapter is structured as follows: in Section 4.1, relevant works are briefly presented and reviewed. Section 4.2 formalizes the G-S-VSRP problem and Section 4.3 presents the building blocks for the evolutionary MOEA schemes under consideration. In Section 4.4, the synthetic instance generation method is briefly explained. Section 4.5 is concerned with the empirical evaluation of our proposed model. Finally, Section 4.6 concludes the chapter.

4.1 Related Works

VSRP has been studied in a few works recently. Brouer et al. in [10], were the first ones modeling the optimal recovery action in container shipping. They examined three strategies to recover from the given disruption which includes speeding up, omitting port calls and swapping port calls. A *mixed-integer*

programming model was formulated to balance the increased fuel consumption and the impact on cargo flows in the shipping network. By testing the model on real scenarios, they reported up to 58% reduction in costs. Another *Mixed Integer Linear Programming* (MILP) model has been proposed by Qi in [58] which is inspired from aircraft recovery models [67, 68, 23]. The work in [58] is modeled with binary variables based on a graph characterization of the vessel plan. The objective function is the overall cost minimization; this objective is composed of two terms: the first is the cost of vessel operation and the second is the penalty cost from delaying or miss-connecting cargo. Weighted sum scalarization and ϵ-constraint methods have been used to solve this MOO problem. The weighted sum problem is that the relative importance of the objectives is not known a priori and does not allow assessing the trade-off between the objectives. Moreover, if the boundary of the objective space is not convex then some solutions could not be found. The ϵ-constraint method overcomes some of the convexity problems introduced by the weighted sum techniques by minimizing a primary objective and expressing the other objectives in the form of inequality constraints. The latter still does not provide a set of alternative solutions but returns a single solution that minimizes the weighted sum of the objectives.

Li et al. in [45] modeled VSRP via nonlinear programming and dynamic programming. They proposed an optimal operational action to mitigate delays in liner shipping schedules. The three disruption recovery strategies including vessel speeding up, port call omission, and port call swapping were analyzed. Based on their experiments, speeding up was determined to be the most effective when experiencing small delays, however, it was determined that major disruptions required skipping and swapping ports to recover from the delayed schedule. Their approaches are deterministic and do not consider future new delays. In addition, scaling the problem into many port calls is not practical due to the long running time.

Moreover, Li et al. in [44] studied real-time modeling of schedule recovery problem under regular uncertainties and disruption events along the voyage. With the similar idea in our work, they divided the journey between two neighbour ports into multiple segments. They assigned a random variable reflecting its geographical characteristics as the uncertainty probability. They proposed a multi-stage stochastic formulation for the problem. Li et al. in [45] modeled VSRP via nonlinear programming and dynamic programming. They proposed an optimal operational action to mitigate delays in liner shipping schedules.

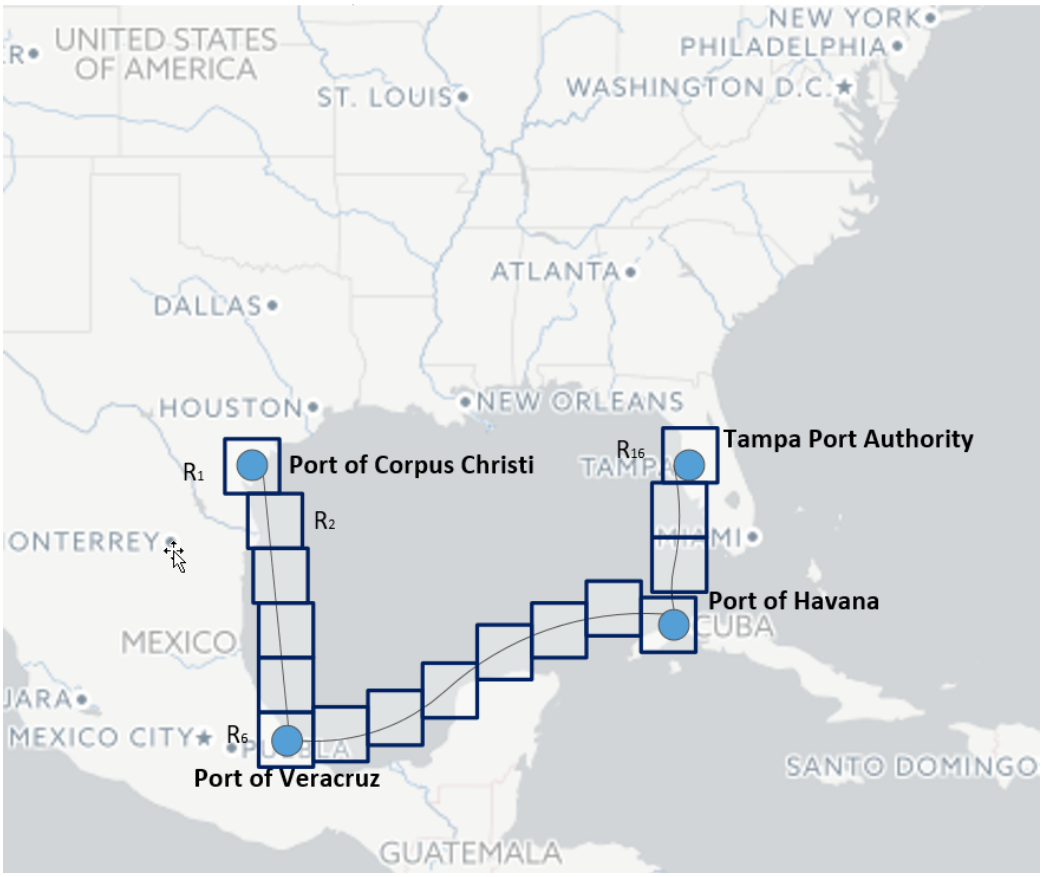

Figure 4.1: The route comprising four ports is represented by geohashed regions. Each region may contain either a port or just a transient location towards the next port call in the route. In this example, R_1, R_6, R_{13} and R_{16} are four port regions, and the remaining regions are transient sea locations.

The opportunities and challenges of using big data in liner shipping are enumerated in [60] [76]. The authors in [60] explain some of the issues they have come across while working on big data and suggest some solutions. They mention many data sources on the ship, potentially high volumes of data and correspondingly many opportunities for errors. They also discuss issues around volume and storage management, data quality and data an-

alytics as three dimensions of big data management. The first dimension considers how data is stored and structured. To deal with errors, data quality control is critical for the correctness of the analysis results. Lastly, data is converted to information via various forms of discovery methods such as machine learning. The authors in [76] put forth a framework to integrate BDA and the IIoT technologies for offshore support vessels based on a high-performance computing platform. Recently, Zhang et al. in [80] developed an analytical approach to analyze ship traffic demand and the spatio-temporal dynamics of ship traffic in port waters using AIS big data. They were able to find the hotspot areas and stable ship activities near Singapore's port waters.

While the above works highlight interesting avenues for exploiting big data in maritime transportation, we have not found any such application within the confines of an optimization problem in the maritime domain. To our knowledge, there is no similar work on VSRP to use AIS big data information in granular speed mining and modeling it as an MOO problem.

4.2 A Granular Multiobjective S-VSRP Formulation

Disruptions during a ship's journey often lead to delays and missing important scheduled deliveries. Approximately $70 - 80\%$ of the vessels experience delays in at least one port during each roundtrip [53]. These delays may happen at sea (e.g., harsh weather) or at ports (e.g., congestion). To manage and mitigate the consequences of delays, a recovery plan should be put in place. The idea behind our proposed G-S-VSRP formulation is that a vessel travels at different speeds throughout the route, even within the same leg (i.e., the distance between two neighbor port calls). This is due to various factors such as maritime traffic or weather conditions. The AIS data source has the capacity to provide us with valuable historical reported speed limits that thousands of vessels have reported in their trajectories worldwide. In our formulation, the trajectory (i.e., route) between ports is divided into sub-regions encoded by the geohashed system invented by Gustavo Niemeyer [133]. In G-S-VSRP, a single value represents the fixed recovery speed the vessel will travel at between two port calls. In our G-S-VSRP formulation, we

consider a sequence of speed values, each clamped to the speed bounds mined from real vessel traffic in each geohash and during a certain time period. According to a predefined schedule, a vessel on a voyage passes through N geohashed regions in its route. Starting at time zero, it departs from region R_1 (origin port) and visits regions $\{R_2, R_3, \cdots, R_N\}$ in the route, respectively. Each region is either a port or just a transient location towards the next port call in the route. An example of regions represented by geohashes is given in Figure 4.1. We assume that we are aware of the delays due to any disruptions throughout the vessel's route while looking for a recovery plan.

We formulate the G-S-VSRP as an MOO problem with three decision objectives. In the first objective, we address the financial loss caused by the delayed schedules. When a disruption happens, a vessel can speed up to mitigate the delay at the expense of higher fuel costs. However, the delay may still not be completely mitigated and it would cause some penalty because of a late arrival at a port. In fact, traveling faster will increase the fuel cost and minimize the delay; whereas traveling at a slow speed will do little to mitigate the delay. This would reduce the fuel cost and maximize the delay, and there is a penalizing cost because of delayed arrivals at a port. In formulating G-S-VSRP, we use the term *loss* instead of *cost* for the first

Table 4.1: G-S-VSRP formulation notations

Symbol	Description
N	The number of regions on a route. The regions are indexed from 1. The $(N+1)$-th region refers to the vessel sailing back to the first region/port.
d_i	The distance (in nautical miles) from region R_i to region R_{i+1}.
d_{max}	The maximum distance in the problem instance used for normalizing the distances.
d_{min}	The minimum distance in the problem instance used for normalizing the distances.
s_i^{const}	The scheduled sailing speed of ships at sea (in knots) in region R_i.
s_i	The sailing speed (in knots) from region R_i to region R_{i+1} under the recovery strategy, which takes a value between the average speed $\overline{s_i}$ and the vessel maximum speed α.
$\overline{s_i}$	The average speed (in knots) at region R_i from historical speeds in AIS data.
s_i^{min}	The minimum speed (in knots) at region R_i reported in historical AIS data.
s_i^{max}	The maximum speed (in knots) at region R_i reported in historical AIS data.
x_i	The delay experienced (in hours) by the ship before it departs from a region R_i due to disruption.
f_i	Fuel consumption cost per a unit of distance (in nautical miles) between region R_i and R_{i+1}.
t_i	The scheduled arrival time (in hours) at region R_i which is a port.
t_i^{RP}	The time of arrival at a port in region R_i with recovery plan and tuning the speeds.
t_i^{region}	The time (in hours) a ship spends in a port in region R_i for loading/unloading. It is zero if region R_i is not a port.
c_i	The port cost (in dollars) due to delay at region R_i. If the region is not a port, c_i is zero.

objective, since there is a fixed cost regarding any planned schedule that accounts fuel cost. The extra cost in case of a disruption would matter for the stakeholders. That said, we call the first objective *financial loss* and aim to minimize it by tuning the speeds in the recovery plan. This financial loss objective is adopted from [41] and [45].

The second objective minimizes *delay*. There are situations in which arriving at a port at any cost is justifiable for the company; in other situations,

the speedup only makes sense up to some expense level (fuel cost plus wait penalty at the port). Moreover, a longer delay might be acceptable, for example, where there is a reported port congestion a priori and there is no rush to get to the port on time. Finally, the third objective controls the compliance of the optimized speed values in the geohashed regions with the statistics from historical AIS data. This objective aims at maximizing *average speed compliance*. We should note that the ships are not restricted to sail at these speeds but historically these speeds are representative of low-risk navigational patterns in the regions.

Enunciating G-S-VSRP as a three-objective optimization problem and solving it in a Pareto-based fashion has the benefit of providing the stakeholders with a tool to inspect the trade-off among these objectives. This allows for a greater flexibility in choosing a particular recovery plan from the ones returned in the Pareto front that matches the company's operational priorities at any point in time. This is possible because the MOO formulation is solved in a Pareto-based-fashion, which returns a set of possible solutions capturing the trade-off among the three objectives. The mathematical notations are defined in Table 4.1. In the following, the objectives' formulations are presented:

Minimize Financial Loss

$$\min_{\overrightarrow{s}} \sum_{i=1}^{N+1} d_i \times \left(f(s_i) - f(s_i^{const})\right) + \sum_{i=1}^{N} c_{i+1} \times t_{i+1}^{region} \times (t_{i+1}^{RP}(s_i) - t_{i+1}) \quad (4.1)$$

where:

$$\overrightarrow{s} = \{s_0, s_1, \ldots, s_N\}$$

subject to:

$$t_i^{RP}(s_i) - t_i \geq 0 \quad \text{if } R_i \text{ is a port}$$

$$\overline{s_i} \leq s_i \leq \alpha$$

$$s_i^{const} = \overline{s_i}$$

$$t_{i+1} = \sum_{j=1}^{i} (t_i^{region} + \frac{d_i}{s_i^{const}})$$

$$t_{i+1}^{RP} = \sum_{j=1}^{i} (t_i^{region} + \frac{d_i}{s_i} + x_i)$$

$$c_i = \begin{cases} c_i > 0 & \text{if } R_i \text{ is a port} \\ c_i = 0 & \text{if } R_i \text{ is not a port} \end{cases}$$

$$t_i^{region} = \begin{cases} t_i^{region} > 0 & \text{if } R_i \text{ is a port} \\ t_i^{region} = 0 & \text{if } R_i \text{ is not a port} \end{cases}$$

In the above formulation, the distance (in km) and speed (in knots) between the regions R_i and R_{i+1} are given as d_i and s_i, respectively. When R_i contains a port to be visited by the vessel, there is a loading/unloading time (port time) t_i^{region} (in hours). $t_i^{region} = 0$ if the region R_i is not a port call in the route. Delays between two consecutive regions are denoted by x_i (in hours) which indicate the extra time imposed to travel from region R_i to R_{i+1}. This objective minimizes the total financial loss of a route from the origin to destination by taking into account the extra cost of the consumed bunker fuel in the first term and the penalty to arrive at a port with a delay in the second term. The delay cost at a port in region R_i according to [45] is $c_i \times t_i^{region} \times (t_i^{RP}(s_i) - t_i)$, where we assume c_i to be normally distributed with regards to port time with mean μ and variance σ^2; the reason behind this assumption is that the cost of delay at specific port in region R_i is linearly

correlated with t_i^{region} which means that the longer the loading/unloading time at a port, the more costly the delay. Also, it is difficult to precisely estimate the bunker fuel consumption of a container ship when preparing a voyage schedule, since during a specific time interval (even one day) it is influenced by many factors, such as its sailing speed, displacement, trim, and prevalent weather conditions. Among these factors, sailing speed is the most important one. Fuel consumption as a function of speed per unit of *knots* has been formulated in [27]. This expression assumes the minimum of the speed limit of 14 knots. We linearly interpolated the above expression for speeds below 14 knots.

In the first constraint $t_i^{RP}(s_i) - t_i \geq 0$, the assumption is that arriving at the port in region R_i earlier than the scheduled time is not beneficial because a vessel usually has to wait to get services at the port. Therefore, it would save fuel by not speeding up more than necessary. The second constraint clamps the speed within the range $[\overline{s_i}, \alpha]$ which is the average speed in the i-th region to the maximum speed limit in normal steaming α which is 25 knots [54].

Minimize Delay Time

$$\min_{\vec{s}} \sum_{i=1}^{N} (t_{i+1}^{RP}(s_i) - t_{i+1}) \quad i = 1, \cdots, N \tag{4.2}$$

The *delay* objective is the accumulation of delays not recovered by speeding up. If the delay at each port is not recovered, it would propagate to the subsequent port calls. The total delay refers to the aggregation of propagated delays associated with enacting a recovery plan. As mentioned earlier, minimizing the delay is only possible at the expense of more fuel consumption to speed up; therefore, this conflict enables the decision maker to gauge the trade-off between cost and delay. It is important to mention that the cost of the delay is partially incorporated in the second term of the loss objective.

Maximize Average Speed Compliance

$$\max_{\vec{s}} \frac{\sum_{i=1}^{N} compliance(s_i)}{N} \quad i = 1, \cdots, N \tag{4.3}$$

where $compliance(s_i)$ is defined as:

$$\begin{cases} 1 & \text{if } s_i \in [\overline{s_i}, s_i^{max}] \\ \exp^{-\frac{(s_i - s_i^{max})^2}{\beta}} \times \exp^{-\frac{(d_i - d_{min})}{d_{max}}} & \text{if } s_i > s_i^{max} \end{cases} \tag{4.4}$$

The formulated problem is powered by a BDA engine that consumes real-world AIS static and dynamic messages and provides $[s_i^{min}, \overline{s_i}, s_i^{max}]$ for each geohashed region along the vessel route, where s_i^{min} and s_i^{max} are the minimum and maximum reported speeds in the region and $\overline{s_i}$ is the average of all reported speeds in that region. We use $\overline{s_i}$ instead of s_i^{min} for the scheduled speed in the region R_i because it is a more robust speed compared to the minimum speed (only one vessel might have reported it) and minimum zero (e.i., a vessel stops for any reason) is not correct for scheduling. If we assume the reported speeds in each geohash reflect the normal traffic in the region and exceeding its speed bounds increases the risk level, this gives an opportunity to the decision maker either stick to the normal speed limits or ignore these indicators and pay more attention to two other objectives, viz loss and delay.

In Equation 4.4, we assign the compliance value to 1 if the speed value lies within the AIS-mined speed bounds in the region R_i. Otherwise, two factors determine the compliance degree: (i) violation from the valid range in the region R_i (i.e. the first *exp* term in 4.4) (ii) the duration of this speed violation as a function of the traveled distance (i.e. the second *exp* term in 4.4). The first factor measures how far the speed value is from the valid speed

range. The larger the difference with the upper bound speed, the lower the compliance level. The β regulates the penalization degree of non-compliant solutions in the formula. In our experiments, we experimentally set β to 200 to get smooth value changes for the speed violation. The second factor weights the speed violation by the ratio of the distance traveled above the upper bound. Traveling longer distance at a non-compliant speed lowers the compliance index.

4.3 MOEA Algorithms for G-S-VSRP

G-S-VSRP is modeled as an MOO problem as discussed in Section 4.2. In this section we unveil the building blocks of the MOEAs used to solve this problem.

Solution Encoding

Each solution is encoded as a real-valued vector with a set of speeds s_i, for $i = 1, 2, \ldots, N$, where N is the number of geohashes representing the regions along the vessel's route. The s_i value is bounded by average speed $\overline{s_i}$ and the maximum of α knots where average speeds are mined from historical AIS data and α is set to 25 knots. M geohashes contain port calls where

$M << N$. Because of the dimensionality of the solution vector corresponds to the number of regions/geohashes in the route, G-S-VSRP could be considered a large-scale optimization problem.

Objective Functions

The three objectives in G-S-VSRP are: *financial loss* in (4.1), *delay* in (4.2), and *average speed compliance* in (4.3, 4.4). The first objective minimizes the financial loss that includes the fuel cost and penalty delay for arriving late at a port call. The second objective minimizes the total delay in arriving at the port calls. The last objective maximizes the compliance of the speed-based recovery plans with the historical navigational patterns.

Population Initialization

The initial population is generated as a collection of N-element vectors drawn from a random uniform distribution $\sim U(\overline{s_i}, \alpha)$. Each vector represents N speeds in N geohashed regions.

MOEA Algorithms

We employ three algorithms in the MOEA framework: namely, Non-dominated Sorting Genetic Algorithm II and III (NSGA-II, NSGA-III) [19] and [20],

from GA category and GDE3 [39] from popular DE category. The G-S-VSRP specially in higher geohash precisions is a complex problem in terms of large speed variables. Algorithms usually fail in finding solutions in such a large solution space. DE category was the most used algorithm in the literature in such problems [30]. NSGA-III was still promising in finding solutions while NSGA-II (an older version) was used as a benchmark for the other two algorithm performances.

Evolutionary Operator

We follow the default algorithms' operators and their default parameter values provided by MOEA Framework [77]. Simulated Binary Crossover (SBX) and PM are the default operators for NSGA-II and NSGA-III. GDE3 uses a different default operator named Differential Evolution (de). Binary tournament selection is the default selection operator for all algorithms. Tuning these operators has been postponed for future work.

Infeasibility Handling

In the G-S-VSRP formulation, three constraints govern the feasibility of solutions. MOEA Framework always generates the elements of a solution within

the given boundary; it also keeps the solution feasible during mutation and crossover by checking the solution after applying each operator and if a value is not in the boundary it sets it to lower or upper bound depending on which side has been violated. The second constraint controls the arrival time at the ports which should not be earlier than the scheduled time. This constraint is controlled by constraint handling in MOEA Framework which assigns zero to feasible solutions and otherwise a penalty value to the non-feasible solutions. The last one is a soft constraint that is controlled by the compliance objective. It allows solutions that are not completely compliant with the historical speed limits.

Stopping Criterion

The stop criterion in our simulations is a set of predefined number of function evaluations $\{50k, 100k, 150k, ..., 3600k\}$. However, time is an important factor in any DSS, since we are validating our new problem formulation, we skip the operational constraints imposed for finding a recovery plan (5 to 10 seconds) in our experiments. This is a large-scale optimization problem for which appropriate techniques that meet the time constraint are required. Improving the time performance is considered as a future work and is therefore

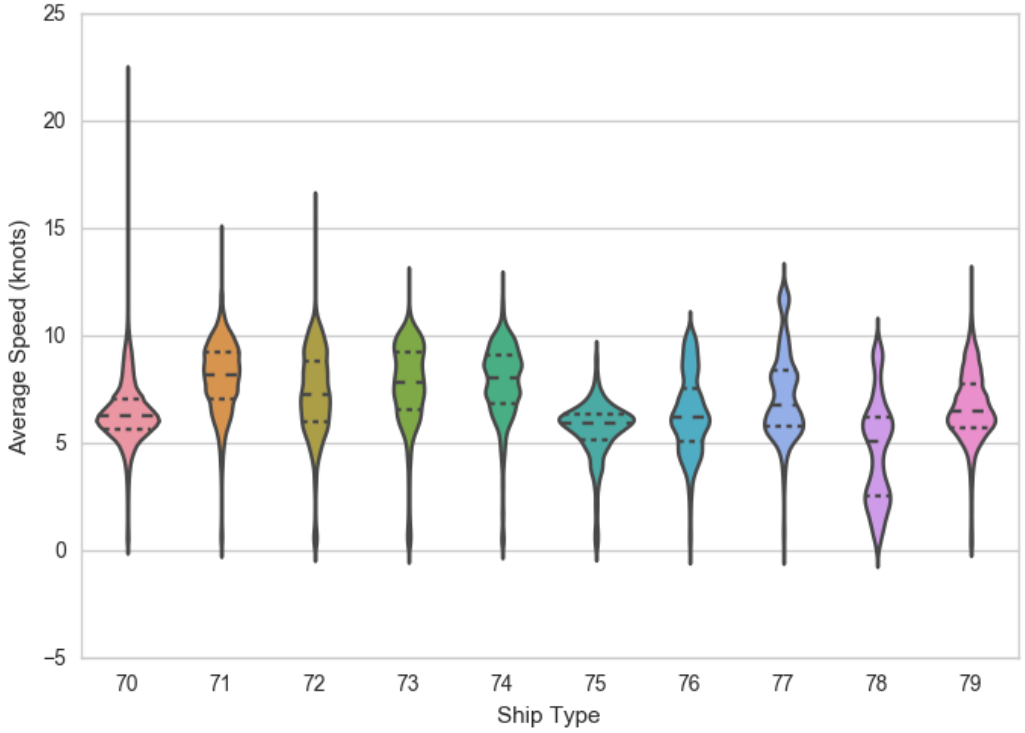

Figure 4.2: Speeds distributions of different cargo ship types in five months of January to May 2015 for geohashes in a route under consideration.

beyond the scope of this work.

4.4 Synthetic Instance Generation

To generate synthetic problem instances, we used Larus Technologies' proprietary routing algorithm API [129] to identify the optimal routes between port calls. The route was divided into multiple subregions (encoded as geohashes of a predefined length). The speed bounds for each geohash in a certain POI and for particular vessel type(s) were mined from the available historical AIS

data. The statistics of each geohash are only computed after five or more reported AIS messages have been received in each geohash during the specified POI. When situations containing less than five reported messages in one

Table 4.2: A sample of a geohashed route in a G-S-VSRP instance.

From	**To**	$\mathbf{D_i}$	...	$\mathbf{s^{mean}}$	$\mathbf{s^{max}}$	$\mathbf{x_i}$
reg_1	reg_2	d_1	...	$\overline{s_1}$	s_1^{max}	x_1
reg_2	reg_3	d_2	...	$\overline{s_2}$	s_2^{max}	x_2
...	...	...	...	...	...	...
reg_{N-1}	reg_N	d_N	...	$\overline{s_N}$	s_N^{max}	x_N

geohash arose, statistics from adjacent geohashes were interpolated. Figure 4.2 demonstrate speed distributions for different cargo ship types (codes 70 - 79) from January to May 2015. Table 4.2 shows a sample of a geohashed instance. The other important columns are port time, port cost and minimum speed. The instances used in the experiments can be found on our github repository [1].

4.5 Empirical Analysis and Discussion

We conducted three experiments to evaluate the solutions of the proposed multiobjective G-S-VSRP model. In the first experiment, the feasibility of the three MOEA solvers in finding Pareto-based solutions is verified. The sec-

[1] `https://github.com/fchgithub/AIS-Enabled-Instances`

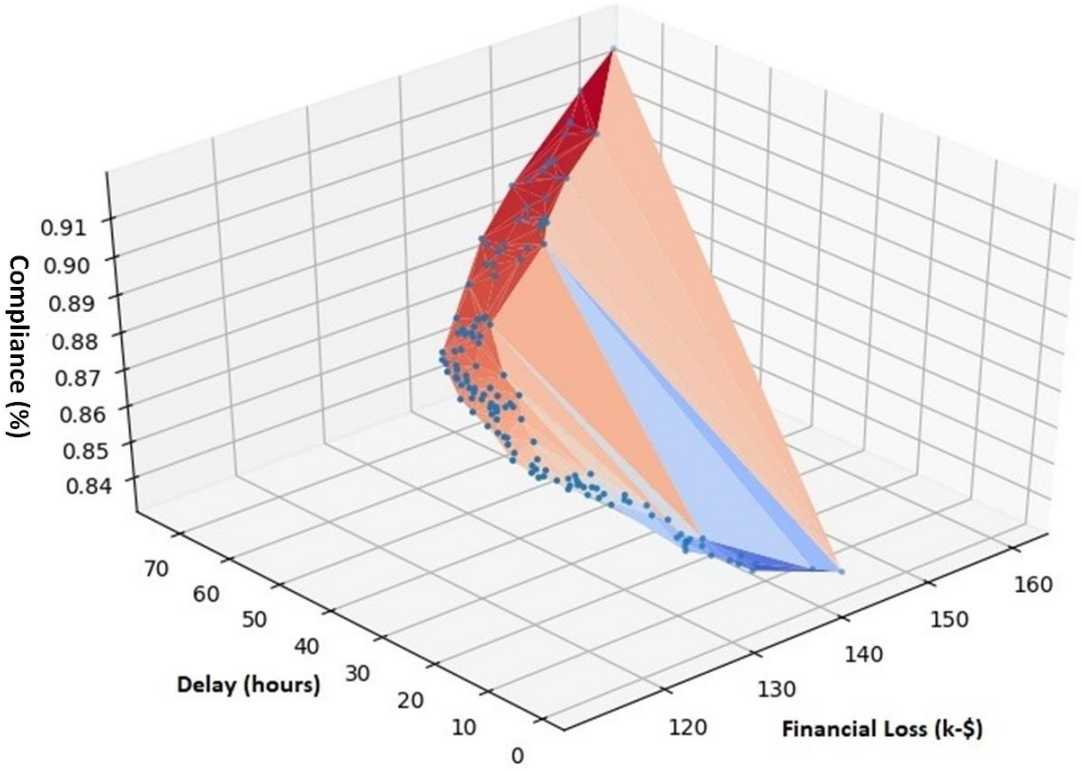

Figure 4.3: 3D presentation of Pareto front with GDE3 on a problem instance with big disruptions. The blue points are the solutions on a non-dominated surface.

ond experiment investigates the effect of dimensionality reduction (through clustering adjacent geohashes) upon the quality of solutions and on the Elapsed Time (ET). In the third experiment, we examine different granularity levels (geohash precisions) and their effects on the ensuing optimization problem.

4.5.1 Experiment I: Pareto Front Analysis

The goal of this experiment is to verify the feasibility of the proposed G-S-VSRP formulation and identify possible limitations and future works. In addition, the experiment aims at evaluating the three MOEAs discussed in Section 4.3, namely NSGA-II, NSGA-III and GDE3. Due to the large number of geohashes between consecutive port calls in the synthetic route, the complexity of the problem instances increases in terms of the number of decision variables (i.e., speed variables) in the geohashed regions. In a six-port-call instance with precision-five geohashes, the number of regions grows up to nearly 1500. Each solution is mapped to a large optimized speed vector in the decision space. This is a large-scale optimization problem where the number of variables to be optimized is large and the size of the solution space grows exponentially [94]. However, due to the spacial characteristics of our problem instance, we still can benefit from regular MOEAs, when we let them run for a very large number of evaluations in order of millions. In G-S-VSRP formulation, the third objective (i.e. compliance) is a soft constraint that inclines the search towards a small range of speed values. The compliance objective is maximized when the speed values are between

the average speed $\overline{s_i}$ and maximum speed s_i^{max}; This impacts the area of interest (AOF) by the optimizer. The average of differences between $\overline{s_i}$ s and s_i^{max} s is small (i.e. 1.15 knots in our problem instance) and this leads to a small AOF. By considering this fact about our problem, however, the G-S-VSRP is a large-scale optimization problem, still the optimizer is interested in looking in not very large area. Therefore, if we give the regular MOEAs the chance to run for many number of function evaluations, they can provide us with good solutions. For this reason, we ran the experiments for different number of function evaluations to show the impact of the number of function evaluation in the performance. Tuning and improving suitable algorithms in this regard is beyond the scope of this work and will be considered as an expanded version of this work. We use three famous MOEAs and identify their limitations.

To evaluate the performance of the algorithms, a reference set for each problem instance was created by running each algorithm 30 times on the instance and merging all solutions into an archive, from which a non-dominated solution set emerges. Also, we used our designed DMOCCA (i.e. it is introduced in the next Chapter 5) in the reference set generation. The three-dimensional Pareto front for an instance with big disruptions (i.e., referring

to Table 4.2, in some geohashes $x_i > (s_i^{max} - \overline{s_i})/d_i$ which means by speeding up in the historical range, the delay will not be fully recovered) is shown in Figure 4.3. The Pareto front provides a trade-off among the three objectives. If high historical speed compliance does not matter, there are solutions that largely mitigate the delay due to speeding up and even though they save in port penalty but the expense of a more fuel consumption leads to higher total loss at the end. On the other hand, solutions with higher compliance which offers less risky speed values (i.e., the normal speeds that many ships have reported could be a good representative of the situation in the region) are at the expense of superior loss and delay. This is a stakeholder's call to pick among the solutions according to the operational priorities of the liner shipping company at that time.

The performance of the three MOEAs and their standard deviation over 30 runs are shown in Figures 4.4, 4.5 and 4.6. Among the three algorithms NSGAII shows better performance with less uncertainty. Also, NSGAIII which is specially designed to deal with many-objective problems, and this is not the case with G-S-VSRP, does not seem to perform well in a large-scale problem. The last algorithm GDE3 uses special combination of individuals in the population to create offsprings, and due to the small range of speed values

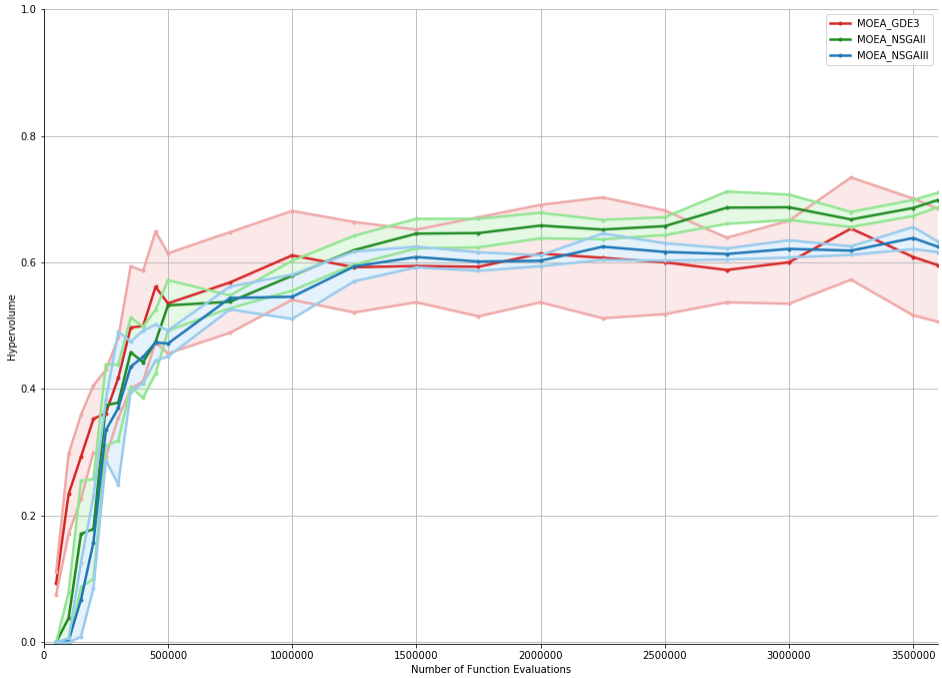

Figure 4.4: Hypervolume of three MOEAs over different number of function evaluation shows an increasing rate by increasing the number of evaluations. These numbers, due to the size of the G-S-VSRP problem, are much bigger than the required evaluations in the regular problems.

(i.e. to maximize the compliance), it is difficult to keep the offsprings in that range which causes a higher standard deviation. However, this suggests us that tuning of the algorithm may have good impact to keep the offspring in the vicinity.

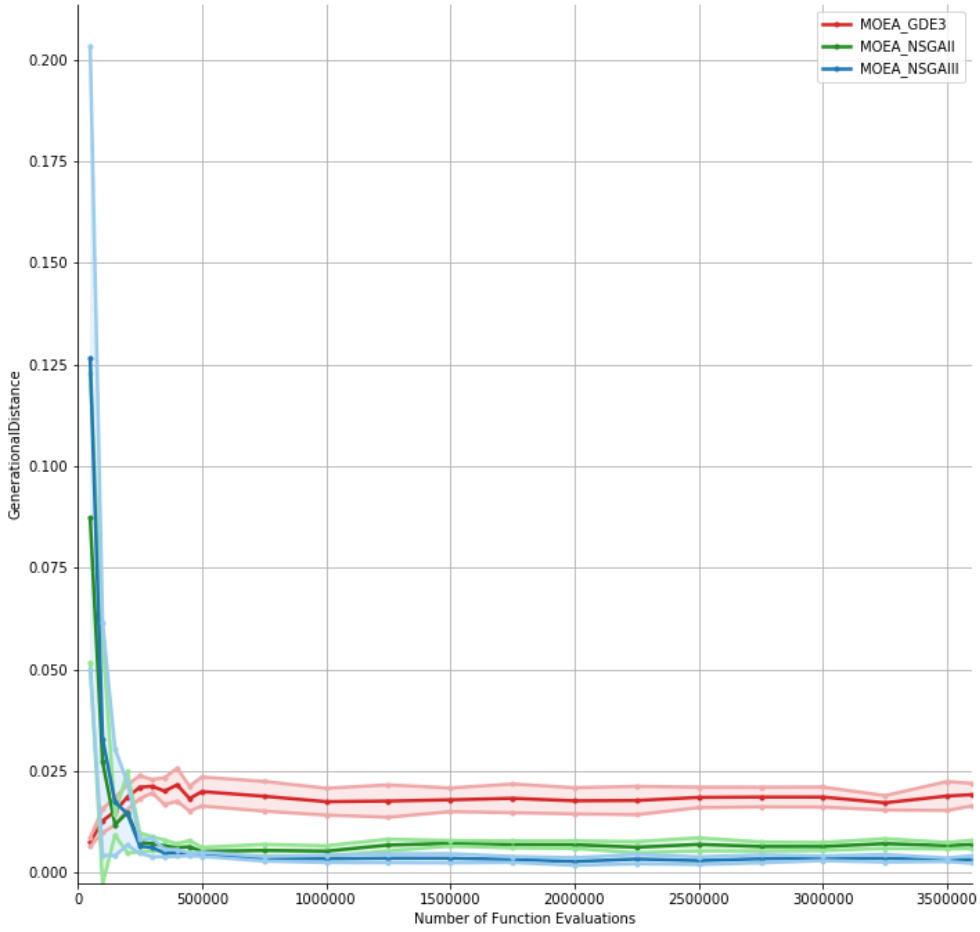

Figure 4.5: Generational Distance of three MOEAs over different number of function evaluation shows an increasing rate by increasing the number of evaluations. These numbers, due to the size of the G-S-VSRP problem, are much bigger than the required evaluations in the regular problems.

4.5.2 Experiment II: Dimensionality Reduction via Geohash Clustering

To reduce the G-S-VSRP problem dimensionality, which has increased by dividing the trajectories between the ports into multiple geohashed regions,

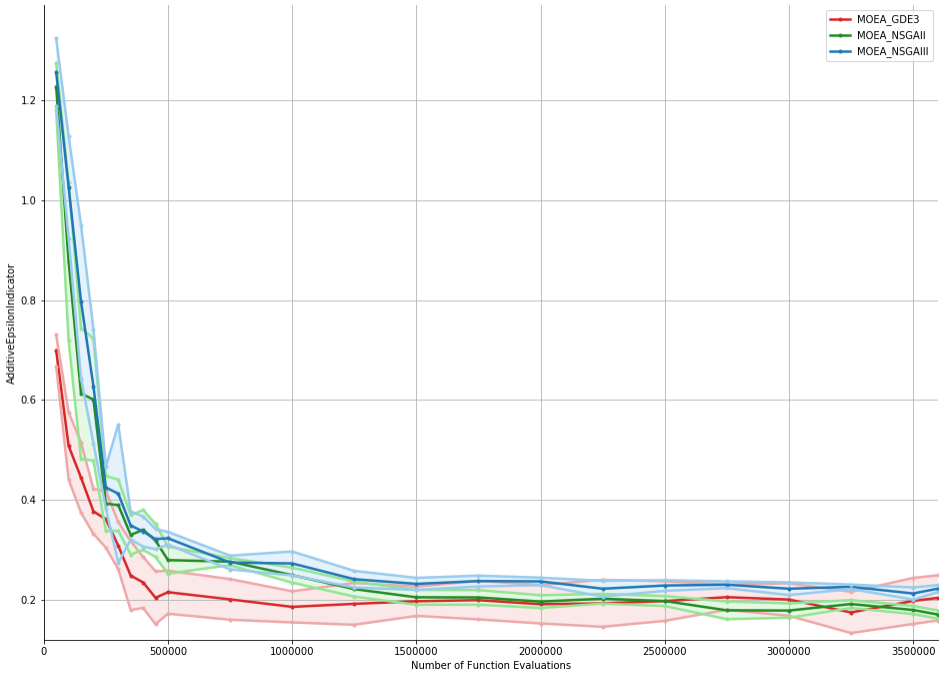

Figure 4.6: Epsilon Indicator of three MOEAs over different number of function evaluation shows an increasing rate by increasing the number of evaluations. These numbers, due to the size of the G-S-VSRP problem, are much bigger than the required evaluations in the regular problems.

the well-known clustering method K-Means [48] was used to merge adjacent geohashed regions with similar speed profiles along the route. The merging method is explained in Subsection 4.5.2. The aim of this experiment is to evaluate how the merged geohashes affect the quality of the solutions in the objective space and the computational efficiency of the three optimizers. To this end, we compared the solutions for the same problem instance in two cases: one when all geohashes are used in the optimization and the other one when clustered geohashes are used in the optimization. The problem

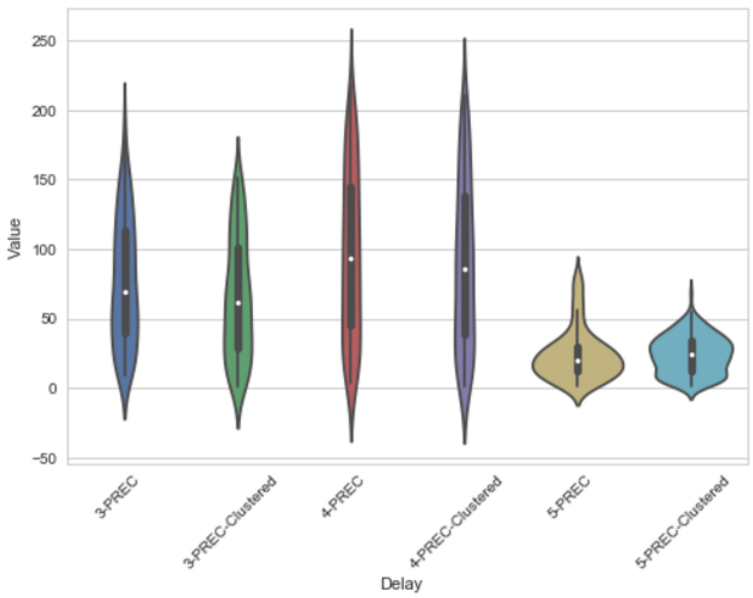

Figure 4.7: Delay objective distribution across multiple geohash precisions and with/without clustering adjacent geohashes.

instance for this experiment has been generated with precision-five geohash.

Figure 4.7 shows that the average delays after clustering changes but have similar distributions and probability densities. However, the loss objective in Figure 4.8 is slightly reduced. When two or more adjacent geohashes merge, the speed statistics are averaged and the distances are added together. A longer distance with a small speed change can recover from the disruption while in dealing with shorter distances (i.e., two or more geohashes before the merge), each speed has to change and this change leads to increased fuel cost compared to a merged instance. Finally, the compliance objective in Figure 4.9 significantly goes up in geohash precision-five clustering compared the

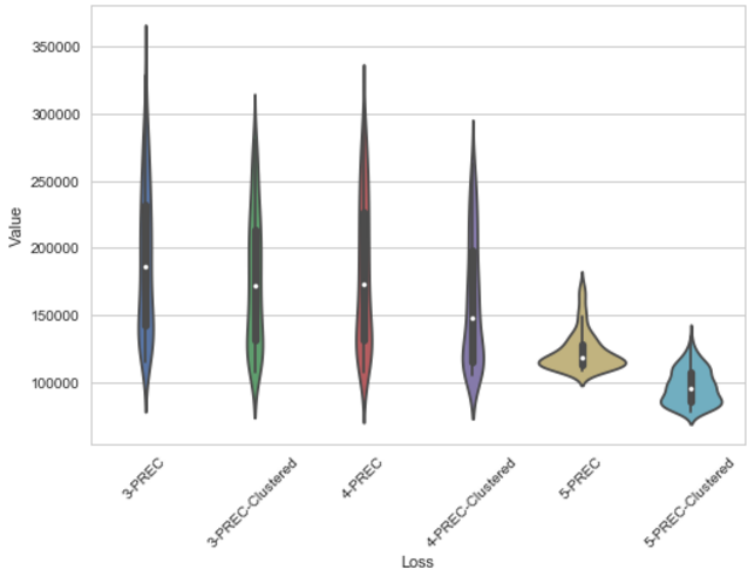

Figure 4.8: Loss objective distribution across multiple geohash precisions and with/without clustering adjacent geohashes.

original, geohash-unclustered. It seems it is easier to follow the historical speed limits in the merged regions which have longer distances. However, as we will discuss in the next experiment, this behaviour still depends on the characteristics of the problem instance.

The clustering in precision-five geohashes saved up to 50% in running time for 250,000 number of function evaluation. This is due to the fact that with five-precision, small adjacent geohashes have similar speed profiles and consequently, a larger number of them could merge together, therefore, the problem size is reduced significantly. Since the smaller size of the problem has less complexity for the optimization algorithm, it can save up to 50% in computation time with the same number of function evaluations.

Merging the Granular Speed Limits

The clustering algorithm is applied to each route leg, i.e., to group geohashed regions between two consecutive port calls. They could merge with adjacent regions if their speed profiles (represented as a pair $< s_i^{min}, \overline{s_i}, s_i^{max} >$) are similar. The number of clusters (K) in K-Means is selected in such a way to minimize the SSE error (Sum of Square Errors) of the clusters. After labeling all regions in the leg according to their cluster ID, adjacent regions with the same cluster labels are merged together. Distances and disruptions of these regions are summed up and the speed statistics are averaged up. A Java machine learning library was used for clustering [1].

4.5.3 Experiment III: Geohash Precision Analysis

In this experiment, we try out various geohash precisions to evaluate how the underlying speed granulation mechanism affects the quality of solutions in the objective space and the computational complexity of the three optimizers. A smaller geohash precision encodes a larger geographical area. We should note that the problem instances generated in the three geohash precisions (i.e., precision-three, precision-four and precision-five) are not exactly the same

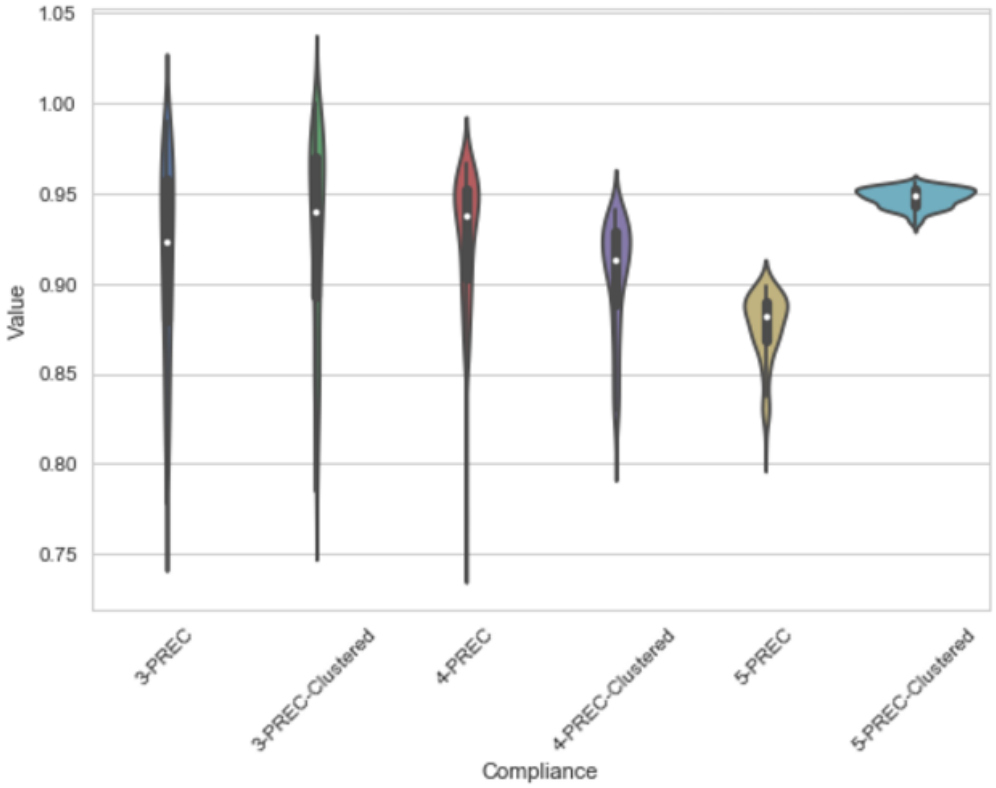

Figure 4.9: Compliance objective distribution across multiple geohash precisions and with/without clustering adjacent geohashes.

but very similar and the reported results are for 250,000 number of function evaluations. They are the same in terms of port-calls, port times and the port costs values, but the distances between the ports have changed slightly. This is because the design of the synthetic data generation method which uses the center of geohashes for the distance between two adjacent geohashes and it may vary across different geohash precisions. Also, the number of reported speeds is higher in larger geohashes, therefore the mined speed statistics are subject to change.

In Figures 4.7, 4.8 and 4.9, we combine the results of precision-three and precision-four geohashes with precision-five in the last experiment. Also, to have a stronger evaluation on the effect of clustering, we apply clustering in problem instances with these precisions as well. Assuming instances from different geohash precisions are presenting close problem instances, the higher precision gives better results (lower loss and delay with almost identical compliance).

In the big disruption problem instance with over 30 runs, clustering geohashes would affect the precision of objectives on an average of 8% change on loss objective and 20% change on delay objective. The compliance objective in clustered geohashes changes on average up to 4%. Also, the clustering method saved 16% (i.e., 6.9 sec to 5.76 sec), 27% (i.e., 19.42 sec to 14.2 sec) and 50% (i.e., 86.25 sec to 42.94 sec) in running time respectively in three, four and five precisions. With an increased precision, the number of geohashes in the route goes up. In the synthetic problem instances, it reaches from 54 geohashes in precision-three and 300 geohashes in precision-four to nearly 1500 geohashes in precision-five. The highest time saving happens in precision-five because the chance of merging similar speed profiles in smaller regions is higher.

4.6 Summary

In this work, a new VSRP generalization has been proposed. In the new formulation, the trajectory between the port calls is divided and encoded into adjacent geohashed regions. In each geohash, the historical speed profiles are extracted from AIS data. The G-S-VSRP is a multiobjective speed optimization problem with three objectives (i.e., loss, delay, and compliance). Three MOEAs tackled the problem and provided the stakeholder with a Pareto front which reflects the trade-off among the three objectives. Geohash granularity and dimensionality reduction techniques were evaluated and discussed on the model.

The limitations of the current work can be considered as the future works. The algorithm parameters such as population size, number of function evaluations, crossover and mutation rates can be tuned regarding the problem instance structure. Also, we did not improve the algorithms to perform better in such a large-scale optimization problem. Other methods such as coevolutionary [57] and memetic [51] algorithms can be used to improve the performance for this types of problems. Parallel and distributed methods may also help boost the time efficiency. Moreover, the proposed geohash

clustering step can be improved by optimizing the choice of the number of the clusters and the chosen clustering algorithm.

Chapter 5

Distributed Multiobjective Cooperative Coevolution Algorithm

Using geohash-based speed mining on AIS data in the granular speed-based vessel schedule recovery problem (G-S-VSRP) gives rise to a large-scale multiobjective optimization problem, where the number of speed variables in geohashed regions grows to the order of thousands. In G-S-VSRP which is searching for a recovery plan to mitigate the disrupted schedule, we want to minimize delay and cost of a vessel's voyage while looking to maximize

the compliance of the speed variables with historical AIS speed statistics. However, due to the small range between the average $\overline{s_i}$ and maximum speed values s_i^{max} (i represents the ith geohash area), which is the range where compliance is maximized, to solve the problem, we did not deal with a very large search space. In fact, a big number of function evaluations can compensate for the difficulty of the problem solved by the optimizers. The big number of function evaluations gives the opportunity to explore in the search space sufficiently. It is not always true that this range has to be so small or having the maximized compliance be a matter of interest. The statistical values of the historical data can change, if we include more data from additional months and years (i.e. for the current experiments, we used the reported AIS data for five months of January to May 2015) and this may not lead to a small range between the average speed values $\overline{s_i}$s and maximum speed valuess_i^{max} (i.e., $s_i^{max} - \overline{s_i}$). Also, the decision maker may not be interested in complying with the historical navigational speed patterns but still care to be aware of his behaviour against historical data. Both of these scenarios prompted us to modify the G-S-VSRP to switch the compliance objective from maximization to minimization. With this change, we can study the behaviour of MOEAs when faced a very large solution space; also, we can

address the request of those stakeholders interested in high speeds and less delays (not necessarily comply with historical speed statistics). According to the speed statistics, maximizing the compliance is linked to low values for both average and maximum speeds in geohashes. Therefore, minimizing the compliance would give the opportunity to find the solutions with higher speeds values (i.e., between s_i^{max}s and 25 knots) which leads to lower delays.

In the new G-S-VSRP, we seek to improve the MOEA's performance by using the concept of cooperative coevolution performed on a distributed platform. In Cooperative Coevolution (CC), the optimization problem is divided into smaller subproblems which are easier to solve (i.e., decomposing a high-dimensional variable vector into multiple low-dimensional sub-vector). Each subproblem is assigned a separate sub-population. Each sub-population evolves a subset of decision variables using a regular MOEA while cooperating with each other to evaluate complete solutions through a cooperative exchange of individuals [57]. In this chapter, a novel distributed multiobjective cooperative coevolutionary algorithm (DMOCCA) is introduced to improve the quality of the solutions found by MOEAs. The algorithm starts with k parallel MOEAs running with different random seeds. Then, DMOCCA breaks down the problem into the subproblems where each subproblem is the

geohashes between every two ports. DMOCCA improves MOEA's solution set by picking base solutions from this set and running a regular MOEA on the subproblems while their populations are evaluated as a complete solution using base solutions to fill up the missing subcomponents. A distributed algorithm is designed to utilize the Apache Spark framework (i.e., a distributed computing engine) [63] to parallelize evolving the subproblems in the algorithm. The experiments show that with DMOCCA there is an improvement in the performance metrics (i.e., hypervolume, generational distance and epsilon indicator) when regular MOEAs (GDE3, NSGAII, and NSGAIII) would stop improving or showing a steady behaviour. These are some of the state of the arts MOEAs and other algorithms can be used as well.

The remainder of this chapter starts with the related works in section 5.1. Section 5.2 explains the algorithm setup for DMOCCA. In section 5.3 DMOCCA algorithm is elaborated. It is followed by the experimental results in section 5.4. The chapter concludes in section 5.5.

5.1 Related Works

The literature related to the G-S-VSRP is already surveyed in the previous Chapters 3 and 4, where the contributions were mostly defining the optimization problems, and solving them using MOEAs, Mixed Integer programing, etc. In this section, we review the distributed optimization algorithms in the literature where Apache Hadoop and Spark are used as tools to parallelize the process in the algorithms. Then, we survey the large-scale optimization and distributed large-scale optimization approaches.

5.1.1 Distributed Evolutionary Algorithms

With the emergence of big data frameworks such as Apache Hadoop and Apache Spark and their success to speed up the algorithms when tackling big data problems (i.e., the problems with high volume of data with variety, veracity and velocity (4Vs)), the optimization research has found its way to these frameworks to take advantage of their parallelism feature. The optimization may [91, 83] or may not [87] be data intensive but is usually computation intensive and therefore suitable to look for distributing the optimizers to speed up the execution and improve the quality of the solutions.

The authors in [84] optimized the Traveling Salesman Problem (TSP) with a MapReduce Ant Colony Optimization (ACO). m number of problem instances in parallel optimize the problem with p number of ants on each instance. The total number of ants are $m \times p$. It is an iterative process and in each iteration the pheromone matrix is updated based on the findings of $m \times p$ ants. Also, the authors in [85] utilize Apache Spark to parallel the ACO. In an iterative process, each worker runs ACO with x number of ants and the optimal path is used to update the pheromone matrix and it is broadcast to all machines for the use of next iteration. In another work [86], the authors compare two parallel ACO approaches. In the first approach, the search space is replicated and each worker runs ACO on a replication of search space. In the second approach, the search space is partitioned and m mappers optimize partial pheromone on a partition of search space and in the reducer, all these partial pheromones are combined which provides a partially optimized pheromone matrix and then ACO is run over the whole search space. Another ACO parallism can be found in [87] using Apache Spark. The authors there combine Min-Max Ant colony with Spark MapReduce to execute the path building and the pheromone operation in a distributed computer cluster to solve TSP.

The authors in [88] use MapReduce to parallelize Artificial Immune System. Detector production is performed in parallel (i.e. it happens in mappers). In another work [89] an AIS-based fraud detection model is built with parallizing the calculation of the affinity threshold and the generation of memory cells on a MapReduce framework. Using distributed frameworks to parallelize optimization algorithms is not limited to these articles. Many more works can be found in the literature. Another example involves Particle Swarm Optimization (PSO) which is done in parallel with MapReduce and Spark [91, 92]. In [93], DE algorithm on MapReduce and Spark has been studied by the authors. Also, authors in [90] used Spark to perform the fitness evaluation in parallel for GA.

5.1.2 Large-scale Optimization

In recent years, the number of real-world problems that require to optimize an increasing number of variables is growing. This type of the optimization, called Large-Scale Global Optimization (LSGO), is especially difficult due to the immense increase of the multi-dimensional solution space. This exponential growth in the size of the search space is known as the curse of dimensionality [94]. The LSGO field has attracted many researchers' attention

recently, especially in the area of big data. It has brought new opportunities and resulted in the increasing number of LSGO in different researches fields [95, 94]. The goal of researchers in this field is to improve the search quality and time efficiency.

There are various evolutionary approaches in the literature to solve LSGOs which are listed here:

1. Sampling and variation operators in which sampling examines the initial population sampling and variation operators considers developing new generic operators or modifying the operators [96, 97, 98, 99].

2. Approximation and surrogate modeling where from a real complex problem an approximation model is built. This model mimics the behaviour of the simulation model as closely as possible while being computationally cost-effective to evaluate [102, 103].

3. Local search and memetic algorithms that are the combination of EA global search with local search operators working within the EA loop [104, 105].

4. Decomposition and divide-and-conquer that is dividing a large problem into subcomponents and solving them independently in order to solve

the large problem [106, 122].

5. Parallelization where GPU, CPU and other parallel frameworks are used to evaluate the problem with higher performance [106, 122].

6. Hybridization which is benefiting from unique features of different optimizers [108, 109, 110].

We tackle the LSGO problem with a divide-and-conquer approach on a distributed framework to benefit from the parallelization of decomposition of a large problem. The G-S-VSRP can also privilege the approximation model by using unsupervised machine learning on the given problem instance to cluster similar adjacent geohashes and merge them to reduce the dimensionality of the problem and creates a close approximation of the real problem. The later has been experienced in the previous Chapter 4.

5.1.3 Distributed Cooperative Coevolutionary Algorithm

Due to the parallel nature of the cooperative coevolutionary (CC) algorithm where each subcomponent evolves separately, a variety of contributions for parallel CC have been studied. The CC paradigm can be applied to both single and multiobjective optimization problems. Authors in [111] proposed

a multiobjective CC genetic algorithm (MOCCGA). They combined MOGA with Cooperative Coevolution Genetic Algorithm (CCGA) in [57]. In CCGA, the best individual from each sub-population cooperates with other sub-populations to complete a solution. Each sub-population evolves and assigns a fitness to its individuals based on their rank in the sub-population's local Pareto front. They evaluated the proposed algorithm with ZDT benchmark functions. Later, authors in [112, 113] proposed an extension on the previous work by using other MOEAs (Niched Pareto Genetic Algorithm (NPGA)[114], NSGA [115], Controlled Elitist NSGA (CNSGA)[116]) and defining a central fixed size archive to store the non-dominated solutions. The evaluation of an individual in the sub-population is obtained by substituting it into its location in a solution that is randomly extracted from the archive.

The work in [119] proposed a different cooperative coevolutionary strategy by dividing the decision variable space. A couple of populations evolve all decision variables at different intervals. This approach includes dynamic population management that eliminates populations not contributing to the search. The results in this approach are similar to state-of-the-art MOEAs on benchmark problems; however, its elitism strategy has a high selection

pressure which could reduce its performance. In another approach [120], the authors propose an adaptive coevolutionary DE algorithm. They use a random grouping strategy to divide the optimization variable into subcomponents. Also, an adaptive weighting mechanism (i.e. apply a weight to each subcomponent and only optimize a much lower dimensional vector) is applied at the end of every cycle to provide an extra chance to evolve all the objective variables at the same time. A coevolutionary extension of NSGAII is presented in [118] which uses representatives/collaborators by choosing a random solution from the best non-dominated level. A multiobjective constructive cooperative coevolutionary (moC^3) was proposed in [121] which introduces a constructive initial optimization of the sub-populations in a co-adaptive fashion. During the initial optimization, only a subset of the other subproblems is considered for the co-adaptation. This subset increases until all subproblems are included.

In another work [117], a coarse-grained model is introduced by the authors. In this model, two representatives (best + random) are exchanged between sub-populations. Therefore, the fitness is evaluated twice per individual. Each sub-population has its own archive and a novel adaptive niching mechanism is used. It is a parallel CC evolutionary algorithm by assigning

sub-populations to different nodes. Furthermore, the authors in [106] propose a parallel multiobjective cooperative coevolution algorithm. The population is split into several sub-populations with each optimizing a part of the global solution in parallel using a MOEA. To evaluate a partial solution and to build a complete solution, every sub-population shares a number of K solutions randomly chosen from the partial Pareto front. This strategy improves the diversity of the results compared to one random selection from the partial Pareto front. They could achieve speedup when running the algorithm on a multicore machine. In another parallel CC [122], authors used MapReduce to parallelize PSO. In their proposed method, in addition to dividing the population, the search space is partitioned into subspaces. Only searching on different subspaces is performed in parallel and the cooperative PSO run is not parallelized.

The proposed DMOCCA's performance is compared with the regular MOEAs'. In comparison with [106], the G-S-VSRP could not achieve a good convergence using their approach. With following their approach, merging solutions from different subcomponents would not guarantee a feasible solution set. The comparison of DMOCCA with other LSGO approaches is left for future work.

5.2 DMOCCA Setup

DMOCCA is designed to address the curse of dimensionality in G-S-VSRP while the compliance objective is minimized. It uses regular MOEAs in different stages. The regular MOEAs initially fill the archives used in DMOCCA. Also, DMOCCA uses MOEAs to evolve its subproblems. The set of MOEAs for the experiments with DMOCCA are similar to Chapter 4, which are GDE3, NSGAII, and NSGAIII. Also, solution encoding, population initialization, generic operators, infeasibility handling follow the same direction as shown in Chapter 4.

Objective Functions

This is a three-objective optimization problem, where the first objective minimizes the *total loss* as in (4.1), and the second objective minimizes the *total delay* as in (4.2). The third objective is the *compliance* as formulated in (4.3) and (4.4) but it minimizes the compliance with historical navigational patterns.

Reference Set Generation

To evaluate the performance of the optimizers, we need a reference set. Since the reference set for the problem is unknown, we generate a reference set by running different MOEAs followed by DMOCCA. Each MOEA is run 30 times and all solutions are collected in a non-dominated archive. Then, DMOCCA is run for 100 iterations, 30 times on the solution in the archive. All DMOCCA solutions are added to the archive as well. Employing three MOEAs (i.e. GDE3, NSGAII, and NSGAIII) generates three non-dominated archives. We refer to the set generated by merging these three archives into one non-dominated set, as the reference set for our experiments.

5.3 DMOCCA: Algorithm Design

DMOCCA is designed to deal with large-scale optimization problems. In DMOCCA, the optimization variables (i.e., speed values in geohashes) are divided into smaller subcomponents. The splitting strategy is to group the geohashes between every two ports into a subcomponent. For example, for a four port-call problem instance with N geohashes, there are three subcomponents. The size of the subcomponents is not necessarily the same due to a

different number of geohashes between ports. DMOCCA leverages a Spark cluster to distribute some parts of its process (i.e. parallel processing) which is explained later in this section. Due to the iterative process in DMOCCA and input/output overhead when collecting the solutions in the driver machine of the cluster after each iteration, it is not considered to be a very fast approach. Therefore, we first benefit utmost from the regular MOEAs, then we improve those solutions by DMOCCA.

The work-flow of DMOCCA is illustrated in Figure 5.1. As shown in the figure, the distributed algorithm starts with a Parallel Multiobjective Evolutionary Algorithm (PMOEA). PMOEA runs the same MOEA on K machines where K is smaller than or equal to the number of machines in the cluster. The purpose of PMOEA is to boost the solutions found using a regular MOEA by running it with different random seeds. The solutions found by the PMOEA are added into two non-dominated archives (i.e. Archive-A and Archive-B). Archive-A only keeps the solutions found by PMOEA, while Archive-B not only keeps PMOEA's solutions but also accumulates the solutions found by DMOCCA in the later iterations. DMOCCA randomly takes a solution called *base solution* from one of the archives with probability 0.5 and explores the search spaces limited to the subcomponents. Through the

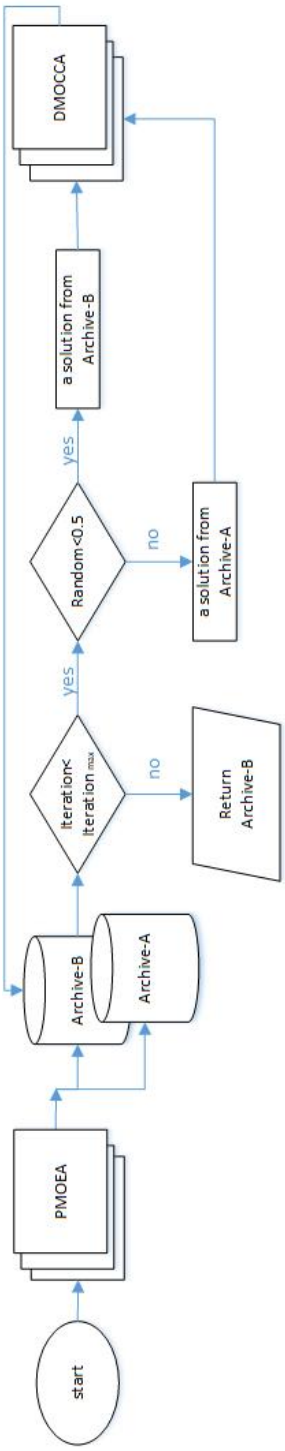

Figure 5.1: The DMOCCA work-flow. The multi-layer boxes show the parallel steps of the algorithm.

experiments, we found out keeping the Archive-A separately results in better solutions which is due to the higher chance of diversity in picking the base solutions. However, selecting a base solution from Archive-B gives more chance to search the subspaces more intensely. Choosing the base solutions from the archives can be repeated as many times as we set the number of iterations $iteration_{max}$ in the algorithm. The more number of iterations gives the more chance to DMOCCA to improve the solutions in the Archive-B. The non-dominated Archive-B after $iteration_{max}$ is the final approximate set found by DMOCCA.

The detailed psudocode of DMOCCA is found in Algorithms 2 and 3. Algorithm 2 starts with creating the Spark context instance in lines 1. According to the experiments with regular MOEAs (i.e. the MOEAs were run for different NFEs $\{50k, 100k, 150k, ..., 3600k\}$), we noticed GDE3 shows a better performance comparing to NSGAII and NSGAIII. The highest average performance for GDE3 happened in $300k$ number of function evaluations. Therefore, to get the most advantage from the regular MOEAs, we choose this algorithm with $300k$ NFE to fill up the archives A and B in lines 2 to 4. In line 5, the subcomponents are generated by grouping the geohashes between every two ports into subcomponents. Each subcomponent just carries

the starting and ending indices of geohashes of the ports. A base solution is selected from one of the archives with probability 0.5 in line 6. Since the problem instance, the base solution, and the choice of MOEA algorithm are needed in the worker machines (i.e. slave machines in the Spark cluster), they are broadcasted respectively in lines 7, 8 and 9. In line 10, we use *parallelize* with subcomponents and the number of subcomponents as the parameters to force the parallelism to the algorithm. With this, if we have at least the same number of machines as the number of subcomponents, then we are sure that they will run in parallel. It is important to note that because the size of subcomponents are not necessarily equal, the parallel tasks are not finished at the same time and the longest task determines the running time of the parallel tasks.

In the following lines of the algorithm, the subcomponents are mapped to a set of solutions found by *findPartialSolution* function and are collected to the driver machine in the iterations. In lines 11 to 12, this is the first time that this map is materialized and then added to the archive-B in the line 13. The remainder of iterations (i.e. choosing the base solution, broadcasting to the workers, running MOEAs on the subcomponents and collecting the solutions in archive-B) happens in the following "While do" loop in line 15.

The output of the algorithm, in line 16, is the non-dominated solution set in archive-B.

Algorithm *findPartialSolution*, as shown in Algorithm 3, is the core part of parallelism that occurs in the DMOCCA algorithm. Each worker machine runs *findPartialSolution* algorithm on the subproblem corresponding to the assigned subcomponent. The problem instance, the base solution and the type of the MOEA algorithm are already copied in the worker machines via broadcasting these variables. The problem instance is broadcasted since the evolution happens in this instance. The base solution keeps the same context among all the distributed subcomponents. In fact, each worker machine locally looks for the alternative (better) solutions in the subproblem by evolving the sub-population limited to the subcomponent subspace. The individuals in each sub-population are evaluated as the full-size solutions when the explicit context from the base solution fills up the rest of the missing subcomponents. Also, the broadcasted MOEA algorithm determines which algorithm should be run on the subproblem.

Algorithm 3 runs on different nodes of the cluster for different subcomponents. In line 1, a G-S-VSRP instance (i.e. with three minimization objectives) is created which is a subproblem associated with the subcompo-

nent. Then, the sub-population corresponding to the subcomponent evolves with the MOEA algorithm. Therefore, the MOEA solves a reduced size G-S-VSRP problem. As mentioned before, to evaluate the individuals in the sub-population, the full-size solutions are build by substituting the missing subcomponents from the broadcasted base solution as shown in Figure 5.2 and called *buildCompleteSolution*. In fact, the subproblems related to the subcomponents are evolved in parallel to find better solutions and improve the final solution set in archive-B. Due to the concern about the overhead of input/output in the cluster, we only return the subcomponent and its corresponding subsolutions. The subcomponent is used to build the full-size solutions in the driver machine and add to the archive as shown in the "While do" loop in Algorithm 2.

5.4 Empirical Analysis and Discussion

To run the experiments with DMOCCA, we setup a Spark cluster with five nodes. The configuration of each node is shown in Table 5.1. We used Java Spark API for the implementation, since the MOEA framework [77] is written in Java. Also, Kyro [131] which is a fast and popular (e.g. used by Apache

Algorithm 2: DMOCCA

```
   Input: instance: G-S-VSRP problem instance; MOEAlgorithm: MOEA algorithm;
          max_iteration: maximum number of iterations;
   Output: Approximation set (Pareto front)
 1 sc = SparkContext();
 2 solutions = GDE3(instance);
 3 Archive-A.add(solutions);
 4 Archive-B.add(solutions);
 5 subcomponents = splitInstance(instance);          ▷ start & end geohash indices between ports
 6 if random ≤ 0.5 then
       base_solution = archive.randomSelection(Archive-A);
   else
       base_solution = archive.randomSelection(Archive-B);
 7 brd_instance = sc.broadcast(instance);
 8 brd_base_solution = sc.broadcast(base_solution);
 9 brd_MOEAlgorithm = sc.broadcast(MOEAlgorithm);
10 subcomponent_rdd = sc.parallelize(subcomponents, subcomponents.size);
11 subsolution_rdd = subcomponent_rdd.map(subcomponent:
     findPartialSolutions(subcomponent));
12 subsolutionsList = subsolution_rdd.collect();
13 archive.add(buildCompleteSolutions(subsolutionsList, base_solution));
14 iteration = 2;
15 while iteration< max_iteration do
       if random ≤ 0.5 then
           base_solution = archive.randomSelection(Archive-A);
       else
           base_solution = archive.randomSelection(Archive-B);
       brd_base_solution = sc.broadcast(base_solution);
       subsolution_rdd = subsolution_rdd.map(x,y: findPartialSolutions(x));
       subsolutionsList = subsolution_rdd.collect();
       archive.add(buildCompleteSolutions(subsolutionsList, base_solution));
16 return archive-B
```

Algorithm 3: findPartialSolutions

```
   Input: subcomponent: start and end index of the geohashes in the subcomponent;
   Output: subcomponent, subsolutions
 1 subproblem = G_S_VSRP(subcomponent, brd_instance.value);
 2 subsolutions = brd_MOEAlgorithm.value.run(subproblem) {
       subsolution.evolve();
       evaluate(buildCompleteSolution(subsolution, subcomponent, brd_base_solution.value));
   }
 3 return (subcomponent, subsolutions);
```

Spark) custom serialization library, was employed in the implementation. In the experiments, we did not conduct any parameter tuning for the MOEAs. We used some fixed values as shown in Table 5.2. The population size for the

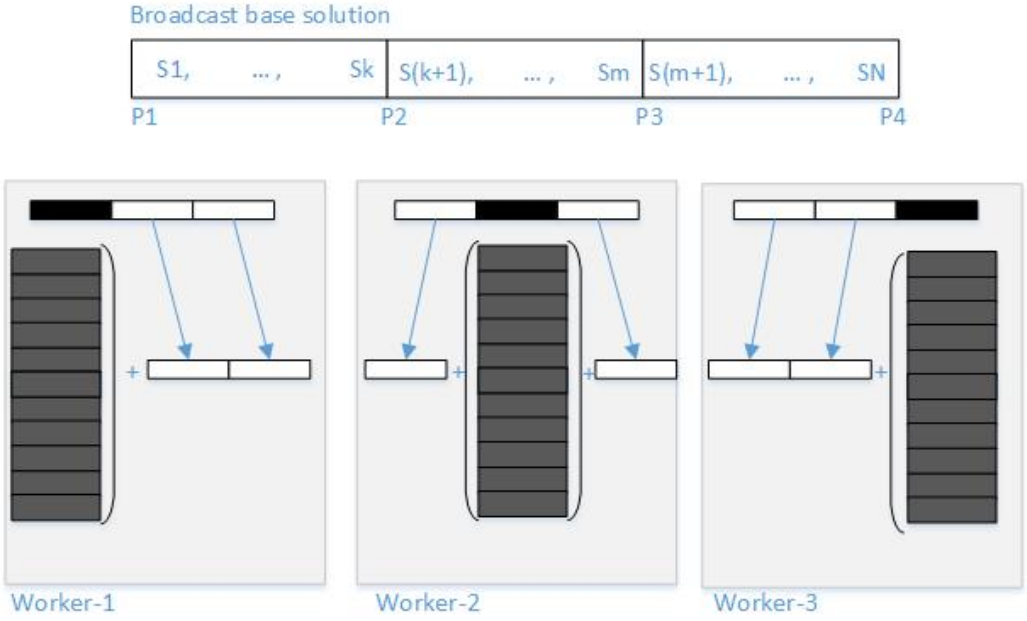

Figure 5.2: For a four port-call instance, there are three subcomponents. The base solution is randomly selected from the archive and broadcasted to the cluster nodes. Each node is working on one subcomponent and runs MOEA by creating a sub-population and evolving the sub-population, but the evaluation of the sub-individuals is done by building the complete size solution using the broadcasted base solution. The missing subcomponents are directly copied from the base solution.

PMOEA is 100, but in the iterations, since the solutions are collected in each iteration, we reduced the size of the population to 75 to lighten the load of the collected data in the network and in the driver machine. Also, the set of number of function evaluations (NFE) in MOEAs is $\{50k, 100k, 150k, ..., 3600k\}$. The number of function evaluation in DMOCCA is $300k$ (in PMOEA) plus $35k$ in *findPartialSolutions* which the latest is repeated in 100 iterations.

Table 5.1: Spark Cluster's Nodes Configuration

OS	Linux 16.04
CPU	4 cores
RAM	16 G
HDD	150 G
Network	1 Gb/s NIC on the internal

Table 5.2: Parameter Setup

Approach	Pop. Size	2x.rate	pm.rate	dist.index
MOEA	100	0.75	0.25	25
DMOCCA	{100, 75}	0.75	0.25	25

In the following experiment, the behaviour of regular MOEAs (i.e. NSGAII, NSGAIII, and GDE3) is studied on G-S-VSRP and compared with DMOCCA.

5.4.1 Experiment I: DMOCCA vs. MOEAs

In this experiment, we evaluate the proposed DMOCCA in comparison with the regular MOEAs (i.e. NSGAII, NSGAIII, and GDE3). We assume there is no time limit for the algorithms. In fact, we loosen up on this limitation to focus the study to the functioning of regular MOEAs and the proposed DMOCCA when dealing with large-scale optimization problems. Also, we examine the speedup of DMOCCA which is checking out the running time of the algorithm when executing on a different number of nodes in the cluster. The performance of the regular MOEAs and DMOCCA are evaluated by three performance metrics HV, GD and Epsilon Indicator (EPI) similar to the experiments in the previous chapters. Figures 5.3, 5.4 and 5.5 illustrate the HV, GD and EPI for the MOEAs and DMOCCA. The graphs also show the standard deviation of the performance with the shadow around the average

performance over 30 runs.

Figure 5.3 represents the HV performance values for the regular MOEAs which start with very low values and improve to some degree by increasing the number of function evaluation; but, they stop improving (i.e. NSGAII and NSGAIII) or the performance degrades (i.e. GDE3) even if we let them run for a big number of function evaluations. This behaviour is because of the difficulty of the problem they try to solve. The functioning of GDE3 (i.e. decreasing the performance with increasing the number of function evaluation) is probably due to its breeding technique in which three individuals in the population are combined to create new offsprings. After some number of function evaluations when the algorithm converges at some level, more evolution does not help to converge any more; in fact, the new generations even diverge (i.e. the combinations of individuals lead to new offsprings that are far from the relatively good parents) from the best approximation set they could have reached. Among the three MOEAs, GDE3 shows better performance with less uncertainty for the hypervolume.

Figure 5.4 reveals that among the MOEAs, except for NSGAIII, the other two algorithms cannot keep the new generations close to the reference set (i.e. GD increases). GDE3 shows the worst GD performance with the highest

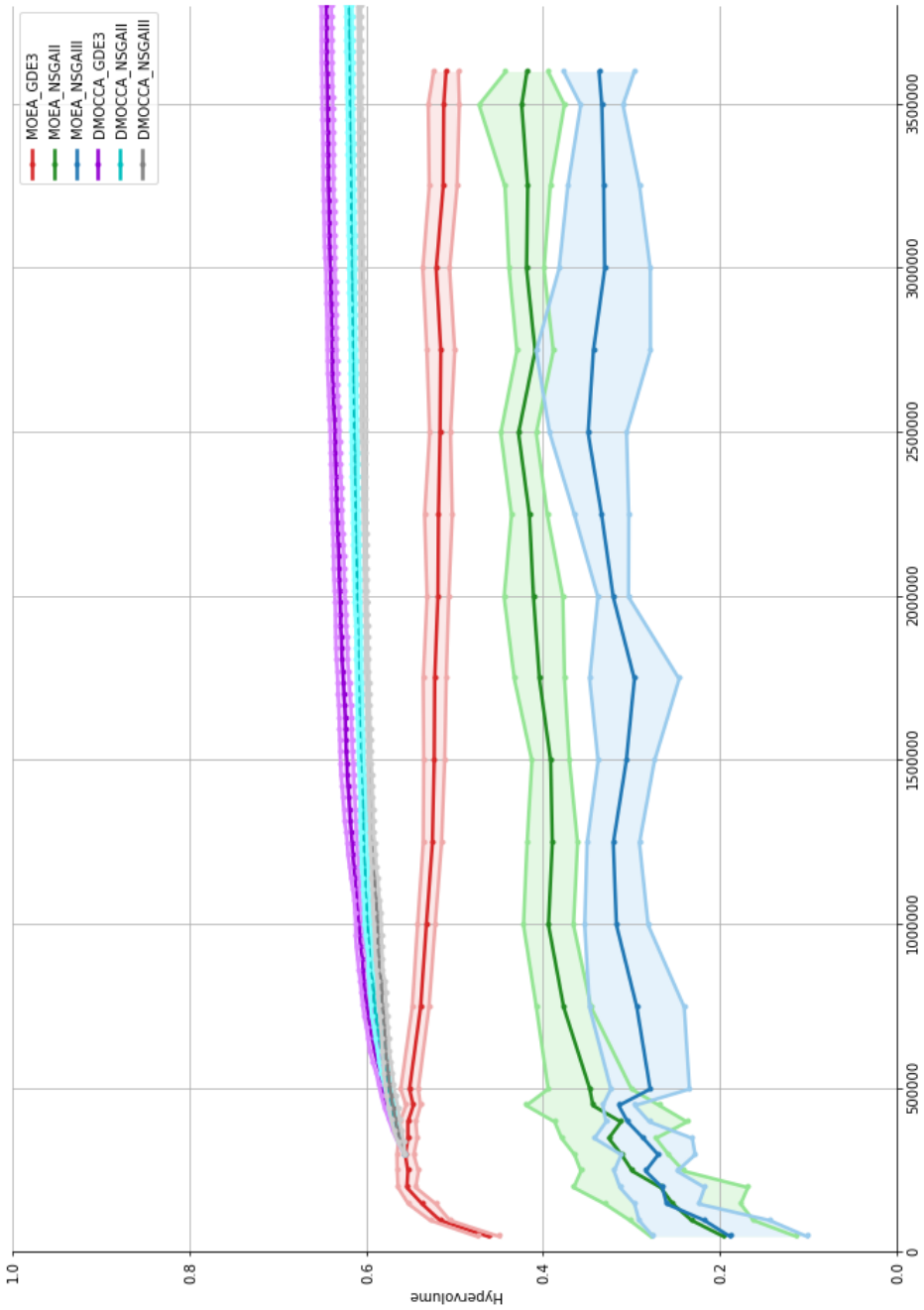

Figure 5.3: Hypervolume improves by more iterations. GDE3 shows higher capability in finding better solutions.

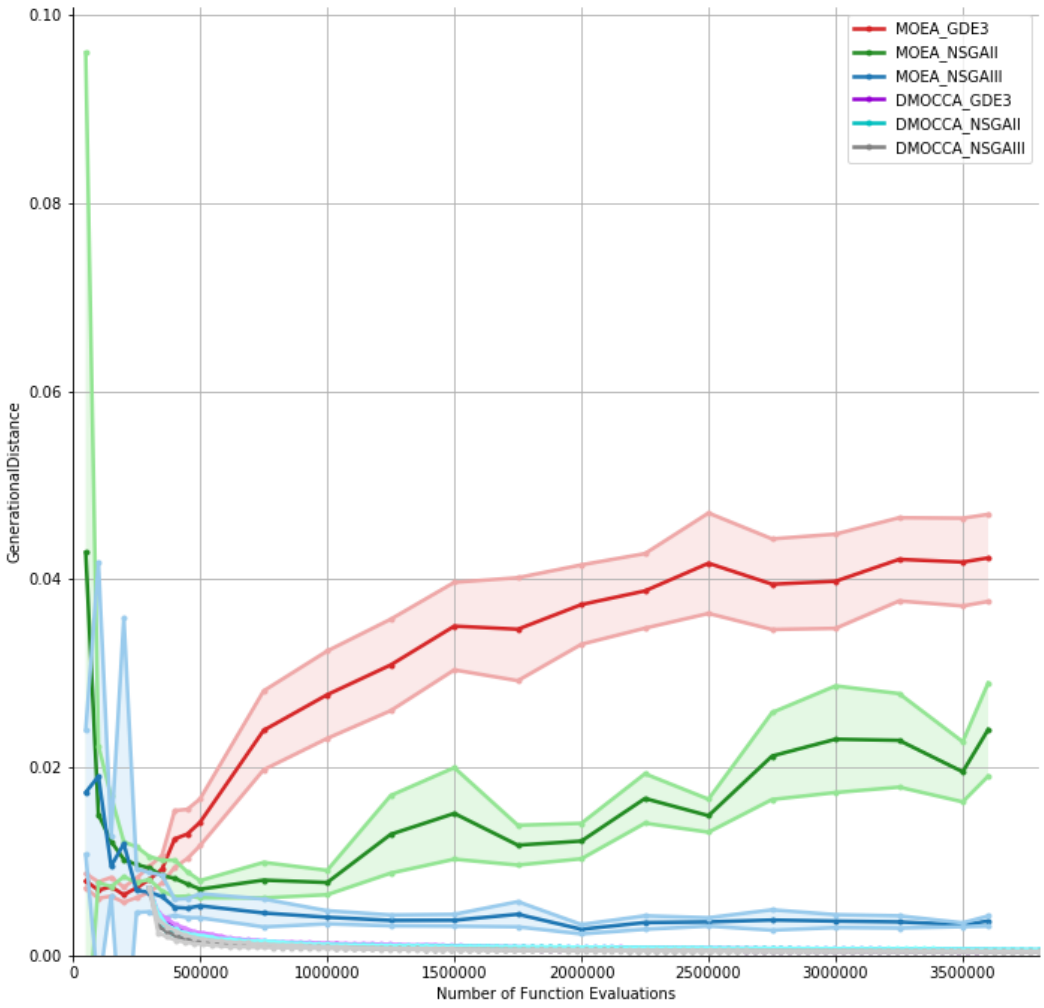

Figure 5.4: Higher number of function evaluation increases Generational Distance in GDE3 and NSGAII while DMOCCA and NSGAIII keep the solution close to the reference set.

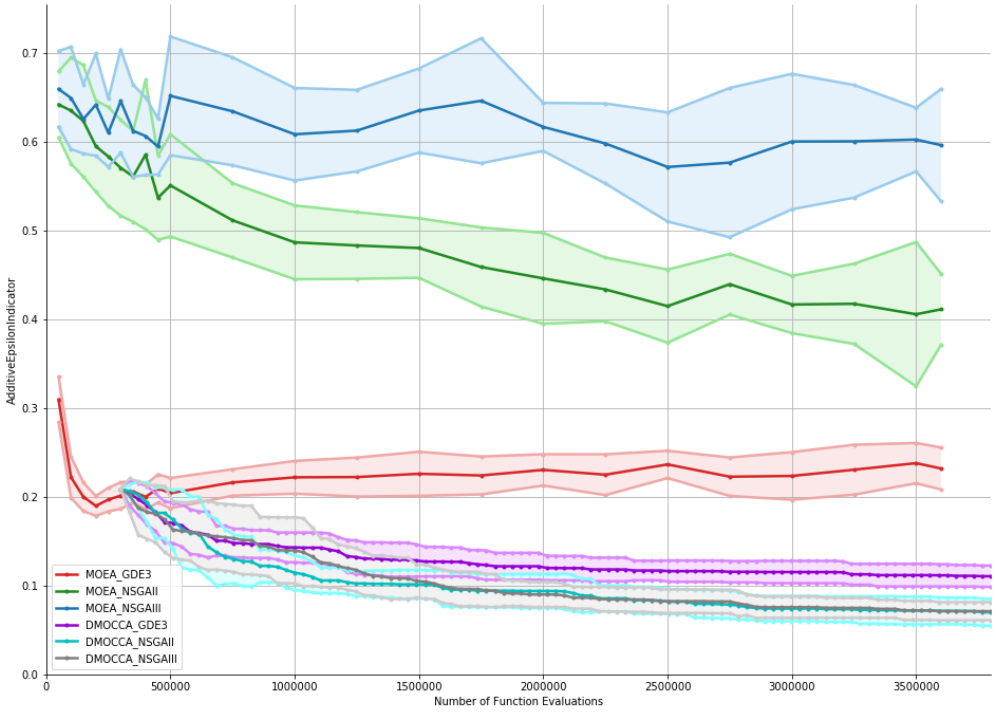

Figure 5.5: While with higher number of function evaluation NSGAII and NSGAIII show low performance with high uncertainties in epsilon indicator, and GDE3 gets a way from its best performance, DMOCCA shows a continues improvement with all three MOEAs.

uncertainty however it was the best in the HV which represents the diversity. For the same reason as mentioned before, due to the breeding technique, there is a high chance that offsprings fall far from their parents. This suggests that a dynamic parameter tuning may help to generate better offsprings in their parents' proximity. Figure 5.5 illustrates the consistency (epsilon indicator) of the approximation sets generated by the algorithms. Here, the GDE3 shows better EPI compared to other two MOEAs however similar behaviour as GD and HV is observed regarding degrading the performance with increasing the number of function evaluations.

Since GDE3 shows better diversity (HV) and consistency (EPI) for the approximation set, we select this algorithm and its best performance (i.e. with $300k$ number of function evaluation) for DMOCCA. However, in the design of DMOCCA, PMOEA runs with different random seeds, to only subject the performance of the cooperative coevolution step of DMOCCA, we found a fixed seed that GDE3 performs near the average performance at $300k$ number of function evaluation. Starting from the best average performance of GDE3, the results show a smooth improvement in terms of HV, GD, and EPI. As shown in the figures, the standard deviation in DMOCCA is small when running DMOCCA with three algorithms NSGAII, NSGAIII or GDE3.

This higher certainty in DMOCCA, is due to the fact that DMOCCA solves a less complex problem (i.e. it deals with subproblems which are simpler problem compared to the original problem), therefore, it is more successful in finding reliable solutions. Among these three MOEAs in DMOCCA, GDE3 gives rather better diversity, the same proximity and slightly lower consistency in the approximation set. NSGAII and NSGAIII have almost the same performance. The running time of the algorithms are showed in Figures 5.6 and 5.7. DMOCCA guarantees boosting of the performance

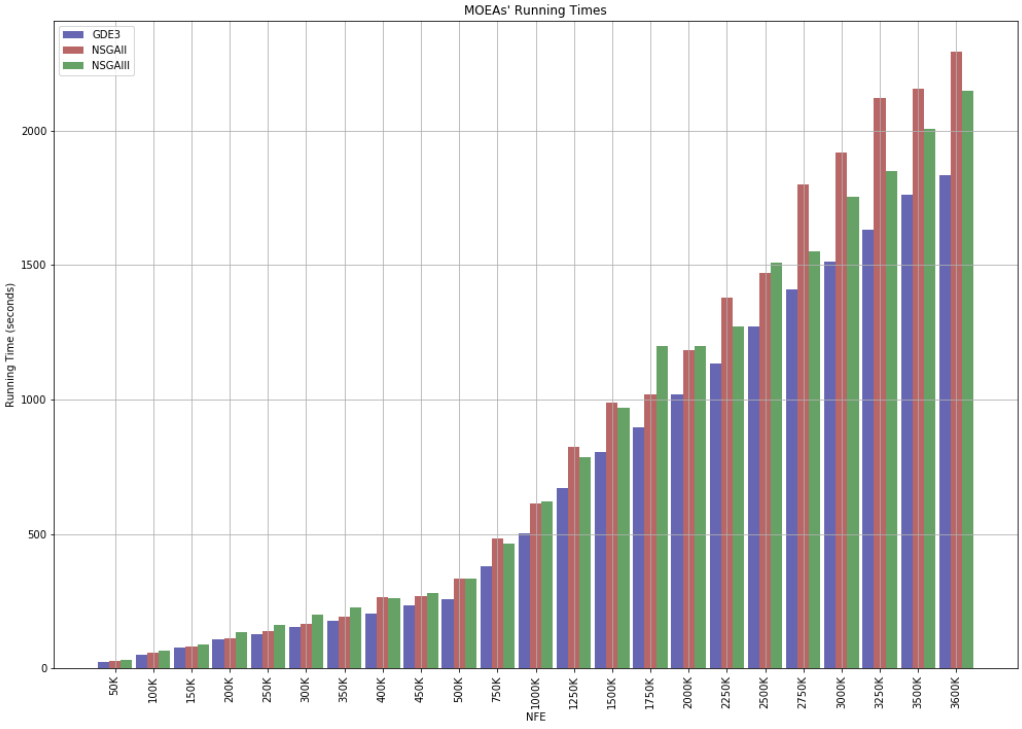

Figure 5.6: With higher number of function evaluations the running time increases in GDE3, NSGAII, and NSGAIII.

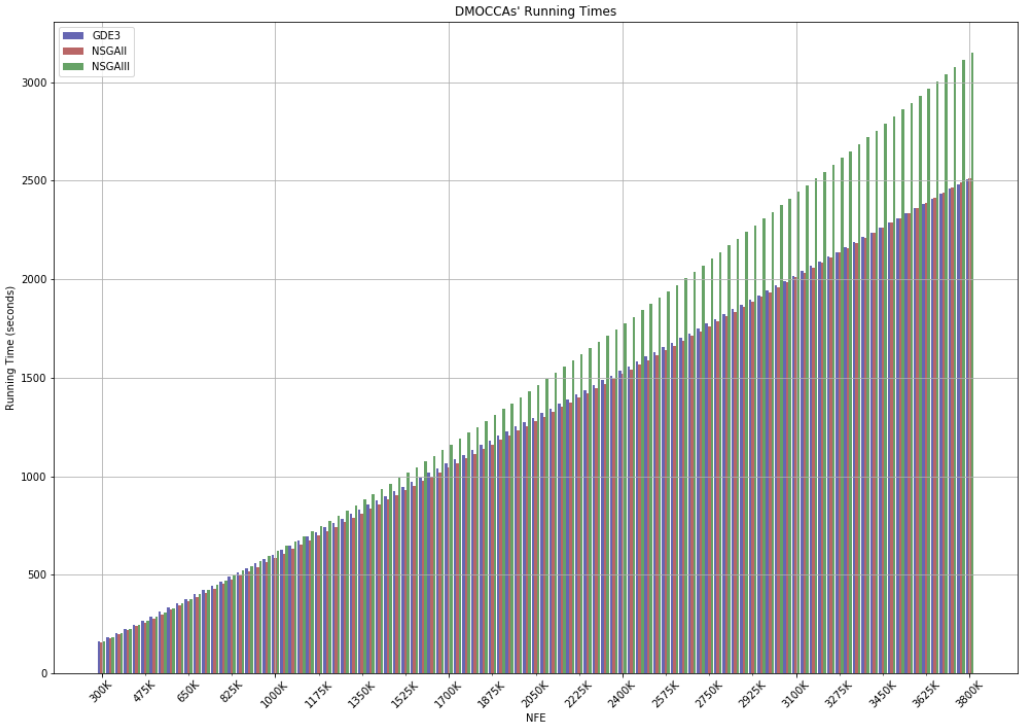

Figure 5.7: Running time in DMOCCA increases by higher NFE. All iterations in DMOCCA take the same amount of time. Therefore, since each NFE represents an iteration, the graph looks like a linear running time.

in the large-scale optimization problem when the search space is huge. But, however DMOCCA is a parallel algorithm and it runs faster than its equivalent serial algorithm (i.e. if running on a single machine), it is not considered a very fast approach.

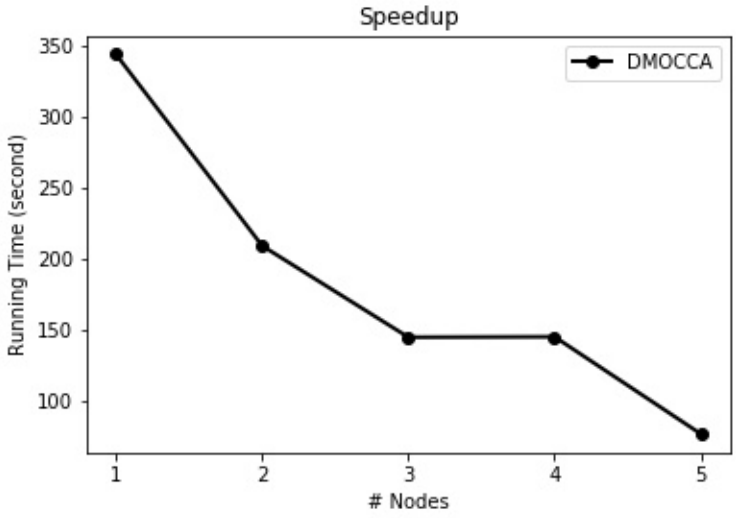

Figure 5.8: By adding more nodes into the cluster, the running time of the algorithm decreases.

Speedup Analysis To study the speedup of the parallel algorithms, we run them on clusters with the different number of nodes. Figure 5.8 proves the speedup of DMOCCA. The running times showed in this figure are for running DMOCCA with GDE3 where PMOEA is run for $100k$ and the number of iterations in DMOCCA is 10. The number of iterations does not make any change in the generality of speedup because the source of parallelism in

DMOCCA is the number of subcomponents. As shown in Figure 5.8, the running times in three and four nodes are the same. This is because there are five subcomponents in our problem instance and therefore there are five tasks and to distribute them between three or four nodes at least one node should work on two tasks one after another (i.e. there is a queue of two tasks for one node). In DMOCCA, it should be noted that due to a special splitting the subcomponents, the maximum number of nodes that are effective in parallelism is the same as the number of subcomponents.

Dimensionality Reduction It is worth to mention that using unsupervised machine learning, which is clustering speed statistics values in geohashes (i.e., minimum, average, and maximum speeds in the geohashes) and merging adjacent geohashes with similar speed profiles, is a promising approach to reduce the size of the problem. This is analogous to the dimensionality reduction approach on the G-S-VSRP in Chapter 4. This approach is not generalizable to all LSGO problems, since grouping sets of variables together and assigning one value to all of them in the same cluster will depend on the problem and hence may not be applicable to all problems. However, for those problems that can still generate valid solutions with regards to clus-

tering similar variables, this is a promising method to reduce the problem complexity and solve them with regular MOEAs.

5.5 Summary

The proposed DMOCCA is a promising algorithm to approach large-scale optimization problems when there is a wide search space for the decision variables (i.e. as in G-S-VSRP with three minimization objectives). When the problem is high-dimensional but there is a small range for each dimension then the DMOCCA can not maneuver that much over the search space since it locally searches for the neighbor solutions and if the range is small, the chance of finding better solutions in the neighbourhood is low.

The DMOCCA is a novel distributed cooperative coevolution algorithm. The idea is that to take the maximum advantage from a regular MOEA to find a good approximation set (i.e. the best algorithm for G-S-VSRP) and add them to two archives (i.e. one just keeps the early MOEA's solutions, the other one, beside those solutions, stores new solutions found in the next steps of the algorithm). Then, we improve these solutions with a distributed cooperative coevolution approach which is breaking down the large vector

into smaller subcomponents where each subcomponent creates a small subproblem running on a node in the cluster. In G-S-VSRP the subcomponents are the geohashes between every two ports. In an iterative process, a base solution is picked from any of the archives with probability 0.5 and broadcasted to the nodes in the cluster. Each node runs a regular MOEA on the subproblem by creating a random population limited to the subcomponent's search space and evolving this population. The individuals in the population are evaluated by building a complete solution using the base solution. The missing subcomponents are replaced by the corresponding subcomponents from the base solution.

We ran an experiment to examine the performance of the regular MOEAs vs. DMOCCA with three state-of-the-art MOEAs: NSGAII, NSGAIII and GDE3. Due to the difficulty of the problem, the regular MOEAs performed poorly, and after some number of function evaluation, stopped improving (i.e. NSGAII and NSGAIII) or the performance decreased (i.e. GDE3). DMOCCA starting from the best performance of GDE3 could improve the results with small uncertainty (small standard deviation showed by shadows in the graphs).

The DMOCCA showed its strength in dealing with large-scale G-S-VSRP

and proved the algorithm's speedup, however, regarding the VSRP application to the maritime sector, we could not meet the time constraint. From the experiments in Chapter 4, dimensionality reduction via clustering is an approach to reduce the difficulty of the problem and hence the computation time required by the algorithm.

In this experiment, we did not do parameter tuning. This step is left for future work. Also, selecting the base solutions from the non-dominated archives can be upgraded to a more advanced method to select the solutions as spread as possible to improve diversity in the final solution set. Moreover, running multiple DMOCCA jobs with different splitting strategies can be experimented in the future. Another possible future work is to apply different clustering techniques to reduce the dimensionality of the problem and quantify the influence of using each of them on the quality of the solutions.

Chapter 6

Conclusion

In this thesis, we addressed an important challenge in maritime disruption management that is vessel schedule recovery problem. We formulated the problem as a bi-objective optimization problem (i.e. minimizing total delay and total financial loss) and used multiobjective evolutionary algorithms as the solvers. Three scenarios were evaluated in order to capture interesting findings in the algorithms' behaviour related to their ability to optimize long vessel routes, with the vessel adopting a particular steaming policy and conducting regional/transoceanic voyages. We also discussed some insights about the optimizers' performance and statistically validate their significance. Then, the problem was extended to a three-objective problem using

automated identification system data. The historical navigational patterns were captured from data, and vessel schedule recovery problem was defined as minimizing total delay and financial loss and maximizing compliance with the historical navigational patterns. The extraction of the speed profiles was geohash-based where the minimum, average, and maximum speed in the encoded sub-regions on the voyage route were captured from the data. Geohash granularity and dimensionality reduction techniques were evaluated and discussed in the model. Using geohash-based speed mining led us to a large-scale optimization problem and a large number of function evaluations was required to find good solutions. By changing the compliance objective to a minimization, due to the curse of dimensionality, regular multiobjective algorithms performed poorly. We proposed a novel distributed cooperative coevolution algorithm which could improve the quality of the solutions with regards to three performance metrics (i.e. hypervolume, generational distance, and epsilon indicator).

In the daily operations of a fleet in the maritime industry, the findings of this research can help the decision-makers to optimize the ships' schedule when facing disruptions in the scheduled plans. Regarding the results in a variety of scenarios and experiments and the circumstances that happened,

the best algorithm and approach can be selected to control the situation.

In summary, the main contributions of the thesis are: defining S-VSRP as a bi-objective optimization problem; evaluating the problem on different scenarios employing our generated and publicly available datasets; extending the VSRP using AIS data and modeling a new big-data-enabled VSRP called G-S-VSRP which exploits historical AIS speed statistics; examining a dimensionality reduction technique (i.e. using k-means clustering) on the quality of the solutions and running time of the optimization algorithms; proposing a novel distributed cooperative coevolution method to deal with the large-scale G-S-VSRP optimization problem. The research in this Ph.D. program resulted in one journal and five conference and workshop publications listed in section 1.4.

Limitations and Future Works: The limitation and future work for each proposed approach are outlined in the following:

VSRP: In this research work, we only considered a speed-based recovery strategy in our formulation. This can be extended by combining other strategies such as port swapping or port skipping in future work. Also, the proposed approach can be extended to the entire fleet instead of a single

vessel. Furthermore, parameter tuning for the multiobjective optimization algorithms can be instance-specific which we skipped in this work since it is computationally expensive.

G-S-VSRP: The other potential for future work is tuning the population size, a number of function evaluations, crossover and mutation rates according to the problem instances in G-S-VSRP. The influence of different clustering methods used as the dimensionality reduction can be examined.

DMOCCA: The DMOCCA can be evaluated using different splitting strategies and a more advanced selection of the base solution. A comparison of DMOCCA with reduced dimension instances (using clustering) can be added to the experiments. Moreover, running multiple DMOCCA jobs with different splitting strategies, and combining their solutions can be experimented in the future. Comparing the results of the proposed approach with other large-scale optimization techniques can be investigated as well.

Finally, and as generalized future works, experiences in this thesis are based on instances from five-month AIS data. In the future, more data can be used to create instances which reflect more knowledge in data. Also, the proposed optimization approaches in this work can be generalized to other

transportation systems such as air, bus, or train.

Bibliography

[1] T. Abeel, Y. V. de Peer, and Y. Saeys, JAVA-ml: A machine learning library, *The Journal of Machine Learning Research*, vol. 10, Apr. 2009.

[2] "AIS experts: Types of AIS information broadcast for each ship. `http://www.milltechmarine.com/`, Accessed: 2017.

[3] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera, Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework., Journal of Multiple-Valued Logic & Soft Computing 17.

[4] A. Azaron, K. Brown, S. Tarim, M. Modarres, A multiobjective stochastic programming approach for supply chain design considering risk, International Journal of Production Economics 116 (1) (2008) 129–138.

[5] A. Bargiela and W. Pedrycz, *Granular computing: an introduction*, vol. 717. Springer Science & Business Media, 2012.

[6] R. Bello, R. Falcon, W. Pedrycz, and J. Kacprzyk, *Granular Computing: at the Junction of Rough Sets and Fuzzy Sets.* Berlin-Heidelberg, Germany: Springer Verlag, 2008.

[7] Mohammadi, Mohammad Mahdi, Bijan Raahemi, Fatemeh Cheraghchi, Wael Obidallah, and Elnaz Bigdeli. Big data analytics using hadoop. In Proceedings of 24th Annual International Conference on Computer Science and Software Engineering, pp. 323-325. IBM Corp., 2014.

[8] de Kok, A.G. and Graves, S.C. eds., 2003. Supply chain management: Design, coordination and operation (Vol. 11). Amsterdam: Elsevier.

[9] Brouer, B.D., Karsten, C.V. and Pisinger, D., 2016. Big data optimization in maritime logistics. In Big data optimization: recent developments and challenges (pp. 319-344). Springer International Publishing.

[10] B. D. Brouer, J. Dirksen, D. Pisinger, C. E. Plum, B. Vaaben, The Vessel Schedule Recovery Problem (VSRP)–a MIP model for handling disruptions in liner shipping, European Journal of Operational Research 224 (2) (2013) 362–374.

[11] E. K. Burke, G. Kendall, et al., Search methodologies, Springer, 2005.

[12] D. M. Cabrera, Evolutionary algorithms for large-scale global optimisation: a snapshot, trends and challenges, *Progress in Artificial Intelligence*, vol. 5, no. 2, pp. 85–89, 2016.

[13] Clausen, J., Larsen, A., Larsen, J. and Rezanova, N.J., 2010. Disruption management in the airline industry—Concepts, models and methods. Computers and Operations Research, 37(5), pp.809-821.

[14] Cariou, P., 2011. Is slow steaming a sustainable means of reducing CO 2 emissions from container shipping?. Transportation Research Part D: Transport and Environment, 16(3), pp.260-264.

[15] C. C. Coello, G. B. Lamont, D. A. Van Veldhuizen, Evolutionary algorithms for solving multiobjective problems, Springer, 2007.

[16] D. W. Corne, N. R. Jerram, J. D. Knowles, M. J. Oates, PESA-II: Region-based selection in evolutionary multiobjective optimization, in: Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation, Morgan Kaufmann Publishers Inc., 2001, pp. 283–290.

[17] C. W. Craighead, J. Blackhurst, M. J. Rungtusanatham, R. B. Handfield, The severity of supply chain disruptions: design characteristics and mitigation capabilities, Decision Sciences 38 (1) (2007) 131–156.

[18] Thai, V.V., 2008. Service quality in maritime transport: conceptual model and empirical evidence. Asia Pacific Journal of Marketing and Logistics, 20(4), pp.493-518.

[19] Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T.A.M.T., A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE transactions on evolutionary computation 6 (2) (2002) 182–197.

[20] K. Deb, H. Jain, An evolutionary many-objective optimization algorithm using reference-point-based non-dominated sorting approach, part i: Solving problems with box constraints., IEEE Trans. Evolutionary Computation 18 (4) (2014) 577–601.

[21] Teijeiro, D., Pardo, X.C., Penas, D.R., González, P., Banga, J.R. and Doallo, R., 2016, August. Evaluation of parallel differential evolution implementations on MapReduce and Spark. In European Conference on Parallel Processing (pp. 397-408). Springer, Cham.

[22] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm and Evolutionary Computation 1 (1) (2011) 3–18.

[23] D. Dienst, Airline disruption management-the aircraft recovery problem, Master's thesis, Technical University of Denmark, DTU, DK-2800 Kgs. Lyngby, Denmark (2010).

[24] Ehrgott, M., 2006. Multicriteria optimization. Springer Science & Business

[25] "Maritime ICT cloud enables ships to join the networked society. `https://www.ericsson.com/en/press-releases/2015/1/maritime-ict-cloud-enables-ships-to-join-the-networked-society`, Accessed: September 2017.

[26] European Commission website `https://ec.europa.eu/transport/modes/maritime_en`, Accessed: September 2017

[27] K. Fagerholt, G. Laporte, I. Norstad, Reducing fuel emissions by optimizing speed on shipping routes, Journal of the Operational Research Society 61 (3) (2010) 523–529.

[28] C. M. Fonseca, P. J. Fleming, On the performance assessment and comparison of stochastic multiobjective optimizers, in: International Conference on Parallel Problem Solving from Nature, Springer, 1996, pp. 584–593.

[29] Esmaeilpour, A., Bigdeli, E., Cheraghchi, F., Raahemi, B. and Far, B.H., 2016, May. Distributed Gaussian Mixture Model Summarization Using the MapReduce Framework. In Canadian Conference on Artificial Intelligence (pp. 323-335). Springer International Publishing.

[30] E. Glorieux, B. Svensson, F. Danielsson, and B. Lennartson, Improved constructive cooperative coevolutionary differential evolution for large-scale optimisation," in *Computational Intelligence, 2015 IEEE Symposium Series on*, pp. 1703–1710, IEEE, 2015.

[31] Lekhavat, S., Aydin, N., Lee, H. and Bilici, A., "DECISION SUPPORT SYSTEM FOR MULTIOBJECTIVE SUSTAINABLE MARINE SHIPPING", European, Mediterranean & Middle Eastern Conference on Information Systems 2015.

[32] D. Hadka, P. Reed, Borg: An auto-adaptive many-objective evolutionary computing framework, Evolutionary computation 21 (2) (2013) 231–259.

[33] Haimes, Y.Y., 1971. On a bicriterion formulation of the problems of integrated system identification and system optimization. IEEE transactions on systems, man, and cybernetics, 1(3), pp.296-297.

[34] Definition supply chain management `https://www.ericsson.com/en/press-releases/2015/1/maritime-ict-cloud-enables-ships-to-join-the-networked-society`, September 2017.

[35] Holland, J.H., 1992. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press.

[36] Deb, K., 2014. Multi-objective optimization. In Search methodologies (pp. 403-449). Springer US.

[37] Kidston, D. and Kunz, T., 2008. Challenges and opportunities in managing maritime networks. IEEE Communications Magazine, 46(10). [**?**] Kjeldsen, K.H., 2012. Routing and scheduling in liner shipping. Aarhus UniversitetAarhus University, School of Business and Social SciencesSchool of Business and Social Sciences, Institut for ØkonomiDepartment of Economics and Business, Institut for Økonomi-CORAL-Centre for Operations Research Applications in LogisticsDepartment of Economics and Business-CORAL-Centre for Operations Research Applications in Logistics.

[38] P. R. Kleindorfer, G. H. Saad, Managing disruption risks in supply chains, Production and operations management 14 (1) (2005) 53–68.

[39] S. Kukkonen, J. Lampinen, GDE3: The third evolution step of generalized differential evolution, in: Evolutionary Computation, 2005. The 2005 IEEE Congress on, Vol. 1, IEEE, 2005, pp. 443–450.

[40] Larus Technologies Inc., `https://www.larus.com/`, note = Accessed: 2017-08-28.

[41] C.-Y. Lee, D.-P. Song, Ocean container transport in global supply chains: Overview and research opportunities, Transportation Research Part B: Methodological 95 (2017) 442–474.

[42] S. Lekhavat, N. Aydin, H. Lee, A. Bilici, Decision support system for multiobjective sustainable marine shipping, European, Mediterranean & Middle Eastern Conference on Information Systems.

[43] Levinson, M. , 2006. The Box: How the Shipping Container Made the World Smaller and the World Economy Bigger. Princeton University Press, Princeton, NJ.

[44] C. Li, X. Qi, D. Song, Real-time schedule recovery in liner shipping service with regular uncertainties and disruption events, Transportation Research Part B: Methodological 93 (2016) 762–788.

[45] C. Li, X. Qi, C.-Y. Lee, Disruption recovery for a vessel in liner shipping, Transportation Science 49 (4) (2015) 900–921.

[46] Lloyds Marine Intelligence Unit `http://www.imsf.info/media/1129/4-wally_mandryk_lmiu_imsf09.pdf` Accessed: September 2017

[47] Mahdavi, S., Shiri, M.E. and Rahnamayan, S., 2015. Metaheuristics in large-scale global continues optimization: A survey. Information Sciences, 295, pp.407-428.

[48] J. MacQueen *et al.*, Some methods for classification and analysis of multivariate observations, in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, pp. 281–297, Oakland, CA, USA., 1967.

[49] Talbi, E.G., 2009. Metaheuristics: from design to implementation (Vol. 74). John Wiley and Sons.

[50] B. L. Miller, D. E. Goldberg, Genetic algorithms, tournament selection, and the effects of noise, Complex systems 9 (3) (1995) 193–212.

[51] P. Moscato and C. Cotta, Memetic algorithms, *Handbook of Applied Optimization*, pp. 157–167, 2002.

[52] J. Mulder, R. Dekker, M. Sharifyazdi, Designing robust liner shipping schedules: Optimizing recovery actions and buffer times, Report/Econometric Institute, Erasmus University Rotterdam (2012) 1–24.

[53] T. E. Notteboom, The time factor in liner shipping services, Maritime Economics & Logistics 8 (1) (2006) 19–39.

[54] T. Notteboom, P. Cariou, Fuel surcharge practices of container shipping lines: Is it about cost recovery or revenue making, in: Proceedings of the 2009 International Association of Maritime Economists (IAME) Conference, IAME, 2009, pp. 24–26.

[55] T. Notteboom, P. Cariou, Slow steaming in container liner shipping: is there any impact on fuel surcharge practices?, The International Journal of Logistics Management 24 (1) (2013) 73–86.

[56] Pantuso, G., Fagerholt, K. and Hvattum, L.M., 2014. A survey on maritime fleet size and mix problems. European Journal of Operational Research, 235(2), pp.341-349.

[57] M. A. Potter and K. A. De Jong, A cooperative coevolutionary approach to function optimization, in *International Conference on Parallel Problem Solving from Nature*, pp. 249–257, Springer, 1994.

[58] X. Qi, Disruption management for liner shipping, in: Handbook of Ocean Container Transport Logistics, Springer, 2015, pp. 231–249.

[59] X. Qi, D.-P. Song, Minimizing fuel emissions by optimizing vessel schedules in liner shipping with uncertain port times, Transportation Research Part E: Logistics and Transportation Review 48 (4) (2012) 863–880.

[60] Rodseth, Ø.J., Perera, L.P. and Mo, B., 2016. Big data in shipping-Challenges and opportunities, 15th International Conference on Computer and IT Applications in the Maritime Industries (COMPIT 2016), 2016.

[61] A. Saltelli, P. Annoni, I. Azzini, F. Campolongo, M. Ratto, S. Tarantola, Variance based sensitivity analysis of model output. design and

estimator for the total sensitivity index, Computer Physics Communications 181 (2) (2010) 259–270.

[62] M. R. Sierra, C. A. C. Coello, Improving pso-based multiobjective optimization using crowding, mutation and epsilon dominance, in: Evolutionary multi-criterion optimization, Springer, 2005, pp. 505–519.

[63] `https://spark.apache.org/`

[64] D.-P. Song, D. Li, P. Drake, Multi-objective optimization for planning liner shipping service with uncertain port times, Transportation Research Part E: Logistics and Transportation Review 84 (2015) 1–22.

[65] Storn, R. and Price, K., 1997. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization, 11(4), pp.341-359.

[66] Tang, C.S., Robust strategies for mitigating supply chain disruptions, International Journal of Logistics: Research and Applications 9 (1) (2006) 33–45.

[67] B. G. Thengvall, G. Yu, J. F. Bard, Multiple fleet aircraft schedule recovery following hub closures, Transportation Research Part A: Policy and Practice 35 (4) (2001) 289–308.

[68] B. G. Thengvall, J. F. Bard, G. Yu, A bundle algorithm approach for the aircraft schedule recovery problem during hub closures, Transportation Science 37 (4) (2003) 392–407.

[69] Notteboom, T.E., 2006. The time factor in liner shipping services. Maritime Economics and Logistics, 8(1), pp.19-39.

[70] S. M. Wagner, N. Neshat, Assessing the vulnerability of supply chains using graph theory, International Journal of Production Economics 126 (1) (2010) 121–129.

[71] Wang, S. and Meng, Q., Liner ship route schedule design with sea contingency time and port time uncertainty, Transportation Research Part B: Methodological 46 (5) (2012) 615–633.

[72] Wang, S. and Meng, Q., Robust schedule design for liner shipping services, Transportation Research Part E: Logistics and Transportation Review 48 (6) (2012) 1093–1106.

[73] Wang, S. and Meng, Q., 2014. Liner shipping network design with deadlines. Computers and Operations Research, 41, pp.140-149.

[74] K. Wang, A. K. Ng, J. S. L. Lam, X. Fu, Cooperation or competition? factors and conditions affecting regional port governance in south china, Maritime Economics & Logistics 14 (3) (2012) 386–408.

[75] Wang, H. and Créput, J.C., 2014, May. Parallel and distributed implementation models for bio-inspired optimization algorithms. In International Conference on Swarm Intelligence Based Optimization (pp. 68-79). Springer, Cham.

[76] Wang, H., Osen, O.L., Li, G., Li, W., Dai, H.N. and Zeng, W., Big data and industrial internet of things for the maritime industry in northwestern Norway, (2015) IEEE (pp. 1-5).

[77] A free and open source JAVA framework for multiobjective optimization, `http://moeaframework.org/`, accessed: 2017-05-08.

[78] Fuel consumption by containership size and speed, the geography of transport systems, `https://people.hofstra.edu/geotrans/eng/ch8en/conc8en/fuel_consumption_containerships.html`, accessed: 2017.

[79] G. Yu, X. Qi, Disruption management: framework, models and applications, World Scientific, 2004.

[80] L. Zhang, Q. Meng, and T. F. Fwa, Big AIS data based spatial-temporal analyses of ship traffic in Singapore port waters,*Transportation Research Part E: Logistics and Transportation Review*, 2017.

[81] E. Zitzler, M. Laumanns, L. Thiele, SPEA2: Improving the strength pareto evolutionary algorithm.(2001)

[82] `http://www.ibmbigdatahub.com/infographic/four-vs-big-data`

[83] Cheraghchi F, Iranzad A, Raahemi B. Subspace selection in high-dimensional big data using genetic algorithm in apache spark. InProceedings of the Second International Conference on Internet of things and Cloud Computing 2017 Mar 22 (p. 54). ACM.

[84] Mohan, A. and Remya, G., 2014. A parallel implementation of ant colony optimization for tsp based on mapreduce framework. International Journal of Computer Applications, 88(8).

[85] Gaifang, D., Xueliang, F., Honghui, L. and Pengfei, X., 2017. Co-operative ant colony-genetic algorithm based on spark. Computers & Electrical Engineering, 60, pp.66-75.

[86] Wu, B., Wu, G. and Yang, M., 2012, May. A mapreduce based ant colony optimization approach to combinatorial optimization problems. In Natural Computation (ICNC), 2012 Eighth International Conference on (pp. 728-732). IEEE.

[87] Wang, L., Wang, Y. and Xie, Y., 2015. Implementation of a parallel algorithm based on a spark cloud computing platform. Algorithms, 8(3), pp.407-414.

[88] Hormozi, E., Akbari, M.K., Javan, M.S. and Hormozi, H., 2013, April. Performance evaluation of a fraud detection system based artificial immune system on the cloud. In Computer Science & Education (ICCSE), 2013 8th International Conference on (pp. 819-823). IEEE.

[89] Halvaiee, N.S. and Akbari, M.K., 2014. A novel model for credit card fraud detection using Artificial Immune Systems. Applied Soft Computing, 24, pp.40-49.

[90] Cheraghchi, F., Iranzad, A. and Raahemi, B., 2017, March. Subspace selection in high-dimensional big data using genetic algorithm in apache spark. In Proceedings of the Second International Conference on Internet of things and Cloud Computing (p. 54). ACM.

[91] Aljarah, I. and Ludwig, S.A., 2012, November. Parallel particle swarm optimization clustering algorithm based on mapreduce methodology. In Nature and biologically inspired computing (NaBIC), 2012 fourth world congress on (pp. 104-111). IEEE.

[92] Cao B, Li W, Zhao J, Yang S, Kang X, Ling Y, Lv Z. Spark-based parallel cooperative coevolution particle swarm optimization algorithm. InWeb Services (ICWS), 2016 IEEE International Conference on 2016 Jun 27 (pp. 570-577). IEEE.

[93] Teijeiro D, Pardo XC, Penas DR, González P, Banga JR, Doallo R. Evaluation of parallel differential evolution implementations on MapReduce and Spark. InEuropean Conference on Parallel Processing 2016 Aug 24 (pp. 397-408). Springer.

[94] Cabrera, D.M., 2016. Evolutionary algorithms for large-scale global optimisation: a snapshot, trends and challenges. Progress in Artificial

Intelligence, 5(2), pp.85-89.

[95] Zhao YOng YAN, Tsang IS. The Emerging "Big Dimensionality". IEEE Computational Intelligence Magazine. 2014 Aug;9(3):14-26.

[96] Hui Wang, Zhijian Wu, and Shahryar Rahnamayan. "Enhanced opposition-based differential evolution for solving high-dimensional continuous optimization problems". In: Soft Computing 15.11 (2011), pp. 2127–2140.

[97] Sedigheh Mahdavi, Shahryar Rahnamayan, and Kalyanmoy Deb. "Center-based initialization of cooperative coevolutionary algorithm for large-scale optimization". In: IEEE Congress on Evolutionary Computation. IEEE. 2016, pp. 3557–3565.

[98] Hongwei Ge et al. "Cooperative differential evolution with fast variable interdependence learning and cross-cluster mutation". In: Applied Soft Computing 36 (2015), pp. 300–314.

[99] Ali Wagdy Mohamed and Abdulaziz S Almazyad. "Differential Evolution with Novel Mutation and Adaptive Crossover Strategies for Solving Large Scale Global Optimization Problems". In: Applied Computational Intelligence and Soft Computing 2017 (2017).

[100] Cheraghchi F, Abualhaol I, Falcon R, Abielmona R, Raahemi B, Petriu E. Modeling the speed-based vessel schedule recovery problem using evolutionary multiobjective optimization. Information Sciences. 2018 Jun 1;448:53-74.

[101] Cheraghchi F, Abualhaol I, Falcon R, Abielmona R, Raahemi B, Petriu E. Big-data-enabled modelling and optimization of granular speed-based vessel schedule recovery problem. In Big Data IEEE International Conference on 2017 Dec 11 (pp. 1786-1794).

[102] Enying Li, Hu Wang, and Fan Ye. "Two-level Multi-surrogate Assisted Optimization method for high-dimensional nonlinear problems". In: Applied Soft Computing 46 (2016), pp. 26–36.

[103] Selvakumar Ulaganathan et al. "A hybrid sequential sampling based meta-modelling approach for high-dimensional problems". In: IEEE Congress on Evolutionary Computation. IEEE. 2016, pp. 1917–1923.

[104] Tseng LY, Chen C. Multiple trajectory search for large scale global optimization. InEvolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on 2008 Jun 1 (pp. 3052-3059). IEEE.

[105] Molina D, Lozano M, Herrera F. Memetic algorithm with local search chaining for large scale continuous optimization problems. InEvolutionary Computation, 2009. CEC'09. IEEE Congress on 2009 May 18 (pp. 830-837). IEEE.

[106] Dorronsoro B, Danoy G, Nebro AJ, Bouvry P. Achieving superlinear performance in parallel multiobjective evolutionary algorithms by means of cooperative coevolution. Computers & Operations Research. 2013 Jun 1;40(6):1552-63.

[107] Wang Y, Li Y, Chen Z, Xue Y. Cooperative particle swarm optimization using MapReduce. Soft Computing. 2017 Nov 1;21(22):6593-603.

[108] Ye S, Dai G, Peng L, Wang M. A hybrid adaptive coevolutionary differential evolution algorithm for large-scale optimization. InEvolutionary Computation (CEC), 2014 IEEE Congress on 2014 Jul 6 (pp. 1277-1284). IEEE.

[109] Deng W, Chen R, He B, Liu Y, Yin L, Guo J. A novel two-stage hybrid swarm intelligence optimization algorithm and application. Soft Computing. 2012 Oct 1;16(10):1707-22.

[110] Bolufé-Röhler A, Fiol-González S, Chen S. A minimum population search hybrid for large scale global optimization. InEvolutionary Computation (CEC), 2015 IEEE Congress on 2015 May 25 (pp. 1958-1965). IEEE.

[111] Keerativuttitumrong N, Chaiyaratana N, Varavithya V. Multi-objective cooperative coevolutionary genetic algorithm. InInternational Conference on Parallel Problem Solving from Nature 2002 Sep 7 (pp. 288-297). Springer, Berlin, Heidelberg.

[112] Maneeratana K, Boonlong K, Chaiyaratana N. Multi-objective optimisation by cooperative coevolution. InInternational Conference on Parallel Problem Solving from Nature 2004 Sep 18 (pp. 772-781). Springer, Berlin, Heidelberg.

[113] Maneeratana K, Boonlong K, Chaiyaratana N. Cooperative coevolutionary genetic algorithms for multiobjective topology design. Computer-Aided Design and Applications. 2005 Jan 1;2(1-4):487-96.

[114] rey Horn J, Nafpliotis N, Goldberg DE. Multiobjective optimization using the niched pareto genetic algorithm. IlliGAL report. 1993 Jul(93005):61801-2296.

[115] Srinivas N, Deb K. Muiltiobjective optimization using non-dominated sorting in genetic algorithms. Evolutionary computation. 1994 Sep;2(3):221-48.

[116] Deb K, Goel T. Controlled elitist non-dominated sorting genetic algorithms for better convergence. InInternational Conference on Evolutionary Multi-Criterion Optimization 2001 Mar 7 (pp. 67-81). Springer, Berlin, Heidelberg.

[117] Tan KC, Yang YJ, Goh CK. A distributed cooperative coevolutionary algorithm for multiobjective optimization. IEEE Transactions on Evolutionary Computation. 2006 Oct;10(5):527-49.

[118] Iorio AW, Li X. A cooperative coevolutionary multiobjective algorithm using non-dominated sorting. InGenetic and Evolutionary Computation Conference 2004 Jun 26 (pp. 537-548). Springer, Berlin, Heidelberg.

[119] Coello CC, Sierra MR. A coevolutionary multiobjective evolutionary algorithm. InEvolutionary Computation, 2003. CEC'03. The 2003 Congress on 2003 Dec 8 (Vol. 1, pp. 482-489). IEEE.

[120] Yang Z, Zhang J, Tang K, Yao X, Sanderson AC. An adaptive coevolutionary differential evolution algorithm for large-scale optimization. InEvolutionary Computation, 2009. CEC'09. IEEE Congress on 2009 May 18 (pp. 102-109). IEEE.

[121] Glorieux E, Svensson B, Danielsson F, Lennartson B. Multi-objective constructive cooperative coevolutionary optimization of robotic press-line tending. Engineering Optimization. 2017 Oct 3;49(10):1685-703.

[122] Wang Y, Li Y, Chen Z, Xue Y. Cooperative particle swarm optimization using MapReduce. Soft Computing. 2017 Nov 1;21(22):6593-603.

[123] Song DP, Li D, Drake P. Multi-objective optimization for a liner shipping service from different perspectives. Transportation research procedia. 2017 Jan 1;25:251-60.

[124] Aydin N, Lee H, Mansouri SA. Speed optimization and bunkering in liner shipping in the presence of uncertain service times and time windows at ports. European Journal of Operational Research. 2017 May 16;259(1):143-54.

[125] Zhao Y, Jia R, Jin N, He Y. A novel method of fleet deployment based on route risk evaluation. Information Sciences. 2016 Dec 1;372:731-44.

[126] Cicerone S, Di Stefano G, Schachtebeck M, Schöbel A. Multi-stage recovery robustness for optimization problems: A new concept for planning under disturbances. Information Sciences. 2012 May 1;190:107-26.

[127] Mirjalili S, Lewis A. Obstacles and difficulties for robust benchmark problems: A novel penalty-based robust optimisation method. Information Sciences. 2016 Jan 20;328:485-509.

[128] Bai R, Kendall G, Qu R, Atkin JA. Tabu assisted guided local search approaches for freight service network design. Information Sciences. 2012 Apr 15;189:266-81.

[129] "Larus Technologies." https://www.larus.com/.

[130] `http://http://www.site.uottawa.ca/~petriu/BigData_mIoT-CSPS2018.pdf`.

[131] Kryo - Java serialization and cloning: fast, efficient, automatic. `https://github.com/EsotericSoftware/kryo`.

[132] `https://spark-summit.org/wp-content/uploads/2015/03/SparkSummitEast2015-AdvDevOps-StudentSlides.pdf`.

[133] Official geohash website: `http://geohash.org/`

[134] Mandaraka-Sheppard A. Modern maritime law and risk management. Informa Law from Routledge; 2014 Feb 4.

[135] Lam, J.S.L., 2012. Risk management in maritime logistics and supply chains. In Maritime logistics: Contemporary issues (pp. 117-131). Emerald Group Publishing Limited.

[136] Fagerholt K, Gausel NT, Rakke JG, Psaraftis HN. Maritime routing and speed optimization with emission control areas. Transportation Research Part C: Emerging Technologies. 2015 Mar 1;52:57-73.

[137] Al-Khayyal F, Hwang SJ. Inventory constrained maritime routing and scheduling for multi-commodity liquid bulk, Part I: Applications and model. European Journal of Operational Research. 2007 Jan 1;176(1):106-30.

[138] Lokuge P, Alahakoon D. Improving the adaptability in automated vessel scheduling in container ports using intelligent software agents. European Journal of Operational Research. 2007 Mar 16;177(3):1985-2015.

[139] Chen ZL, Lei L, Zhong H. Container vessel scheduling with bidirectional flows. Operations Research Letters. 2007 Mar 1;35(2):186-94.

[140] Meng Q, Wang S, Andersson H, Thun K. Containership routing and scheduling in liner shipping: overview and future research directions. Transportation Science. 2013 May 10;48(2):265-80.

[141] Shvachko K, Kuang H, Radia S, Chansler R. The hadoop distributed file system. InMSST 2010 May 3 (Vol. 10, pp. 1-10).

[142] `https://aws.amazon.com/s3/`

[143] `http://cassandra.apache.org/`

[144] Miqing, Li, Zheng Jinhua, and Luo Biao. "A multi-objective evolutionary algorithm based on minimum spanning tree [J]." Journal of Computer Research and Development 5 (2009).

[145] `https://en.wikipedia.org/wiki/Geohash`

[146] Shvachko, K., Kuang, H., Radia, S. and Chansler, R., 2010, May. The hadoop distributed file system. In MSST (Vol. 10, pp. 1-10).