



Norwegian University of
Science and Technology

The Multi-objective Supply Vessel Planning Problem

A Hybrid Genetic Search Approach

Thomas Borthen
Henrik Loennechen

Industrial Economics and Technology Management

Submission date: June 2016

Supervisor: Kjetil Fagerholt, IØT

Co-supervisor: Xin Wang, IØT

Norwegian University of Science and Technology
Department of Industrial Economics and Technology Management

Problem description from the master's thesis agreement

The purpose of this thesis is to study and implement solution methods for the supply vessel planning problem (SVPP) with multiple objectives. The SVPP is a planning problem that occurs within the offshore petroleum industry and consists of deciding a fleet of platform supply vessels (PSVs) and the voyages the PSVs shall sail in order to service a set of offshore installations. The problem definition and test data are based on experiences and data from Statoil.

Preface

This master's thesis is written as a part of our MSc. in Industrial Economics and Technology Management at the Norwegian University of Science and Technology, Department of Industrial Economics and Technology Management. The thesis is a continuation of the work done in our specialization project during the fall of 2015.

The master's thesis focuses on tactical planning of Statoil's offshore supply operations, which is a part of their upstream supply chain. The work is done as a part of the MOLO (Maritime Offshore Logistics Optimization) project, a collaboration between Statoil, the Norwegian University of Science and Technology, the Norwegian Marine Technology Research Institute (MARINTEK), and Molde University College.

We would like to express our gratitude towards our supervisors Professor Kjetil Fagerholt and Postdoctoral Fellow Xin Wang for valuable guidance, interesting discussions and precise and constructive feedback.

Thomas Borthen and Henrik Loennechen

Trondheim, June 2016

Abstract

The supply vessel planning problem (SVPP) is a problem faced by Statoil, the largest operator on the Norwegian continental shelf. Offshore installations need supplies from an onshore supply depot in order to operate, and the supplies are transported by platform supply vessels (PSVs). The objective of the SVPP is to minimize the costs related to the chartering and operation of PSVs, while maintaining a reliable supply service. In addition to minimal costs, the decision makers at Statoil have requested solutions that are persistent and robust. Persistent solutions are solutions that have few changes from the previous solution, while robust solutions are solutions that are capable of handling unforeseen events, such as delays. The SVPP with multiple objectives is referred to as the multi-objective supply vessel planning problem (MSVPP), and this thesis considers the MSVPP with cost, persistence and robustness as objectives.

A mathematical formulation for the SVPP is presented, as well as formulations of how to measure persistence and robustness. Exact methods are unable to solve real-size instances of the SVPP and the MSVPP due to the complexity of the problem. A hybrid genetic search heuristic with adaptive diversity control (HGSADC) is therefore developed for the SVPP, based on the work of Vidal et al. (2014). The heuristic is extended to solve the MSVPP, generating Pareto fronts that illustrate the trade-off between cost, persistence and robustness.

The heuristic is tested on problem instances provided by Statoil. The results from solving the instances with cost as the only objective show that the heuristic is able to find the optimal solution to the SVPP for all problem instances where exact methods can prove optimality, and that it seems to find high-quality solutions for real-size problem instances. The heuristic has stable performance for all problem instances, and the running time increases linearly with the size of the problem instance. For the MSVPP, three different sets of objectives were used: cost and persistence, cost and robustness and finally cost, persistence and robustness. The results show that the heuristic finds Pareto fronts that are identical or close to the optimal fronts for problem instances that can be solved by exact methods. The cost of all the solutions found for the real-size problem instances are less than 1% higher than the cost of the solutions found by the single-objective heuristic, indicating that adding additional objectives does not impair the quality of solutions much. The heuristic has stable performance for the MSVPP, and in the tests, the running time increases linearly with the number of objectives.

Pareto fronts generated for real-size problem instances show that the persistence and robustness of the solutions can be improved to optimal or near-optimal values for a cost increase of less than 1% from the lowest known cost. The Pareto fronts also illustrate the trade-off between cost and the other objectives, improving the decision makers insights of how the objectives are related. Decision support tools that implement the solution methods presented in this thesis are therefore expected to improve the supply service by reducing costs and improving reliability.

Sammendrag

Denne oppgaven handler om taktisk planlegging i forbindelse med transport av forsyninger til Statoils offshoreinstallasjoner. Offshoreinstallasjoner trenger forsyninger for å kunne utvinne olje og gass, og forsyningene fraktes fra land med forsyningsfartøy. Supply vessel planning-problemet (SVPP) går ut på å bestemme en flåte av forsyningsfartøy og hvilke turer fartøyene skal seile for å forsyne en mengde med offshoreinstallasjoner. Målet med SVPP er å ha en pålitelig forsyningstjeneste med minimale kostnader relatert til leie og bruk av forsyningsfartøy. Beslutningstakere hos Statoil har bedt om løsninger som ikke bare har minimale kostnader, men også er persistente og robuste. Persistente løsninger har få endringer fra forrige løsning, og robuste løsninger takler forsinkelser og uforutsette hendelser. SVPP med flere objektiver omtales som det multi-objektive supply vessel planning-problemet (MSVPP), og denne oppgaven omhandler MSVPP med kostnad, persistens og robusthet som objektiver.

En matematisk formulering for SVPP presenteres, samt formuleringer av hvordan persistens og robusthet måles. Eksakte metoder klarer ikke å løse probleminstanser av realistisk størrelse grunnet kompleksiteten til problemet. En hybrid-genetisk søkeheuristikk med adaptiv mangfoldskontroll utvikles derfor for å løse SVPP, basert på arbeidet til Vidal et al. (2014). Heuristikken utvides til å løse MSVPP, og genererer Paretofronter som illustrerer avveiningen mellom kostnad, persistens og robusthet.

Heuristikken er testet på probleminstanser fra Statoil. Resultatene viser at heuristikken for SVPP finner optimal løsning for alle probleminstanser der optimalitet kan bevises av eksakte metoder, og at den ser ut til å finne løsninger av høy kvalitet for probleminstanser av realistisk størrelse. Heuristikken er stabil for alle probleminstanser og løsningstiden ser ut til å øke lineært med problemstørrelse.

Den utvidede heuristikken for MSVPP er testet med både to og tre objektiver. Resultatene viser at heuristikken finner Paretofronter som er identiske eller er svært like de optimale frontene for probleminstanser som kan løses av eksakte metoder. Kostnaden til løsningene for MSVPP er under 1% høyere enn kostnaden til løsningene som ble funnet med kun kostnad som objektiv, noe som indikerer at de ekstra objektivene ikke forringer kvaliteten til løsningene i særlig grad. Heuristikken er stabil for probleminstansene som løses, og kjøretiden ser ut til å øke lineært med antall objektiver.

Paretofronter for probleminstanser av realistisk størrelse viser at persistens og robusthet kan økes til optimale eller nær optimale verdier for en kostnadsøkning på under 1% av laveste kjente kostnad. I tillegg kan de bidra til å øke beslutningstakeres innsikt ved å illustrere avveiningen mellom kostnad og andre objektivene. Det er derfor forventet at beslutningsstøtteverktøy basert på løsningsmetodene som presenteres i denne oppgaven kan bidra til å forbedre forsyningstjenesten ved å både redusere kostnader og øke påliteligheten.

Contents

1	Introduction	1
2	Problem Description	3
2.1	Planning requirements	5
2.2	Objectives	5
2.3	Handling multiple objectives	7
3	Literature Review	9
3.1	The supply vessel planning problem	9
3.2	The periodic vehicle routing problem	11
3.3	Persistence	12
3.4	Robustness	13
3.5	Multi-objective optimization	14
3.5.1	Genetic algorithms for MOPs	16
3.5.2	Multi-objective approaches to the SVPP	17
4	Mathematical Model	19
4.1	Previous work	19
4.2	Remarks and assumptions	20
4.3	Mathematical formulation of the SVPP	21
4.4	Measuring persistence	25
4.5	Measuring robustness of schedules	26
5	Hybrid genetic search with adaptive diversity control for the SVPP	29
5.1	Overview	30
5.2	Individual representation	30
5.3	Search space	33
5.4	Evaluation of individuals	33
5.5	Constructing the initial population	35
5.6	Parent selection and crossover	36
5.7	Education	38
5.7.1	Voyage improvement	40
5.7.2	Pattern improvement	40
5.7.3	Repair	41
5.8	Population management	43

5.8.1	Survivor selection	43
5.8.2	Penalty parameter adjustment	43
5.8.3	Diversification	43
5.9	Optimizing the fleet size and mix	44
5.10	Comparison with Vidal et al. (2012a)	45
6	Hybrid genetic search with adaptive diversity control for the MSVPP	51
6.1	Adapting from single-objective to multi-objective	51
6.1.1	Domination criterion and Pareto archive	51
6.1.2	Stopping criterion	52
6.1.3	Modified biased fitness function	53
6.1.4	Fitness evaluation, diversity control and elitism	53
6.2	Education for persistence	54
6.3	Optimizing the fleet size and mix	55
7	Computational Study	59
7.1	Test instances	59
7.2	Calibration of parameters for the HGSADC	60
7.2.1	The parameters of the HGSADC	61
7.2.2	Parameter calibration	61
7.2.3	Calibration of education and repair	70
7.3	Results - SVPP	72
7.3.1	Comparison with the VBM	72
7.3.2	Results for all instances	73
7.3.3	Stability of the HGSADC	77
7.4	Results - MSVPP with cost and persistence	80
7.4.1	Results compared with optimal fronts	80
7.4.2	Results for real-size problem instances	84
7.5	Results - MSVPP with cost and robustness	89
7.5.1	Results compared with optimal fronts	90
7.5.2	Results for real-size problem instance	91
7.6	Results - MSVPP with cost, persistence and robustness	92
7.6.1	Results compared with optimal fronts	92
7.6.2	Results for real-size problem instance	93
7.6.3	Running times	94
8	Economic Implications	97
8.1	The advantages of persistent schedules	97
8.2	The advantages of robust schedules	99
8.3	Combining low cost, high persistence and high robustness	100
8.4	Adding additional objectives	100
8.5	Limitations of the results	101
9	Conclusion	103

A	Voyage-based model	109
A.1	Voyage-based formulation	109
A.1.1	Voyage generation	109
A.1.2	Voyage-based model	110
A.1.3	Measuring persistence	112
A.1.4	Measuring robustness	113
B	Voyage generation using dynamic programming	115
B.1	The voyage generation procedure	115
B.2	Label data	117
B.3	Label extension	117
B.4	Label domination	120
B.5	The labelling algorithm	121
C	Code and test instances	125
C.1	Exact methods	125
C.1.1	Voyage generation	125
C.1.2	VBM	125
C.2	HGSADC	126

List of Figures

2-1	Illustration of a solution to the SVPP	4
2-2	Photo of a PSV and an offshore installation	4
2-3	Illustration of persistent schedules	7
3-1	Illustration of Pareto fronts	15
5-1	Illustration of a schedule	30
5-2	Illustration of an individual and its tour chromosome	33
7-1	Plot of objective value development for 13-44 with $I^{NI} = 5\,000$. . .	66
7-2	Plot of objective value development for 27-80 with $I^{NI} = 5\,000$. . .	67
7-3	Plot of objective value development for 27-80 with $I^{NI} = 10\,000$. . .	67
7-4	Plot of penalty parameters with initial values of 1	69
7-5	Plot of penalty parameters with initial values of 1 000	70
7-6	Plot of sailing cost found by HGSADC vs problem size	74
7-7	Comparison of the running time of the VBM and the HGSADC . . .	78
7-8	Optimal and heuristic fronts for a medium-sized problem instance with variation case 1	81
7-9	Optimal and heuristic fronts for a medium-sized problem instance with variation case 2	82
7-10	Optimal and heuristic fronts for a medium-sized problem instance with variation case 3	82
7-11	Optimal and heuristic fronts for a medium-sized problem instance with variation case 4	83
7-12	Heuristic fronts for a real-sized problem instance with variation case 1	85
7-13	Heuristic fronts for a real-sized problem instance with variation case 2	86
7-14	Heuristic fronts for a real-sized problem instance with variation case 3	87
7-15	Heuristic fronts for a real-sized problem instance with variation case 4	88
7-16	Optimal and heuristic fronts for a medium-sized problem instance with robustness	90
7-17	Optimal and heuristic fronts for a medium-sized problem instance with robustness	91
7-18	Heuristic fronts for a real-sized problem instance with robustness . . .	92
7-19	Optimal and heuristic fronts for a medium-sized problem instance with robustness and variation case 1	93

7-20	Heuristic fronts for a real-sized problem instance with robustness and variation case 3	95
------	---	----

List of Tables

5.1	Installation chromosome	32
5.2	PSV chromosome	32
7.1	Parameter values	62
7.2	Calibration of ξ^{REF}	63
7.3	Calibration of η^{ELI}	64
7.4	Calibration of μ and λ	65
7.5	Calibration of I^{NI} and η^{DIV}	68
7.6	Evaluation of education procedures for single-objective HGSADC . .	71
7.7	Calibration of ρ^{REP}	72
7.8	Comparison of results of the HGSADC and the VBM for the SVPP .	75
7.9	Fleet size and objective value found by the HGSADC for all problem instances of the SVPP	76
7.10	Running time and coefficients of variation for the HGSADC for all problem instances of the SVPP	79
7.11	Running times for medium-sized instances of the MSVPP with cost and persistence	84
7.12	Running times for real-size instances of the MSVPP with cost and persistence	89
7.13	Comparison of running times for different objectives	95

List of Algorithms

1	HGSADC	31
2	Construction heuristic	37
3	Crossover operator	39
4	Reducing the number of voyages	42
5	Penalty parameter adjustment	43
6	Optimizing the fleet size and mix	46
7	Installation pattern improvement for persistence	55
8	Moving PSV departures to improve persistence	56
9	Multi-objective HGSADC with variable fleet	58
10	Pseudocode for voyage generation	116
11	Pseudocode for voyage generation for one PSV	122
12	Pseudocode for extending a label	123
13	Pseudocode for adding a label to stage	123

Chapter 1

Introduction

The petroleum industry is Norway's largest industry in terms of investments and value creation, and the petroleum production on the Norwegian continental shelf (NCS) has created values of more than NOK 12 000 billion in present value since the start in 1971, according to the Norwegian government (2015). The petroleum industry is also Norway's largest industry in terms of government revenues, and the Norwegian Petroleum Directorate (2016) reports that the net government cash flow from petroleum activities in 2015 was about NOK 218 billion, or about 20% of total government revenues.

Three areas of the NCS are open to petroleum production: the North Sea, the Norwegian Sea and the Barents Sea. Most of the fields in production are located in the North Sea, but there are also fields in production further north, in the Norwegian Sea. Despite more than 40 years of production, the Norwegian government (2015) estimates that only 42% of the total expected resources on the NCS have been extracted.

The Norwegian State Oil Company, Statoil, was formed in 1972 and has been one of the most important players in the Norwegian petroleum industry. Statoil is the leading operator on the NCS, having an annual total revenue of more than NOK 622 billion, operating 45 of the fields in production and being responsible for approximately 70% of the total production in 2014 (Statoil, 2015).

Statoil operates from offshore installations at the fields. Offshore installations are normally self-sufficient with regards to water and energy, but require supplies like drilling mud and pulverized cement in order to operate. The supplies are stored at an onshore supply depot and transported from the supply depot to the installations by platform supply vessels (PSVs), vessels specially designed to supply offshore installations.

The costs in the petroleum production in the North Sea has risen faster than any other industrial sector in the region, according to McKinsey & Company (2014). The increased costs, combined with the drop in oil prices in late 2014, have made operator companies more aware of the need for cost reductions. Costs related to the chartering

and operating of PSVs are a major part of the costs in the upstream supply chain of operator companies. More efficient use of the PSVs can reduce these costs.

The use of PSVs is planned at strategic, tactical and operational levels. This thesis addresses the planning problem at a tactical level, more specifically, deciding how many and which PSVs to charter, which voyages the PSVs should sail and when they should sail them. Statoil is currently using optimization-based decision support tools to aid this planning. The current decision support tools generate fleet compositions and weekly schedules with minimal PSV chartering and sailing costs as the only objective.

The planners at Statoil have experienced that the total cost related to the supply service is affected by more than the chartering and sailing costs. One example is that the tough weather conditions in the North Sea often cause delays to the PSVs, and the planners might have to send out extra PSVs, or even helicopters, to avoid shutdown of the installations. Sending out an extra PSV is costly, and the extra cost could probably be avoided if a more *robust* schedule was selected, i.e. a schedule that handles delays better. The planners therefore prefer schedules that are both robust and have minimal chartering and sailing costs. The planners have also experienced that changes in the schedule propagate to other parts of the supply chain, in some cases causing disruptions and extra costs. Hence, they request schedules that require few changes from the previous schedule, referred to as *persistent* schedules. Including these two additional objectives is expected to reduce the total costs related to the supply service. The objectives are often conflicting, meaning that improving one objective worsens another. Hence, the choice of schedule is a trade-off between the different objectives that depends on the decision makers' preference for each objective. In this thesis, it is argued that the best approach for solving problems with multiple conflicting objectives is to present decision makers with a set of schedules, and let them use their experience to select the best schedule given the current circumstances. This allows decision makers to account for factors that are not included in the model. The set of schedules also gives decision makers better insight into the relationship between the different objectives, which improves the decision making process.

A detailed description of the problem is given in Chapter 2. Literature relevant to the problem is reviewed and discussed in Chapter 3, and in Chapter 4 a mathematical formulation of the problem is presented. Chapter 5 presents a heuristic for solving the single-objective problem, based on the Unified Hybrid Genetic Search framework of Vidal et al. (2014). In Chapter 6, the heuristic is extended to handle multiple objectives. Chapter 7 shows how the parameters for the heuristic are calibrated and the results from solving test instances provided by Statoil. The test instances are solved with different combinations of three objectives: minimizing cost, maximizing persistence and maximizing robustness. The economic implications of the results and the usefulness to the decision makers at Statoil are discussed in Chapter 8, before Chapter 9 concludes the thesis and presents suggestions for further research.

Chapter 2

Problem Description

This master's thesis addresses the multi-objective supply vessel planning problem (MSVPP), an extension of the supply vessel planning problem (SVPP). The SVPP is a problem faced by Statoil, the largest operator on the Norwegian continental shelf. Halvorsen-Weare et al. (2012) present a model and method for solving the SVPP. Shyshou et al. (2012) present a Large Neighbourhood Search heuristic for solving the same problem, and Borthen and Loennechen (2015) present a model for the SVPP that considers persistence as an additional objective. All these studies are based on the model introduced by Halvorsen-Weare et al. (2012).

The following paragraphs are based on these three studies, and some parts are taken directly from the original texts. The definition of the SVPP used in the studies is:

Identifying the optimal fleet composition of supply vessels that are to service a given number of offshore installations from one common onshore depot while at the same time determining the weekly routes and schedules for these vessels. A route in this setting is a combination of one or more voyages, starting and ending at the supply depot, which a vessel sails during a week. During a voyage, the vessels may visit one or more offshore installations.

A solution to the SVPP determines (1) the size and mix of the fleet of PSVs, (2) the voyages sailed by each PSV and (3) the departure time(s) from the supply depot for each PSV. The objective of the SVPP is to minimize the costs related to the usage and operation of the PSVs while providing a reliable supply service. Two costs are considered in this problem: the time charter cost of PSVs and the sailing cost of the voyages sailed by the PSVs.

Figure 2-1 shows an example of a solution to a problem instance with four installations and two PSVs. In this solution, each PSV sails two voyages. PSV 1 departs on day 1 to service installations 1 and 2 and departs on day 3 to service installations 4, 3 and 2, in that order. PSV 2 departs on day 2 to service installations 3 and 4 and on day 4 to service installations 1 and 2.

The MSVPP is defined as any variant of the SVPP with multiple objectives. The

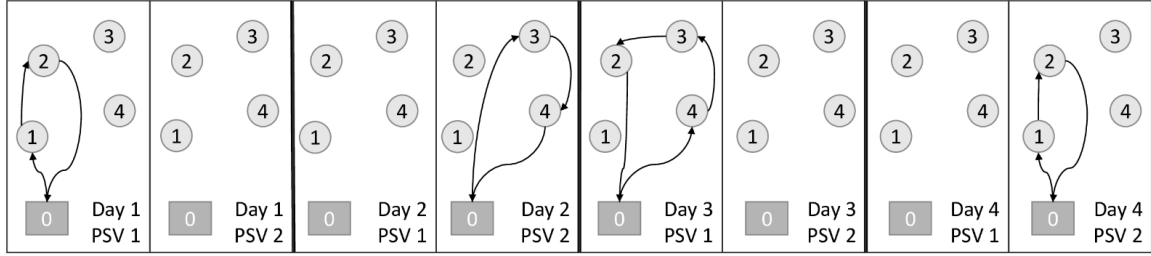


Figure 2-1: Example of a solution to the SVPP for an instance with two PSVs and four installations.

MSVPP has exactly the same decision variables and constraints as the SVPP, and differs only in the objectives. This thesis studies three objectives: (1) minimizing the sum of time charter cost and sailing cost, (2) maximizing the persistence of the schedule and (3) maximizing the robustness of the schedule. The first of these, minimizing the sum of the PSV charter and sailing costs, is the same objective as in the SVPP. The two other objectives are based on the experience and preferences of the decision makers at Statoil. The constraints, referred to as *planning requirements*, are described in Section 2.1, and the objectives and the rationale behind them are described in Section 2.2. Section 2.3 describes how to handle multiple objectives.



Figure 2-2: PSV and offshore installation. Photo: Business Day News (2011)

2.1 Planning requirements

The planning requirements are written in italics in the following. The schedules are weekly plans that are repeated over several weeks, i.e. each PSV sails the same voyage(s) each week. *PSVs have limited load capacity* varying with each PSV type. Each installation needs to be *serviced a given number of times each week*, and also has a *weekly demand for supplies* that must be satisfied. The demand for supplies of a single visit is estimated as the weekly demand divided by the number of weekly visits. The *departures from the supply depot are spread evenly* over the week in order to ensure a steady supply. The departures to offshore installations are spread rather than the visits to these installations, since requests from offshore installations may be unpredictable and should be met without too much delay. As a result, it is important to ensure that the time of the next departure to an offshore installation is never too far in the future. For example, suppose an installation requires three visits per week and the PSVs visiting the installation are scheduled to leave the base on three consecutive days. If the installation makes a request just after the third PSV has left, then almost five days will elapse until the next departure, which can be too long. In such a case, it may be necessary to reroute another PSV to the installation or send out a helicopter to fulfil its request. This will in most cases be very costly, and can to a large extent be avoided if the departures are evenly spread. The *service time* needed to unload supplies at the installations and the *sailing time* may vary with each PSV type. A PSV needs a given number of *hours at the supply depot* in order to prepare for a new voyage. Both the supply depot and some of the offshore installations are subject to *opening hours*, meaning a PSV may have to stay idle for a while without being serviced, either if it arrives outside the opening hours or if the service is not finished by the closing time. Voyages have a *minimum and maximum duration*, as well as a *maximum number of installations visited*, in order to avoid both short voyages - which may lead to unexploited PSV capacity - and long voyages - which makes sailing times uncertain. The *capacity at the supply depot* is limited, meaning that only a limited number of PSVs can be prepared for a new voyage each day.

2.2 Objectives

This section describes the three objectives studied: minimizing the cost, maximizing the persistence of the schedule and maximizing the robustness of the schedule.

Cost

Two costs are considered in the SVPP, the time charter cost of PSVs and the sailing cost of the voyages sailed by the PSVs. The time charter cost of a PSV is the cost of chartering the PSV in the spot market for the duration of the planning period. The sailing cost of a voyage is the sum of the variable costs that occurs when the voyage is sailed. Of these costs, the fuel consumption cost of sailing the voyage is the largest. The main objective of the SVPP is to minimize the sum of the time charter cost of PSVs and the sailing cost of the voyages sailed by the PSVs.

Persistence

The weekly schedule of the PSVs (the solution to the SVPP) is usually used by Statoil until there are large changes in demand. The typical lifetime of a schedule is a few months. When a new schedule is needed, the input is updated and the model is solved to optimality. The new schedule has the lowest cost possible, but can be very different from the existing schedule. This leads to difficulties, as the offshore installations are organized and adapted to the existing schedule. Changes in the schedule might disrupt the supply chain in unpredictable ways. Statoil has therefore requested a model that emphasizes persistent solutions, i.e. they prefer new schedules that do not differ from the existing schedule when there is a need to change the input and re-optimize the problem. The word *persistent* is used to indicate that a new schedule contains few *changes* from the existing schedule. In this thesis, it is distinguished between *persistence* and *change*, the former being a general concept and the latter a well-defined value which can be measured. The most important factor for Statoil to keep unchanged is when the installations need to place their supply orders. This is determined by when the voyages to an installation depart from the supply depot. Thus, only the days of departure to each installation are considered when measuring the number of changes. Persistent schedules make the operation of the installations more predictable for both managers and workers, who can focus on working as efficiently as possible instead of adapting to changes. For an example of how persistence influences the choice of schedule, see Figure 2-3, which shows three schedules for a problem with four installations and one PSV. Each box represents a voyage and the numbers represent the installations that are part of the voyage and the order of visits. The top line is the old schedule, i.e. the one used until now, while the bottom two are two new cost-optimal schedules. Note that the new schedules consist of the same voyages, thus they have the same schedule cost. The first new schedule contains no changes in the ordering routines for the installations, while in the second schedule, installations 1 and 4 need to place their orders on different days than in the existing schedule. Therefore, schedule 1 is preferred, as it is the most persistent one.

Robustness

Disruptions to the planned schedule can result in installations not receiving the demanded supplies, and without the required supplies, an installation can be forced to do a temporary shut-down of production. A temporary shut-down can result in millions of USDs in lost income, according to Halvorsen-Weare and Fagerholt (2011). To avoid costly shut-downs, operators can transport critical supplies by changing the voyage of a nearby PSV, send out an available PSV from the supply depot or transport the supplies by helicopter, but all of these options result in additional effort for the planners and increased costs. Disruptions to the planned schedule are mainly caused by the adverse weather conditions in the North Sea and the Norwegian Sea, especially during the winter. According to Statoil, PSVs have to reduce the sailing speed when the wave height exceeds 3.5 meters, and the service time at installations increases when the wave height exceeds 2.5 meters. Installations cannot service PSVs

Day	Mon	Tues	Wed	Thurs	Fri	Sat	Sun
Old schedule		2 1 3			4 2		
New schedule 1		3 2 1			4 2		
New schedule 2		4 2			3 2 1		

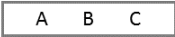
 Boxes represent voyages, numbers indicate which installations are visited in the voyage

Figure 2-3: Example where two new schedules are generated. The two schedules have equal cost, since they consist of the same voyages, but the first solution is preferred, because it does not involve changes in the departures to the installations. In the second alternative the PSVs visiting installations 1 and 4 depart on different days than in the existing schedule, which is unwanted.

if the wave height exceeds 4.5 meters, and the PSVs will have to wait for service until the weather conditions improve. Disruptions are also caused by the fact that the demand at the installations may vary. Unexpected orders or increased volume of the existing orders can cause disruptions the same way bad weather does. For example, an increase in the volume delivered to an installation will increase the service time at that installation, which can lead to delays in the schedule.

Due to these uncertain factors, the planners at Statoil have experienced that schedules that do not consider robustness are difficult to execute in real life. They therefore request that the schedules generated by solving the SVPP are more robust. In this thesis, robustness is defined as by Halvorsen-Weare and Fagerholt (2011):

By robustness we mean the capability for a voyage or schedule to allow for unforeseen events during execution. Robust solutions will reduce the actual costs of the supply vessel service by avoiding expensive and unplanned means of bringing critical demand to the offshore installations.

2.3 Handling multiple objectives

Three objectives for the MSVPP were presented in the previous section: minimizing cost, maximizing persistence and maximizing robustness. In practice, the balancing of these represents a trade-off, meaning that one cannot optimize them all at the same time. A robust schedule is likely to be more expensive than an unrobust one, since robust schedules need more slack, which lead to less efficient utilization of the PSVs. Similarly, a persistent schedule is likely to be more expensive and less robust than an impersistent one, since the demand for persistence restricts the problem. Identifying the optimal solution requires some weighting or prioritization of each objective. In

this case, the expected cost of using an impersistent and unrobust schedule is difficult to quantify, thus it is difficult to state a general weighting of, or preference for, each objective that can be used for all instances of the MSVPP. The preference for each objective is likely to vary from time to time, depending on both the experience of the decision maker and factors which are not captured in the model of the MSVPP. Based on this, it will be argued that the most appropriate approach is to generate a set of so-called Pareto-optimal solutions from which the decision maker can choose. The concept of Pareto-optimality will be elucidated later, but in short it means that a solution is such that one objective cannot be improved without impairing another objective.

As mentioned, the objective of the SVPP is to minimize the costs related to the usage and operation of the PSVs while providing a reliable supply service. It is worth noting that all the mentioned objectives seek to minimize the realised costs related to the supply service, i.e. the costs that will actually occur. Maximizing persistence and robustness is done to avoid unplanned costs and inefficiency, so even though the objectives seem conflicting, they all seek to minimize the realised costs, either directly or indirectly.

Chapter 3

Literature Review

This chapter reviews the existing literature related to the MSVPP. Section 3.1 summarizes previous studies of the SVPP and some closely related maritime routing problems. Section 3.2 gives a brief introduction to the periodic vehicle routing problem (PVRP) and the current state-of-the-art algorithms for solving different variations of it, while Section 3.3 describes some relevant studies of persistence in optimization models. Section 3.4 reviews previous studies on robustness, both generally for vehicle routing problems and specifically for the SVPP. Finally, a short introduction to multi-objective optimization and the most popular solution methods is given in Section 3.5, including the most important properties of genetic algorithms for multi-objective optimization and a summary of earlier work on multi-objective variants of the SVPP. Borthen and Loennechen (2015) present a literature review of many of the same topics, and parts of that literature review is used here.

3.1 The supply vessel planning problem

Fagerholt and Lindstad (2000) were among the first to study the supply service in the Norwegian Sea, solving a relaxed version of the SVPP. The relaxed version excludes the constraints on supply depot capacity, voyages not starting before they have returned from the previous voyage and evenly spread departures. Halvorsen-Weare et al. (2012) present a mathematical model and solution method for the SVPP. The model has been implemented in a decision support system that is used by Statoil to plan the supply service. The problem is solved using a voyage-based formulation and is solved in two steps: (1) Generating feasible *candidate voyages* for each PSV and (2) Using the voyages to solve a voyage-based model (VBM) to find the optimal fleet composition and weekly routes and schedule. A candidate voyage is the voyage with shortest duration that visits a given set of installations, starting and ending at the depot. Step 1 consists of generating the candidate voyage for all subsets of installations that satisfy the constraints on PSV capacity and number of installations per voyage. This corresponds to solving a travelling salesman problem (TSP) with multiple time windows for each subset. Halvorsen-Weare et al. (2012) also present some extensions

of the model, including departures on specific days, robust schedules and collision avoidance. Shyshou et al. (2012) build on the model presented by Halvorsen-Weare et al. (2012), but solve the problem heuristically with a large neighbourhood search (LNS) instead of using the voyage-based model. The LNS heuristic is not guaranteed to find an optimal solution, but outperforms the VBM on large problem instances; it is able to find feasible solutions to problem instances with up to 31 installations.

Halvorsen-Weare and Fagerholt (2011) combine the VBM presented by Halvorsen-Weare et al. (2012) with weather simulations to study the impact of weather more in depth. They aim to create more robust schedules for the SVPP, that is, schedules that can withstand changes in the environment (e.g. changing weather conditions). Halvorsen-Weare and Fagerholt (2016) present an arc-flow formulation for the SVPP, and compare it with the work of Halvorsen-Weare and Fagerholt (2011) and Halvorsen-Weare et al. (2012). Test results show that the arc-flow model is outperformed by the voyage-based model, but the authors argue that the arc-flow formulation has some value, since it provides a more precise description of the problem. Norlund and Gribkovskaia (2013) study the reduction of CO₂-emissions by PSVs through speed optimization, where speed optimization consists of finding voyages with waiting time and reducing the speed of the PSVs of these in order to reduce fuel consumption. The study is based on the VBM presented by Halvorsen-Weare et al. (2012), and adds speed optimization in the voyage generation phase. Test results show an average reduction of fuel consumption by 10%, without increasing the fleet size. Norlund et al. (2015) build on the work by Norlund and Gribkovskaia (2013), considering the robustness of schedules and the reduction of emissions. They identify the trade-off between low emissions and robust schedules, and solve the problem multiple times with different robustness requirements, in order to present the decision makers with a set of schedules for different trade-offs between emissions and robustness.

The SVPP is a type of Fleet Size and Mix Vehicle Routing Problem (FSMVRP), where one decides which vehicles to use and the routing of the vehicles simultaneously. These types of problems have been studied by Christiansen et al. (2006), who discuss various models of FSMVRPs arising in the shipping industry. Studies have also been made on the problem of routing a single PSV, modelled as a Single Vehicle Pickup and Delivery Problem with Capacitated Customers (SVPDPCC). In this model, the storage capacity on the installations is included as a constraint, as well as the amount of used supplies that are to be delivered back to the depot from the installations. The objective of the problem is to design a least cost vehicle route visiting each customer, starting at the depot. There must be sufficient capacity both in the vehicle and at each customer location to perform the pickup and delivery operations. The limited capacity of the customers make the problem different from a standard multi-trip Vehicle Routing Problem (VRP). Jahre et al. (2007) present the SVPDPCC and a mathematical formulation of it, and solve small instances of the problem. Gribkovskaia et al. (2008) build on this work and present multiple construction heuristics and a tabu search algorithm for creating voyages for a single PSV.

Aas et al. (2009) focus on the PSVs and their role in the offshore logistics. They argue that a thorough understanding of one’s logistics is necessary to determine the best sourcing strategy of PSVs. They further investigate the properties of a PSV, and point out that a heterogeneous fleet often performs better than a homogeneous one.

The SVPP is a special case of the periodic vehicle routing problem (PVRP), which will be discussed in the next section.

3.2 The periodic vehicle routing problem

The PVRP is a type of VRP where the planning period lasts more than one unit of time. Each vehicle can service one route per unit of time, which is normally one hour, day, week or month. A description of the PVRP can be found in Francis et al. (2008). The problem consists of constructing and assigning vehicle routes to a set of vehicles which are to service a set of customers. Each customer has a required number of visits and a total demand over the planning period which needs to be fulfilled. The objective is to minimize the cost while still satisfying the demand and the required number of visits. For each unit of time one has to decide the route each vehicle uses. There are usually constraints making sure the delivery to a customer is spread evenly over the planning period. The SVPP is a special case of the PVRP. In both the PVRP and the SVPP, the planning period spans several time units, and each customer (installation) requires a given number of visits. The difference is that in the SVPP, a route (voyage) can span multiple days, meaning constraints that ensure that a vehicle finishes a route before starting a new one are needed. The possibility of routes spanning multiple days is studied by Savelsbergh and Song (2008), who study the Inventory Routing Problem with Continuous Moves (IRP-CM). The goal of the IRP-CM is to minimize transportation costs while avoiding stockouts of the customers’ inventory, and incorporates routes spanning several days. It differs from the SVPP because it has multiple pick-up points, while the SVPP only considers one supply depot. The authors use a combination of a randomized greedy heuristic and local search to solve the IRP-CM.

Vidal et al. (2012a) present a hybrid algorithm for solving a large class of VRPs, including the PVRP. They refer to the algorithm as a *Hybrid Genetic Search Algorithm with Advanced Diversity Control (HGSADC)*, where the term *hybrid genetic algorithm* refers to a special class of *genetic algorithms*. Genetic algorithms are described in Section 3.5.1. Hybrid genetic algorithms differ from normal genetic algorithms in that they utilize problem-specific knowledge by ”incorporating heuristics, approximation algorithms, local search techniques, specialized recombination operators, truncated exact methods, etc.” (Moscato and Cotta, 2003). Hybrid algorithms are also referred to as *memetic algorithms*. Vidal et al. (2014) generalize the HGSADC to a Unified Hybrid Genetic Search (UHGS) metaheuristic for solving numerous variants of multi-attribute vehicle routing problems. The framework matches or outperforms the current state-of-the-art algorithms for many benchmark instances, including the

benchmark instances for PVRP.

The SVPP differs from a standard PVRP due to two factors: The vehicle fleet is heterogeneous and the routes can span multiple time periods. The fact that routes can span multiple time periods makes them interdependent, since one needs to make sure that a vessel has returned from its previous voyage before embarking on a new one. Vidal et al. (2014) study a large class of multi-attribute VRPs, but interdependent services are mentioned as a possible extension of the framework. The general class of VRPs with interdependencies between routes is referred to as the Vehicle Routing Problem with Multiple Synchronization Constraints (VRPMS). Drexler (2014) presents an overview of the most studied variants of VRPMSs, and provides a general heuristic for solving these problems.

3.3 Persistence

The weekly schedule generated by solving the SVPP is used by Statoil every week as long as there are no major changes in the structure of the problem, and is re-optimized whenever a major change occurs. When new schedules are generated, Statoil prefers the new schedules to require small changes in routines from the ones in place, even if it means that the schedule is suboptimal with regards to cost. In other words, they are willing to use a schedule with a higher cost if it involves less change. This preference is common among decision makers, and has been studied by researchers in the context of other problems. As noted earlier, it is emphasized that the word *persistent* is used to indicate that a new plan contains few *changes* from a previous plan. It is distinguished between *persistence* and *change*, the former being a general concept and the latter a well-defined value which can be measured. Deciding the most appropriate method for measuring change is problem-specific and depends on factors such as the preferences of decision makers and the complexity and structure of the model. The following section presents some studies which consider persistence in optimization models.

Brown et al. (1997b) argue that lack of persistence is a major source of complaints when optimization models are used in real life. Persistence here refers to when new plans "retain the features of prior published plans". Persistent plans increase trust in and usage of decision support systems. The study provides multiple case examples where persistence have successfully been included in the optimization model. One of these is presented by Brown et al. (1996), who find solutions to a ship scheduling problem that are both low-cost and similar to a previous solution. The new solution is found by changing the objective function to minimize changes from the previous solution and setting an upper limit for the cost. Another approach for achieving persistence is used by Brown et al. (1997a), who use Lagrangian relaxation to penalize changes from the baseline solution over a specified threshold. Persistence in ship routing is studied by Fagerholt et al. (2009), who handle the problem by finding a set of high-quality solutions that are similar to the initial solution in the beginning of the planning period, and diverse at the end. This is then presented to the decision

makers, who can use their experience to select their preferred solution. This approach reduces the need to quantify the costs related to lack of persistence.

3.4 Robustness

Unforeseen events like adverse weather conditions and changes in demand at installations can delay voyages and cause costly disruptions to the planned schedule. The *robustness* of a voyage or schedule is the capability to allow for unforeseen events during execution. The planners at Statoil have experienced that schedules that do not consider robustness are difficult to execute in real life, and have therefore requested schedules that are more robust. This section presents studies that consider robustness, both in routing problems in general and in the SVPP.

Gendreau et al. (1996) provide a comprehensive summary of the literature on stochastic VRPs, including VRPs with stochastic demand, customers and travel times. Potvin et al. (2006) solve a vehicle routing and scheduling problem with stochastic travel times and stochastic customer demand. Agra et al. (2013) solve the robust VRP with time windows, inspired by a maritime transportation problem where delays due to unforeseen events are common. The problem "falls into the framework of robust programming, where a solution is said to be feasible only if it is feasible for all realizations of the data in a predetermined uncertainty set". They utilize various robust optimization tools to handle the uncertainty. Agra et al. (2015) also study a maritime transportation problem with uncertain sailing and waiting times. They present a two-stage stochastic programming model with recourse, and use a decomposition approach similar to the L-shaped algorithm to solve it. Fischer et al. (2016) present multiple slack-based strategies to increase the robustness of voyages sailed in a fleet deployment problem for roll-on roll-off liner ships.

Robustness in the SVPP

Halvorsen-Weare et al. (2012) suggest a simple method for extending the SVPP to create robust and cost-efficient schedules. They define the *slack* of a voyage as "hours available after finishing a voyage before starting to prepare for the next voyage at the supply depot". To increase the robustness of the generated schedule, a minimum requirement of slack for each voyage sailed is set. Halvorsen-Weare and Fagerholt (2011) extend this model further and add a robustness profit that rewards solutions with equal distribution of slack in the objective function. They also present a solution method based on simulation and optimization. They add an extra step to the solution method that assigns a robustness value to each voyage, based on how the voyage performs when different weather conditions are simulated. The robustness value is used to estimate the average demand not delivered by each voyage, and a penalty for missed demand is added to the objective function. The additional cost incurred by fulfilling the missed demand is referred to as the *extra cost*. The total cost of a solution is calculated as the sum of the schedule cost and the extra cost. The results show that solutions that consider robustness has a slightly higher schedule cost, but

a significantly lower extra cost, resulting in a reduced total cost. It is worth to note that even though the simulation approach yields the biggest reduction of total cost ($\sim 3\%$), the simpler, slack-based approaches also perform well ($\sim 2\%$).

Norlund and Gribkovskaia (2013) use speed optimization to minimize the emissions of CO₂ from the PSVs used by reducing the sailing speed of PSVs. Robustness is not considered in the solution method, but they expect robust schedules as a side-effect of reducing the speed, since the PSVs can adapt to unforeseen events by increasing the speed. Norlund et al. (2015) solve the SVPP with cost, environment and robustness considerations. A new step is added to the two-step approach presented by Halvorsen-Weare et al. (2012), in order to make the schedules more robust. Simulations of the weather conditions in the North Sea are used to obtain a probability distribution of the voyage duration of each candidate voyage. If the probability that a candidate voyage is feasible within its assigned duration is less than the robustness parameter p , the candidate voyage is removed from the set of candidate voyages.

3.5 Multi-objective optimization

The field of multi-objective optimization provides methods for solving problems with multiple objectives, also known as MOPs (Multi-Objective Problems). The following section provides some background on multi-objective optimization and some examples of how MOPs can be solved.

Deb (2014) gives an introduction to multi-objective optimization and the most commonly used solution methods. In multi-objective problems, the optimal solutions make what is known as a Pareto front in the objective space, a set of solutions all having the property that one cannot improve one objective without impairing other objectives. Figure 3-1 shows an illustration of Pareto fronts for different objectives. A solution on the Pareto front is referred to as *Pareto-optimal* or *non-dominated*. Thus, the Pareto front leads to a trade-off situation where the decision makers have to use their preference for each objective to select which solution to use. Feasible solutions that are not on the Pareto front are referred to as *dominated* solutions, since there exists at least one solution on the Pareto front that is better (dominates) with respect to one or more of the objectives and at least as good with respect to the rest of the objectives. The advantage of presenting a Pareto front is that decision makers can easily see how much an improvement in one objective affects the remaining objectives. The nature of the Pareto front implies that any solution on the Pareto front may be considered optimal, depending on your preferences of the objectives. These preferences may vary from decision maker to decision maker or even from day to day for the same decision maker. The main purpose of creating a Pareto front is to illustrate the trade-offs between different solutions, and let the decision maker use his or her judgment to select the preferred solution. Often it is computationally infeasible to find the entire Pareto front, and a best-known Pareto front is found instead. Konak et al. (2006) list three goals that a good best-known Pareto front should achieve: (1) be as close to the true Pareto front as possible, (2) solutions should be uniformly

spread over the Pareto front and (3) capture the whole spectrum of the true Pareto front.

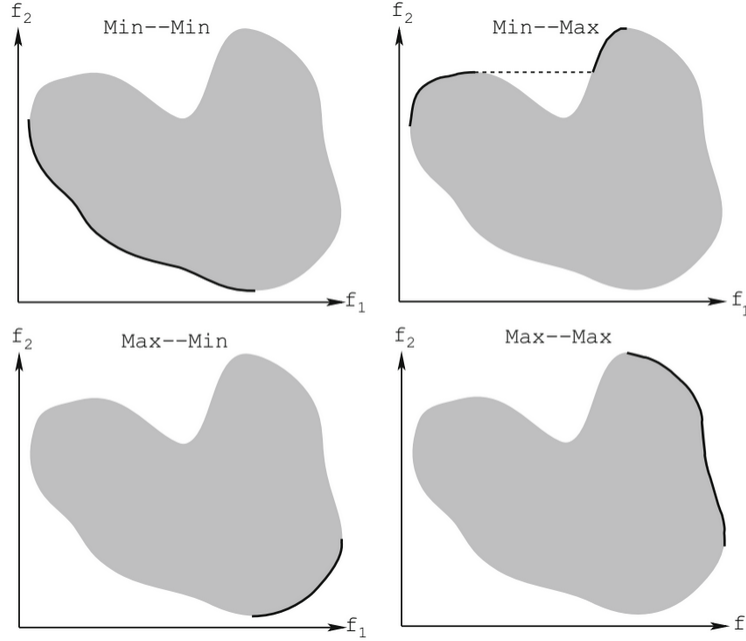


Figure 3-1: Illustration of Pareto fronts for different bi-objective problems (Taken from Deb (2014)).

In general there are two main approaches to solving MOPs, and Deb (2014) refers to them as the *preference-based approach* and the *ideal approach*. In the preference-based approach, information about the preference for each objective is used to give a weight to each objective. This vector of weights is called the preference vector. Using the weights, one can combine the objectives into a single objective function and solve the problem using only this single objective function. The result is the optimal solution for the given preference vector, which is a Pareto-optimal solution. The challenge of this method is to determine the preference vector, since this is often difficult to quantify in real life. In addition, the method only finds one of the solutions on the Pareto front. In the real world, decision makers often want to have alternatives to compare and choose from. In the ideal approach, on the other hand, one starts by finding the Pareto front, and then selects the preferred solution from the Pareto front. Deb (2014) argues that this approach is more methodical, more practical and less subjective than the preference-based approach. The three main methods for finding the Pareto front are:

- The weighted-sum method
- The ϵ -constraint method
- Genetic algorithms

In the weighted-sum method the preference-based approach is repeated n times, with a different preference vector each time, resulting in n solutions, all lying on the Pareto

front. The main issue with this method is that it cannot find solutions lying on the non-convex regions of the Pareto front. The ϵ -constraint method can be used to avoid this issue. The idea of the ϵ -constraint method is to optimize only one of the objectives while adding the remaining objectives as constraints, each limited by some value. The vector of these values is referred to as ϵ . One objective is optimized while the constraints make sure the other objectives are kept within reasonable limits. One solution on the Pareto front is found for each value of ϵ . Since the problem needs to be solved once for each ϵ , a challenge of this method is finding ϵ -values that give enough granularity without taking too much time to solve. In both the weighted-sum and the ϵ -constraint method the single-objective problem has to be solved once for every solution on the Pareto front.

3.5.1 Genetic algorithms for MOPs

In a study of metaheuristics used for solving multi-objective problems performed by Jones et al. (2002), 70 % of the studied articles used genetic algorithms as the primary metaheuristic. According to Konak et al. (2006), the main reason for the popularity of genetic algorithms in multi-objective optimization is that they are population-based and can thus search multiple regions of the search space simultaneously. In addition, they require neither convex, continuous nor unimodal (having only one maximum) solution spaces.

Konak et al. (2006) give a succinct introduction to multi-objective optimization using genetic algorithms. Genetic algorithms are inspired by the theory of evolution from the field of biology. Genetic algorithms start out with an initial population of solutions, referred to as *individuals* or *chromosomes*. Each chromosome is evaluated according to a fitness function. In order to generate a new generation of individuals, the two genetic operators *mutation* and *crossover* are used. Crossover combines two individuals, known as *parents*, into a new solution. Chromosomes with higher fitness are more likely to be selected as parents, making each generation more fit than the previous one. Mutation makes small random changes to the individuals, thus exploring new parts of the solution space.

Several different genetic algorithms have been used for multi-objective optimization, Konak et al. (2006) list 13 of them. They are all based on the general genetic algorithm framework described above, but differ in terms of the fitness function used for evaluating individuals, how population diversity is preserved and how *elitism* is used. The main approaches to the fitness function is either to use a weighted objective function, to alter the objective function during the search or to use Pareto ranking, i.e. to sort the population into several fronts based on dominance. Preserving a diverse population of individuals is important for any genetic algorithm, in order to explore as much of the search space as possible. This is especially true when solving multi-objective problems, since diversity is necessary not only during the search, but also in the final solutions. If no mechanism for preserving diversity is included, the individuals tend to form clusters close to each other, referred to as *genetic drift* by Konak et al. (2006). Genetic drift is undesirable, since one of the goals of the Pareto

front is to be as uniformly distributed over the front as possible. Elitism means that the best individuals, i.e. the non-dominated individuals always survive to the next generation. This can for example be done by having an external population which always contains the non-dominated individuals.

3.5.2 Multi-objective approaches to the SVPP

Some multi-objective extensions of the SVPP have been studied, as described in Section 3.1. All of the mentioned studies have cost as the primary objective, but enhance the other objectives, e.g. robustness, by adding steps to the two-step solution method presented by Halvorsen-Weare et al. (2012). Halvorsen-Weare and Fagerholt (2011) use a weighted-sum approach where a penalty is given to unsatisfied demand, where the expected unsatisfied demand is found through simulation. The method provides a single optimal solution, where the weighting of robustness is determined by the penalty given to unsatisfied demand.

The most extensive study of multiple objectives is the model presented by Norlund et al. (2015), which aims to minimize cost, maximize robustness and minimize CO₂-emissions. They solve the problem in three steps, one for each objective. First they use the method of Norlund and Gribkovskaia (2013) to generate the optimal voyages, with the PSV speed set such that the CO₂-emissions are minimized. Then simulation is performed to find the probability distribution of the duration of each voyage. The voyages are assigned a duration such that the probability that they actually are finished within the assigned duration is larger than a given value p . Finally, the voyage-based model is solved in order to find the minimum cost. An important decision in this model is to set the value of p . The authors plot the cost of the solution against this p -value, illustrating the trade-off between cost and robustness, since a higher p -value implies more robust schedules. The method of solving the model for different values of p is equivalent to the ϵ -constraint method, and the resulting plots are Pareto-fronts, clearly showing how the two conflicting objectives of robustness and cost are related.

Borthen and Loennechen (2015) solve the SVPP with persistence as an additional objective, using the ϵ -constraint method to generate the Pareto fronts for cost and the number of changes from a given baseline solution. The method is able to generate the exact Pareto front for problems with up to 11 installations. Heuristics are presented to reduce the running time, but the method is still unable to solve large problems, i.e. problems with more than 15 installations.

Chapter 4

Mathematical Model

This chapter presents a mathematical formulation of the MSVPP. Section 4.1 briefly summarizes previous formulations of the SVPP and compares them to the formulation presented in this chapter. Remarks and assumptions are described in 4.2, and Section 4.3 presents a mathematical formulation of the SVPP. Sections 4.4 and 4.5 describe how persistence and robustness are measured, respectively, and present an objective function for each objective.

4.1 Previous work

Different mathematical models and solution methods have been used to solve the SVPP. Halvorsen-Weare et al. (2012) present a voyage-based model, i.e. a path-flow model, where the problem is solved in two steps: First generating all voyages (paths) that may be part of an optimal schedule, then solving the voyage-based model using the pre-generated voyages. The same model is used by Halvorsen-Weare and Fagerholt (2011) and slightly modified variants of it is presented by Shyshou et al. (2012), Norlund et al. (2015) and Borthen and Loennechen (2015). An arc-flow model of the same problem is presented by Halvorsen-Weare and Fagerholt (2016), and results show that it is more efficient to solve the voyage-based model, including the generation of voyages, than to solve the arc-flow model. The mathematical model presented below represents a new approach to modelling the SVPP, based on the model of the PVRP presented by Vidal et al. (2012b). It replaces the constraints on evenly spread departures and overlapping voyages with predefined *patterns* defining both on what days there are departures to each installation and on what days each PSV departures from the depot. Apart from this, the model has many similarities with the arc-flow model of Halvorsen-Weare and Fagerholt (2016), which has been shown to be less efficient than the voyage-based model. For this reason, it is likely that solving the model using exact methods would be less efficient than solving a voyage-based model. Note also that parts of the formulation is non-linear. The formulation is presented, however, since it relates more closely to the solution methods that will be used in Chapters 5 and 6 than the other formulations mentioned. The voyage-based model

(VBM) presented by Borthen and Loennechen (2015) is used to find the optimal solutions presented in Chapter 7. The formulation of the VBM can be found in Appendix A and a description of the procedures used by Borthen and Loennechen (2015) to generate the candidate voyages for the VBM can be found in Appendix B.

4.2 Remarks and assumptions

This section describes some properties of the problem that are important to note, and that makes the SVPP different from standard PVRPs. Some reflect the reality for Statoil in the North Sea, while others are simplifications. The mathematical model presented in Sections 4.3-4.5 are based on these properties.

Remarks and assumptions about the depot

The supply depot is closed at night, and a full working day is needed to prepare a PSV for a voyage. This has two implications. The first is that all PSVs that depart on a certain day has to be at the depot when it opens, since the entire day is needed to prepare the PSV for a new voyage. The other implication is that all PSVs depart from the depot when it closes. It is also assumed that servicing multiple PSVs at the same time does not affect the service time at the depot, as long as the depot capacity constraint is not violated. In reality, the service time at the depot will vary, but for planning purposes this is a reasonable assumption.

Remarks and assumptions about the installations

No installations require more than one service per day. It is assumed that the installations are available for PSV visits at all times, i.e. that none of the installations have time windows. In reality, this is the case for 23 out of 27 installations, thus this is considered a reasonable assumption for the purpose of this thesis. It is also assumed that an installation can be serviced by multiple PSVs at the same time, without affecting the service time. Another assumption is that the demand of supplies delivered in each visit is constant and equal for each visit. In reality, the installations report demand continuously and the demand is subject to variations. The demand is up-scaled by a load factor to allow some variation, and based on the experience of the planners at Statoil, the adjustment is sufficient to satisfy demand.

Remarks and assumptions about the PSVs

The sailing speed of all PSVs is assumed to be constant. In reality, the speed will vary with weather conditions. Including the uncertainty in weather conditions in the model is a complicated task and outside the scope of this thesis.

Comparison with previous work

Most of the assumptions above are made by previous studies of the SVPP. Halvorsen-Weare et al. (2012) make all of the same assumptions, except that they do have time

windows for the installations. They also present an extension that avoids that two PSVs visit the same installation at the same time. Halvorsen-Weare and Fagerholt (2011), Norlund and Gribkovskaia (2013) and Norlund et al. (2015) make many of the same assumptions, but they consider the PSV speed to be affected by the weather conditions.

4.3 Mathematical formulation of the SVPP

This section presents a mathematical formulation of the SVPP. The formulation is based on the PVRP formulation used by Vidal et al. (2012a), and constraints that are specific to the SVPP are added. The formulation uses departure patterns to decide both on which days the PSVs depart and on which days there are departures to each installation. The *installation departure patterns* define the set of days on which there should be a departure to the installation. For example, the pattern $\{0, 3, 5\}$ indicates that there is a voyage that services the installation departing on Monday, Thursday and Saturday (days are zero-indexed). Similarly, the *PSV departure patterns* define the set of days on which a PSV departs from the depot. The use of patterns replaces some of the constraints used by Halvorsen-Weare et al. (2012): the installation departure patterns replaces the constraints for ensuring evenly spread departures, while the PSV departure patterns replaces the constraints on overlapping voyages.

Sets

\mathcal{N}	- The set of installations and the depot
$\mathcal{N}^{CUST} = \mathcal{N} \setminus \{0\}$	- The set of installations
\mathcal{V}	- The set of PSVs available
\mathcal{T}	- The set of days in the planning period
\mathcal{P}_i^{CUST}	- The set of feasible departure patterns for installation i
\mathcal{P}_v^{PSV}	- The set of feasible departure patterns for PSV v

Variables

There are four main decisions to be made: (1) Which PSVs should be chartered, (2) on which days there should be a departure to each installation, (3) which PSV should depart to each installation on these days and (4) in what sequence should each PSV service the installations. These decisions are reflected in the following variables in the model:

$$\delta_v = \begin{cases} 1, & \text{if PSV } v \text{ is chartered} \\ 0, & \text{otherwise} \end{cases}$$

$$x_{ijvt} = \begin{cases} 1, & \text{if PSV } v \text{ sails directly from } i \text{ to } j \text{ on a voyage beginning on day } t \\ 0, & \text{otherwise} \end{cases}$$

$$y_{ivt} = \begin{cases} 1, & \text{if PSV } v \text{ begins a voyage that services installation } i \text{ on day } t \\ 0, & \text{otherwise} \end{cases}$$

$$z_{ip} = \begin{cases} 1, & \text{if installation departure pattern } p \text{ is used for installation } i \\ 0, & \text{otherwise} \end{cases}$$

$$u_{vp} = \begin{cases} 1, & \text{if PSV departure pattern } p \text{ is used for PSV } v \\ 0, & \text{otherwise} \end{cases}$$

The variable δ_v is defined for $v \in \mathcal{V}$, while x_{ijvt} and y_{ivt} are defined for $i, j \in \mathcal{N}, v \in \mathcal{V}$ and $t \in \mathcal{T}$. z_{ip} is defined for $i \in \mathcal{N}^{CUST}, p \in \mathcal{P}_i^{CUST}$, and u_{vp} is defined for $v \in \mathcal{V}, p \in \mathcal{P}_v^{PSV}$. An important constraint in the SVPP is that the departures to each installation should be evenly spread throughout the planning period. This is handled in the model by predefining all feasible installation departure patterns for each installation. Similarly, all feasible departure patterns are predefined for each PSV. The PSV departure patterns are necessary in order to prevent a PSV from departing on a new voyage before it has returned to the supply depot. It is important to note the distinction between when a PSV departs to an installation and when the installation is actually serviced. Since a voyage can last more than one day, the visit will not necessarily be on the same day as the departure. The index t in x_{ijvt} and y_{ivt} indicates that the voyage begins on day t , but on what exact day the PSV services the installation and on what day the PSV sails from i to j depends on the order the installations in the voyage are serviced. The feasible installation departure patterns are contained in the set $\mathcal{P}_i^{CUST}, i \in \mathcal{N}^{CUST}$, and the feasible PSV departure patterns are contained in the set $\mathcal{P}_v^{PSV}, v \in \mathcal{V}$. The parameter $A_{pt} = 1$ if and only if pattern p contains the day t , where p is any feasible pattern, both for installations and PSVs. Installations with the same required visit frequency will have the same set of feasible installation departure patterns.

The following variables are included to keep track of the quantity delivered, duration and number of installations visited in each voyage:

- q_{vt} - Quantity delivered on the voyage sailed by PSV v beginning on day t
- τ_{vt} - Duration of the voyage sailed by PSV v beginning on day t
- n_{vt} - Number of installations visited in the voyage sailed by PSV v beginning on day t

Parameters

C_v^{TC}	-	Cost of chartering PSV v for the length of the planning period
C_{ijv}^S	-	Cost of PSV v sailing from i to j , including service cost at j
T_{ijv}	-	Time taken for PSV v to sail from i to j , including service time at j
A_{pt}	-	Equals 1 if departure pattern p contains the day t , 0 otherwise
B_t	-	Depot capacity on day t
D_i	-	Quantity of supplies demanded per visit at installation i
Q_v^{MIN}	-	Minimum quantity of supplies delivered per voyage for PSV v
Q_v^{MAX}	-	Maximum quantity of supplies delivered per voyage for PSV v
T^{MIN}	-	Minimum duration per voyage
T_{pt}^{MAX}	-	Maximum duration for the voyage beginning on day t if pattern p is used
N^{MIN}	-	Minimum number of installations serviced per voyage
N^{MAX}	-	Maximum number of installations serviced per voyage

Objective function

The objective function (4.1) minimizes the sum of chartering cost and sailing cost, where C_v^{TC} is the time charter cost of PSV v , and C_{ijv}^S is the cost for PSV v of sailing from i to j .

$$\min \sum_{v \in \mathcal{V}} C_v^{TC} \delta_v + \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} \sum_{v \in \mathcal{V}} \sum_{t \in \mathcal{T}} C_{ijv}^S x_{ijvt} \quad (4.1)$$

Constraints

The objective function is subject to the following constraints:

$$\sum_{p \in \mathcal{P}_i^{CUST}} z_{ip} = 1, \quad i \in \mathcal{N}^{CUST} \quad (4.2)$$

$$\sum_{p \in \mathcal{P}_v^{PSV}} u_{vp} = 1, \quad v \in \mathcal{V} \quad (4.3)$$

$$y_{ivt} = \sum_{j \in \mathcal{N}} x_{ijvt}, \quad i \in \mathcal{N}, v \in \mathcal{V}, t \in \mathcal{T} \quad (4.4)$$

$$\sum_{v \in \mathcal{V}} y_{ivt} - \sum_{p \in \mathcal{P}_i^{CUST}} A_{pt} z_{ip} = 0, \quad i \in \mathcal{N}^{CUST}, t \in \mathcal{T} \quad (4.5)$$

Constraints (4.2) and (4.3) ensure that exactly one departure pattern is chosen for each installation and PSV, respectively. Constraints (4.4) connect the x -variables with the y -variables, while constraints (4.5) make sure that there is a departure to each installation only on days that are defined in the selected departure pattern.

$$\sum_{j \in \mathcal{N}} x_{0jvt} - \sum_{p \in \mathcal{P}_v^{PSV}} A_{pt} u_{vp} = 0, \quad v \in \mathcal{V}, t \in \mathcal{T} \quad (4.6)$$

$$\sum_{j \in \mathcal{N}} x_{0jvt} - y_{ivt} \geq 0, \quad i \in \mathcal{N}, v \in \mathcal{V}, t \in \mathcal{T} \quad (4.7)$$

$$y_{ivt} \leq \delta_v, \quad i \in \mathcal{N}, v \in \mathcal{V}, t \in \mathcal{T} \quad (4.8)$$

$$\sum_{j \in \mathcal{N}} x_{jivt} - \sum_{j \in \mathcal{N}} x_{ijvt} = 0, \quad i \in \mathcal{N}, v \in \mathcal{V}, t \in \mathcal{T} \quad (4.9)$$

$$\sum_{i \in S} \sum_{j \in S} x_{ijvt} \leq |S| - 1, \quad S \subset \mathcal{N} \setminus \{0\}, |S| \geq 2, v \in \mathcal{V}, t \in \mathcal{T} \quad (4.10)$$

Constraints (4.6) make sure the PSVs depart on the days given by their PSV departure pattern. Constraints (4.7) ensure that a PSV that services an installation leaves the depot, and constraints (4.8) make sure that only chartered PSVs can service installations. Constraints (4.9) conserve the flow in and out of all installations and the depot. Constraints (4.10) eliminate subtours.

$$q_{vt} = \sum_{i \in \mathcal{N}^{CUST}} D_i y_{ivt}, \quad v \in \mathcal{V}, t \in \mathcal{T} \quad (4.11)$$

$$Q_v^{MIN} \sum_{p \in \mathcal{P}_v^{PSV}} A_{pt} u_{vp} \leq q_{vt} \leq Q_v^{MAX}, \quad v \in \mathcal{V}, t \in \mathcal{T} \quad (4.12)$$

$$\tau_{vt} = \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} T_{ijv} x_{ijvt}, \quad v \in \mathcal{V}, t \in \mathcal{T} \quad (4.13)$$

$$T^{MIN} \sum_{p \in \mathcal{P}_v^{PSV}} A_{pt} u_{vp} \leq \tau_{vt} \leq \sum_{p \in \mathcal{P}_v^{PSV}} T_{pt}^{MAX} u_{vp}, \quad v \in \mathcal{V}, \quad t \in \mathcal{T} \quad (4.14)$$

$$n_{vt} = \sum_{i \in \mathcal{N}^{CUST}} y_{ivt}, \quad v \in \mathcal{V}, \quad t \in \mathcal{T} \quad (4.15)$$

$$N^{MIN} \sum_{p \in \mathcal{P}_v^{PSV}} A_{pt} u_{vp} \leq n_{vt} \leq N^{MAX}, \quad v \in \mathcal{V}, \quad t \in \mathcal{T} \quad (4.16)$$

Constraints (4.11), (4.13) and (4.15) define the value of the variables q_{vt} , τ_{vt} and n_{vt} . Constraints (4.12), (4.14) and (4.16) define the limits on these three variables. The sum on the left-hand side of these constraints indicates whether PSV v departs from the depot on day t . This sum is multiplied with the lower limits, since the limits only apply if PSV v departs on a voyage on day t . If it does not depart on that day, the variables q_{vt} , τ_{vt} and n_{vt} will be equal to 0. Note that the upper limit on duration of a voyage, T_{pt}^{MAX} , depends on the departure pattern of the PSV sailing it. For example, if a PSV has a departure on both Monday and Wednesday, the duration of the voyage beginning on Monday cannot be longer than two days, since it needs to be ready for a new voyage on Wednesday. The voyage departing on Wednesday, on the other hand, needs only to be finished before the next Monday, i.e. after five days. In practice, an upper limit lower than five days will usually be set to reduce delays, since longer voyages are more prone to delays.

$$\sum_{j \in \mathcal{N}} \sum_{v \in \mathcal{V}} x_{0jvt} \leq B_t, \quad t \in \mathcal{T} \quad (4.17)$$

Constraints (4.17) ensure that the depot capacity is not exceeded.

4.4 Measuring persistence

Persistence indicates that a new solution has few changes from the existing solution. The following section describes how to measure persistence in the SVPP. Change is defined and it is described how to maximize persistence by minimizing the number of changes.

The existing solution is referred to as the *baseline solution*. The most appropriate method for measuring changes depends on the preferences of decision makers. In the following model, only the changes in departures to an installation are used to measure

change. This is based on information from Statoil that the most important factor to be persistent is the supply ordering routines, that is, at what times the installations need to report their demand. This is determined by the time the PSVs depart from the supply depot. Let \mathcal{N}^B be the set of installations in the baseline solution B . Change from baseline solution B to a new solution is then measured as:

$$\sum_{i \in \mathcal{N}^B \cap \mathcal{N}^{CUST}} \sum_{t \in \mathcal{T}} |\sigma_{it} - \sigma_{it}^B|, \quad (4.18)$$

where σ_{it} is 1 if a PSV that services installation i leaves the supply depot on day t in the current solution and 0 otherwise. σ_{it}^B has the corresponding values for the baseline solution B . Note that σ_{it} is a variable, while σ_{it}^B is a parameter. Also note that σ_{it} is binary, since an installation cannot have more than one departure per day. This is enforced through constraints (4.5) and the installation departure patterns. The sum is only including the installations which are both in the current and the baseline problem to avoid counting the changes that inevitably arise when adding or removing installations. σ_{it} can be calculated using the following formula:

$$\sigma_{it} = \sum_{v \in \mathcal{V}} y_{ivt}, \quad i \in \mathcal{N}^B \cap \mathcal{N}^{CUST}, \quad t \in \mathcal{T}. \quad (4.19)$$

To simplify the objective function, a new variable is introduced

$$\gamma_{it} = \begin{cases} 1, & \text{if there is a change in departures to installation } i \text{ on day } t \\ 0, & \text{otherwise} \end{cases}$$

which is defined for all $i \in \mathcal{N}^B \cap \mathcal{N}^{CUST}, t \in \mathcal{T}$. In other words, $\gamma_{it} = |\sigma_{it} - \sigma_{it}^B|$. Expression (4.18) uses the absolute value operator, which is non-linear. The solution methods that are used in this thesis do not require linearity, but Borthen and Loennechen (2015) present a possible linearization. After inserting the new variable into Equation (4.18), the objective function becomes:

$$\min \sum_{i \in \mathcal{N}^B \cap \mathcal{N}^{CUST}} \sum_{t \in \mathcal{T}} \gamma_{it} \quad (4.20)$$

4.5 Measuring robustness of schedules

As shown in Section 3.4, several approaches to robustness have been used earlier: approaches based on the slack of voyages, approaches that combine simulation and optimization, and more advanced approaches like stochastic programming and robust optimization techniques. Comparing the slack-based approaches and combinations of

simulation and optimization, the latter seem to perform better, but are also more complex. The slack-based approaches are simpler than combining simulation and optimization, but still perform well. Since the objective of this thesis is to solve the MSVPP, not study robustness in detail, a slack-based approach is used due to its simplicity. Specifically, the robustness measure that will be used is the *share of sailed voyages that have at least a minimum number of hours of slack*. Using the share of sailed voyages instead of number of robust voyages makes it easy to compare robustness across different problem instances, since the robustness value is in the same range (i.e. between zero and one) regardless of problem size. The following section describes how robustness is measured.

A voyage is considered a *robust voyage* if it has at least a given number of hours of slack. The variable d_{vt} is introduced to keep track of the duration of a voyage in days:

d_{vt} - The duration, in days, of the voyage sailed by PSV v beginning on day t

d_{vt} is calculated from τ_{vt} using the formula

$$d_{vt} = \lceil \frac{\tau_{vt} + T^{DEPOT}}{24} \rceil, \quad (4.21)$$

where T^{DEPOT} is the time taken to prepare a PSV at the depot. Let the parameter $T_{d_{vt}}^{ROB}$ define the longest duration, in hours, a voyage lasting d_{vt} days can have and still be considered robust. For instance, if the depot opens 16 hours after a PSV departs and the required slack is 4 hours, a voyage that lasts one day needs to be shorter than 12 hours in order to arrive in time for the opening of the depot and have the required slack. Similarly, a voyage lasting two days needs to be shorter than 36 hours. Note that the expression above is non-linear, but since the solution methods in this thesis do not require linearity it can be used here. The required slack can vary with the duration of the voyage, e.g. two-day voyages can require less slack than three-day voyages, since longer voyages are more prone to disruptions. Whether a voyage is robust or not is represented by the variable r_{vt} :

$$r_{vt} = \begin{cases} 1, & \text{if the voyage sailed by PSV } v \text{ on day } t \text{ is robust} \\ 0, & \text{otherwise} \end{cases}$$

The connection between the duration of a voyage τ_{vt} and the variable r_{vt} can be enforced by the following constraints:

$$\tau_{vt} - T_{d_{vt}}^{ROB} \leq M(1 - r_{vt}), \quad v \in \mathcal{V}, t \in \mathcal{T}, \quad (4.22)$$

where M is a sufficiently large number to ensure that the constraint is not binding for any value of τ_{vt} when $r_{vt} = 0$. Note that the variable d_{vt} is used as an index, making the expression non-linear. A linear, voyage-based model of the problem can be found in Appendix A. The robustness of a schedule is defined as the share of voyages sailed that are robust. Since the goal is to maximize robustness, the new objective can be written as

$$\max \frac{\sum_{v \in \mathcal{V}} \sum_{t \in \mathcal{T}} r_{vt}}{\sum_{v \in \mathcal{V}} \sum_{p \in \mathcal{P}_v^{PSV}} \sum_{t \in \mathcal{T}} A_{pt} u_{vp}}, \quad (4.23)$$

where the numerator is the number of robust voyages and the denominator is the total number of voyages sailed in the solution, calculated by summing up all departures in the PSV patterns used.

A weakness of this robustness measure is that it does not distinguish between a voyage having zero hours of slack and one that has slack one hour below the threshold, when in reality there can be a significant difference between the two. Also, the setting of the slack thresholds is not straightforward. However, the measure is easy to understand and improving it will improve the robustness of a solution. The measure satisfies the goal of robustness: reducing the probability that the demand at an installation cannot be met on time using the planned schedule. Therefore it is used as the robustness measure here.

Chapter 5

Hybrid genetic search with adaptive diversity control for the SVPP

This chapter presents the solution method used for solving the SVPP with minimum cost as the only objective. The solution method is extended to handle multiple objectives in Chapter 6. The method is based on the Unified Hybrid Genetic Search (UHGS) framework for solving multi-attribute VRPs, developed by Vidal et al. (2014). UHGS is a generic framework that does not specify all procedures, e.g. local search and crossover, meaning they need to be tailored to best fit the problem at hand. Vidal et al. (2014) identify four key concepts that make the UHGS framework successful:

1. Hybridization of a genetic algorithm with efficient local search procedures
2. A solution representation without trip delimiters
3. Allowing and penalizing infeasible solutions
4. Population management by evaluating solutions by both cost and diversity contribution

One implementation of the UHGS is the Hybrid Genetic Search with Adaptive Diversity Control (HGSADC), first presented by Vidal et al. (2012a), which is used for solving PVRPs and Multi-Depot PVRPs (MDPVRPs). The heuristic used for solving the SVPP draws heavily on this implementation, although some parts have been modified to fit the SVPP. The HGSADC is a non-deterministic heuristic, meaning that it neither guarantees optimal solutions nor the same solutions when run multiple times. The HGSADC takes a fleet of PSVs as input, and generates a *schedule*, consisting of a set of voyages each PSV shall sail and when they shall sail them, as output. A *solution* to the SVPP consists of an optimal fleet of PSVs and an optimal schedule given that fleet. By running the HGSADC multiple times with different fleets as input, a best-known fleet can be found. Note the distinction between a solution and

a schedule: a solution consists of both a fleet and a schedule. In the following, the terms *individual* and schedule will be used interchangeably, while solution refers to a combination of a fleet and a schedule.

The following chapter describes the methods used for solving the SVPP in detail. The adapted version of the HGSADC for solving the SVPP with a fixed fleet of PSVs is described in Sections 5.1 - 5.8 and Section 5.9 describes how the HGSADC can be used to optimize the fleet size and mix. The chapter concludes with a comparison with the work of Vidal et al. (2012a) and Vidal et al. (2014) in Section 5.10.

5.1 Overview

Algorithm 1 describes the overall structure of the HGSADC proposed for solving the SVPP. The algorithm works on a *population* of individuals. It keeps the population \mathcal{S} separated in two disjoint subpopulations: The subpopulation of feasible individuals, $\mathcal{S}^{FEASIBLE}$, and the subpopulation of infeasible individuals, $\mathcal{S}^{INFEASIBLE}$. The algorithm keeps breeding new individuals until there have been I^{NI} iterations without improvement, or the maximum running time limit T^{MAXRUN} is reached. One *iteration* refers to the breeding and education of a new individual, i.e. one iteration of the while-loop starting on Line 2 in Algorithm 1. The size of each subpopulation is limited by the parameters μ and λ . μ is the minimum size and λ is the generation size, such that the maximum subpopulation size is $\mu + \lambda$. When the maximum subpopulation size is reached, individuals are removed until there are only μ individuals left in the subpopulation, a process referred to as *survivor selection*. The initial population is created using the construction heuristic described in Section 5.5.

5.2 Individual representation

An individual is a schedule that determines which voyages each of the PSVs should sail, and at what day the PSV should start the voyage. An example of a schedule for a problem spanning four days with four installations and two PSVs is shown in Figure 5-1. In this example, each voyage has a duration of at most two days.

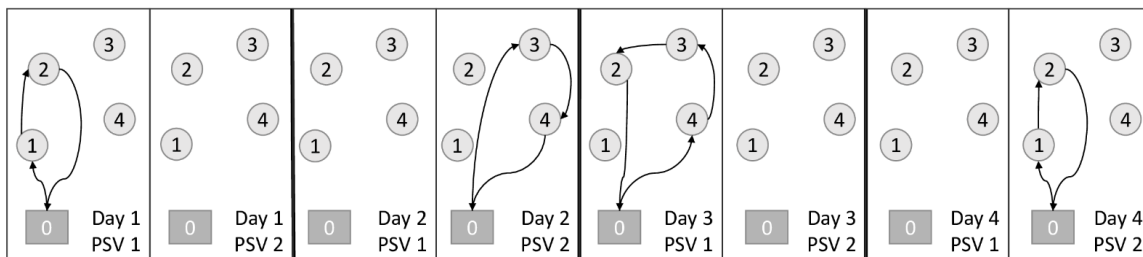


Figure 5-1: Example of a schedule for a problem with four days, four installations and two PSVs.

Algorithm 1 Hybrid Genetic Search with Adaptive Diversity Control (HGSADC)

```
1: Initialize population ▷ Section 5.5
2: while Iterations without improvement  $< I^{NI}$  and time  $< T^{MAXRUN}$  do
3:   Select parent individuals  $s_1$  and  $s_2$  ▷ Section 5.6
4:   Generate offspring  $s_{new}$  from  $s_1$  and  $s_2$  (crossover) ▷ Section 5.6
5:   Educate offspring  $s_{new}$  with probability  $\rho^{EDU}$  ▷ Section 5.7
6:   if  $s_{new}$  is infeasible then
7:     Repair  $s_{new}$  with probability  $\rho^{REP}$  ▷ Section 5.7.3
8:   end if
9:   if  $s_{new}$  is still infeasible then
10:    Insert  $s_{new}$  into infeasible subpopulation
11:   else
12:    Insert  $s_{new}$  into feasible subpopulation
13:   end if
14:   if maximum subpopulation size  $\mu + \lambda$  reached then
15:     Select survivors ▷ Section 5.8.1
16:   end if
17:   Adjust penalty parameters for violating feasibility conditions ▷ Section 5.8.2
18:   if best individual not improved for  $I^{DIV}$  iterations then
19:     Diversify population ▷ Section 5.8.3
20:   end if
21:   Return best feasible individual
22: end while
```

Each individual is represented by three separate *chromosomes*. The first chromosome is the *installation chromosome*, which for each installation defines at what days the voyages that visit the installation depart from the depot, referred to as the *installation pattern* $\pi_i(s)$ for installation i . The installation chromosome for the schedule in Figure 5-1 is shown in Table 5.1.

Inst i	1	2	3	4
Pat $\pi_i(s)$	{1, 4}	{1, 3, 4}	{2, 3}	{2, 3}

Table 5.1: Installation chromosome corresponding to the individual in Figure 5-1. The top row shows the installation and the bottom row shows the days on which there is a departure to the installation.

The second chromosome is the *PSV chromosome*, which for each PSV defines at what days the PSV departs from the depot, referred to as the *PSV pattern* $\beta_v(s)$ for PSV v . The PSV chromosome for the schedule in Figure 5-1 is shown in Table 5.2.

PSV v	1	2
Pat $\beta_v(s)$	{1, 3}	{2, 4}

Table 5.2: PSV chromosome corresponding to the individual in Figure 5-1. The top row shows the PSV and the bottom row shows the days on which the PSV departs from the depot.

The installation patterns and PSV patterns correspond directly with the patterns described in Chapter 4, but are represented in a different way here. If installation i uses pattern p in individual s , meaning that $z_{ip} = 1$, $\pi_i(s)$ is given by Equation (5.1):

$$\pi_i(s) = \{t \mid t \in \mathcal{T} \wedge A_{pt} = 1\}. \quad (5.1)$$

Similarly, if PSV v uses pattern p , meaning that $u_{vp} = 1$, $\beta_v(s)$ is given by Equation (5.2):

$$\beta_v(s) = \{t \mid t \in \mathcal{T} \wedge A_{pt} = 1\}. \quad (5.2)$$

The final chromosome is the *tour chromosome*, which defines the complete schedule, with a voyage r_{vt} assigned to each $(v, t), v \in \mathcal{V}, t \in \mathcal{T}$. A voyage is a sequence of installations that are to be visited, in the order given by the sequence. If voyage r_{vt} is empty, PSV v does not depart on a voyage on day t . Figure 5-2 shows a schedule s and its tour chromosome. The sequence in voyage r_{13} indicates that PSV 1 departs on day 3, visiting the installations 4 - 3 - 2, in that order.

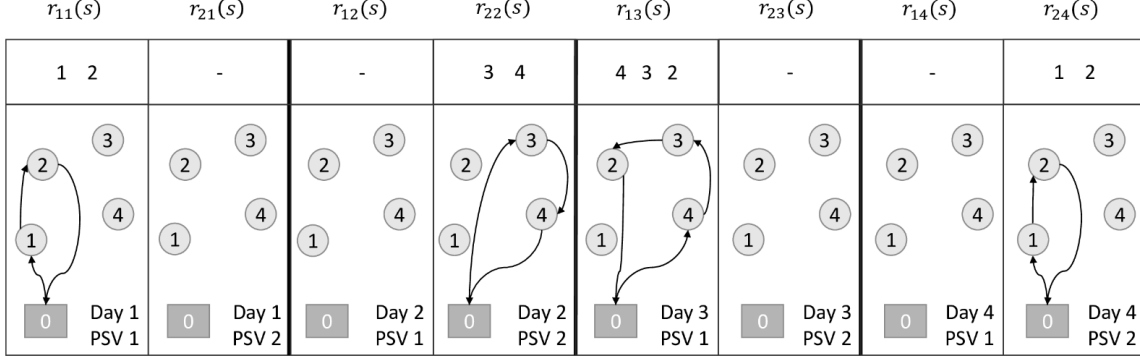


Figure 5-2: Example of an individual and its corresponding tour chromosome.

5.3 Search space

Optimal individuals often lie at the boundary of feasibility, hence both Korsvik et al. (2011) and Vidal et al. (2014) argue that allowing infeasible individuals improves the performance of the search. Therefore, infeasible individuals are included in the search, but penalized according to how far they are from being feasible. However, the infeasible individuals are only allowed to violate some of the constraints. All individuals in the search space, both feasible and infeasible, satisfy the constraints on (1) number of visits to each installation, (2) spread of departures to each installation and (3) depot capacity. In addition, all voyages start and end at the depot, and the individuals do not contain any subtours. The construction heuristic, education and crossover are all designed to ensure this. The constraints that can be violated are the lower and upper limits on duration of voyages, capacity of PSVs and the number of installations visited in each voyage, i.e. constraints (4.12), (4.14) and (4.16).

5.4 Evaluation of individuals

Individuals are evaluated based on their cost, how much they violate the constraints and how much they contribute to the diversity of the population. Let $\mathcal{R}(s)$ be the set of voyages sailed in individual $s \in \mathcal{S}^{FEASIBLE} \cup \mathcal{S}^{INFEASIBLE}$. Let c_{vt} be the cost of sailing voyage r_{vt} , that is, the voyage sailed by PSV v beginning on day t . The *penalized cost* ϕ_{vt} of voyage r_{vt} is defined as the cost of the voyage plus any penalties if the voyage is infeasible, calculated as

$$\begin{aligned}
\phi_{vt} &= c_{vt} \\
&+ \omega^D \max\{0, T^{MIN} - \tau_{vt}, \tau_{vt} - T_{vt}^{MAX}\} \\
&+ \omega^Q \max\{0, Q^{MIN} - q_{vt}, q_{vt} - Q_v^{MAX}\} \\
&+ \omega^N \max\{0, N^{MIN} - n_{vt}, n_{vt} - N^{MAX}\}.
\end{aligned} \tag{5.3}$$

ω^D , ω^Q and ω^N are the *penalty parameters* per unit violation of the constraints on duration, capacity and number of installations visited, respectively. τ_{vt} , q_{vt} and n_{vt} are the duration, utilized capacity on the PSV and the number of installations in voyage r_{vt} , respectively. These are equal to the variables τ_{vt} , q_{vt} and n_{vt} presented in Chapter 4. The maximum duration T_{vt}^{MAX} of a voyage is determined by what pattern PSV v uses. Recall that $u_{vp} = 1$ if PSV v uses pattern p and that T_{pt}^{MAX} is the maximum duration of a voyage departing on day t when using PSV pattern p . The maximum duration of voyage r_{vt} can be calculated as

$$T_{vt}^{MAX} = \sum_{p \in \mathcal{P}_v^{PSV}} u_{vp} T_{pt}^{MAX}. \tag{5.4}$$

The penalized cost of an individual s is the sum of the penalized cost of every voyage in the individual, $\phi(s) = \sum_{(v,t) \in \mathcal{R}(s)} \phi_{vt}$. The penalized cost is used to evaluate the fitness of the individual. One of the key success factors of a population-based metaheuristic like the HGSADC is to maintain a diverse population of individuals. Therefore, each individual s is evaluated in terms of its *diversity contribution*, $\Delta(s)$, defined as the average distance to its n^{CLO} closest neighbours. Let \mathcal{N}^{CLO} be the set containing these neighbours. Then the diversity contribution of individual s becomes

$$\Delta(s) = \frac{1}{n^{CLO}} \sum_{s_2 \in \mathcal{N}^{CLO}} \delta^H(s, s_2), \tag{5.5}$$

where $\delta^H(s_1, s_2)$ is the normalized Hamming distance between individuals s_1 and s_2 , based on the Hamming distance first presented by Hamming (1950). The normalized Hamming distance counts the number of installations that have different departure patterns and the number of installations that are serviced by a different set of PSVs. The sum is normalized by dividing it by two times the number of installations, giving a value between 0 and 1. Let $\mathcal{V}_i(s)$ be the set of PSVs servicing installation i in

individual s . Then the normalized Hamming distance becomes

$$\delta^H(s_1, s_2) = \frac{1}{2|\mathcal{N}^{CUST}|} \sum_{i \in \mathcal{N}^{CUST}} (\mathbf{1}(\pi_i(s_1) \neq \pi_i(s_2)) + \mathbf{1}(\mathcal{V}_i(s_1) \neq \mathcal{V}_i(s_2))), \quad (5.6)$$

where $\mathbf{1}(cond) = 1$ if condition *cond* is true and 0 otherwise. Every individual is *ranked* based on its penalized cost and its diversity contribution. Let $Rank^C(s)$ and $Rank^D(s)$ be the rank of individual s in terms of penalized cost and diversity contribution, respectively. The individual with the lowest penalized cost will have $Rank^C(s) = 1$, and the individual with the highest penalized cost will have $Rank^C(s) = |\mathcal{S}^{FEASIBLE}| + |\mathcal{S}^{INFEASIBLE}|$. The ranks are further used to calculate the *biased fitness*, given by Equation (5.7), where n^{ELI} is the number of elite individuals wanted to survive to the next generation and \mathcal{S} is the current population of individuals.

$$BF(s) = Rank^C(s) + (1 - \frac{n^{ELI}}{|\mathcal{S}|}) Rank^D(s) \quad (5.7)$$

5.5 Constructing the initial population

The initial population is initialized by creating $K^{INIT}\mu$ individuals and assigning each individual to the appropriate subpopulation, where μ is the minimum number of individuals in each subpopulation. The number of individuals created should be high enough to contribute sufficiently to the diversity of the population, but not too high, as survivor selection (explained in Section 5.8.1) will restrict the size of each subpopulation to $\mu + \lambda$. Note that during the first iterations of the HGSADC, the balance between feasible and infeasible individuals may be skewed such that one of the subpopulations contains less than μ individuals.

The initial population is created using the construction heuristic described in Algorithm 2. An individual s is created in three steps, one for each of the chromosomes used to represent an individual. The first step is to create the installation departure chromosome by randomly assigning an installation pattern π_i to each installation i . The installation pattern is selected from the set of feasible installation patterns for installation i , \mathcal{P}_i^{CUST} . The union of all the assigned installation patterns is the set of days that need a departure, denoted $\mathcal{T}^{DEP}(s)$. The second step is to create the PSV departure chromosome by randomly assigning a PSV pattern $\beta_v(s)$ to each PSV v . The PSV pattern is selected from the set of possible PSV patterns for each PSV v , \mathcal{P}_v^{PSV} , that visit at least one day in $\mathcal{T}^{DEP}(s)$. When a PSV pattern is assigned, the days in the pattern are removed from the set of days that need a departure: $\mathcal{T}^{DEP}(s) = \mathcal{T}^{DEP}(s) \setminus \beta_v(s)$. If $\mathcal{T}^{DEP}(s) \neq \emptyset$ after all PSVs have been assigned a PSV pattern, at least one installation pattern have days with no PSV departing,

and the second step is restarted. The PSV pattern could be selected randomly from \mathcal{P}_v^{PSV} , but making sure that the selected pattern includes at least one day in $\mathcal{T}^{DEP}(s)$ reduces the number of restarts significantly, as the chance of having a PSV departing on all days in $\mathcal{T}^{DEP}(s)$ increases. The second step of the construction heuristic is also restarted if the depot capacity constraint is violated on any day of the planning period. In other words, the second step is completed when there is at least one departure on all days in $\mathcal{T}^{DEP}(s)$ and the depot capacity constraint is not violated on any day. The third step is to create the tour chromosome by allocating installation departures to a combination of day and PSV. The set of installations that have a departure on day t , denoted $\mathcal{N}_t(s)$, can be generated from the installation departure chromosome. Likewise, the set of PSVs that have a departure on day t , denoted $\mathcal{V}_t(s)$, can be generated from the PSV departure chromosome. The tour chromosome is created by iterating through all days in the planning period, $t \in \mathcal{T}$, and for each day allocate each of the installations in $\mathcal{N}_t(s)$ to a PSV, randomly selected from $\mathcal{V}_t(s)$. After the three steps are completed, all of the chromosomes that represent an individual have been created. The individual then undergoes education, as described in Section 5.7, and is assigned to the appropriate subpopulation.

5.6 Parent selection and crossover

Crossover is the process where the chromosomes of two parent individuals are combined into a new individual. Two parents, s_1 and s_2 are chosen and the result of the crossover is a new individual (also known as *offspring*) s_{new} . Each parent is selected by a binary tournament, i.e. randomly picking two individuals from the entire population and choosing the one with the best biased fitness as the parent. The crossover procedure is described in Algorithm 3. The algorithm begins by selecting which parts of the chromosome should be inherited from which parent. This is done by randomly dividing the set of (PSV v , day t) couples into three disjoint sets: Λ_1 , Λ_2 and Λ_{mix} , containing the (PSV, day) couples that shall inherit data from s_1 , s_2 and both, respectively. The next step (step 1) is to inherit data from s_1 . For the (v, t) couples that are in Λ_1 , all departures are simply copied from s_1 to s_{new} . For the (v, t) couples that are in Λ_{mix} , two random cut-off points, α_{vt}^1 and α_{vt}^2 , are picked, and the installation sequence between α_{vt}^1 (inclusive) and α_{vt}^2 (exclusive) is copied from s_1 to s_{new} . Note that α_{vt}^1 may be larger than α_{vt}^2 , in which case the copied sequence is formed by removing the sequence between α_{vt}^2 and α_{vt}^1 .

Step 2 consists of inheriting data from s_2 . Recall that the constraints on (1) number of visits to each installation, (2) spread of departures to each installation and (3) depot capacity are not allowed to be violated by any individual. Since each individual complies with these constraints, the departures copied from s_1 are guaranteed to not violate them. When combining departures from two different individuals, however, one needs to check that these constraints are not violated before copying departures. These constraints are enforced by the installation patterns and PSV patterns, therefore the algorithm checks that both the installation pattern and PSV pattern resulting from copying an installation departure i into a voyage are feasible. Line 13 checks

Algorithm 2 Construction heuristic

1: $individualsCreated \leftarrow 0$
2: **while** $individualsCreated < K^{INIT}\mu$ **do**

 STEP 1: SELECT RANDOM INSTALLATION PATTERN

3: **for** each installation $i \in \mathcal{N}^{CUST}$ **do**
4: $\pi_i(s) \leftarrow$ random pattern in \mathcal{P}_i^{CUST}
5: **end for**

 STEP 2: CREATE PSV PATTERN

6: $\mathcal{T}^{DEP}(s) \leftarrow \bigcup_{i=1}^{|\mathcal{N}^{CUST}|} \pi_i(s)$
7: **for** each PSV $v \in \mathcal{V}$ **do**
8: $\beta_v(s) \leftarrow$ random pattern in \mathcal{P}_v^{PSV} that contains at least one day in $\mathcal{T}^{DEP}(s)$
9: $\mathcal{T}^{DEP}(s) \leftarrow \mathcal{T}^{DEP}(s) \setminus \beta_v(s)$
10: **end for**
11: **if** $\mathcal{T}^{DEP}(s) \neq \emptyset$ or *depot capacity constraint is violated* **then**
12: **go to** 6 ▷ restart step 2
13: **end if**

 STEP 3: CREATE TOUR CHROMOSOME

14: **for** each day $t \in \mathcal{T}$ **do**
15: **for** each installation $i \in \mathcal{N}_t(s)$ **do**
16: $v \leftarrow$ random vessel in $\mathcal{V}_t(s)$
17: Add i to the end of voyage r_{vt}
18: **end for**
19: **end for**
20: Educate individual s with probability ρ^{EDU}
21: **if** s is infeasible **then**
22: Repair with probability ρ^{REP}
23: **end if**
24: **if** s is still infeasible **then**
25: Insert s into infeasible subpopulation
26: **else**
27: Insert s into feasible subpopulation
28: **end if**
29: $individualsCreated \leftarrow individualsCreated + 1$
30: **end while**

that there is no departure to i on day t already and that the resulting installation pattern is part of at least one of the feasible patterns for i . If the condition is not satisfied, i cannot be copied to the voyage. If the PSV v already departs on day t , there is no change in the PSV pattern, and the installation can safely be copied. If v does not depart, Line 16 checks that there is available depot capacity on that day and that the resulting PSV pattern is part of at least one feasible PSV pattern. Here, $n_t^{PSV}(s_{new})$ equals the number of PSVs departing on day t . If both of these requirements are satisfied, a new voyage is created, and the installation is copied from s_2 to s_{new} .

After all feasible installation departures have been copied from both s_1 and s_2 , there may still be installations that require more visits. These are inserted in Step 3. The missing departures are inserted in random order. Each is inserted into the voyage where it is cheapest, in terms of penalized cost, to insert it, at the cheapest position in the voyage. The cheapest voyage and position is found by iterating through all possible positions in all feasible voyages, calculating the increase in penalized cost by inserting the installation at the given position in the given voyage.

Due to the design of the crossover operator, the offspring individual s_{new} has the required number of installation visits, has a feasible installation pattern and does not exceed the depot capacity. It may, however, contain voyages that makes it infeasible in terms of number of installations per voyage, capacity of the PSVs and the maximum or minimum duration of the voyages. These three constraints are the only constraints that are allowed to be violated, and are handled by penalizing violations in the penalized cost function. It is important to note that the maximum duration of a voyage depends on the PSV departure pattern of the PSV sailing that voyage. For example, if a PSV departs on Monday, Wednesday and Saturday, the maximum duration of the voyage departing on Monday and Saturday will be two days in order to be ready for its next departure. The voyage departing on Wednesday, on the other hand, can last three days, since there are three days from Wednesday to Saturday.

5.7 Education

Education enhances the individuals found and is performed whenever a new individual is generated, either by the construction heuristic or by combining two parents using the crossover operator. Education is performed in three steps:

1. Voyage improvement
2. Pattern improvement
3. Voyage improvement once more

Voyage improvement refers to methods aiming to improve the quality of voyages by reordering the sequence of visits within a voyage. In terms of the chromosomes, this means that voyage improvement neither changes the installation patterns nor the PSV patterns. Pattern improvement, on the other hand, tries to improve the individual by

Algorithm 3 Crossover operator

STEP 0: INHERITANCE RULE

- 1: Pick two random numbers between 0 and $|\mathcal{T}| \times |\mathcal{V}|$ according to a uniform distribution. Let n_1 and n_2 be the smallest and the largest of these numbers, respectively
- 2: Randomly select n_1 (PSV, day) couples to form the set Λ_1
- 3: Randomly select $n_2 - n_1$ remaining couples to form the set Λ_2
- 4: The remaining $|\mathcal{V}| \times |\mathcal{T}| - n_2$ couples make up the set Λ_{mix}

STEP 1: INHERIT DATA FROM s_1

- 5: **for** each (PSV, day) (v, t) belonging to set Λ_1 **do**
- 6: Copy the sequence of installation departures from $r_{vt}(s_1)$ to $r_{vt}(s_{new})$
- 7: **end for**
- 8: **for** each (PSV, day) (v, t) belonging to set Λ_{mix} **do**
- 9: Randomly (uniform distribution) select two chromosome-cutting points α_{vt}^1 and α_{vt}^2 and copy the α_{vt}^1 to α_{vt}^2 substring of $r_{vt}(s_1)$ to $r_{vt}(s_{new})$
- 10: **end for**

STEP 2: INHERIT DATA FROM s_2

- 11: **for** each (PSV, day) $(v, t) \in \Lambda_2 \cup \Lambda_{mix}$ selected in random order **do**
- 12: **for** each installation departure i in $r_{vt}(s_2)$ **do**
- 13: **if** $t \notin \pi_i(s_{new})$ **and** $\exists p \in \mathcal{P}_i^{CUST}(p \supset (t \cup \pi_i(s_{new})))$ **then**
- 14: **if** $t \in \beta_v(s_{new})$ **then**
- 15: Copy installation i at the end of $r_{vt}(s_{new})$
- 16: **else if** $n_t^{PSV}(s_{new}) < B_t$ **and** $\exists p \in \mathcal{P}_v^{PSV}(p \supset (t \cup \beta_v(s_{new})))$ **then**
- 17: Create new voyage $r_{vt}(s_{new})$ and insert installation i
- 18: **end if**
- 19: **end if**
- 20: **end for**
- 21: **end for**

STEP 3: COMPLETE INSTALLATION SERVICES

- 22: **while** there are installations with unsatisfied service frequency requirements **do**
 - 23: $i \leftarrow$ random installation for which service frequency requirements are not satisfied
 - 24: Let \mathcal{F} be the set of feasible (PSV, day) combinations (v, t) with respect to the feasible installation patterns, feasible PSV patterns and depot capacity based on the visits already in s_{new} .
 - 25: **if** $\mathcal{F} = \emptyset$ **then**
 - 26: **go to** 1 \triangleright no feasible insertion, restart the procedure
 - 27: **else**
 - 28: Let $\psi(i, v, t)$ be the minimum penalized cost for the insertion of installation i into the voyage sailed by PSV v on day t .
 - 29: Insert i at least cost position in r_{vt} , where $(v, t) = \operatorname{argmin}_{(v, t) \in \mathcal{F}} \psi(i, v, t)$.
 - 30: **end if**
 - 31: **end while**
-

changing these patterns. Note that some of the pattern improvement methods used may change the order of visits within a voyage.

5.7.1 Voyage improvement

Let the neighborhood of an installation u be defined as the hn closest installations, where n is the number of installations visited on the voyage and $h \in [0, 1]$ is a parameter restricting the size of the neighborhood. The idea of the voyage improvement is to evaluate each installation u and all of its neighbors in random order. Let v be a neighbor of u , and x and y be the successors of u and v , respectively. The evaluation assesses the following moves in random order, and performs the first move that improves the penalized cost of the voyage:

- (M1) Remove u and place it after v
- (M2) Remove u and x and place u and x after v
- (M3) Remove u and x and place x and u after v
- (M4) Swap the position of u and v
- (M5) Swap the position of u and x with v
- (M6) Swap the position of u and x with v and y
- (M7) Swap the position of x and v

The voyage improvement stops when all installations and their neighborhoods have been evaluated.

5.7.2 Pattern improvement

Three different methods are used in the pattern improvement: (1) Changing installation departure patterns, (2) merging multiple voyages departing on the same day and (3) moving installation departures from short voyages to other voyages. The goal of the last two methods is to reduce the cost by reducing the number of voyages sailed. The reason for this is that the distance between the depot and the installations often are longer than the distance between installations, resulting in a high start-up cost for a voyage. Therefore, fewer voyages often lead to cheaper schedules.

Changing installation departure patterns

The installation departure patterns are improved by iterating through all installations, looking for better patterns for each installation. This is done by removing all departures to the installation and looping through all feasible departure patterns for the installation, reinserting the departures on the least cost positions on the days given by the installation pattern. The pattern resulting in the lowest penalized cost is chosen.

Merging voyages

The goal of this procedure is to reduce the number of voyages sailed by attempting to merge two voyages departing on the same day into one voyage. This is done by looping through all days that have more than one voyage and calculating the penalized cost of merging all combinations of two voyages on that day. The merge which reduces the penalized cost the most, if any, is performed.

Reducing the number of voyages

The goal of this procedure is to reduce the number of voyages sailed by moving installation departures from days with short voyages to other days, and it is described in Algorithm 4. The procedure for changing installation patterns described above changes the pattern of one installation at a time and evaluates whether the penalized cost is reduced when the pattern is changed. The new procedure differs in that it changes the pattern of multiple installations and then evaluates whether the new individual is improving or not. This is done because sometimes the individual might be stuck in a local optimum where changing one installation pattern does not improve the individual, but the individual could be improved by changing multiple installation patterns. The procedure works as follows: let $n_t^{REM}(s)$ be the smallest number of installation departures that needs to be moved to another day in order to reduce the number of voyages sailed on day t , calculated as $n_t^{REM}(s) = |\mathcal{N}_t(s)| \bmod N^{MAX}$. The procedure selects the day t with lowest $n_t^{REM}(s)$ and changes the installation pattern of installations that have a departure on t . Let \mathcal{P}_{it}^{MOV} be the set of feasible patterns for installation i that do not contain day t . If this set is non-empty, i is *movable* and i is included in the set of movable installations with departures on day t , $\mathcal{N}_t^{MOV}(s)$. If $n_t^{REM}(s) > |\mathcal{N}_t^{MOV}(s)|$, it is not possible to move enough of the installations to remove a voyage, and the procedure stops. Else, the procedure continues by finding the (installation, pattern)-combination (i, p) that results in the lowest penalized cost, where $i \in \mathcal{N}_t^{MOV}(s), p \in \mathcal{P}_{it}^{MOV}$. After the pattern is changed, the installation is no longer part of $\mathcal{N}_t^{MOV}(s)$, hence $n_t^{REM}(s)$ has decreased by one. The procedure keeps changing the pattern of the installation with the lowest move cost until $n_t^{REM}(s) = 0$. If the penalized cost of the individual is reduced as a result of the procedure, the changes are saved, if not, the procedure does not change anything and individual s remains as before the procedure started.

5.7.3 Repair

If the resulting individual after education is feasible, it is referred to as *naturally feasible*. If it is infeasible, *repair* is performed with probability ρ^{REP} . Repair attempts to make infeasible individuals feasible by multiplying the penalty parameters by 10 and running the education again. If the individual still is infeasible, the penalty parameters are multiplied by 100 and the education is run again.

Algorithm 4 Reducing the number of voyages

STEP 1: SELECT DAY TO MOVE DEPARTURES FROM

- 1: **for** $t \in \mathcal{T}$ **do**
- 2: $n_t^{REM}(s) \leftarrow |\mathcal{N}_t(s)| \bmod N^{MAX}$
- 3: **end for**
- 4: $t \leftarrow \operatorname{argmin}_{t \in \mathcal{T}} n_t^{REM}(s)$

STEP 2: MOVE INSTALLATION DEPARTURES

- 5: $\phi^{BEF}(s) \leftarrow$ penalized cost of individual s before voyage reduction
- 6: **for** $i \in \mathcal{N}_t(s)$ **do**
- 7: $\mathcal{P}_{it}^{MOV} \leftarrow \{p \mid p \in \mathcal{P}_i^{CUST} \wedge t \notin p\}$ \triangleright Feasible patterns for i without t
- 8: **end for**
- 9: $\mathcal{N}_t^{MOV}(s) \leftarrow \{i \mid i \in \mathcal{N}_t(s) \wedge |\mathcal{P}_{it}^{MOV}| > 0\}$
- 10: **while** $0 < n_t^{REM}(s) \leq |\mathcal{N}_t^{MOV}(s)|$ **do**
- 11: Let $\psi(i, p)$ be the minimum penalized cost of i changing pattern to p
- 12: $(i, p) \leftarrow \operatorname{argmin}_{(i \in \mathcal{N}_t^{MOV}(s), p \in \mathcal{P}_{it}^{MOV})} \psi(i, p)$
- 13: $\pi_i(s) \leftarrow p$
- 14: $\mathcal{N}_t^{MOV}(s) \leftarrow \mathcal{N}_t^{MOV}(s) \setminus \{i\}$
- 15: $n_t^{REM}(s) \leftarrow n_t^{REM}(s) - 1$
- 16: **end while**

STEP 3: EVALUATE CHANGE IN PENALIZED COST

- 17: $\phi^{AFT}(s) \leftarrow$ penalized cost of individual s after voyage reduction
 - 18: **if** $\phi^{AFT}(s) < \phi^{BEF}(s)$ **then**
 - 19: Keep changes
 - 20: **else**
 - 21: Revert changes and keep input individual s
 - 22: **end if**
-

5.8 Population management

Three different population management mechanisms are used to improve the search. These are the *survivor selection*, *penalty parameter adjustment* and *diversification* mechanisms, all explained below. The aim of these mechanisms is to maintain a given balance between feasible and infeasible individuals, to maintain the diversity of the population and to keep high-quality individuals. They have in common that they affect the entire population simultaneously, contrary to crossover, education and repair, which only affect one individual at a time.

5.8.1 Survivor selection

The goal of survivor selection is to increase the quality of the population by discarding both bad quality individuals and *clones*, i.e. individuals that have a Hamming distance of zero to another individual. Survivor selection is performed on a subpopulation whenever the size of the subpopulation reaches its maximum limit, given by $\mu + \lambda$. The mechanism removes individuals until there are μ individuals left. The individuals are removed in order of decreasing biased fitness, first removing clones and when no clones are left, removing the individuals with highest biased fitness.

5.8.2 Penalty parameter adjustment

The *penalty parameters* are adjusted every 100 iterations, in order to attain a desired share of naturally feasible individuals. The last 100 individuals generated are considered, and each constraint is considered separately. Let ξ^D , ξ^Q and ξ^N be the share of naturally feasible individuals among the last 100 with respect to duration, capacity and number of installations, respectively. Let ξ^{REF} be the target ratio of naturally feasible individuals. The penalty parameters are adjusted as shown in Algorithm 5, where ζ^{UP} and ζ^{DOWN} are the adjustment factors for the penalties.

Algorithm 5 Penalty parameter adjustment

```
1: for  $p = D, Q, N$  do
2:   if  $\xi^p \leq \xi^{REF} - 0.05$  then
3:      $\omega^p = \omega^p \zeta^{UP}$ 
4:   else if  $\xi^p \geq \xi^{REF} + 0.05$  then
5:      $\omega^p = \omega^p \zeta^{DOWN}$ 
6:   end if
7: end for
```

5.8.3 Diversification

Diversification is used to prevent the algorithm from getting stuck in a local optimum. In case no improvement has been made to the best individual the last I^{DIV} iterations, diversification is performed. Diversification works by removing all but the best third of each subpopulation and generating $K^{DIV} \mu$ new individuals.

5.9 Optimizing the fleet size and mix

The algorithm described in Sections 5.1 - 5.8 minimizes the sailing cost using a given, *fixed* fleet. The objective of the SVPP is to minimize all costs related to the usage and operation of the PSVs, thus a solution to the SVPP also determines the fleet size and mix, as explained in Chapter 2. The method used to optimize the fleet is based on two additional assumptions. The first is that the time charter cost of every PSV is larger than the total sailing cost of all PSVs. The second is that the sum of the charter cost of the two cheapest PSVs is higher than the charter cost of the most expensive PSV. The first assumption makes sure that the reduction in sailing cost by increasing the fleet size will never offset the increase in charter cost. The second assumption makes sure that if a feasible fleet is found, the charter cost of that fleet is lower than the charter cost of all fleets containing more PSVs. The two assumptions combined imply that if there is a feasible fleet of size k , the size of the optimal fleet will be at most k . This can be utilized to optimize the fleet size and mix in two separate stages. Both assumptions are realistic and based on input from the decision makers at Statoil. The differences in charter cost between different PSVs are small as of April 2016, and the total sailing cost is around 50% of the charter cost of one PSV in the largest problem instances solved by the decision makers.

Algorithm 6 describes a method for solving the SVPP with optimal fleet size and mix, given the two assumptions. The fleet optimization algorithm consists of two steps: (1) Optimizing the fleet size and (2) Optimizing the fleet mix. The `FEASIBILITYCHECK(\mathcal{V})` procedure consists of running the HGSADC with the input fleet \mathcal{V} . The procedure stops when the algorithm finds a feasible individual or if no feasible individual is found within I^{NI} iterations. The fleet optimization algorithm starts by running the `FEASIBILITYCHECK` procedure with the entire available fleet as input. If a feasible individual is found, the PSV with lowest capacity is removed from the fleet, and `FEASIBILITYCHECK` is run with the resulting fleet as input. This is done until no feasible individual is found for the input fleet. Let k denote the size of the infeasible fleet. Even though no feasible individual was found for the input fleet, other fleets of size k can be feasible, since removing the PSV with the lowest capacity is a heuristic to reduce the fleet size. Therefore, the algorithm checks the feasibility of all fleets of size k . If a feasible fleet is found, the algorithm goes back to the loop of removing the PSV with the lowest capacity and checking the feasibility. If none of the fleets of size k are feasible, the size of the optimal fleet has to be at least $k + 1$. Since a feasible fleet of size $k + 1$ has been found, the two assumptions imply that the fleet size is at most $k + 1$. In other words, the optimal fleet size must be $k + 1$. The second step of the fleet optimization algorithm is to run HGSADC for all fleets of size $k + 1$. The total cost of a solution is defined as the sum of the charter and sailing costs. The solution with the lowest total cost is the optimal solution, with optimal schedule s^{OPT} and optimal fleet \mathcal{V}^{OPT} .

Note that in both Line 8 and 14 of the algorithm, all possible fleets of a certain size are tested. The number of fleets that are tested can be reduced by analyzing the input data. One improvement is to create an ordering of PSVs, as some PSVs may

have properties such that they will always be preferred over another. For instance, if PSV v has a lower capacity and sailing speed than all the other PSVs, it will never be preferred with regards to the feasibility of a fleet. If a fleet of size k that does not contain v is infeasible, a fleet of size k that contains v will also be infeasible. This means that the fleets containing v does not have to be tested in Line 8. If v also has a higher charter cost than all of the other PSVs, it does not need to be checked in Line 14. Another way to use the input data is to calculate the required capacity of the fleet and let `FEASIBILITYCHECK` check that the fleet has sufficient capacity before it runs `HGSADC`.

Homogeneous fleet

Even though the `HGSADC` considers a heterogeneous fleet, the algorithm can be used with a homogeneous fleet. The fleet optimization algorithm is the same, but the performance will improve significantly. The reason is that the for-loop described in Line 8 - 13 will only have one iteration, since there is only one fleet of size k . The second step of the fleet optimization algorithm is also a for-loop that will be reduced to one iteration. Note that the assumption about the relationship of the charter and sailing costs is still needed to determine the optimal fleet size. The second assumption, that the charter cost of any two PSVs exceeds the charter cost of each single PSV, will always hold for a homogeneous fleet, since the charter cost is the same for all PSVs.

5.10 Comparison with Vidal et al. (2012a)

The SVPP differs from the PVRP in two aspects: The fleet is heterogeneous and the routes can span multiple time periods. These two aspects require the `HGSADC` to be changed from that of Vidal et al. (2012a) and Vidal et al. (2014), who solve both the PVRP and the multi-depot PVRP (MDPVRP). The main difference is that the individual representation is changed: the *depot chromosome* used by Vidal et al. (2012a) is replaced by the PSV (vehicle) chromosome. Since these two chromosomes are different in structure, additional adaptations are necessary in the construction of the initial population, crossover and education. In addition, two new education operators tailored to solve the SVPP are introduced and the fleet optimization algorithm is adapted to handle a heterogeneous fleet. The following section goes through all the parts of the heuristic presented above and describes the similarities and differences with the heuristic presented by Vidal et al. (2012a) and Vidal et al. (2014). All references to the PVRP, MDPVRP and methods used for solving these refer to these two papers.

Overview

The overall structure of the `HGSADC` for the SVPP is the same as the `HGSADC` for the MDPVRP, and Algorithm 1 differs from the overall algorithm presented by Vidal et al. (2012a) only in notation and wording.

Algorithm 6 Optimizing the fleet size and mix

STEP 1: OPTIMIZE THE FLEET SIZE

```
1:  $feasibleSize \leftarrow \text{FEASIBILITYCHECK}(\mathcal{V})$ 
2: while  $feasibleSize$  do
3:    $v \leftarrow$  The PSV in  $\mathcal{V}$  with lowest capacity
4:    $\mathcal{V} \leftarrow \mathcal{V} \setminus \{v\}$ 
5:    $feasibleSize \leftarrow \text{FEASIBILITYCHECK}(\mathcal{V})$ 
6: end while
7:  $k \leftarrow |\mathcal{V}|$  ▷ The size of the infeasible fleet
8: for all fleets  $\mathcal{V}_k$  of size  $k$  do
9:    $feasibleSize \leftarrow \text{FEASIBILITYCHECK}(\mathcal{V}_k)$ 
10:  if  $feasibleSize$  then
11:    go to 2
12:  end if
13: end for
```

STEP 2: OPTIMIZE THE FLEET MIX

```
14: for all fleets  $\mathcal{V}_{k+1}$  of size  $k + 1$  do
15:    $s \leftarrow \text{HGSADC}(\mathcal{V}_{k+1})$ 
16:    $C^{TOT} \leftarrow \sum_{v \in \mathcal{V}_{k+1}} C_v^{TC} + \phi(s)$ 
17:   if  $C^{TOT} < C^{OPT}$  then
18:      $s^{OPT} \leftarrow s$ 
19:      $\mathcal{V}^{OPT} \leftarrow \mathcal{V}_{k+1}$ 
20:      $C^{OPT} \leftarrow C^{TOT}$ 
21:   end if
22: end for
```

Individual representation

In the MDPVRP, an individual is represented by three chromosomes: an installation pattern chromosome, a depot chromosome and a giant tour chromosome. For the SVPP, the depot chromosome is exchanged with a PSV pattern chromosome. The MDPVRP can be split into independent subproblems, one for each (day, depot)-combination, each subproblem being a VRP for a given depot, a set of customers to service and a number of available vehicles. The SVPP is split into one voyage (route) for each (day, PSV)-combination, each subproblem being a TSP for a given set of installations (customers) to service and a given PSV (vehicle). The PSV chromosome is necessary in order to make sure a PSV has returned to the depot before it departs on a new voyage. Since installations are assigned to a specific PSV, a heterogeneous fleet can be used. The PSV chromosome is necessary even for a homogeneous fleet, due to the possibility of multi-day voyages.

The fact that voyages in the SVPP can last multiple days creates interdependencies between the days, since the availability of a given PSV on a given day depends on what voyages it sails on other days. Vidal et al. (2014) lists interdependent services as an extension of the UHGS framework, but in fact the solution method for solving the PVRP takes care of one interdependency: the spread of departures. This interdependency is taken care of by the installation pattern representation, explained in Section 5.2. Each installation is assigned a departure pattern, a set of departure days that does not violate the constraints on spread of departures. Spread of departures is an important part of the SVPP as well, therefore installation patterns are used in the same way to solve the SVPP. Pattern representation is also used for the PSVs in order to solve the interdependencies created by multi-day voyages. A set of PSV departure patterns is generated, where each pattern is a set of departure days for one PSV. The upper limit on voyage duration depends on the pattern chosen for the PSV, thus preventing overlapping voyages.

The change from depot chromosome to PSV chromosome has consequences for the tour chromosome as well. The MDPVRP is solved by splitting the problem into a subproblem for each (day, depot)-combination and solving each subproblem separately. The set of installations is represented as a *giant tour*, which is a sequence of installations to be visited, in the order given by the sequence. In the SVPP, each subproblem is defined by a (day, PSV)-combination, meaning that each subproblem is a TSP rather than a VRP, since there is only one PSV available. The tour chromosome defines the set of installation visits and the order of visits for each (day, PSV)-combination. The name has been changed from giant tour chromosome to tour chromosome, because giant tour is used to signify that the sequence of installation visits can be split into multiple voyages, while in the case of SVPP, the sequence represents a single voyage.

It is important to note that a given installation chromosome and a given depot chromosome results in only one possible allocation of installations to (day, depot)-combinations, while there are multiple possible allocations for a given installation chromosome and a given PSV chromosome. This fact necessitates changes to the con-

struction of the initial population, crossover and education, which will be described below.

Search space

Vidal et al. (2012a) also allow infeasible individuals during the search. The additional constraints in the SVPP restrict the search space somewhat: the constraints on depot capacity, overlapping voyages and number of installations visited in a voyage are all new constraints that are added for the SVPP. The constraints on overlapping voyages are enforced by the duration constraints, since the upper limit on duration is determined by which PSV pattern the relevant PSV uses. The constraints on the number of installations visited in a voyage are handled in the same way as the constraints on the quantity delivered and duration, i.e. by allowing and penalizing violations.

Evaluation of individuals

The use of penalized cost, ranking of individuals and the inclusion of a diversity contribution rank are all used by Vidal et al. (2012a). The last term of the penalized cost (see Equation (5.3)), i.e. the penalty for violating the constraints on the number of installation visits in a voyage is added for the SVPP. The normalized Hamming distance is changed slightly due to the new individual representation. The difference is that instead of comparing the depot assignment of the installation, the set of PSVs that service the installation is compared. The calculation of biased fitness is identical, except for the calculation of ranks. In the solution method for the MDPVRP, the rank is calculated within each subpopulation, while in the solution method for the SVPP, the rank is calculated across subpopulations, resulting in a global rank instead of a local rank. Both local and global rank were tested, and global rank gave the best results.

Constructing the initial population

The construction heuristic is similar to the construction heuristic used for the MDPVRP, but with some modifications due to the changes in individual representation. The first step of the construction heuristic is identical, selecting a random departure pattern from the set of feasible departure patterns for each installation. The second step in the construction heuristic of the MDPVRP is to randomly assign each installation to a depot. Equivalently for the SVPP, each PSV is randomly assigned a PSV pattern. However, for the SVPP, not all combinations of installation pattern and PSV pattern are feasible. If there is an installation departure on a day without a PSV departure after all PSVs have been assigned a PSV pattern, the individual is infeasible and the second step is restarted. The same happens if the depot capacity constraint is violated. In order to reduce the number of restarts, the PSV pattern is randomly selected from the set of feasible departure patterns that contain at least one day with an installation departure and no PSV departure. This modification

greatly reduces the number of restarts. Note that the construction heuristic for the MDPVRP does not restart, since there are no infeasible combinations of departure patterns and depot assignments, and no constraints on the depot capacity.

The changes in individual representation also affects the third step of the construction heuristic. In the case of the MDPVRP, the departure pattern and depot assignment of an installation uniquely identifies which (day, depot)-combination each installation visit is assigned to. In the case of the SVPP, the installation departure and PSV departure chromosomes define which day all of the installation visits are assigned to, and which PSVs that depart on each day. If two PSVs depart on the same day, there are two possible allocations for all installations having a departure on that day. To solve this, each installation visit is randomly allocated to one of the available PSVs in step three of the construction heuristic.

Parent selection and crossover

Binary tournament is used as the parent selection strategy both for the MDPVRP and the SVPP. The crossover procedure used for the SVPP is strongly inspired by the *periodic crossover with insertions (PIX)* that is used by Vidal et al. (2012a), but has been modified to fit the new chromosomes and the additional constraints of the SVPP. All (day, depot)-combinations in PIX have been replaced by (day, PSV)-combinations, to reflect the changes in individual representation. Step two have been modified to ensure the validity of the PSV patterns and the depot capacity constraint. Step three is similar to step three in PIX, but with two important modifications. The first modification is that the SVPP does not require the *Split*-algorithm of Prins (2004), which is used to split the sequence of customers into separate routes for different vehicles. Since there is only one PSV available for each (day, PSV)-combination, there is no need to split the voyage. The second modification is that the crossover procedure for the SVPP will restart if it is not possible to complete step three. The option to restart is only needed for the SVPP, and will happen when an unserved installation visit cannot be inserted into any (day, PSV)-combination without violating the installation patterns, PSV patterns or depot capacity constraints. This is likely to happen if the individual has multiple PSV departing on some days, with no PSVs available to create new departures on other days. Since the cause of the infeasibility is the genetic material inherited in the first two steps, the entire crossover procedure is restarted.

Education

The voyage improvement procedure aims to improve the sequence of a given set of installation visits. This procedure is the same as the route improvement procedure used for the MDPVRP, except for the inter-route operators, which are not needed in the SVPP, since there is only one voyage per subproblem.

The pattern improvement procedure, on the other hand, depends heavily on the individual representation, and since the chromosomes are changed, the pattern im-

provement procedure is changed as well. The pattern improvement procedure for the MDPVRP iterates through all customers and changes the depot and departure pattern to the cheapest combination, in effect moving visits between (day, depot)-combinations. Similarly, the installation pattern improvement procedure of the SVPP attempts to move installation visits between (day, PSV)-combinations. For each installation i it finds the installation pattern p resulting in the lowest penalized cost and changes the installation pattern $\pi_i(s)$ to p .

Two additional pattern improvement procedures are introduced to solve the SVPP: one that merges voyages starting on the same day and one that attempts to reduce the number of voyages. Both of the procedures were developed based on initial tests using only the voyage improvement and installation pattern improvement procedures. The optimal individuals had fewer and longer voyages than the individuals found by the HGSADC. These two procedures aim to reduce the number of voyages by moving multiple installations at the same time, in contrast to the installation pattern improvement procedure, which only considers one installation at a time.

Population management

The population management consists of three mechanisms: penalty parameter adjustment, diversification and survivor selection. The SVPP has a constraint on the number of installations per voyage, which is not a part of the MDPVRP. Thus, an extra penalty parameter is added to penalize violations of this constraint, and subsequently added to the penalty parameter adjustment. Diversification and survivor selection are identical to those used for the MDPVRP.

Optimizing the fleet

The SVPP considers a heterogeneous fleet, while the MDPVRP assumes a homogeneous fleet. The procedure used for optimizing the fleet for the SVPP is based on some additional assumptions that are relevant for the SVPP, and is more complicated than the procedure presented by Vidal et al. (2012a). When a homogeneous fleet is used, it simplifies to the same procedure.

Chapter 6

Hybrid genetic search with adaptive diversity control for the MSVPP

The solution method described in the previous chapter assumes a single objective: to minimize total cost. In order to solve problems with multiple objectives, some adjustments and additions are needed. The extension from single-objective to multi-objective is done by adding the new objectives in the biased fitness function. In addition, some modifications to the way the best individuals are saved and the stopping criterion are needed. For persistence as an objective, two new education procedures are added to enhance the search. Section 6.1 describes the changes necessary to add additional objectives, Section 6.2 describe the new education procedures for persistence and Section 6.3 describes how the fleet can be optimized for multi-objective problems.

6.1 Adapting from single-objective to multi-objective

Three changes are necessary when adapting the HGSADC from solving single-objective problems to multi-objective problems. The first is that the output is a set of non-dominated individuals rather than a single individual. Individuals need to be compared based on *dominance* and stored in a *Pareto archive*. The second and third change is that the stopping criterion and the biased fitness function are changed.

6.1.1 Domination criterion and Pareto archive

When optimizing multiple objectives, the result is not one single optimal individual, but a front of Pareto-optimal individuals, constituting the Pareto front. When comparing the individuals, the concept of domination is used. An individual s_1 dominates another individual s_2 , written as $s_1 \prec s_2$, if and only if s_1 is at least as good as s_2 with

respect to all objectives, and is better than s_2 with respect to at least one objective. Mathematically, $s_1 \prec s_2$ if and only if

$$\forall k \in \mathcal{O} \text{ (} Objective_k(s_1) \preceq Objective_k(s_2) \text{)}, \quad (6.1)$$

$$\exists k \in \mathcal{O} \text{ (} Objective_k(s_1) \prec Objective_k(s_2) \text{)}, \quad (6.2)$$

where \mathcal{O} is the set of objectives to be optimized, $Objective_k(s_1) \preceq Objective_k(s_2)$ indicates that s_1 is better than or equal to s_2 with respect to objective k , and $Objective_k(s_1) \prec Objective_k(s_2)$ indicates that s_1 is strictly better than s_2 with respect to objective k . For example, consider the following individuals, where cost and number of changes are two objectives to be minimized:

s_1 : Cost = 600000 NOK, Number of changes = 4

s_2 : Cost = 600000 NOK, Number of changes = 2

s_3 : Cost = 500000 NOK, Number of changes = 4

Here, $s_2 \prec s_1$, since s_2 has the same cost as s_1 , but fewer changes (fewer is better). Also, $s_3 \prec s_1$, since s_3 has the same number of changes as s_1 , but lower cost. Neither s_2 nor s_3 dominates the other, since s_2 has fewer changes while s_3 has lower cost. Each is better than the other with respect to one objective. Since neither s_2 nor s_3 are dominated by any individual, they constitute the Pareto front.

During the execution of the HGSADC, the best individuals found so far are saved in an external population \mathcal{S}^{PARETO} , or *Pareto archive*, containing all individuals that are not dominated by any other individual found so far. Whenever a new individual is created, it is compared to the individuals in \mathcal{S}^{PARETO} . If the new individual is not dominated by any of the individuals in \mathcal{S}^{PARETO} , the new individual is added to \mathcal{S}^{PARETO} . If \mathcal{S}^{PARETO} already contains a individual with the same objective values as the new individual, the new individual is not added. If the new individual dominates any of the individuals in \mathcal{S}^{PARETO} , the dominated individuals are removed from \mathcal{S}^{PARETO} .

6.1.2 Stopping criterion

For the single-objective problem, the algorithm stops when an improving individual has not been found during the last I^{NI} iterations or the maximum time limit is reached. For the multi-objective problem, I^{NI} is the maximum number of iterations without adding a new individual to the Pareto archive, i.e. the maximum number of iterations without improving the Pareto front.

6.1.3 Modified biased fitness function

The biased fitness function is used to evaluate the quality of an individual, and determines which individuals survive during survivor selection. A low (lower is better) biased fitness also increases the probability that the individual will be selected as a parent. For the single-objective problem, the biased fitness of an individual s is the sum of two parts: The cost rank $Rank^C(s)$, and the diversity contribution rank, $Rank^D(s)$. The fact that the fitness is based on two different ranks means that the algorithm already uses multiple objectives when searching. Additional objectives can simply be added to the biased fitness function. Let $Rank^k(s)$ be the rank of individual s with respect to objective k . The individual with the best value of objective k will have $Rank^k(s) = 1$, and the individual with the worst value will have $Rank^k(s) = |\mathcal{S}|$. The general biased fitness function for multi-objective problems then becomes

$$BF(s) = \sum_{k \in \mathcal{O}} Rank^k(s) + (1 - \frac{n^{ELI}}{|\mathcal{S}|}) Rank^D(s), \quad (6.3)$$

where \mathcal{O} is the set of objectives to be optimized. Using this function one can select objectives to optimize as desired. Adding objectives to the biased fitness function increases the probability that individuals with good values for these objectives survive, and that their genetic material is passed on to new individuals. Results that will be described in Chapter 7 show that this simple modification is able to provide good Pareto fronts for the tested problems and objectives, even without education procedures improving the new objectives.

6.1.4 Fitness evaluation, diversity control and elitism

Konak et al. (2006) list three main components that separate different multi-objective genetic algorithms from each other: evaluation of individuals, diversity control and elitism. In the multi-objective HGSADC presented above, the evaluation of individuals is done by ranking each individual in terms of each objective value. Ranking is useful when the objective values have different ranges or orders of magnitude, since it avoids the problem of scaling objectives to the same level. The HGSADC handles diversity in two ways: by rewarding diverse individuals in the biased fitness function and by replacing most of the genetic material in the population when no improving individual has been found for a while, a process referred to as diversification. The degree of elitism is defined by the parameter n^{ELI} , which defines how strongly diversity contribution should be weighted in the biased fitness function. A high value of n^{ELI} gives low weight to diversity, meaning that the biased fitness of an individual depends mostly on the objective values of the individual. Elitism is also handled by the external Pareto archive, which saves the best found individuals during the whole run. The external archive is necessary because a non-dominated individual does not necessarily have the best biased fitness, meaning that it may be removed from the

population during the search.

6.2 Education for persistence

Much of the success of the HGSADC comes from the education procedures. In order to effectively find persistent schedules, two additional education procedures for persistence are implemented. The first one changes the installation pattern of one installation, the second one moves a PSV departure from one day to another, both with the goal of reducing the number of changes from the baseline schedule, i.e. increasing persistence. In order to find a Pareto front that has individuals in all parts of the front, individuals go through different combinations of cost-education and persistence-education. Each individual goes through one of four types of education, chosen randomly with equal probability. The four types are:

- Cost-education
- Persistence-education
- First cost-education, then persistence-education
- First persistence-education, then cost-education

Cost-education refers to the procedures described in Section 5.7, while persistence-education refers to the procedures described in this section. If an individual needs to be repaired, as described in Section 5.7.3, the same type of education is performed again, this time with higher penalty parameters.

Installation pattern improvement for persistence

The goal of this procedure is to improve the persistence of the individual by changing the departure pattern of one of the installations. The procedure is described in Algorithm 7. It starts by selecting a random installation and changing the departure pattern of the installation to the one given by the baseline pattern. If the selected installation is not in the baseline schedule, or already uses the baseline pattern, a new random installation is selected. For each day in the new pattern, the installation visit is inserted into the voyage and position in the voyage that results in the lowest penalized cost. As in the crossover algorithm, $\psi(i, v, t)$ is the minimum penalized cost for the insertion of installation i into the voyage sailed by PSV v on day t .

Moving PSV departures to other days

The feasibility of a individual depends heavily on the combination of the installation and PSV patterns: in order for an individual to be feasible, there needs to be enough PSVs departing on each day to service all the installations requiring a visit on that day, as specified by the installation patterns. The goal of this procedure is to ensure that there are enough PSVs departing on each day, by moving a PSV departure from one day to another, and moving installation visits along with the PSV. The installations

Algorithm 7 Installation pattern improvement for persistence

```
1:  $i \leftarrow$  installation randomly picked from  $\mathcal{N}^{CUST}$ 
2: if  $i \notin \mathcal{N}^B$  or  $\pi_i(s) = \pi_i^B$  then
3:   go to line 1
4: else
5:   Remove all visits to installation  $i$  in individual  $s$ 
6:    $\pi_i(s) \leftarrow \pi_i^B$ 
7:   for  $t \in \pi_i^B$  do
8:      $v = \operatorname{argmin}_{v \in \mathcal{V}} \psi(i, v, t)$ 
9:     Insert  $i$  into  $r_{vt}$  at least cost position
10:  end for
11: end if
```

that are moved are picked from all voyages departing on the given day, not only the voyage that is serviced by the PSV that is moved. The procedure is described in Algorithm 8 and consists of three steps: (1) deciding a day t^- to remove a PSV departure and a day t^+ to add a new PSV departure, (2) moving installation visits from day t^- to the new voyage on t^+ and (3) merging the remaining installation visits on t^- into the remaining voyages. The procedure begins by calculating the minimum number of PSVs that need to depart on each day in order to attain the baseline pattern and the number of PSVs departing in the current individual. It then selects a random pair of days (t^-, t^+) , where day t^- has more PSVs departures than required to attain zero changes from the baseline schedule and day t^+ has fewer. Line 9 checks that there is at least one PSV departure on t^- that can be moved to t^+ , i.e. that the previous voyage of the PSV is finished before t^+ . It also checks that there is sufficient depot capacity for another PSV to depart on t^+ . If no such PSV exists, a new pair of days is selected. The PSV with the most time from t^+ until its next departure is chosen as the PSV v to move, since it can sail the longest voyage, and a new voyage r_{v,t^+} is created for PSV v on day t^+ . In Step 2, randomly selected installations are removed from the set of installations with a departure on t^- , \mathcal{N}_{t^-} , and added to the end of r_{v,t^+} . This is done until the new voyage visits the maximum number of installations, or there are no more installation visits left on t^- . An installation i is moved only if it has no departure on t^+ already and if t^- is not in the baseline pattern for i . Finally, the remaining installations from the removed voyage are inserted into the least cost positions in the other voyages on that day.

6.3 Optimizing the fleet size and mix

Finding the optimal fleet size and mix becomes more complicated when including persistence and robustness as objectives. The fleet optimization procedure described in Section 5.10 is based on two assumptions: the first is that the time charter cost of every PSV is larger than the total sailing cost of all PSVs. The second is that the sum of the charter cost of the two cheapest PSVs is higher than the charter cost of

Algorithm 8 Moving PSV departures to improve persistence

STEP 1: SELECT DAYS TO MOVE DEPARTURE FROM AND TO

```
1: for  $t \in \mathcal{T}$  do
2:    $n_t^B \leftarrow$  Number of PSV departures required on day  $t$  to attain baseline pattern
3:    $n_t(s) \leftarrow$  Number of PSV departures on day  $t$  in current individual  $s$ 
4: end for
5:  $\mathcal{T}^- \leftarrow \{t \in \mathcal{T} \mid n_t(s) < n_t^B\}$   $\triangleright$  Days with too few PSVs departing
6:  $\mathcal{T}^+ \leftarrow \{t \in \mathcal{T} \mid n_t(s) > n_t^B\}$   $\triangleright$  Days with excess PSVs departing
7:  $\mathcal{T}^C \leftarrow \mathcal{T}^- \times \mathcal{T}^+$ 
8:  $(t^-, t^+) \leftarrow$  Random pair of days in  $\mathcal{T}^C$ 
9: if not feasible to move PSV departure from  $t^-$  to  $t^+$  then
10:    $\mathcal{T}^C \leftarrow \mathcal{T}^C \setminus \{(t^-, t^+)\}$ 
11:   if  $\mathcal{T}^C \neq \emptyset$  then
12:     go to 8
13:   else
14:     terminate procedure
15:   end if
16: end if
17:  $v \leftarrow$  PSV with highest number of days from  $t^+$  until next departure
```

STEP 2: MOVE INSTALLATION VISITS TO NEW DAY

```
18: Create new voyage  $r_{v,t^+}$ 
19:  $\mathcal{N}_{t^-} \leftarrow$  installations with departure on  $t^-$ 
20: while  $|r_{v,t^+}| < N^{MAX}$  and  $|\mathcal{N}_{t^-}| > 0$  do
21:    $i \leftarrow$  random installation in  $\mathcal{N}_{t^-}$ 
22:   if  $i$  has no departure on  $t^+$  already and  $t \notin \pi_i^b$  then
23:     Remove  $i$  from its voyage on  $t^-$ 
24:     Add  $i$  at the end of  $r_{v,t^+}$ 
25:   end if
26: end while
```

STEP 3: MERGE REMAINING INSTALLATION VISITS

```
27: for  $i \in \mathcal{N}_{t^-}$  do  $\triangleright$  Remaining installations in the removed voyage,  $r_{v,t^-}$ 
28:    $v = \operatorname{argmin}_{v \in \mathcal{V}_{t^-}} \psi(i, v, t^-)$ 
29:   Insert  $i$  into  $r_{v,t^-}$  at least cost position
30: end for
```

the most expensive PSV. When minimizing cost is the only objective, these assumptions imply that there is only one optimal fleet size. When maximizing persistence and maximizing robustness are added as objectives, larger fleets can provide Pareto-optimal individuals, meaning that one needs to solve the problem for multiple fleet sizes. Algorithm 9 describes a procedure for doing this. The minimum fleet size is found the same way as for the single-objective problem, but the HGSADC needs to be run on increasingly large fleets until the guaranteed maximum fleet size is found. All Pareto-optimal individuals from each run are stored together in a common Pareto archive, and the function MERGEPARETOFRONTS merges two Pareto-fronts into one and removes all individuals dominated by any individual in the combined front. The maximum fleet size is found when a individual in which *all objectives except cost have their optimal value* is found. Optimal value here refers to the best possible value for that objective. For example, when an individual with zero changes from the baseline and a robustness value of 1.0 is found, one can conclude that no fleet larger than the current one can dominate this individual. This is because an individual with optimal persistence and robustness can only be dominated by an individual with lower cost, and due to the assumptions about PSV charter cost, a larger fleet cannot provide an individual with lower cost.

Algorithm 9 Multi-objective HGSADC with variable fleet

STEP 1: FIND MINIMUM FLEET SIZE

```
1:  $feasibleSize \leftarrow \text{FEASIBILITYCHECK}(\mathcal{V})$ 
2: while  $feasibleSize$  do
3:    $v \leftarrow$  The PSV in  $\mathcal{V}$  with lowest capacity
4:    $\mathcal{V} \leftarrow \mathcal{V} \setminus \{v\}$ 
5:    $feasibleSize \leftarrow \text{FEASIBILITYCHECK}(\mathcal{V})$ 
6: end while
7:  $k \leftarrow |\mathcal{V}|$   $\triangleright$  The size of the infeasible fleet
8: for all fleets  $\mathcal{V}_k$  of size  $k$  do
9:    $feasibleSize \leftarrow \text{FEASIBILITYCHECK}(\mathcal{V}_k)$ 
10:  if  $feasibleSize$  then
11:    go to 2
12:  end if
13: end for
```

STEP 2: FIND NON-DOMINATED INDIVIDUALS FOR VARIOUS FLEETS

```
14:  $i \leftarrow 1$ 
15: while not  $MaxFleetSizeFound$  do
16:   for all fleets  $\mathcal{V}_{k+i}$  of size  $k + i$  do
17:      $\mathcal{S}^{PARETO}(\mathcal{V}_{k+i}) \leftarrow \text{HGSADC}(\mathcal{V}_{k+i})$   $\triangleright$  Pareto front found for fleet  $\mathcal{V}_{k+i}$ 
18:      $\mathcal{S}^{PARETO} \leftarrow \text{MERGEPARETOFRONTS}(\mathcal{S}^{PARETO}, \mathcal{S}^{PARETO}(\mathcal{V}_{k+i}))$ 
19:     if  $\exists s \in \mathcal{S}^{PARETO}$  where all objectives except cost have optimal value then
20:        $MaxFleetSizeFound \leftarrow \text{TRUE}$ 
21:     end if
22:   end for
23:    $i \leftarrow i + 1$ 
24: end while
25: return  $\mathcal{S}^{PARETO}$ 
```

Chapter 7

Computational Study

This chapter contains a computational study of the HGSADC for the SVPP, as described in Chapters 5 and 6. The results are compared with the results from the voyage-based model (VBM) of Borthen and Loennechen (2015), where such results are available. The complete VBM can be found in Appendix A, and a detailed description of the algorithms used to generate the voyages can found in Appendix B.

The HGSADC and the voyage generation procedure was implemented using Java 1.8 and run on a MacBook Pro with a 2.4 GHz Intel Core i5 processor and 8 GB of RAM, running OS X El Capitan. The voyage-based model, including the epsilon-constraint method, was implemented in the commercial optimization software Xpress IVE and run on a computer with a 3.4 GHz Intel Core i7 processor and 16 GB of RAM running Windows 7 Enterprise. Version 1.24.06 of the Xpress IVE was used, with version 3.8.0 of Xpress Mosel and version 27.01.02 of Xpress Optimizer.

Section 7.1 describes the test instances used and Section 7.2 presents the parameter values used for the HGSADC and how they were calibrated. Section 7.3 presents the results of solving the SVPP with minimum cost as the only objective. The MSVPP was solved for three different sets of objectives: (1) cost and persistence, (2) cost and robustness and (3) cost, persistence and robustness. Sections 7.4 - 7.6 present the results of these tests.

7.1 Test instances

25 different problem instances were tested, all of them based on the situation for Statoil in the North Sea as of April 2016. The instances contain one supply depot, the Mongstad supply depot, and between three and 27 offshore installations. All consider a planning period of one week, i.e. a weekly schedule that can be used multiple weeks in a row is to be generated.

The names of the test instances have the format X-Z, where X is the number of installations and Z is the total number of installation visits per week. For example,

the test instance named 12-40 has 12 installations and a total of 40 visits. Note that Halvorsen-Weare et al. (2012) and Borthen and Loennechen (2015) use the name format X-Y-Z, where X and Z are the same as in this thesis, and Y is the number installations with time windows. The name format is changed in this thesis, since only problem instances without time windows are considered.

All test instances consider a planning period of one week. The number of visits required by each installation ranges from one to five per week while the total number of required visits ranges from 10 to 80 per week. The service time at each installation ranges from one and a half to four hours, while the service time at the supply depot is eight hours. Each voyage is constrained to visit minimum one and maximum eight installations, and can last maximum three days. This includes the eight hour service time at the supply depot, so in practice, a voyage can last maximum 64 hours. There is no lower limit on the duration of a voyage and no constraints on the capacity of the PSVs in any of the test instances. This is based on the practice at Statoil, where the maximum limit of eight installations is found to be sufficient to ensure that the capacity is not exceeded. The depot is closed on Sundays in all of the test instances, based on input from Statoil.

A fleet of six PSVs have been used in all test instances, as it was provided by Statoil as the maximum number of available PSVs. All of the PSVs are identical, with a sailing speed of 10 knots and capacity of 600 m². In other words, all instances have been solved with a homogeneous fleet. Even though the solution method assumes a heterogeneous fleet, no changes are needed for a homogeneous fleet. The only part of the HGSADC that is affected is the fleet optimization algorithm, as explained in Section 5.9.

7.2 Calibration of parameters for the HGSADC

The performance of metaheuristics usually depends on the values of the input parameters, and this is especially true for evolutionary algorithms. Vidal et al. (2012a) perform extensive parameter testing for the HGSADC, using a metacalibration approach to find the optimal set of parameter values. They found that all of the optimal parameter values appeared to be independent of the problem class, except for the generation size, λ . Since most of the parameter values appeared to be independent of the problem class, and implementing a metacalibration approach is outside the scope of this thesis, a simpler approach to parameter testing is used in this thesis. The HGSADC for the SVPP has 18 parameters, all listed and explained in Section 7.2.1 along with their chosen values. Parameter testing was done to decide the values of the parameters that are expected to affect the performance the most. The results and choice of values are explained in Section 7.2.2. Education and repair are calibrated in Section 7.2.3.

Note that no parameter testing was done for the multi-objective HGSADC, in order to limit the time spent on parameter testing and because good results were obtained without a new calibration of the parameters. The values listed in Section 7.2.1 are

used for all instances solved by the multi-objective HGSADC. The only exception is that a repair rate of 0.5 is used for some of the larger instances, as it increased the number of feasible solutions. This indicates that a parameter test probably will improve the performance of the multi-objective HGSADC, and that no set of parameter values works best in all cases. The fact that the multi-objective HGSADC works well with the same parameter values do however indicate that the heuristic is quite robust with regards to parameter values.

7.2.1 The parameters of the HGSADC

The parameters of this implementation of the HGSADC are listed in Table 7.1. Most of the parameters affect the evaluation and education of a single individual, such as h , defining the neighborhood size in the voyage improvement procedure. μ , λ , I^{NI} , η^{DIV} , η^{ELI} and ξ^{REF} are parameters that affect the search of the algorithm and the number of iterations, and are therefore expected to affect the performance the most. Different values have been tested for these parameters. The values of the constraint penalty parameters ω^T , ω^Q and ω^N are also different from the values used by Vidal et al. (2012a), based on observations of the performance of the algorithm. The testing of all of the mentioned parameters is explained in Section 7.2.2. K^{INIT} , K^{DIV} , ζ^{UP} and ζ^{DOWN} are not treated as parameters by Vidal et al. (2012a), but rather as constant values. The values used by Vidal et al. (2012a) are therefore expected to work well in our implementation as well, and no testing was conducted for these parameters. The rest of the parameters are not expected to affect the performance much, and therefore the values that worked well for Vidal et al. (2012a) are used for these parameters without further testing. The only exception is that different values are tested for the repair rate in Section 7.2.3, since it is related to education, and a different value is chosen. The maximum running time is set to 3 600 seconds. This limit was found to allow the algorithm to converge and terminate due to maximum number of iterations without improvement in most cases, while still providing good results in these cases where it was reached. All parameter tests were performed with cost as the only objective.

7.2.2 Parameter calibration

Different values were tested for the parameters that were expected to affect the performance most. Problem instances 12-40, 20-68 and 27-80 were selected as *parameter test instances*. 12-40 was selected since it is the largest instance that the VBM is able to solve to optimality, and 27-80 was selected because it reflects the real problem faced by Statoil as of April 2016. 20-68 was selected to avoid selecting parameter values that only work well for 27-80. Since the HGSADC is non-deterministic, all parameter values were tested five times. The average running time is reported, as well as the average gap from the best known objective value. The average gap is calculated as the gap between the average objective value of five runs and the best known objective value. The objective value is the sum of the time charter cost and sailing

Parameter	Value	Description
μ	25	Minimum subpopulation size
λ	75	Generation size
I^{NI}	5 000	Max. number of iterations without improvement
η^{DIV}	0.1	Proportion of I^{NI} , such that $I^{DIV} = \eta^{DIV} I^{NI}$
η^{ELI}	0.4	Proportion of elite individuals, such that $n^{ELI} = \eta^{ELI} \times \mathcal{S} $
η^{CLO}	0.2	Proportion of individuals considered in diversity contribution, such that $n^{CLO} = \eta^{CLO} \mu$
ρ^{EDU}	1	Education rate
ρ^{REP}	0.5 / 0	Repair rate in construction heuristic / in normal iteration
K^{INIT}	4	Construction heuristic size factor
K^{DIV}	4	Diversification size factor
ξ^{REF}	0.6	Target ratio of feasible individuals
ω^Q	1 000	Capacity violation penalty
ω^T	1 000	Duration violation penalty
ω^N	1 000	Number of installations violation penalty
ζ^{UP}	1.2	Penalty adjustment factor, up
ζ^{DOWN}	0.85	Penalty adjustment factor, down
h	0.4	Neighbourhood size in voyage improvement
T^{MAXRUN}	3 600	Maximum running time (seconds)

Table 7.1: Parameters used in the HGSADC and their values after parameter testing.

cost. In Tables 7.2 - 7.5, the average running time and objective gap for different parameter settings are reported. When assessing the results of different parameter values, the objective gap is considered more important than the running time, since most of the running times are far from the maximum time limit. In addition, the results of instance 27-80 are considered more important than the results of the other instances, since it is the largest instance and the instance that is actually solved by Statoil. The values used by Vidal et al. (2012a) were used as the starting point for the parameter testing. The parameters that were tested were λ , η^{DIV} , ξ^{REF} , ω^Q , ω^T , ω^N and ρ^{REP} . All other parameters were set to the same values as those used by Vidal et al. (2012a).

Calibrating the target ratio of feasible individuals ξ^{REF}

Recall that ξ^{REF} is the target ratio of feasible individuals used in the penalty parameter adjustment mechanism, described in Section 5.8.2. A high value for ξ^{REF} will result in higher penalties and guide the HGSADC towards more feasible solutions.

Table 7.2 shows the average running time and objective gap for different values of ξ^{REF} . $\xi^{REF} = 0.6$ gives the lowest average objective gap for both 20-68 and 27-80, hence it is chosen as the value. The running times for $\xi^{REF} = 0.6$ are also good, with the lowest running time for 27-80 and low running times for the two other instances. The high objective gap for $\xi^{REF} = 0.2$ and 20-68 is caused by one run where the solution uses a fleet of five PSVs, where all of the other solutions use four. This happened because no feasible solution was found for a fleet of four PSVs within $I^{NI} = 5\ 000$. Since the charter cost is high, increasing the fleet size has a big impact on the objective gap. Also note that one of the other runs for $\xi^{REF} = 0.2$ and 20-68 did not find a feasible solution at all. This happened because no feasible solution was found within $I^{NI} = 5\ 000$ for a fleet of four PSVs, and when the fleet size was increased to five PSVs, no feasible solution was found within $I^{NI} = 5\ 000$. Note that the algorithm did find a feasible solution for a fleet of five PSVs before the fleet size was reduced to four, but not when the fleet size was increased back to five. An explanation of why these two rare cases happen for the same parameter value and instance, is that a low value for ξ^{REF} guides the search towards more infeasible solutions, and in some cases results in no feasible solution found within I^{NI} .

	$\xi^{REF} = 0.2$	$\xi^{REF} = 0.4$	$\xi^{REF} = 0.6$	$\xi^{REF} = 0.8$
Instance	Time (s) / Gap	Time (s) / Gap	Time (s) / Gap	Time (s) / Gap
12-40	99.1 / 0.00%	93.1 / 0.00%	85.9 / 0.00%	82.2 / 0.00%
20-68	733.6 / 5.65%*	779.7 / 0.11%	639.0 / 0.06%	578.6 / 0.08%
27-80	902.8 / 0.23%	778.9 / 0.22%	713.9 / 0.18%	949.0 / 0.28%

* The average of four runs

Table 7.2: The average running time and gap from the best known objective value for different values of ξ^{REF} . The average gap from the best known objective value is calculated as the gap between the average objective value and the lowest known objective value for the instance. All averages are calculated from five runs, except for $\xi^{REF} = 0.2$ for instance 20-68, where one of the runs did not find a feasible solution.

Calibrating the proportion of elite individuals η^{ELI}

η^{ELI} is the proportion of elite individuals and is used in the biased fitness measure, described in Section 5.4. A higher value of η^{ELI} increases the importance of the penalized cost rank and decreases the importance of the diversity rank, resulting in increased elitism in the HGSADC. Table 7.3 shows the average running time and objective gap for different values of η^{ELI} . $\eta^{ELI} = 0.4$ results in the lowest objective gaps and the highest running times, and since the objective gaps are considered more important than the running times, 0.4 is chosen as the value of η^{ELI} .

	$\eta^{ELI} = 0.2$	$\eta^{ELI} = 0.4$	$\eta^{ELI} = 0.6$	$\eta^{ELI} = 0.8$
Instance	Time (s) / Gap	Time (s) / Gap	Time (s) / Gap	Time (s) / Gap
12-40	141.3 / 0.00%	149.5 / 0.00%	140.5 / 0.00%	131.2 / 0.00%
20-68	947.3 / 0.11%	1 079.9 / 0.05%	948.5 / 0.05%	762.6 / 0.11%
27-80	829.9 / 0.38%	1 348.5 / 0.20%	1 045.7 / 0.25%	778.3 / 0.27%

Table 7.3: The average running time and gap from the best known objective value for different values of η^{ELI} . The average gap from the best known objective value is calculated as the gap between the average objective value and the lowest known objective value for the instance. All averages are calculated from five runs.

Calibrating the population size μ and the generation size λ

The optimal values for some of the parameters are likely to be correlated, meaning that the optimal value of one parameter depends on the values of the other parameters. This implies that each parameter cannot be tested separately. Some of the parameters are expected to be more correlated than others, and are therefore tested together. The minimum subpopulation size, μ , and the generation size, λ , are expected to have a high correlation, since both adjust the size of the population. The population size affects the diversity of the population and the search of the HGSADC. Table 7.4 shows the average running time and objective gap for different values of μ and λ . $\mu = 25$ and $\lambda = 75$ gives the lowest objective gap for both 20-68 and 27-80, and are therefore selected as values. The running times are not among the highest or lowest for the two instances. For 12-40, increasing the value of μ and λ increases the running time, but the objective gap of the larger instances is considered more important.

Calibrating the stopping criterion I^{NI} and the diversification criterion η^{DIV}

Recall that I^{NI} is the maximum number of iterations that the HGSADC can do without improvement. If I^{NI} iterations are done without finding any improving solution, the algorithm terminates. Figure 7-1 shows the objective value of the best found solution throughout a run of HGSADC for instance 13-44 with $I^{NI} = 5\,000$. The plot shows that in this particular run, I^{NI} could be reduced to $\sim 1\,000$ without affecting the objective value, since the largest number of iterations between finding two improving solutions is less than 1 000. Reducing I^{NI} would reduce the number of iterations by about 50% and thus also the running time. Since HGSADC is non-deterministic, reducing I^{NI} to 1 000 might affect the objective value for some runs. To be sure not to affect the objective value, I^{NI} should not be reduced to the minimum possible value found based on one run. The value of I^{NI} could be adapted to the problem size in order to find the most effective value for I^{NI} for a given problem instance. This would require the problem instance to be tested several times and the search to be analysed, like in Figure 7-1. This is a time consuming process that would have to be done for all instances. Selecting a constant value for I^{NI} is a simpler and more robust

Instance	μ	$\lambda = 25$	$\lambda = 50$	$\lambda = 75$	$\lambda = 100$
		Time (s) / Gap	Time (s) / Gap	Time (s) / Gap	Time (s) / Gap
12-40					
	15	77.0 / 0.00%	95.8 / 0.00%	95.9 / 0.00%	120.9 / 0.00%
	25	86.8 / 0.00%	100.6 / 0.00%	113.3 / 0.00%	148.8 / 0.00%
	35	95.4 / 0.00%	110.8 / 0.00%	130.9 / 0.00%	182.3 / 0.00%
20-68					
	15	567.2 / 0.13%	743.9 / 0.10%	708.2 / 0.06%	651.4 / 0.08%
	25	960.5 / 0.05%	724.8 / 0.05%	707.0 / 0.03%	814.0 / 0.06%
	35	780.7 / 0.05%	963.2 / 0.04%	845.6 / 0.05%	1089.1 / 0.05%
27-80					
	15	818.3 / 0.32%	678.6 / 0.20%	868.0 / 0.19%	852.3 / 0.20%
	25	746.5 / 0.22%	691.2 / 0.21%	867.0 / 0.15%	875.8 / 0.25%
	35	899.2 / 0.17%	1230.9 / 0.23%	781.6 / 0.26%	1655.6 / 0.20%

Table 7.4: The average running time and gap from the best known solution for different values of μ and λ . The average gap from the best known objective value is calculated as the gap between the average objective value and the lowest known objective value for the instance. All averages are calculated from five runs.

approach, where both the problem of affecting the objective value and the problem of extensive testing are avoided. The downside of this approach is that the running times are higher than necessary, especially for the smaller instances. The constant value approach is chosen in this thesis, due to the simplicity and robustness. Note that Vidal et al. (2012a) also use a constant value for I^{NI} .

If the best found solution has not improved for I^{DIV} , the diversification procedure is called. Since $I^{DIV} = \eta^{DIV} \times I^{NI}$, the values of both I^{NI} and η^{DIV} were tested together. Table 7.5 shows the results for different values of I^{NI} and η^{DIV} . I^{NI} has a big impact on the running time, which makes sense, since I^{NI} is a stopping criterion. The impact on the objective gap, however, is lower. Keep in mind that increasing I^{NI} only makes the algorithm search longer, so the expected objective value will be at least as good. However, the lowest objective gap is the same for the two values of I^{NI} , indicating that improving solutions are rarely found after $I^{NI} = 5\,000$. Since increasing I^{NI} to 10 000 does not seem to improve the solutions, 5 000 is chosen as the value. $\eta^{DIV} = 0.1$ gives the best objective gap for 27-80, and is therefore chosen as the value. The high objective gap for $\eta^{DIV} = 0.7$ and $I^{NI} = 10\,000$ for instance 27-80 is caused by one run where the solution uses a fleet of five PSVs, where all of the other solutions use four. This happened because no feasible solution was found for a fleet of four PSVs within $I^{NI} = 10\,000$. A poor initial population and a high value of η^{DIV} can explain why no feasible solution was found. A high value of η^{DIV}

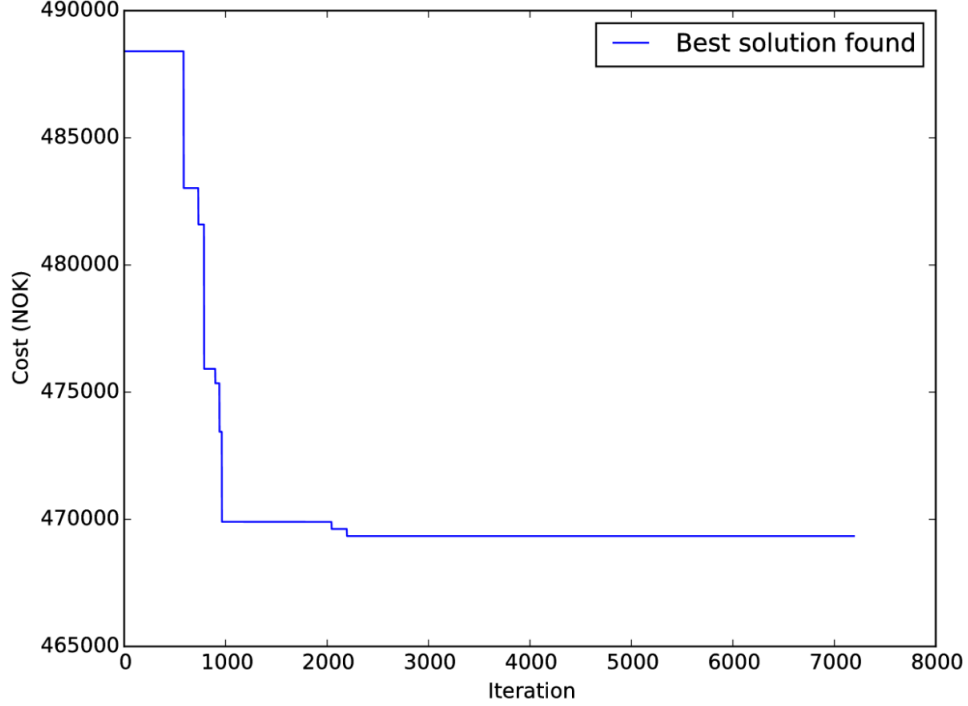


Figure 7-1: Plot of the objective value during the run of the HGSADC for problem instance 13-44 with $I^{NI} = 5\,000$.

leads to few diversifications and increases the impact of the initial population on the search of the HGSADC.

Figures 7-2 and 7-3 show the plot of the objective value of the best found solution throughout a run of HGSADC for instance 27-80 with $I^{NI} = 5\,000$ and $10\,000$, respectively. Figure 7-2 shows that in that particular run the longest period without improvement is around $4\,000$ iterations, and therefore $I^{NI} = 5\,000$ is a good choice. Figure 7-3 shows that increasing I^{NI} to $10\,000$ only affected the last $10\,000$ iterations of the previous improving solution. This supports the choice of $I^{NI} = 5\,000$ as a good trade-off between solution quality and running time. Note that I^{NI} will also affect the time used to optimize the fleet, since the algorithm will do I^{NI} iterations if no feasible solution is found. It is therefore an incentive to keep I^{NI} as low as possible, without affecting the objective value.

Calibrating the starting values of the penalty parameters ω^Q, ω^T and ω^N

The values of the penalty parameters ω^Q, ω^T and ω^N are adjusted dynamically by the algorithm, in order to find solutions on the border between feasible and infeasible solutions. Vidal et al. (2012a) use initial values $\omega^Q = \frac{\bar{c}}{\bar{q}}$ and $\omega^T = 1$, where \bar{c} and \bar{q} are the average distance between all pairs of customers and the average demand, respectively. Figures 7-4 and 7-5 show plots of subpopulation size and penalty parameter values for initial penalty values of 1 and $1\,000$, respectively. The plots show

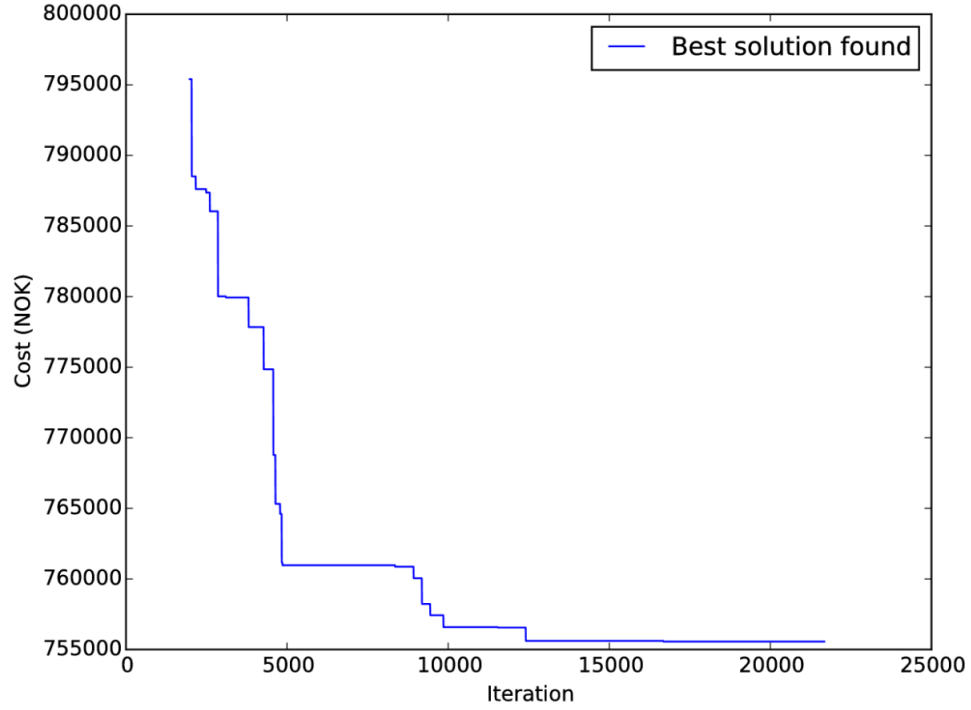


Figure 7-2: Plot of the objective value during the run of the HGSADC for problem instance 27-80 with $I^{NI} = 5\,000$.

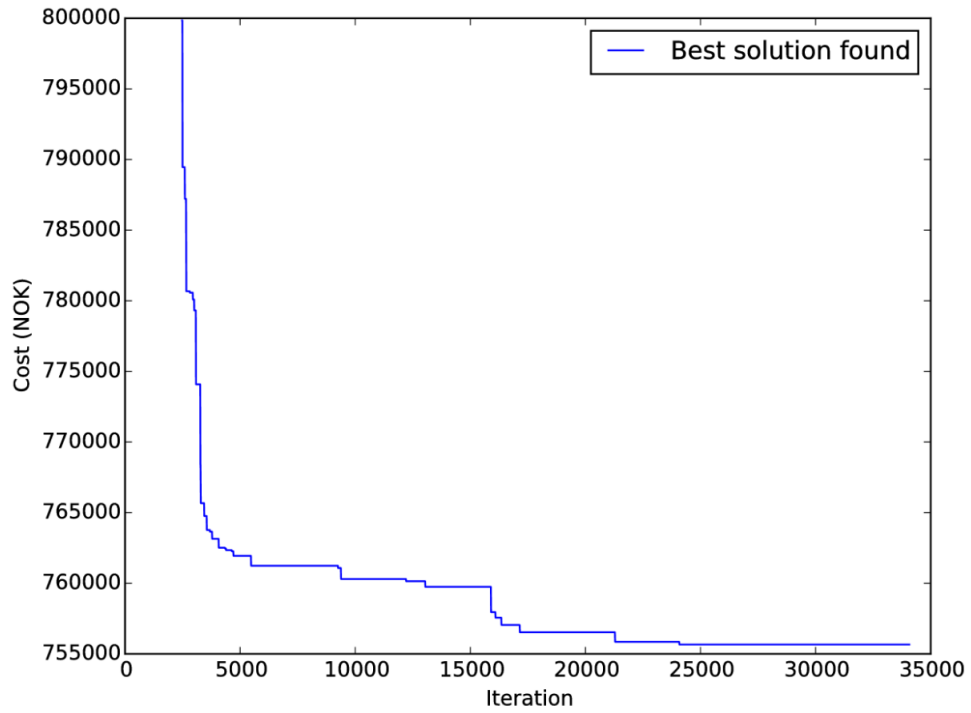


Figure 7-3: Plot of the objective value during the run of the HGSADC for problem instance 27-80 with $I^{NI} = 10\,000$.

Instance	η^{DIV}	$I^{NI} = 5\ 000$	$I^{NI} = 10\ 000$
		Time (s) / Gap	Time (s) / Gap
12-40			
	0.1	89.6 / 0.00%	152.1 / 0.00%
	0.4	81.9 / 0.00%	146.5 / 0.00%
	0.7	86.5 / 0.00%	147.3 / 0.00%
20-68			
	0.1	753.5 / 0.05%	1 077.7 / 0.07%
	0.4	781.5 / 0.05%	1 196.7 / 0.05%
	0.7	602.2 / 0.09%	1 024.8 / 0.05%
27-80			
	0.1	781.3 / 0.18%	1 229.8 / 0.21%
	0.4	1 023.2 / 0.21%	1 061.8 / 0.18%
	0.7	789.9 / 0.22%	1 257.8 / 4.60%

Table 7.5: The average running time and gap from the best known solution for different values of I^{NI} and η^{DIV} . The average gap from the best known objective value is calculated as the gap between the average objective value and the lowest known objective value for the instance. All averages are calculated from five runs.

that the first feasible solution is found much later when the initial penalties are 1 than when they are 1 000. The reason is that no feasible solutions are found when the penalties are low, and the penalty values are adjusted to an appropriate level in the first phase of the algorithm. Observing that no feasible solutions are found when the penalty values are less than 1 000, the time spent on this first phase can be reduced by increasing the initial penalty values. Initial tests showed that the penalty values stabilize at different levels for different problem instances and for each penalty, but the levels do not differ significantly, thus 1 000 is chosen as the initial value for all of the penalty parameters. This simple approach avoids calibrating the initial penalty values for all problem instances. Both of the plots show that ω^Q stabilize at very low levels. As explained in Section 7.1, the test instances have no constraint on the capacity, hence it is expected that ω^Q will decrease.

Concluding remarks on parameter calibration

Three of the tested parameters, η^{ELI} , μ and λ , have the same or nearly the same values as the values used by Vidal et al. (2012a), while the three other parameters, ξ^{REF} , I^{NI} and η^{DIV} , have different values. The different value of ξ^{REF} can be explained by the additional constraints of the SVPP, making it harder to find feasible solutions. Vidal et al. (2012a) do not perform parameter testing for I^{NI} and η^{DIV} , but adjust I^{NI} based on the instances and the desired running time, and keep $\eta^{DIV} = 0.4$. They often

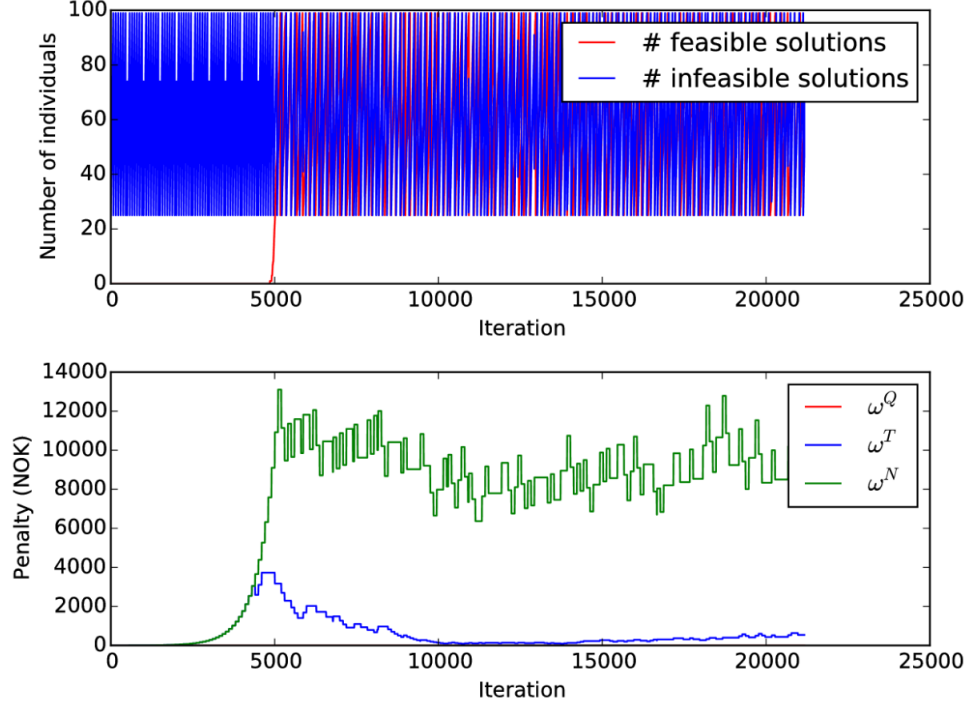


Figure 7-4: Plots showing the size of the subpopulations and the values of the penalty parameters during the run of the HGSADC for problem instance 27-80 with initial penalty values of 1.

use $I^{NI} = 10\,000$, but $I^{NI} = 5\,000$ proved sufficient for the test instances considered in this thesis. The reason might be that additional education procedures are used for the SVPP, which leads to finding better solutions faster. The lower value of η^{DIV} for the SVPP can be explained by the differences in the construction heuristic. The construction heuristic for the SVPP is less random than the one used by Vidal et al. (2012a), as explained in Section 5.5, which makes the generated solutions less diverse and might make it necessary to perform the diversification more often.

The parameter testing is made up of 390 runs of the HGSADC for different parameter settings and problem instances. In two of the runs, the fleet size of the solution is larger than the fleet size of the best known solution, and in one run, no feasible solution was found. This happened for the lowest tested value of ξ^{REF} and the highest tested value of η^{DIV} . Increasing the fleet size has a big effect on the objective value, and parameter values that can result in increased fleet size should be avoided. The HGSADC is stable with regards to fleet size for the other tested parameter settings, finding the same optimal fleet for all runs. For these parameter settings, the largest deviation in average objective gap is 0.10% and 0.23% for instance 20-68 and 27-80, respectively. None of the parameter settings affected the objective gap for instance 12-40. Note that a deviation in objective gap is expected for the larger instances, since the HGSADC is non-deterministic and different objective values are found in different runs even with the same parameter settings. In other words, the HGSADC

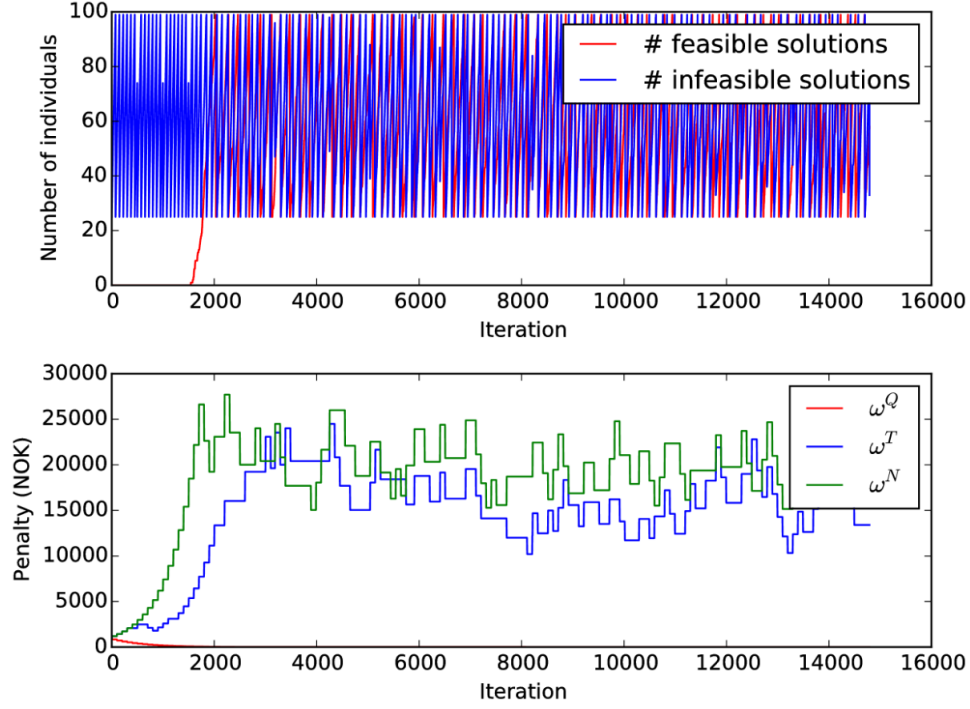


Figure 7-5: Plots showing the size of the subpopulations and the values of the penalty parameters during the run of the HGSADC for problem instance 27-80 with initial penalty values of 1 000.

seems stable with regards to different parameter settings, except for settings that makes it hard to find feasible solutions.

7.2.3 Calibration of education and repair

This section describes the results of different levels of education and repair in the HGSADC. The importance of education is discussed first, followed by a test of different values for the repair rate, ρ^{REP} .

Calibration of education

Vidal et al. (2012a) present a sensitivity analysis of the main components of the HGSADC, and report that the education component affects the objective value the most. The education component in this implementation of the HGSADC consists of voyage improvement and pattern improvement, where pattern improvement consists of three procedures. One of the pattern improvement procedures improves the individual by considering the installation departure patterns, while the two other procedures improves the individual by attempting to increase the length of voyages and reduce the number of voyages. The first procedure is similar to the pattern improvement procedure proposed by Vidal et al. (2012a), while the two other procedures are developed for the SVPP. Table 7.6 shows the effect of different levels education. No

education means that no parts of the education component is used, neither voyage improvement nor pattern improvement procedures. Simple education includes the same education as used by Vidal et al. (2012a), i.e. the voyage improvement procedure and only the first pattern improvement procedure. Full education includes all of the described education procedures. The objective gap is a lot higher for no education than the other two levels of education. Both instance 12-40 and 20-68 have runs that do not return a feasible solution, and the objective gaps of the solutions with feasible solutions are high. The high gaps are caused by an increased fleet size for some of the runs. The difference between simple and full education is smaller. Simple education leads to a lower running time than full education, while full education has a lower objective gap. The difference in objective gap between simple and full education is small, but since the objective gap is considered more important than the running time, full education is chosen. The results of the HGSADC with no education clearly illustrates the importance of the education component.

	No education	Simple education	Full education
Instance	Time (s) / Gap	Time (s) / Gap	Time (s) / Gap
12-40	84.2 / 14.17%*	116.6 / 0.00%	129.1 / 0.00%
20-68	140.8 / 1.06%**	835.1 / 0.03%	1 125.1 / 0.02%
27-80	184.7 / 23.98%	681.8 / 0.24%	824.4 / 0.18%

* The average of three runs

** The average of four runs

Table 7.6: The average running time and gap from the best known solution for no, simple and full education. The average gap from the best known solution is calculated as the gap between the average objective value and the lowest known sailing cost for the instance. All averages are calculated from five runs, unless otherwise specified.

Calibration of repair rate ρ^{REP}

Table 7.7 shows the results for different values of the repair rate ρ^{REP} . Note that the repair rate in the construction heuristic has a value of 0.5 in all of the tests. in order to limit the scope of the testing. During preliminary testing, the repair rate seemed to have a big impact on instance 19-65, so the instance was included in the parameter test. The reason might be that 19-65 is the largest instance that is serviced by a fleet of three PSVs, making it more vulnerable to disturbances in the search. The results in Table 7.7 show that the effect of the repair rate is largest for 19-65. Increasing the repair rate increases the running time for all instances, which makes sense, since it increases the number of calls of the education procedures. The objective gap, on the other hand, does not seem to reduce with increased repair rate. Even though $\rho^{REP} = 1$ gives the lowest objective gap for 27-80, the objective gap

for 19-65 is very high for both $\rho^{REP} = 0.5$ and $\rho^{REP} = 1$. One of the runs does not find a feasible solution and one finds a solution with increased fleet size, which causes high objective gaps. This is the case for both of the values. Since $\rho^{REP} = 0$ provides low objective gaps for all of the instances, it is chosen as value. This means that the repair mechanism is only used in the construction heuristic.

	$\rho^{REP} = 0$	$\rho^{REP} = 0.5$	$\rho^{REP} = 1$
Instance	Time (s) / Gap	Time (s) / Gap	Time (s) / Gap
12-40	86.2 / 0.00%	104.6 / 0.00%	127.8 / 0.00%
19-65	428.1 / 0.07%	534.0 / 7.18%*	671.9 / 7.16%*
20-68	573.2 / 0.06%	694.1 / 0.06%	907.2 / 0.07%
27-80	727.5 / 0.17%	759.0 / 0.27%	1 270.0 / 0.12%

* The average of four runs

Table 7.7: The average running time and gap from the best known objective value for different values of ρ^{REP} . The average gap from the best known objective value is calculated as the gap between the average objective value and the lowest known objective value for the instance. All averages are calculated from five runs, unless otherwise specified.

7.3 Results - SVPP

This section discusses the performance of the HGSADC with a variable fleet and minimum cost as the only objective. First, the performance of the HGSADC is compared to the performance of the VBM presented by Borthen and Loennechen (2015), then the performance of the HGSADC on larger problem instances is described and discussed. The VBM was implemented in the commercial optimization software Xpress IVE and the candidate voyages used as input to the VBM were generated using dynamic programming. The VBM can be found in Appendix A and a detailed description of the dynamic programming algorithm for generating the candidate voyages can be found in Appendix B.

7.3.1 Comparison with the VBM

The results from running the HGSADC are compared with the results from solving the voyage-based model (VBM). Table 7.8 compares the results of the HGSADC with the results of solving the VBM with a commercial solver. The running times reported on the VBM include the time used to generate voyages and solving the voyage-based model. The HGSADC is a non-deterministic algorithm, meaning that it can behave differently on different runs for the same input. The reported results are therefore an

average of ten runs. The commercial solver was able to solve all problem instances up to 12-40 to optimality. For problem instance 13-44 and 14-48, the commercial solver was stopped after 10 000 seconds, indicated by a running time of >10 000 in the table. The commercial solver was not able to find a feasible solution to bigger problem instances than 14-48, so the HGSADC is only compared to instances 3-10 to 14-48. The high optimality gaps of problem instances 13-44 and 14-48 are probably caused by the increase in fleet size from two to three PSVs. The commercial solver is probably not able to prove that there does not exist a solution with two PSVs within the time limit of 10 000 seconds, resulting in low dual bounds and high optimality gaps. The HGSADC finds the optimal solution for all of the problem instances that the commercial solver is able to solve to optimality, even though the running time is higher for the small instances. For instance 13-44 and 14-48, the HGSADC finds a better solution than the commercial solver in less than 3% of the time used by the solver. To further compare the HGSADC and the VBM, the VBM was solved with a fixed fleet of three PSVs. The commercial solver was able to solve 13-44 and 14-48 to optimality with a fixed fleet of three PSVs, and the optimal solutions were the same as the solutions found by the HGSADC for a variable fleet. The results show that the HGSADC finds at least as good solutions as the commercial solver in all runs, and that it outperforms the VBM for the instances that the solver is unable to solve to optimality. The running time of the VBM increases rapidly when increasing problem size from 12 to 13 installations, while the running time of the HGSADC only increases slightly.

7.3.2 Results for all instances

Table 7.9 shows the number of PSVs, the number of voyages and the costs of the solution found by the HGSADC for all problem instances. Each problem instance was solved ten times and the results are the average of the ten solutions. The solutions to each problem instance have the same number of PSVs and voyages, e.g. all of the solutions to problem instance 3-10 have two PSVs and four voyages. A stable fleet size is important for the stability of the HGSADC with regards to cost, since the charter cost is a lot higher than the sailing cost. Figure 7-6 shows a plot of the average sailing cost found by the HGSADC and the number of visits for all problem instances. For problem instances with up to 40 visits, exact methods are able to prove that the HGSADC finds the optimal solution. The plot shows that the sailing cost seems to increase linearly with the number of visits for all instances, indicating that the HGSADC finds good solutions to larger problem instances as well. Note that a new installation is added when the number of visits increases, and that the variation in how much the sailing cost increases is partly caused by the location of the added installation. The sailing cost is plotted instead of the total cost to make it easier to see the trend. The fleet size increases in a natural way for larger problem instances, and since the HGSADC finds the same fleet size for all runs, the charter cost of the larger instances is also assumed to be good. The sailing cost is plotted against the number of visits instead of the number of installations, since the number of visits affects the cost more.

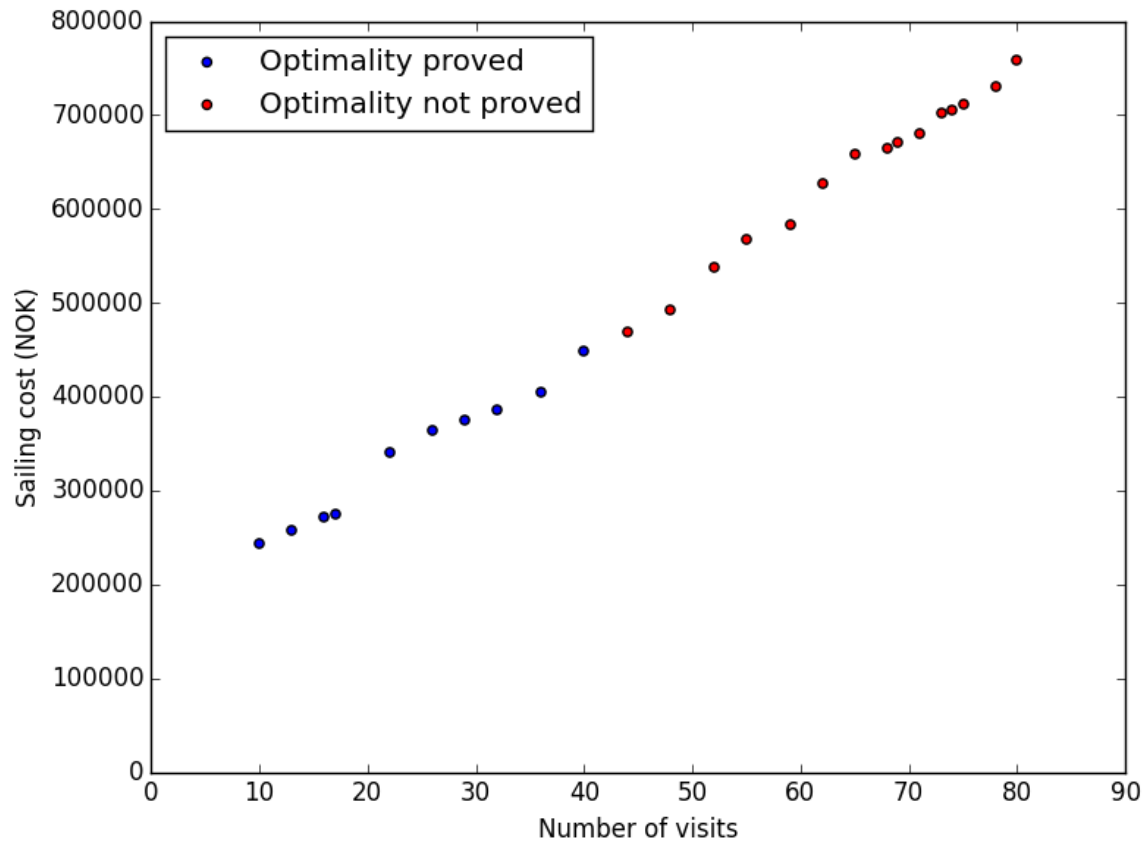


Figure 7-6: Plot of the average sailing cost found by the HGSADC versus the number of visits for all problem instances. Exact methods are only able to solve instances with up to 40 visits to optimality.

Problem instance	VBM		HGSADC	
	Optimality	Time (s)	Gap from	Time (s)
	gap		VBM	
3-10	0.0%	0.5	0.00%	17.2
4-13	0.0%	0.5	0.00%	26.5
5-16	0.0%	0.6	0.00%	60.4
6-17	0.0%	0.7	0.00%	60.6
7-22	0.0%	1.6	0.00%	66.4
8-26	0.0%	3.8	0.00%	74.1
9-29	0.0%	12.9	0.00%	96.4
10-32	0.0%	69.9	0.00%	93.5
11-36	0.0%	197.9	0.00%	118.4
12-40	0.0%	43.1	0.00%	129.6
13-44	22.3%	>10 000	-0.03%	238.5
14-48	19.4%	>10 000	-0.60%	166.9

Table 7.8: Results from solving the SVPP with minimum cost as the only objective. The running times on the VBM include both generating voyages and solving the voyage-based model. Optimality gaps are calculated from the best lower bound reported by the VBM. The results for the HGSADC are the average of ten runs.

Problem instance	PSVs (#)	Voy. (#)	Total cost (NOK)	Charter cost (NOK)	Sailing cost (NOK)
3-10	2	4	3 043 629	2 800 000	243 629
4-13	2	4	3 058 070	2 800 000	258 070
5-16	2	4	3 071 043	2 800 000	271 043
6-17	2	4	3 074 204	2 800 000	274 204
7-22	2	5	3 141 015	2 800 000	341 015
8-26	2	5	3 163 891	2 800 000	363 891
9-29	2	5	3 174 896	2 800 000	374 896
10-32	2	5	3 185 820	2 800 000	385 820
11-36	2	5	3 205 145	2 800 000	405 145
12-40	2	6	3 249 252	2 800 000	449 252
13-44	3	6	4 669 339	4 200 000	469 339
14-48	3	6	4 692 392	4 200 000	492 392
15-52	3	7	4 737 684	4 200 000	537 684
16-55	3	8	4 767 888	4 200 000	567 888
17-59	3	8	4 782 530	4 200 000	582 530
18-62	3	8	4 826 968	4 200 000	626 968
19-65	3	9	4 859 061	4 200 000	659 061
20-68	4	9	6 265 321	5 600 000	665 321
21-69	4	9	6 271 154	5 600 000	671 154
22-71	4	9	6 280 756	5 600 000	680 756
23-73	4	10	6 302 165	5 600 000	702 165
24-74	4	10	6 305 637	5 600 000	705 637
25-75	4	10	6 311 997	5 600 000	711 997
26-78	4	10	6 330 891	5 600 000	730 891
27-80	4	11	6 359 404	5 600 000	759 404

Table 7.9: Results from using the HGSADC to solve the SVPP with minimum cost as the only objective. The total cost is the sum of charter and sailing cost. All results are the average of ten runs. # PSVs and # Voy. are the number of PSVs and voyages used in the solution, respectively.

7.3.3 Stability of the HGSADC

Table 7.10 shows the average running time of each problem instance, calculated from ten runs. In addition, the coefficient of variation (CV) of the total cost, sailing cost and running time are shown for each problem instance. The CV is calculated as the standard deviation divided by the mean. The sailing cost has a CV of 0 for all problem instances up to 14-48, indicating that the HGSADC is very stable for the smaller instances. As explained, the HGSADC finds the optimal solution to all problem instances up to 12-40. Tests not presented here showed that problem instance 15-52 is the largest problem instance the commercial solver is able to solve with a fixed fleet, and the optimal solution to 15-52 with a fixed fleet is the same as the best solution found by the HGSADC. In other words, the HGSADC finds the optimal solution for the problem instances up to 14-48 with a CV of 0, and seems to find good solutions for the larger problem instances with small variations. The average CV of the total and sailing cost are 0.02% and 0.19%, respectively, so the HGSADC is considered very stable with regards to solution quality. The average CV of the running time is 16.37%, and the highest standard deviation of running time is about five minutes. The HGSADC is therefore considered stable with regards to solution time for the tested problem instances. The running times of the HGSADC and VBM are plotted for all problem instances in Figure 7-7. The plot clearly illustrates how much lower the running time of the HGSADC is for the larger problem instances. The plot indicates that the running time of the HGSADC increases linearly with the number of installations. Based on the results, it seems that the HGSADC finds good solutions to all problem instances, the running time increases linearly, and it seems stable with regards to both solution quality and running time.

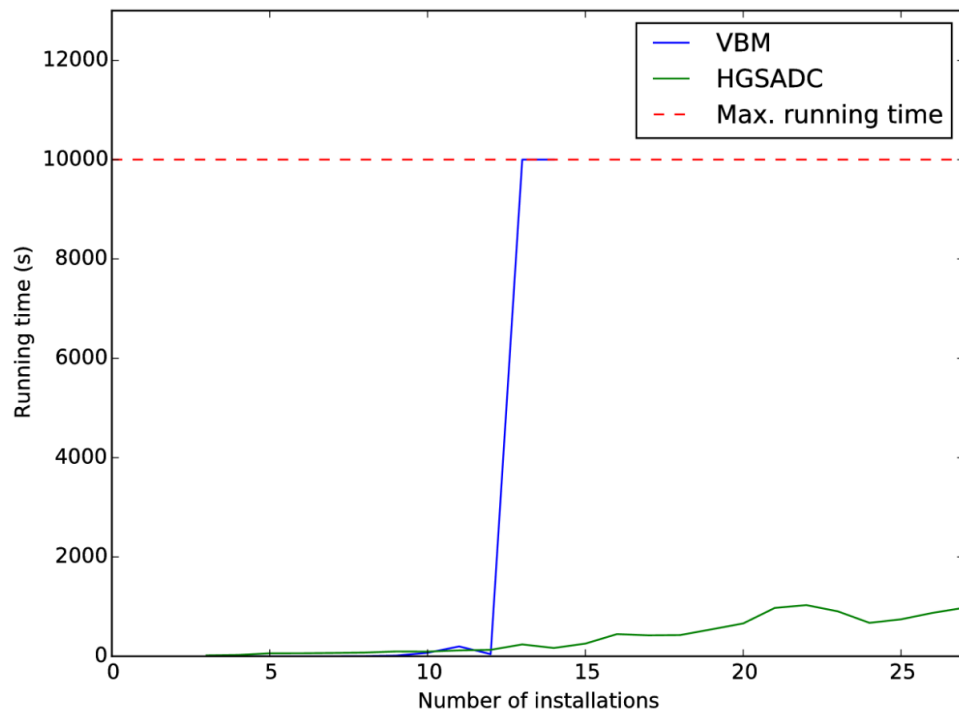


Figure 7-7: Plots of the running time of the HGSADC and the VBM for all of the test instances. The maximum running time was 10 000 seconds for the VBM and 3 600 seconds for the HGSADC. The VBM was not able to solve instances with more than 14 installations, hence no running time is reported for these instances.

Problem instance	Time (s)	Coefficient of variation		
		Total cost	Sailing cost	Time
3-10	17.2	0.00%	0.00%	8.18%
4-13	26.5	0.00%	0.00%	7.22%
5-16	60.4	0.00%	0.00%	7.37%
6-17	60.6	0.00%	0.00%	0.91%
7-22	66.4	0.00%	0.00%	0.58%
8-26	74.1	0.00%	0.00%	2.38%
9-29	96.4	0.00%	0.00%	6.85%
10-32	93.5	0.00%	0.00%	5.46%
11-36	118.4	0.00%	0.00%	14.11%
12-40	129.6	0.00%	0.00%	7.60%
13-44	238.5	0.00%	0.00%	4.49%
14-48	166.9	0.00%	0.00%	11.86%
15-52	254.9	0.00%	0.01%	24.51%
16-55	445.7	0.02%	0.18%	27.48%
17-59	421.2	0.01%	0.12%	28.12%
18-62	426.4	0.16%	1.22%	16.66%
19-65	542.3	0.04%	0.31%	16.67%
20-68	661.0	0.04%	0.34%	17.06%
21-69	973.5	0.04%	0.37%	26.09%
22-71	1 030.0	0.03%	0.28%	22.73%
23-73	904.1	0.02%	0.21%	31.88%
24-74	672.7	0.03%	0.24%	35.55%
25-75	743.9	0.03%	0.26%	21.54%
26-78	873.8	0.05%	0.43%	36.38%
27-80	976.6	0.08%	0.68%	27.59%
Average	N/A	0.02%	0.19%	16.37%

Table 7.10: Results from using the HGSADC to solve the SVPP with minimum cost as the only objective. All numbers are calculated from the results of ten runs.

7.4 Results - MSVPP with cost and persistence

This section presents the results from solving the multi-objective SVPP with two objectives: Minimizing cost and minimizing the number of changes from a given baseline solution. The latter is equivalent to maximizing persistence and the number of changes is measured as described in Section 4.5. Four different *variation cases* have been tested. These cases represent typical scenarios where the model is re-optimized, and are based on the experience of the decision makers at Statoil. For the multi-objective problem, the fleet is considered fixed, based on the preferences of Statoil. The reason for this preference is that the time charter cost of one PSV is significantly higher than the total sailing cost of a schedule in all of the test instances. Thus, the increased cost of chartering additional PSVs is so large that it is not desirable, even if it means that one needs to have changes in the schedule. Fixing the fleet results in smaller Pareto fronts than if the fleet is variable, since the number of PSVs limits the solution space. In order to remove all changes and acquire the entire Pareto front, it may be necessary to charter additional PSVs, as explained in Section 6.3.

7.4.1 Results compared with optimal fronts

The problem instances with 13 or fewer installations can be solved to optimality with a fixed fleet using the ϵ -constraint method, as presented by Borthen and Loennechen (2015). This section compares the Pareto fronts found by the HGSADC (*heuristic fronts*) with the *optimal fronts* found by the ϵ -constraint method for four different variation cases. For each case, the fronts from three runs of the HGSADC are shown, since the HGSADC is non-deterministic. The horizontal axis shows the number of changes from the baseline solution, while the vertical axis shows the total cost of the solution, expressed as percentage increase from the lowest known total cost of the problem instance. Note that when counting the number of changes, only the installations that are both in the current instance and the baseline solution are considered. This means, for example, that when adding a new installation, the departures to this installation do not count towards the number of changes.

Case 1: Add new installations

Figure 7-8 shows the Pareto fronts for problem instance 12-40 using the optimal solution to instance 9-29 as the baseline. In other words, three new installations have been added. The optimal front consists of four solutions. The number of changes can be reduced from 22 to 16 by increasing the cost by less than 0.35%. The HGSADC finds three of the optimal solutions in all runs, the last solution is only slightly more expensive than the optimal one for two of the runs. The fronts also show that it is not possible to reduce the number of changes below 16 without increasing the fleet size.

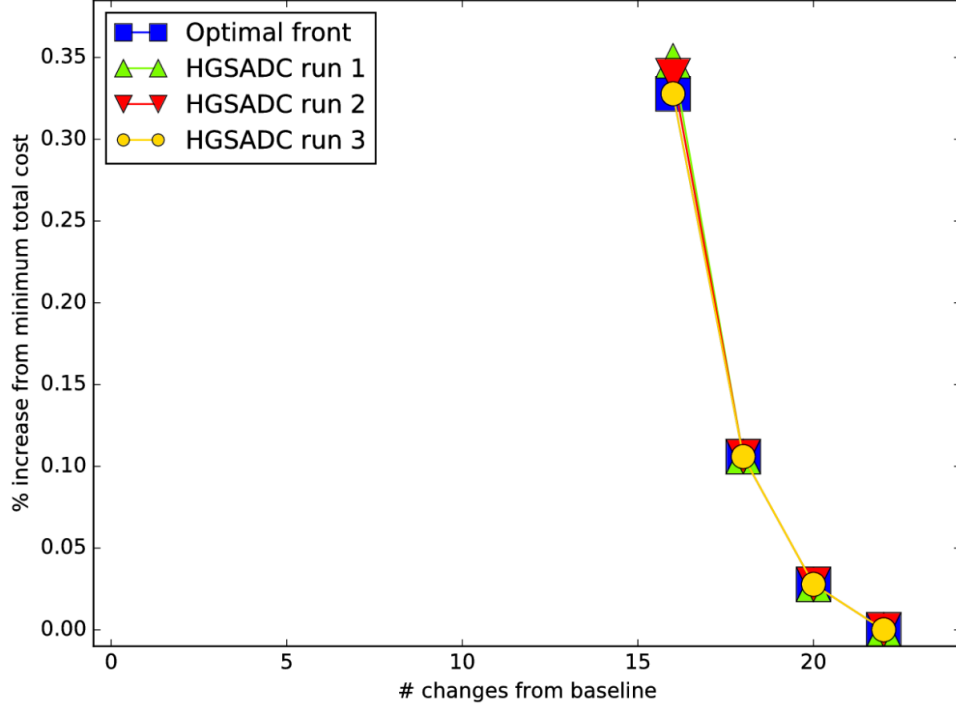


Figure 7-8: Optimal and heuristic fronts for case 1 - add new installations. Problem instance 12-40 is solved and problem instance 9-29 is used as baseline.

Case 2: Load reduction

Figure 7-9 shows the Pareto fronts for problem instance 12-40, modified by reducing the demand and service time at each installation by 25%. The optimal solution to 12-40 with normal demand and service time is used as the baseline. The optimal front consists of five points, with the number of changes ranging from zero to 18. For a cost increase of less than 0.7%, the number of changes can be reduced to zero. The HGSADC found four of the optimal solutions in all runs, but failed to find the last solution on the optimal front in all runs.

Case 3: Shutdown installations

Figure 7-10 shows the Pareto fronts for problem instance 12-40, using the optimal solution to instance 15-52 as the baseline. In other words, three installations have been shut down and removed from the problem. The optimal front consists of five solutions, with number of changes between 30 and 40. For a cost increase of less than 0.3%, the number of changes can be reduced from 40 to 30. The HGSADC finds the optimal front in two out of three runs, the last one has a slightly higher cost for one of the solutions, shown by the green triangle in the figure. The reason for the high number of changes and that no solution with zero changes can be found is that problem instance 12-40 requires two PSVs and instance 15-52 requires three PSVs.

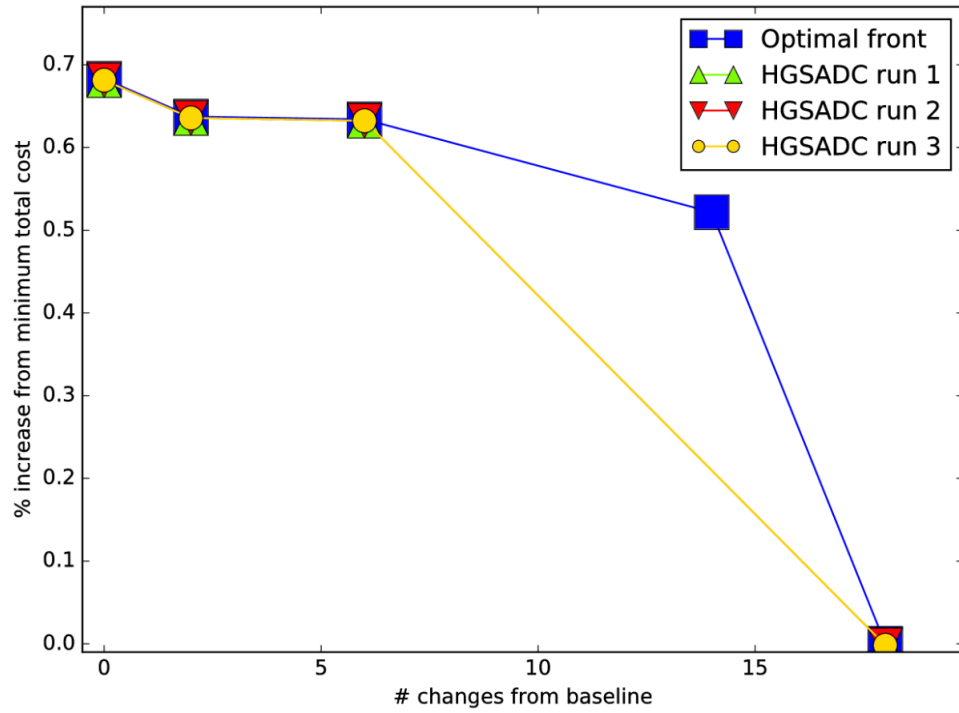


Figure 7-9: Optimal and heuristic fronts for case 2 - load reduction. Problem instance 12-40 with demand and service time reduced by 25% is solved and problem instance 12-40 with normal demand and lay time is used as baseline.

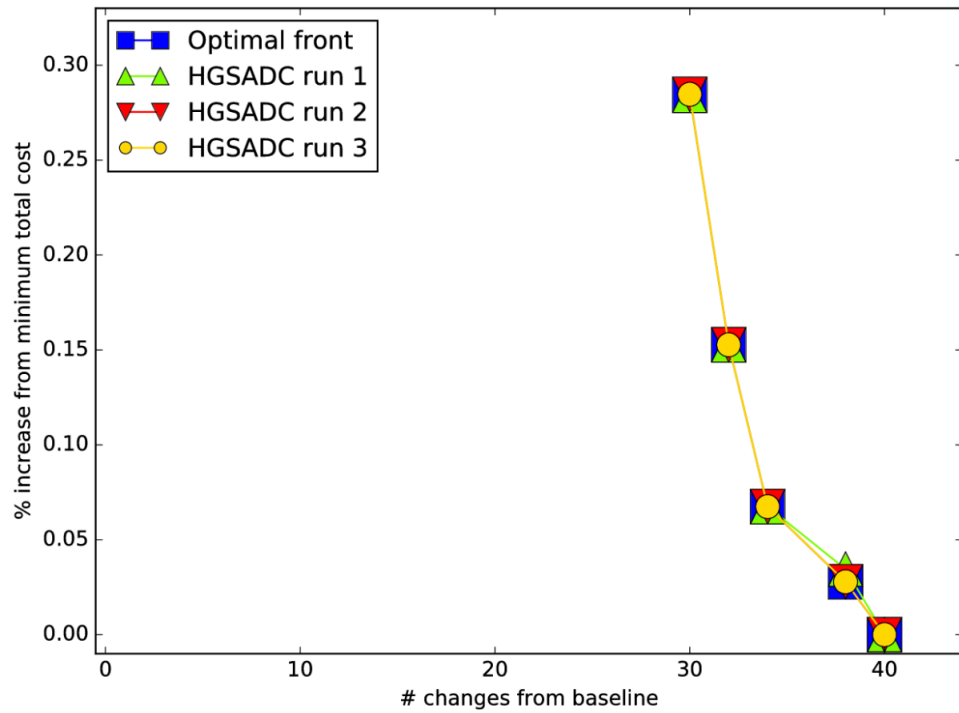


Figure 7-10: Optimal and heuristic fronts for case 3 - shutdown installations. Problem instance 12-40 is solved and problem instance 15-52 is used as baseline.

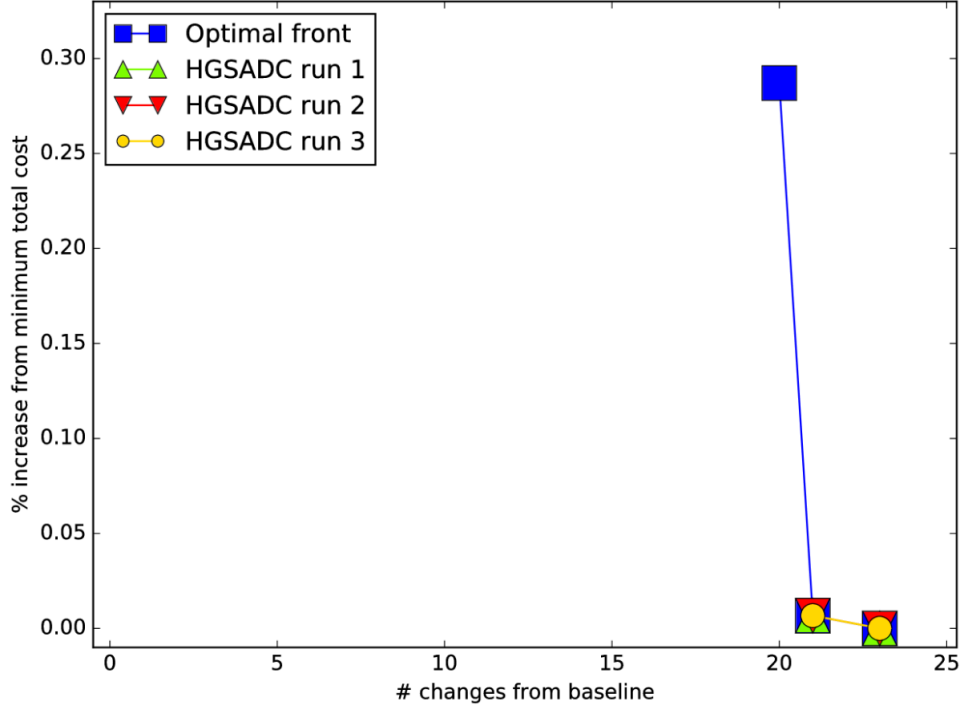


Figure 7-11: Optimal and heuristic fronts for case 4 - add visits to installations. Problem instance 11-39 is solved and problem instance 11-36 is used as baseline.

Case 4: Add visits to installations

Figure 7-11 shows the Pareto fronts for problem instance 11-39, which is the same as instance 11-36, with the addition of one extra visit to three of the installations. The optimal solution to 11-36 is used as the baseline. The optimal front consists of three solutions, with the number of changes ranging from 20 to 23, and a maximum cost increase of $\sim 0.3\%$. All three heuristic fronts are equal and consist of two points, both lying on the optimal front. All heuristic runs fail to find the solution with the minimum number of changes possible. It is impossible to have less than 20 changes without increasing the fleet size.

Running times

Table 7.11 shows the time used to solve the cases described above, both using the ϵ -constraint method and the HGSADC. The HGSADC is slower than the ϵ -constraint method for all of the cases. As for the single-objective problems, the exact method using the VBM is faster than the HGSADC for the small problems. The running times for the HGSADC increase when adding persistence as an objective, the average running time of the HGSADC for the single-objective instance of 12-40 is 98 seconds, compared to 387 seconds for the multi-objective instance. The coefficient of variation of the running times is 3.8% for the single-objective instance of 12-40, while Table 7.11 shows much higher CVs. The reason for the increase in running time and CV is that the search space has been expanded to two dimensions, such that the search

is looking for multiple optimal solutions, using different fitness measures. A higher CV is expected, due to the already mentioned reasons, but also because the single-objective running times are the average of five runs, while the multi-objective ones are the average of three runs.

		ϵ -constraint	HGSADC	
Problem instance	Variation case	Time (s)	Average time (s)	CV of time
12-40	Case 1 - Add installations	142	583	17%
12-40	Case 2 - Load reduction	138	290	15%
12-40	Case 3 - Shutdown installations	183	439	6%
11-39	Case 4 - Add visits	49	236	17%
Avg.		128	387	14%

Table 7.11: Running times for medium-sized instances of the MSVPP with cost and persistence as objectives. The running times for the HGSADC are the average of three runs.

Overall performance on medium-sized instances

In the results above, the HGSADC consistently found optimal or close to optimal fronts for all of the four cases given by Statoil. From these results, it seems that the method finds high-quality fronts in reasonable time, and that the results are stable, i.e. that the method finds approximately the same front for each run. Thus, the HGSADC achieves the first goal when generating best-known Pareto fronts: be as close to the true Pareto front as possible (Konak et al., 2006). The HGSADC is slower than the ϵ -constraint method, which guarantees optimal fronts. This means that for the medium-sized problem instances, the ϵ -constraint method is both faster and provides better solutions. The ϵ -constraint method is, however, not able to solve problem instances with more than 13 installations. Since the ϵ -constraint method is solving the VBM multiple times, the running time is expected to increase faster than for the VBM. The HGSADC is therefore expected to outperform the ϵ -constraint method on all instances larger than 12-40. The next section shows the results of applying the HGSADC on real-size problem instances.

7.4.2 Results for real-size problem instances

This section shows the non-dominated fronts found by running the HGSADC on larger problem instances. The instances represent the situation for Statoil in the North Sea as of April 2016, and the variation cases are typical changes that Statoil face in their operations. As of April 2016, Statoil has 27 installations that are serviced by the same depot, hence the optimal solution to 27-80 is used as baseline for all variation cases.

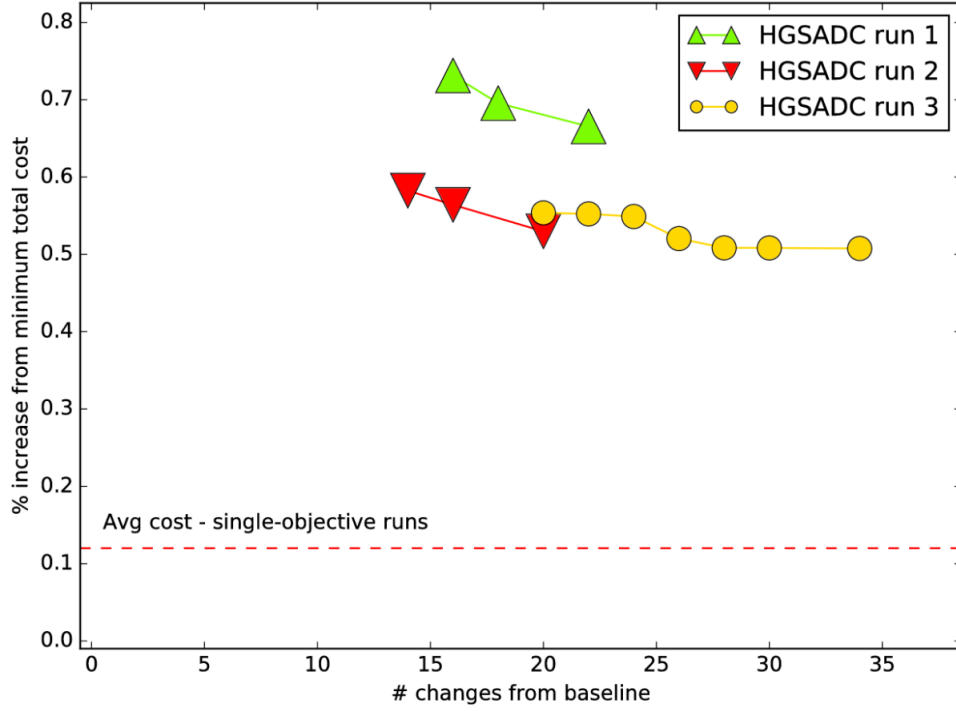


Figure 7-12: Heuristic fronts for case 1 - add new installations. Problem instance 30-87 is solved and problem instance 27-80 is used as baseline.

For these problem instances, no optimal fronts can be obtained due to the problem size, thus only the heuristic fronts are shown. The percentage increase in total cost is calculated from the minimum total cost found by the single-objective HGSADC for each variation case. The dashed line shows the average cost of the solutions found by the single-objective HGSADC.

Case 1: Add new installations

Figure 7-12 shows the Pareto fronts for problem instance 30-87 using a solution to 27-80 as the baseline. In other words, three new installations have been added. Two of the fronts have three solutions each, while the last one consists of seven solutions. The cheapest solution found costs about 0.5% more than the minimum cost solution found by the single-objective HGSADC, but the fronts are relatively flat, meaning that the number of changes can be reduced cheaply.

Case 2: Load reduction

Figure 7-13 shows the Pareto fronts for problem instance 27-80, modified by reducing the demand and lay time at each installation by 25%. A solution to 27-80 with normal demand and service times is used as the baseline. All three fronts found are exactly the same, and consist of two solutions. The number of changes can be reduced from six to zero for a cost increase of about 0.01 percentage points. However, the solutions

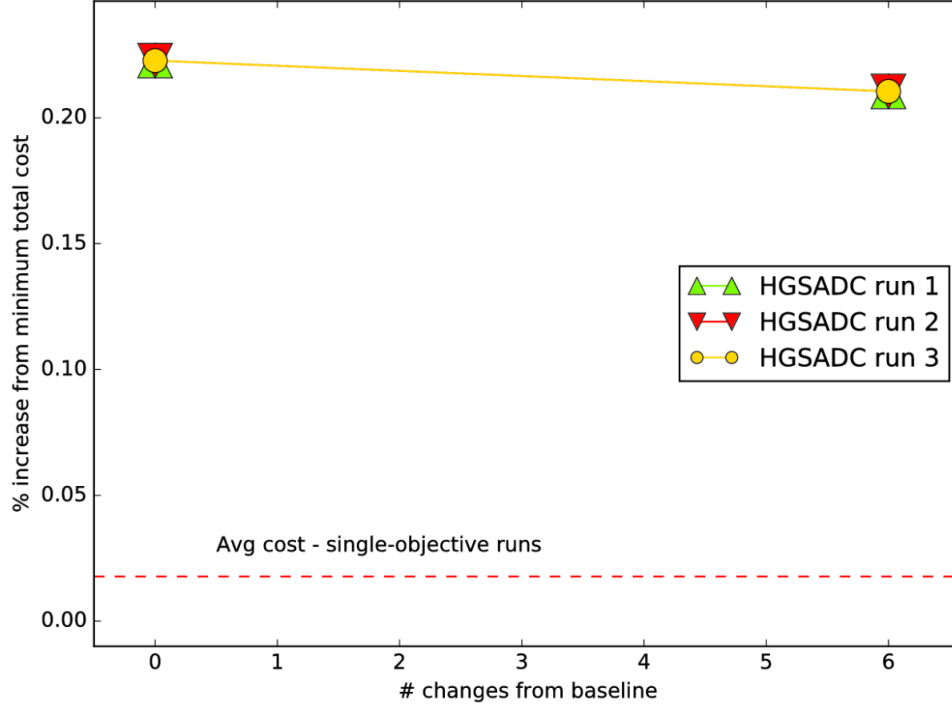


Figure 7-13: Heuristic fronts for case 2 - load reduction. Problem instance 27-80 with demand and lay time reduced by 25% is solved and problem instance 27-80 with normal demand and lay time is used as baseline.

have higher cost than the cheapest solution found by the single-objective HGSADC. The solution with zero changes is $\sim 0.21\%$ more expensive than the minimum cost solution.

Case 3: Shutdown installations

Figure 7-14 shows the Pareto fronts for problem instance 22-71, using the solution to instance 27-80 as the baseline. Five installations have been shut down and removed from the problem. The yellow front shows that the number of changes can be reduced from eight to zero for a cost increase of less than 0.6%. All solutions are slightly more expensive than the average solution found using single-objective HGSADC. None of the fronts dominates any of the others, so in practice, the best possible front is found by combining the best solutions from all three fronts.

Case 4: Add visits to installations

Figure 7-15 shows the Pareto fronts for problem instance 27-83, which is the same as instance 27-80, but with the addition of one extra visit to three of the installations. A solution to 27-80 is used as the baseline. The fronts consist of two and three solutions, all with less than 0.4% increase in cost from the minimum cost solution found by the single-objective HGSADC. The number of changes can be reduced for a cost increase

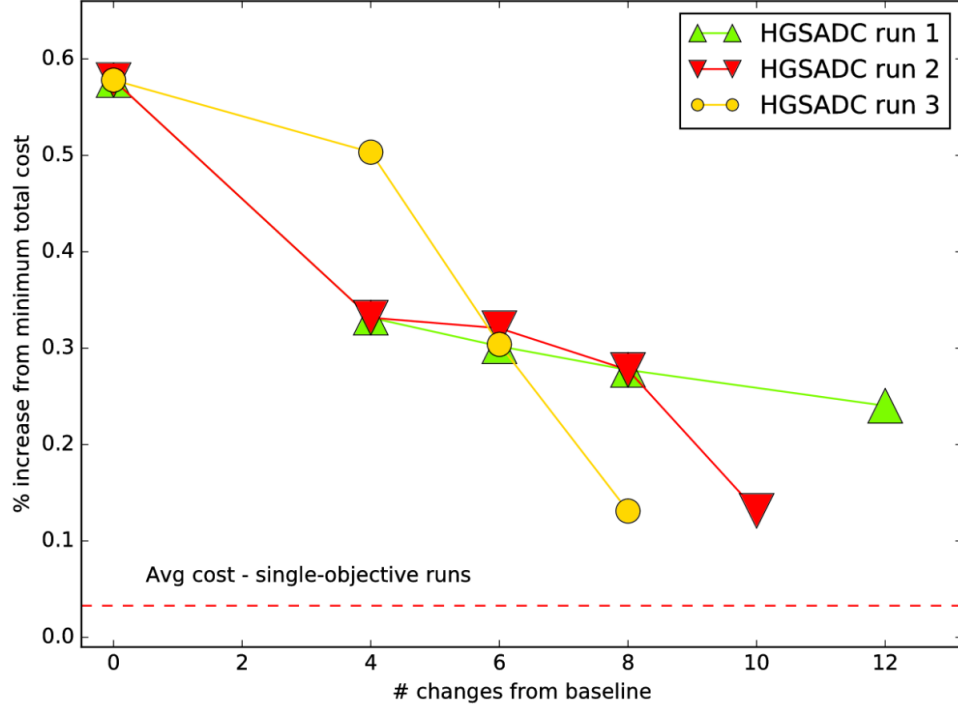


Figure 7-14: Heuristic fronts for case 3 - shutdown installations. Problem instance 22-71 is solved and problem instance 27-80 is used as baseline.

of around 0.1% for all of the fronts. The front shown as green triangles dominates the two others, and the cheapest solution is as cheap as the average solution found using single-objective HGSADC.

Running times

The running times for solving the real-size problems with cost and persistence as objectives are presented in Table 7.12. The rightmost column shows the running times for solving the same problems with cost as the only objective. On average, the running time increases by $\sim 22\%$ from single-objective to multi-objective. The running times are not directly comparable between single- and multi-objective, since the runs do not find the same solutions. Nonetheless, the data show that the running time does not increase significantly when including an additional objective. The results illustrate the usefulness of using a genetic algorithm when solving for multiple objectives: since the HGSADC already works with a population of solutions, it is a small step to change the search from finding one high-quality solution to finding a set of high-quality solutions. This is not the case for methods like the ϵ -constraint, which requires the problem to be solved one time for each solution on the Pareto front, meaning that the running time increases rapidly as objectives are added and the optimal solution space grows. Recall that the algorithm terminates after 5 000 iterations without adding a new solution to the non-dominated front. The maximum time limit was 3 600 seconds, a time limit that was reached by two of the runs

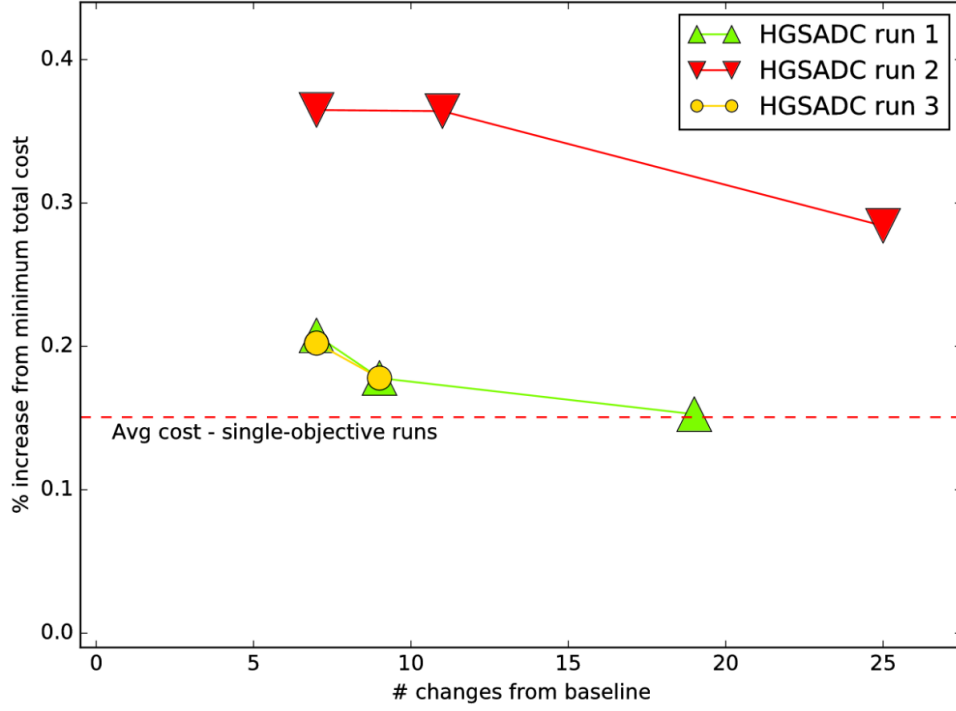


Figure 7-15: Heuristic fronts for case 4 - add visits to installations. Problem instance 27-83 is solved and problem instance 27-80 is used as baseline.

on problem instance 30-87. In these cases, the results may have improved if the maximum time limit was increased. Overall, the running times are reasonable, given the frequency at which the problem needs to be solved, which is less than once a week.

Overall performance on real-size instances

The HGSADC is less stable for the real-size instances than the medium-sized ones, but it is still considered stable. Even though the fronts found in different runs for the same problem instance and variation case differ both in number of solutions and quality of solutions, they are similar with regards to both the number of changes and cost. No fronts have a difference of more than 0.3% cost for the same number of changes. The cost of the cheapest solution of the front is less than 0.7% higher than the lowest cost found by the single-objective heuristic for all cases and runs, indicating that the inclusion of another objective does impair the quality of the solutions much. The algorithm was able to find a solution with zero changes in only two of the cases, probably due to the fixed fleet. As explained, one often needs to charter more PSVs to reduce the number of changes to zero. The HGSADC is stable for the real-size instances and provides Pareto fronts with sufficiently many and diverse solutions to give the decision makers both flexibility in choosing a schedule and insight into the trade-off between cost and persistence. In all of the cases studied, the number of changes could be reduced significantly for an increase in cost of less than 0.7%. The

Problem instance	Variation case	Cost and persistence		Cost
		Average time (s)	CV of time	Average time(s)
30-87	Case 1 - Add installations	3136	26%	1814
27-80	Case 2 - Load reduction	757	3%	1536
22-71	Case 3 - Shutdown installations	1672	3%	1068
27-83	Case 4 - Add visits	972	16%	939
Avg.		1634	12%	1339

Table 7.12: Running times for real-size instances of the MSVPP with cost and persistence as objectives using the HGSADC and using only cost as the objective. All numbers are calculated from the results of three runs, all solved with a fixed fleet.

fact that the heuristic returns different Pareto fronts for different runs implies that the best way to use the heuristic is to run it multiple times in order to get multiple Pareto fronts. These fronts can then be combined to find the best-known Pareto front based on multiple runs. For example, consider the fronts in Figure 7-14. The best possible front based on these solutions is a combination of solutions from all three fronts. Recall the three goals of a best-known Pareto front, listed by Konak et al. (2006): (1) be as close to the true Pareto front as possible, (2) the solutions should be uniformly spread over the Pareto front and (3) capture the whole spectrum of the true Pareto front. For the real-size instances, all the fronts contain at least one solution with no more than half the number of changes of the cheapest one found, with multiple solutions in between these two. This indicates that the fronts capture the spectrum of the true Pareto front and that the solutions are uniformly spread. None of the fronts find the minimum cost solution found by the single-objective HGSADC, but all solutions are less than 0.8% more expensive than it, supporting that the fronts are close to the true Pareto front. In other words, all of the three goals seem to be achieved.

7.5 Results - MSVPP with cost and robustness

This section presents the results from solving three problem instances with two objectives: minimizing cost and maximizing robustness. Robustness is defined and measured as described in Section 4.5. For the first two instances, the fronts generated by the HGSADC are compared with the optimal front, found using the ϵ -constraint method. As in the previous section, all instances are solved with a fixed fleet, due to the assumption that chartering additional PSVs is not a viable option.

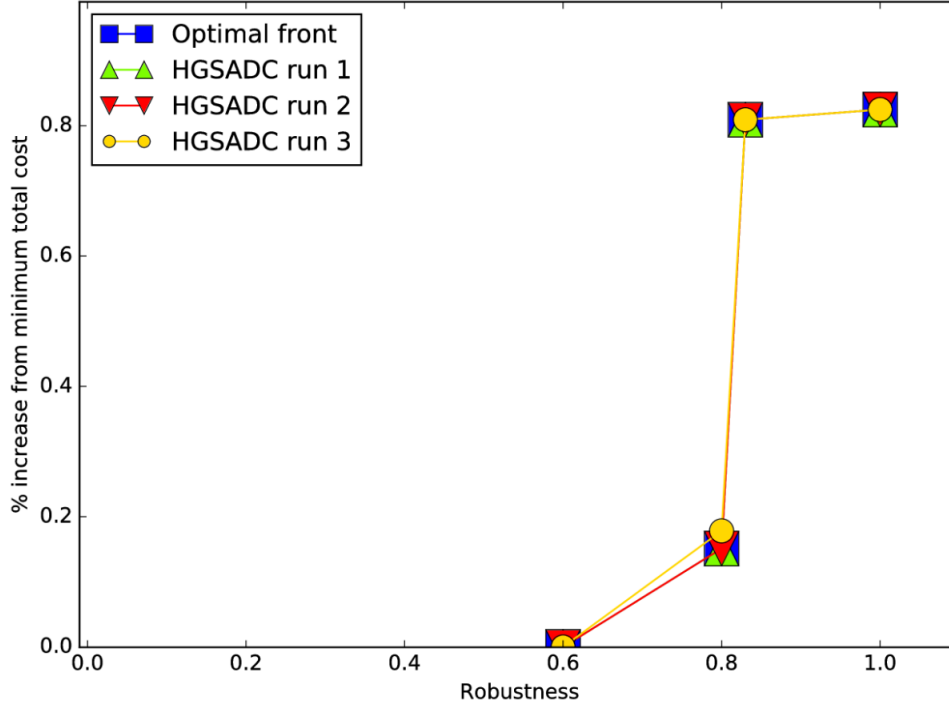


Figure 7-16: Optimal and heuristic fronts for 11-36 with robustness.

7.5.1 Results compared with optimal fronts

Problem instance 11-36 and 13-44 are solved with both the ϵ -constraint method and the HGSADC. The horizontal axis shows the robustness value, which is the share of voyages sailed that have the minimum required slack, while the vertical axis shows the total cost of the solution, expressed as percentage increase from the lowest known total cost of the problem instance. Four hours of slack is required for voyages lasting one or two days to be considered robust, and six hours are required for three-day voyages. No voyages longer than three days are allowed.

11-36 with robustness

Figure 7-16 shows the optimal and heuristic fronts for problem 11-36 with robustness. The optimal front has four solutions, and the heuristic finds all of the optimal solutions in two of three runs, and a front where one solution is slightly more expensive in the third run. The fronts show that all voyages can be made robust for less than 1% increase in costs.

13-44 with robustness

Figure 7-17 shows the optimal and heuristic fronts for instance 13-44 with robustness. The optimal front has two solutions, and the heuristic finds one of the optimal solutions in all of the three runs. The second solution found by the heuristic has the same cost as the optimal solution, but is slightly less robust. Note that all of

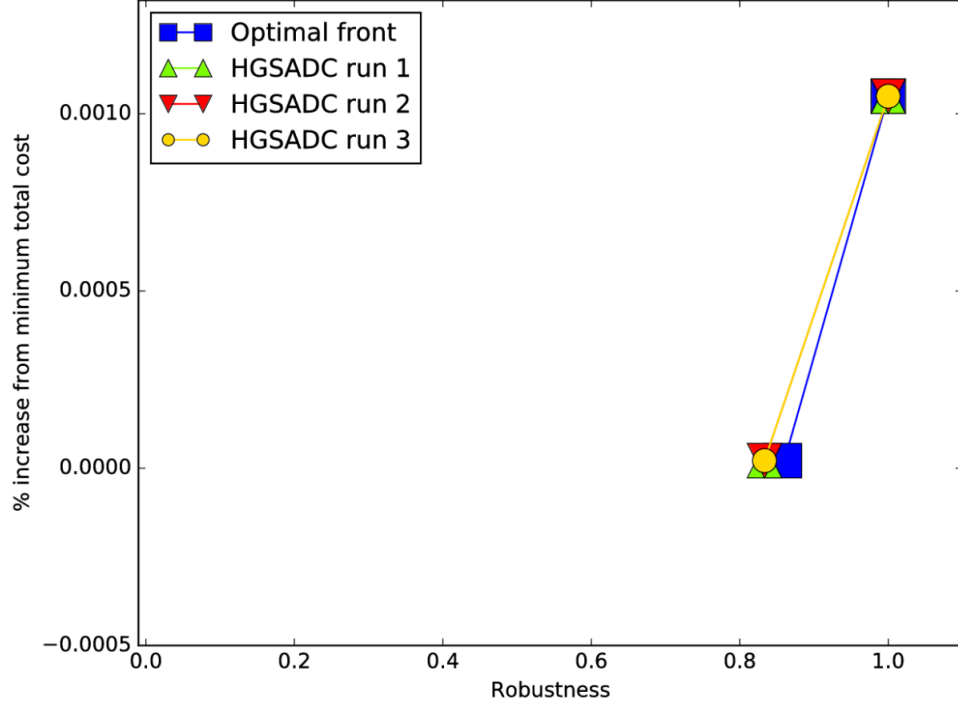


Figure 7-17: Optimal and heuristic fronts for 13-44 with robustness.

the fronts show that the robustness can be increased to 1, meaning that all of the voyages are robust for a cost increase of 0.001%, or about 47 NOK. It is worth noting that a solution method than only considers cost would return the cheapest solution, even though the solution with best robustness is probably preferred by the decision makers.

7.5.2 Results for real-size problem instance

Figure 7-18 shows the heuristic fronts found when solving problem 27-80 with cost and robustness as objectives. The solutions have low cost, in fact two of the runs have solutions with lower cost than the average cost found by the single-objective HGSADC. Adding robustness as an objective does not seem to impair the quality of the solutions. The fronts show that robustness can be increased from 0.73 to 0.91 for a cost increase of around 0.3%. The HGSADC is only tested for one large instance, but the results for 27-80 are promising given that no education is used, and the performance is stable. The fronts are likely to be close to the true Pareto front, since cheap solutions are found; they cover a wide range of robustness values and the solutions are uniformly distributed on the fronts, with the exception of the third front, which only contains two solutions.

It is interesting to note that even though no education is used for robustness, the HGSADC finds the entire or almost the entire optimal Pareto front for the small instances and finds high-quality fronts for the large instance. The changes made to

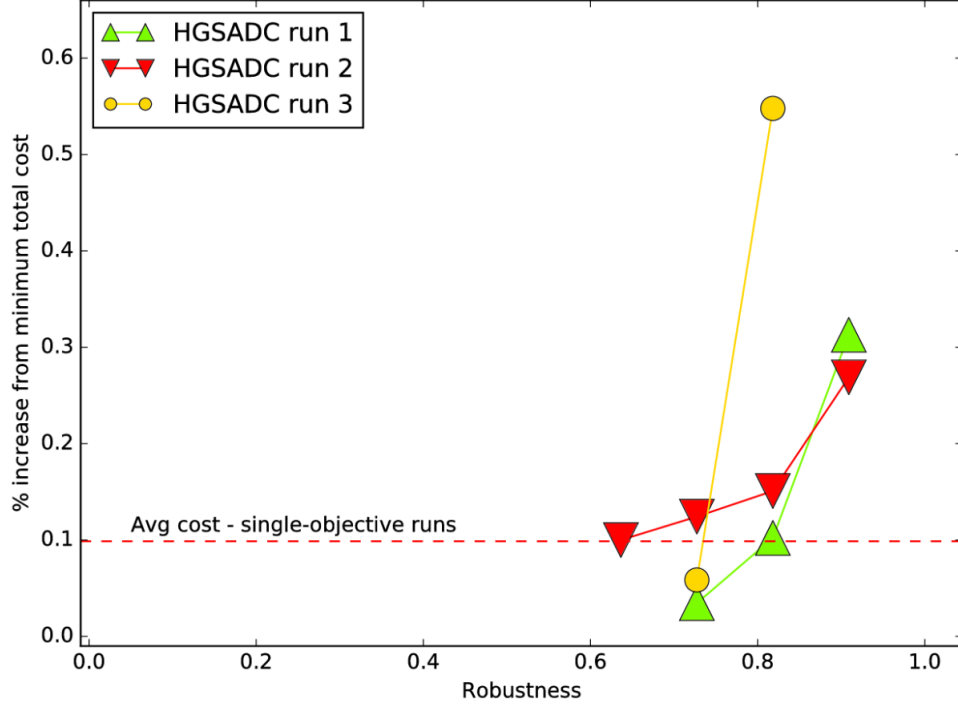


Figure 7-18: Heuristic fronts for 27-80 with robustness.

the algorithm in order to include robustness as an objective consist only of adding the robustness rank to the biased fitness function. This shows that adding a new objective to the HGSADC is very simple and can be effective, even without complicated education procedures, opening up for a wide range of possibilities when it comes to extending the problem.

7.6 Results - MSVPP with cost, persistence and robustness

This section presents the results from solving two problem instances with three objectives: minimizing cost, maximizing persistence and maximizing robustness. The fronts generated by the HGSADC are compared with the optimal front for the first instance. The optimal front is found using the ϵ -constraint method and the VBM found in Appendix A. Exact methods are unable to solve the second instance due to the problem size. Both instances are solved with a fixed fleet, due to the assumption that chartering additional PSVs is not a viable option.

7.6.1 Results compared with optimal fronts

Figure 7-19 shows the optimal and heuristic fronts for instance 11-36, with robustness and three installations added as variation case. Since the MSVPP is solved with three objectives, cost, persistence and robustness, the solution is a three-dimensional Pareto

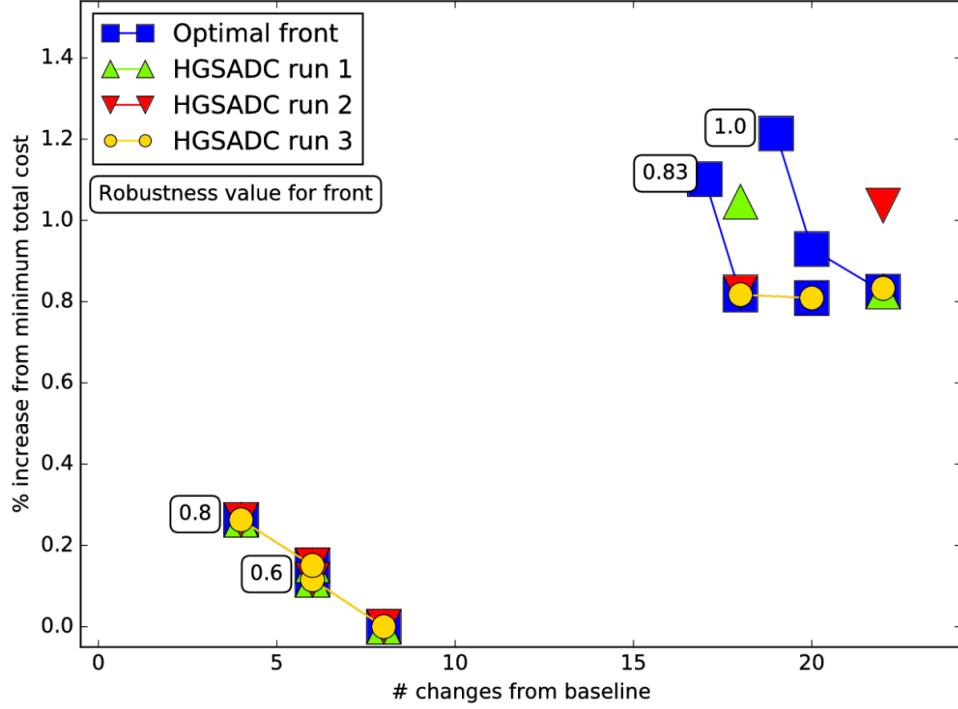


Figure 7-19: Optimal and heuristic fronts for variation case 1 - add new installations and robustness. Problem instance 11-36 is solved and problem instance 8-26 is used as baseline. Solutions that are connected represent one front, the boxes next to the fronts show the robustness value for the front.

front. The three-dimensional front is represented in two dimensions by plotting one front for each level of robustness. This means that the four optimal fronts in Figure 7-19 are the representation of one optimal front in three dimensions. The figure shows that the heuristic finds the same solutions as the optimal front for two of the levels of robustness. For the two other levels, the heuristic finds solutions that are close to or part of the optimal front, but none of the heuristic runs find all of the solutions on the optimal front. The fronts clearly illustrate a trade-off between robustness and persistence, as higher robustness value increases the number of changes. For all levels of robustness, the number of changes can be reduced for a small increase in cost. The fronts are close to the true Pareto fronts, but for high robustness levels they do not cover the entire spectrum of the true Pareto front, since they contain too few solutions.

7.6.2 Results for real-size problem instance

Figure 7-20 shows the heuristic front for problem instance 22-71 with robustness and the shutdown of five installations as variation case. 27-80 is used as baseline and the variation case is defined by Statoil, so the front shows solutions for a real-size problem with a realistic variation case. Only one front is shown due to the high number of solutions on the front, making it difficult to visually compare fronts from different

runs. Results show that similar fronts are found for additional runs, indicating that the HGSADC is stable with regards to solution quality for three objectives and real-size problem instances as well. The three-dimensional front is represented in two dimensions by plotting one front for each value of robustness. The front consists of 22 non-dominated solutions, with the number of changes ranging from 19 to zero, the robustness ranging from 0.64 to 1 and the cost ranging between 0.1% and 0.8% of the lowest known cost found by the single-objective heuristic. The front shows that the robustness can be increased from 0.78 to 1 for a cost increase of around 0.6%, and that for a robustness of 1, the number of changes can be reduced from nine to two for a cost increase in 0.01%. The front also shows that a solution with zero changes and a robustness level of 0.91 can be obtained by increasing the cost by around 0.7%. For all levels of robustness except one, the number of changes can be reduced for a small increase in cost. In many cases the robustness can be increased without affecting the number of changes for a small increase in cost. The front presents decision makers with a diverse set of solutions and clearly illustrates the trade-offs between the three objectives. Note that the two solutions in the upper left corner have a high quality with regards to persistence and cost, and only cost around 0.7% more than the lowest cost found by the single-objective HGSADC. The low cost of the solutions indicate that the front is close to the true front; the solutions contain solutions with a wide range of persistence and robustness values and the solutions are relatively uniformly distributed. In other words, the goals of a best-known Pareto front seem to be achieved.

7.6.3 Running times

The running times of the HGSADC increases, on average, for every objective added. Two samples of how the average running time for solving an instance changes with choice of objectives are shown in Table 7.13. The running time increases as objectives are added, and these samples indicate that it increases linearly with number of objectives. For both these examples, the running time with three objectives is about three times as long as with one objective. The running time will of course depend on the objectives, both on how computationally heavy it is to calculate the objective values and other factors affecting the search space. The numbers show an important benefit of using genetic algorithms for multi-objective optimization: they are scalable and can easily be extended by adding more objectives without increasing the running time by much. This is a huge benefit compared to a method such as the ϵ -constraint method, where both the number of iterations and the time spent on each iteration increase rapidly with each additional objective.

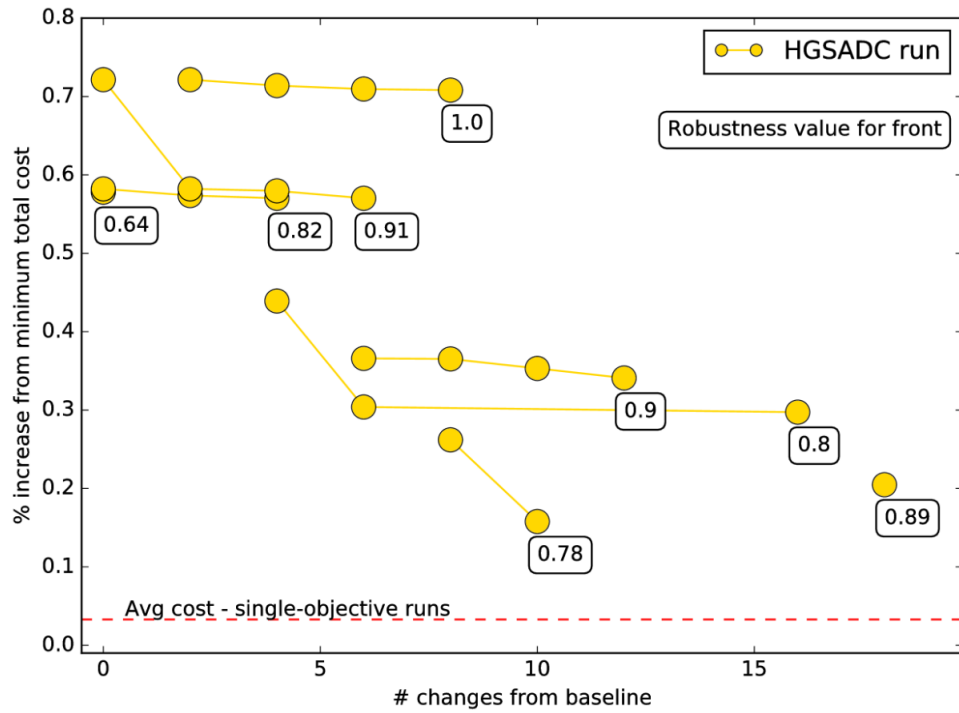


Figure 7-20: Heuristic front for variation case 3 - shutdown installations and robustness. Problem instance 22-71 is solved and problem instance 27-80 is used as baseline. Solutions that are connected represent one front, the boxes next to the fronts show the robustness value for the front.

Problem instance	Variation case	Cost	Cost and persistence	Cost and robustness	Cost, pers. and rob.
22-71	Shutdown 5 installations	1068	1672	2504	2667
27-80	Add 3 installations	977	2211	1737	3374

Table 7.13: Average running times in seconds for different choices of objectives.

Chapter 8

Economic Implications

The objective of the SVPP is to minimize the costs related to the usage and operation of the PSVs while providing a reliable supply service. This section describes why solving the MSVPP instead of the SVPP and presenting the decision makers with Pareto fronts leads to reduced costs, a more reliable supply service and better insights for the decision makers.

One way to solve the SVPP is to identify the costs related to the PSVs, e.g. the cost of chartering and sailing the PSVs, and to minimize these costs while satisfying the relevant constraints. This approach will produce schedules that minimize the *direct costs* related to the usage and operation of the PSVs, the costs that are directly attributable to the PSVs. However, since the supply service is part of a supply chain, there are many *indirect costs* related to the usage and operation of the PSVs, costs that are affected by the PSVs, but not directly attributable. These indirect costs may be difficult to quantify or predict and are therefore unsuitable to be included in an optimization model. Experienced decision makers can, however, to some extent predict what factors will lead to high indirect costs. Including these factors in the optimization model and presenting decision makers with multiple solutions combines the computational efficiency of optimization models with the experience of decision makers. Based on the experience of planners at Statoil, the two most important drivers of indirect costs are persistence and robustness. Sections 8.1-8.3 describe how including these factors as objectives benefit planners at Statoil, Section 8.4 explains the opportunities for including additional objectives and Section 8.5 summarizes some limitations of the results.

8.1 The advantages of persistent schedules

According to the decision makers at Statoil, persistence is one of the most important characteristics of a schedule, apart from the direct costs. The different parts of the upstream petroleum supply chain are interconnected, and rely on each other in a way such that changes in one part, e.g. in the PSV schedules, cause unpredictable

consequences in other parts. Changes in the PSV schedule need to be communicated to and implemented by all relevant parties, a process that takes time and resources. The crews at installations often create their schedules and shifts based on when the PSVs depart from the supply depot, meaning that changes in the PSV schedule cause disturbances in the operations on the installations. From this, it is clear that persistent schedules can make for both smoother operations and reduced costs, which means decision makers should be willing to accept somewhat higher direct costs in order to increase the persistence of a schedule. The decision makers are able to quantify persistence as the number of changes from the previous schedule, but it is difficult to convert the number of changes into cost, since the relationship between cost and the number of changes depends on external factors, like how long it has been since the previous change of schedule. In addition, the relationship between cost and the number of changes is non-linear. For example, a reduction from 25 to 20 changes is worth considerably less than a reduction from five to zero changes, since staffing is likely to change for both 25 and 20 changes, while it is not affected when the schedule has no changes. A Pareto front clearly illustrates the relationship between direct costs and persistence, making it easier for the decision makers to select the solutions that lead to the best supply service.

The decision makers at Statoil have come up with four common variation cases where the input is changed and a new schedule is needed. Figures 7-12 - 7-15 in Chapter 7 show the Pareto fronts for the variation cases on real-size problem instances. The fronts show that persistence can be improved by a small increase in cost for all of the cases. In the load reduction case, the number of changes can be reduced to zero by increasing the direct cost by less than 0.3%. Given the increase of costs in other parts of the supply chain and the extra work to inform all affected parts about the new schedule, the solution with zero changes is probably preferred by decision makers. The other variation cases have similar fronts, where the number of changes can be reduced for a modest increase in cost. All solutions have a cost that is less than 1% higher than the lowest known cost for the problem instance, meaning that generating a Pareto front does not impair the quality of the solutions much, as compared to finding only a single solution.

It is assumed that increasing the persistence of schedules significantly reduces indirect costs, since the decision makers emphasize it as one of the most important characteristics of a schedule. Under this assumption, the total cost can be reduced by presenting the decision makers with Pareto fronts instead of just the minimum cost solution. This is supported by the results of the variation cases for real-size problem instances, where the number of changes can be reduced to zero for a cost increase of less than 0.6% in two of the variation cases. A small increase of the direct costs leads to a large reduction of the indirect costs, and hence reduces the total cost.

In some of the variation cases, no solution with zero changes is found. The reason is that additional visits to installations are added, which can make it impossible to find a solution with zero changes without increasing the fleet size. The same problem can arise when shutting down installations, if the baseline solution uses more PSVs than

the new problem requires. The charter cost of a PSV is $\sim 200\%$ of the total sailing cost for the real-size problem instances, so it is assumed that the decision makers would never prefer to increase the fleet size to improve persistence. Even if the Pareto front does not present a solution with zero changes, it may be of value to the decision makers. The fronts show that a large reduction of the number of changes is possible for a small increase in cost, even though it cannot be reduced zero. For instance, in the real-size problem with variation case 1 - adding installations - the number of changes can be reduced from 35 to 15 for a cost increase of less than 0.6% compared to the minimum cost solution found, while for variation case 4 - adding visits - the number of changes can be reduced from 19 to seven for a cost increase of 0.2%. In addition, a Pareto front can help the decision makers to justify and communicate their decisions to other parts of the organization, such as explaining that there will be changes to the schedule, since chartering another vessel to avoid the changes is too expensive, but that it is minimized as much as possible with the current fleet size.

8.2 The advantages of robust schedules

The objective of the SVPP is not only to minimize costs, but also to provide a reliable supply service. The supply service is considered reliable if the required amount of supplies is delivered at the planned time. The service and sailing times are affected by the rough weather conditions in the North Sea, and to ensure that the supplies are delivered at the planned time, the planners decide the appropriate level of robustness. More robust schedules have higher direct costs, since robustness is increased by adding slack to the voyages, which decreases the number of installations that can be serviced in one voyage. However, robust schedules are expected to have less of the indirect costs that arise when unexpected events necessitate re-routing or chartering additional PSVs. A solution to the SVPP is thus a trade-off between cost and robustness. The value of robustness depends on many factors, for example, the weather conditions in the North Sea are significantly tougher during the winter season, increasing the value of robustness in the winter. Instead of attempting to quantify the value of robust voyages, the decision makers are presented with a Pareto front that illustrates the trade-off between robustness and cost. This way, decision makers can use their own judgment of both the likelihood of delays and the importance of robustness to select the best schedule.

Figure 7-18 shows Pareto fronts for problem instance 27-80, representing Statoil's supply service as of April 2016. The fronts show that the share of robust voyages can be increased from 0.73 to 0.91 for a cost increase of around 0.3%. A cost increase of 0.3% is very little compared to the cost of chartering an additional PSV in case of delay, and the decision makers will probably accept a small cost increase to improve the reliability of the schedule.

8.3 Combining low cost, high persistence and high robustness

Above it has been argued that the increased direct costs of both persistent schedules and robust schedules can be offset by the reduction in indirect costs that persistence and robustness provide. The discussions above have explained the trade-off between direct costs and persistence and between direct costs and robustness, but decision makers also face a trade-off between persistence and robustness. Thus, the most interesting Pareto front is one where all three objectives are included: cost, persistence and robustness. Figure 7-20 in Chapter 7 shows such a Pareto front for a real-size problem instance. The front shows that the number of changes can be reduced for a cost increase of around 0.1% for all levels of robustness. Two of the solutions are probably very interesting to decision makers. One of the solutions have zero changes from baseline and a robustness of 0.91, and another solution has three changes from baseline and a robustness of 1. These two solutions are optimal with regards to one of the objectives and close to optimal for the other, and both cost only around 0.7% more than the best solution found by the single-objective HGSADC. These two solutions are probably preferred by decision makers and are likely to both reduce the realised costs and improve the reliability of the supply service. One should also note the trade-off between persistence and robustness: no solution which maximizes both persistence and robustness is found for the given fleet.

All of the objectives described above aim to reduce the sum of direct and indirect costs. Persistence and robustness are increased in order to avoid unplanned costs and extra work, and to increase the reliability of the supply service. A solution with low persistence and robustness would have lower sailing and charter costs on paper, but the realised costs for Statoil would be much higher. Finding the best solution to the SVPP is therefore done by finding the most appropriate trade-off between the objectives. Presenting the decision makers with Pareto fronts that clearly illustrates the trade-offs gives them valuable insights into how these objectives are related. Combining these insights with their knowledge and experience is likely to provide solutions that more effectively balance the objectives, resulting in lower realised costs. The results from real-size problem instances also show that both persistence and robustness can be improved to optimal or near optimal values for a very small increase in sailing and chartering costs. Decision support tools based on the solution method presented in this thesis are therefore expected to be valuable to the decision makers at Statoil, leading to reduced costs, a more reliable supply service and better insights for the decision makers.

8.4 Adding additional objectives

The solution method for the MSVPP presented in this thesis can easily be extended to other objectives. If an objective can be quantified, it can be added to the solution method. The presented solution method can therefore be used to add additional

objectives and further improve the solutions of the MSVPP. The decision makers can use the solution method to get insights about the trade-off between cost and different characteristics of the schedules, and to get better and more realistic solutions to the MSVPP.

8.5 Limitations of the results

The results from solving the MSVPP using the solution methods presented in this thesis are promising. There are, however, some limitations of the results. The problem considered in this thesis is a simplified variant of the problem solved by the decision makers at Statoil, because it does not include time windows or handle collisions at the installations, i.e. PSVs arriving at an installation at the same time. Additional constraints and adaptations are necessary in order to use the solution methods to solve the actual problem. The effect of the additional constraints and adaptations on solution quality and running time is not known. Another limitation of the results is that no comparable results are available for the larger problem instances. The cost of the solutions on the heuristic fronts are close to the cost found for the single-objective problem, but the quality of the solutions to the single-objective problem is not verified, due to the lack of comparable results. However, the results for smaller instances are good, both for the SVPP and the MSVPP, and the objective value seems to increase at a constant rate as the problem size increases, indicating that the solutions found are high-quality solutions.

Chapter 9

Conclusion

This master’s thesis has discussed the planning of Statoil’s offshore supply service at a tactical level, formulated as the multi-objective supply vessel planning problem (MSVPP). The MSVPP is an extension of the supply vessel planning problem (SVPP), presented by Halvorsen-Weare et al. (2012), where multiple objectives are considered.

The decision makers at Statoil have experienced that several characteristics of a solution affects how it performs in reality, and have therefore requested solutions that consider multiple objectives. More specifically, they have requested solutions with few changes from the previously used solution and solutions that can handle delays. These two characteristics were defined as the persistence and robustness of a solution, respectively. A formulation of the SVPP was presented, along with formulations of persistence and robustness.

For the single-objective SVPP, exact solution methods are not able to solve problem instances with more than 14 installations, and a hybrid genetic search with adaptive diversity control (HGSADC) was developed to solve real-size problem instances, based on the work of Vidal et al. (2012a). The HGSADC was able to find the optimal solution for all problem instances where optimality was proven by exact methods, and seemed to find good solutions for all larger problem instances. Solutions to real-size problem instances were found within 1 000 seconds on average, and the cost of these solutions correlated with problem size in a manner that indicated that they were high-quality solutions. The running time increased linearly with the size of the problem instance, which is a large improvement compared to the exact methods, for which the running time increases exponentially. The HGSADC showed stable performance, having small variations in objective value between each run.

The solutions to the MSVPP were presented as Pareto fronts. Exact solution methods are unsuited for large instances of the MSVPP, as they have to solve the SVPP once for each solution on the Pareto front. The HGSADC, on the other hand, maintains a population of solutions during the search, and is therefore well suited to solve multi-objective problems. The HGSADC was adapted to solve the MSVPP, and was able

to find fronts identical or close to the optimal fronts for the problem instances that could be solved by exact methods. The HGSADC found Pareto fronts for real-size problem instances for all of the objectives, and all solutions had a cost that was less than 1% higher than the best solutions found by the single-objective HGSADC. This indicates that adding additional objectives does not impair the quality of solutions much. In the tests performed, the running time of the HGSADC increased linearly for each objective added. The results on real-size problem instances show that both robustness and persistence can be improved to optimal or near-optimal values for a cost increase of less than 1%. Solving the MSVPP with the proposed solution method is therefore expected to be a valuable tool for decision makers, and implementing it in decision support tools is expected to improve the supply service of Statoil.

The HGSADC utilizes the Unified Hybrid Genetic Search (UHGS) framework of Vidal et al. (2014), which can be applied to many different classes of multi-attribute VRPs. The work in this thesis displays the usefulness of this framework and is an example of how it can be applied to a real-life problem, with special constraints that separate it from a standard PVRP. Previous work with the UHGS has mainly been focused on single-objective problems, but the results from this thesis have shown that the framework can easily be extended to handle multiple objectives, and that these extensions perform well, both in terms of the quality of solutions and the running time of the heuristic. This supports earlier work showing that genetic algorithms are both efficient and effective for multi-objective optimization.

Future research

There are many possible areas of future research for the MSVPP and the HGSADC. The MSVPP can be extended by adding elements such as time-windows for the service of installations, multiple depots, variable departure time from the depot and collision avoidance at the installations. This can make the solution method more applicable for the decision makers at Statoil.

Another possible area of research is to improve the performance of the HGSADC. One way of improving the performance is to add a decomposition phase to the HGSADC, as described by Vidal et al. (2013). The HGSADC can also be adapted to use the Split-algorithm for a homogeneous fleet or for heterogeneous fleets with multiple PSVs of each PSV type. Testing different variations of the multi-objective HGSADC might also improve the performance of it, for instance, changing fitness evaluation to use Pareto ranking or measuring distance between individuals in objective space when calculating the diversity contribution.

Adapting the multi-objective HGSADC to other types of VRPs is also a possible area of future research. The framework presented by Vidal et al. (2014) can be used to adapt the HGSADC to a large range of VRPs, and by using the adaptations presented in this thesis, the VRPs can be solved with multiple objectives. Generating Pareto fronts for different objectives can improve the solutions and the insights of managers for many real-life problems.

Bibliography

- B. Aas, Ø. Halskau Sr, and S. W. Wallace. The role of supply vessels in offshore logistics. *Maritime Economics & Logistics*, 1(3):302–325, 2009.
- A. Agra, M. Christiansen, R. Figueiredo, L. M. Hvattum, M. Poss, and C. Requejo. The robust vehicle routing problem with time windows. *Computers & Operations Research*, 40(3):856–866, 2013.
- A. Agra, M. Christiansen, A. Delgado, and L. M. Hvattum. A maritime inventory routing problem with stochastic sailing and port times. *Computers & Operations Research*, 61:18–30, 2015.
- T. Borthen and H. Loennechen. The Supply Vessel Planning Problem with Persistence, Project Report. NTNU, 2015.
- G. G. Brown, R. F. Dell, and R. A. Farmer. Scheduling coast guard district cutters. *Interfaces*, 26:59–72, 1996.
- G. G. Brown, K. J. Cormican, S. Lawphongpanich, and D. B. Widdis. Optimizing submarine berthing with a persistence incentive. *Naval Research Logistics*, 44: 301–318, 1997a.
- G. G. Brown, R. F. Dell, and R. K. Wood. Optimization and persistence. *Interfaces*, 27:15–37, 1997b.
- Business Day News. Rolls-royce secures order for platform supply vessels from Paxocean Engineering, 2011. Retrieved 21 September 2015, from www.bday.net/rolls-royce-secures-order-for-platform-supply-vessels-from-paxocean-engineering/.
- M. Christiansen, K. Fagerholt, B. Nygreen, and D. Ronen. Maritime transportation. *Transportation*, 14:189–284, 2006.
- K. Deb. Multi-objective optimization. In E. K. Burke and G. Kendall, editors, *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, pages 403–449. Springer US, 2014. ISBN 978-1-4614-6939-1.
- M. Drexler. A generic heuristic for vehicle routing problems with multiple synchronization constraints. Technical report, Johannes Gutenberg University Mainz, 2014.

- K. Fagerholt and H. Lindstad. Optimal policies for maintaining a supply service in the Norwegian Sea. *Omega*, 28(3):269–275, 2000.
- K. Fagerholt, J. Korsvik, and A. Løkketangen. Ship routing scheduling with persistence and distance objectives. *Innovations in Distribution Logistics*, 619:89–107, 2009.
- A. Fischer, H. Nokhart, H. Olsen, K. Fagerholt, J. G. Rakke, and M. Stålhane. Robust planning and disruption management in roll-on roll-off liner shipping. *Transportation Research Part E: Logistics and Transportation Review*, 91:51–67, 2016.
- P. M. Francis, K. R. Smilowitz, and M. Tzur. The period vehicle routing problem and its extensions. In *The vehicle routing problem: Latest advances and new challenges*, pages 73–102. Springer, 2008.
- M. Gendreau, G. Laporte, and R. Séguin. Stochastic vehicle routing. *European Journal of Operational Research*, 88(1):3–12, 1996.
- I. Gribkovskaia, G. Laporte, and A. Shlopak. A tabu search heuristic for a routing problem arising in servicing of offshore oil and gas platforms. *Journal of the Operational Research Society*, 59(11):1449–1459, 2008.
- E. Halvorsen-Weare and K. Fagerholt. Optimization in offshore supply vessel planning. *Optimization and Engineering*, pages 1–25, 2016.
- E. E. Halvorsen-Weare and K. Fagerholt. Robust supply vessel planning. In *Network optimization*, pages 559–573. Springer, 2011.
- E. E. Halvorsen-Weare, K. Fagerholt, L. M. Nonås, and B. E. Asbjørnslett. Optimal fleet composition and periodic routing of offshore supply vessels. *European Journal of Operational Research*, 223(2):508–517, 2012.
- R. W. Hamming. Error detecting and error correcting codes. *Bell System technical journal*, 29(2):147–160, 1950.
- M. Jahre, G. Persson, B. Aas, I. Gribkovskaia, Ø. Halskau Sr, and A. Shlopak. Routing of supply vessels to petroleum installations. *International Journal of Physical Distribution & Logistics Management*, 37(2):164–179, 2007.
- D. F. Jones, S. K. Mirrazavi, and M. Tamiz. Multi-objective meta-heuristics: An overview of the current state-of-the-art. *European journal of operational research*, 137(1):1–9, 2002.
- A. Konak, D. W. Coit, and A. E. Smith. Multi-objective optimization using genetic algorithms: A tutorial. *Reliability Engineering & System Safety*, 91(9):992–1007, 2006.
- J. E. Korsvik, K. Fagerholt, and G. Laporte. A large neighbourhood search heuristic for ship routing and scheduling with split loads. *Computers & Operations Research*, 38(2):474–483, 2011.

- J. Lundgren, M. Rönnqvist, and P. Värbrand. Dynamic programming. In *Optimization*, pages 481–494. Studentlitteratur AB, 2010.
- McKinsey & Company. Meeting the challenge of increasing North Sea costs, 2014. Retrieved 9 December 2015, from http://www.mckinsey.com/~media/mckinsey/dotcom/client_service/oil%20and%20gas/pdfs/meeting_the_challenge_of_increasing_north_sea_costs.ashx.
- P. Moscato and C. Cotta. A gentle introduction to memetic algorithms. In *Handbook of metaheuristics*, pages 105–144. Springer, 2003.
- E. K. Norlund and I. Gribkovskaia. Reducing emissions through speed optimization in supply vessel operations. *Transportation Research Part D: Transport and Environment*, 23:105–113, 2013.
- E. K. Norlund, I. Gribkovskaia, and G. Laporte. Supply vessel planning under cost, environment and robustness considerations. *Omega*, 2015.
- Norwegian government. Norway’s oil history in 5 minutes, 2015. Retrieved 8 December 2015, from <https://www.regjeringen.no/en/topics/energy/oil-and-gas/norways-oil-history-in-5-minutes/id440538/>.
- Norwegian Petroleum Directorate. The government’s revenues, 2016. Retrieved 31 May 2016, from <http://www.norskpetroleum2.no/en/economy/governments-revenues/>.
- J.-Y. Potvin, Y. Xu, and I. Benyahia. Vehicle routing and scheduling with dynamic travel times. *Computers & Operations Research*, 33(4):1129–1137, 2006.
- C. Prins. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, 31(12):1985–2002, 2004.
- M. Savelsbergh and J.-H. Song. An optimization algorithm for the inventory routing problem with continuous moves. *Computers & operations research*, 35(7):2266–2282, 2008.
- A. Shyshou, I. Gribkovskaia, G. Laporte, and K. Fagerholt. A large neighbourhood search heuristic for a periodic supply vessel planning problem arising in offshore oil and gas operations. *INFOR: Information Systems and Operational Research*, 50(4):195–204, 2012.
- Statoil. Annual report on form 20-F, 2015. Retrieved 9 December 2015, from http://www.statoil.com/no/InvestorCentre/AnnualReport/AnnualReport2014/Documents/DownloadCentreFiles/01_KeyDownloads/20-F_2014.pdf.
- T. Vidal, T. G. Crainic, M. Gendreau, N. Lahrichi, and W. Rei. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, 60(3):611–624, 2012a.

- T. Vidal, T. G. Crainic, M. Gendreau, N. Lahrichi, and W. Rei. Electronic companion - a hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, 2012b. URL <http://dx.doi.org/10.1287/opre.1120.1048>.
- T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research*, 40(1):475–489, 2013.
- T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins. A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operational Research*, 234(3):658–673, 2014.

Appendix A

Voyage-based model

The following model is the voyage-based model (VBM) presented by Borthen and Loennechen (2015). All results referred to in the text as from the VBM are from an implementation of this model. Section A.1.4 contains new variables and constraints that add robustness to the model.

A.1 Voyage-based formulation

The voyage-based formulation decomposes the problem into two separate problems, where the first problem is to generate all voyages that can be part of an optimal solution and the second problem is to decide a fleet of PSVs and assign the voyages to the PSVs using a voyage-based model. The problems are solved in sequence, that is, all voyages are first generated, and then the voyage-based model assigns these voyages to PSVs. Note the distinction between the voyage-based formulation, referring to the overall problem, and the voyage-based model, referring only to the model used in assigning voyages.

A.1.1 Voyage generation

The set of all voyages that can be part of an optimal solution, hereafter referred to as the *candidate voyages*, consists of the feasible voyage with lowest cost and the feasible voyage with shortest duration for each subset of installations that satisfy the constraints on PSV capacity and number of installations per voyage. Note that the cheapest voyage can also be the voyage with shortest duration. The voyage generation handles constraints on opening hours, capacity, minimum and maximum duration of a voyage and maximum number of installations visited in a voyage. The voyage generation uses the following parameters and variables:

T_{ij}^S	- Sailing time from installation i to installation j
T_i^{SER}	- Service time at installation i
T_{id}^O	- Opening hour of installation i on day d of the voyage
T_{id}^C	- Closing hour of installation i on day d of the voyage
\overline{T}^{VOY}	- Maximum duration of a voyage
\underline{T}^{VOY}	- Minimum duration of a voyage
FC_{ij}^S	- Fuel cost of sailing from i to j
FC^I	- Fuel cost per hour idle at installation
F_i	- The number of weekly visits demanded by installation i
D_i	- The weekly demand of installation i
Q	- The total loading capacity of the PSV
\overline{M}	- Maximum number of installations in a voyage
h_i^W	- Waiting time at installation i
t_i^A	- Arrival time at installation i

Note that in the voyage generation, the opening and closing hour of an installation is defined as the number of hours after the voyage departs from the supply depot. Therefore, the opening and closing hours are indexed by installation and the day of the voyage. For example, an opening hour of 07:00 of installation i corresponds to $T_{i1}^O = 15$ and $T_{i2}^O = 39$, as the installation opens 15 hours after the departure from the supply depot on the first day of the voyage, and 39 hours after the departure on the second day.

A.1.2 Voyage-based model

The voyage-based model looks as follows: Let \mathcal{V} be the set of all PSVs available for time charter, indexed by v , let \mathcal{N} be the set of all offshore installations serviced from the supply depot, indexed by i , and \mathcal{F} the possible required number of visits to an installation, indexed by f . \mathcal{N}_f are the subsets of \mathcal{N} containing the installations that require $f \in \mathcal{F}$ visits. Let \mathcal{T} be the set of days in the planning period, indexed by t , \mathcal{L} be the set of all possible voyage durations (in days), indexed by l , and \mathcal{R} be the set of all candidate voyages, indexed by r . \mathcal{R}_v is the set of all candidate voyages that PSV $v \in \mathcal{V}$ may sail. From \mathcal{R}_v the subsets \mathcal{R}_{vi} and \mathcal{R}_{vl} are defined. \mathcal{R}_{vi} contains the candidate voyages of PSV v that visits installation $i \in \mathcal{N}$ and \mathcal{R}_{vl} contains the candidate voyages of PSV v that has duration $l \in \mathcal{L}$.

C_v^{TC} is the cost of chartering PSV v the whole planning period, and C_{vr}^S the sailing and service costs of PSV v sailing voyage r . S_i is the number of visits required by installation i , F_v is the number of days PSV v may be used during the planning period, and B_t is the maximum number of PSVs that may be serviced at the supply depot on day t . In order to ensure that departures are spread evenly throughout the planning period, the approach presented by Shyshou et al. (2012) is used, where a sub-horizon h_f is defined for installations requiring $f \in \mathcal{F}$ visits. For each sub-horizon h_f there must be at least \underline{P}_f and no more than \overline{P}_f departures to each relevant installation. For example, if the planning period is seven days, an installation which requires three

visits (i.e. $f = 3$) needs at least one and no more than two departures to it every three days, corresponding to $h_3 = 3$, $\underline{P}_f = 1$ and $\overline{P}_f = 2$.

The two decision variables are:

$$\delta_v = \begin{cases} 1, & \text{if PSV } v \text{ is chartered} \\ 0, & \text{otherwise} \end{cases}$$

$$x_{vrt} = \begin{cases} 1, & \text{if PSV } v \text{ sails voyage } r \text{ starting on day } t \\ 0, & \text{otherwise} \end{cases}$$

The objective function

$$\min \sum_{v \in \mathcal{V}} C_v^{TC} \delta_v + \sum_{v \in \mathcal{V}} \sum_{r \in \mathcal{R}_v} \sum_{t \in \mathcal{T}} C_{vr}^S x_{vrt} \quad (\text{A.1})$$

minimizes the sum of chartering cost and sailing cost. This objective is subject to the following constraints:

$$\sum_{v \in \mathcal{V}} \sum_{r \in \mathcal{R}_{vi}} \sum_{t \in \mathcal{T}} x_{vrt} \geq S_i, \quad i \in \mathcal{N} \quad (\text{A.2})$$

ensure all installations are serviced the required number of times.

$$\sum_{l \in \mathcal{L}} \sum_{r \in \mathcal{R}_{vl}} \sum_{t \in \mathcal{T}} l x_{vrt} - F_v \delta_v \leq 0, \quad v \in \mathcal{V} \quad (\text{A.3})$$

ensure each PSV does not sail more days than allowed.

$$\sum_{v \in \mathcal{V}} \sum_{r \in \mathcal{R}_v} x_{vrt} \leq B_t, \quad t \in \mathcal{T} \quad (\text{A.4})$$

constrain the number of PSVs at the supply depot at a given day.

$$\sum_{r \in \mathcal{R}_{vl}} x_{vrt} + \sum_{r \in \mathcal{R}_v} \sum_{\tau=1}^{l-1} x_{vr, (t+\tau) \bmod |\mathcal{T}|} \leq \delta_v, \quad v \in \mathcal{V}, \quad t \in \mathcal{T}, \quad l \in \mathcal{L} \quad (\text{A.5})$$

ensure a PSV does not begin a new voyage before it has returned from its previous voyage.

$$\underline{P}_f \leq \sum_{v \in \mathcal{V}} \sum_{r \in \mathcal{R}_{vi}} \sum_{h=0}^{h_f} x_{vr, (t+h) \bmod |\mathcal{T}|} \leq \overline{P}_f, \quad i \in \mathcal{N}_f, \quad f \in \mathcal{F}, \quad t \in \mathcal{T} \quad (\text{A.6})$$

make sure departures from the supply depot are evenly spread throughout the planning period. Due to the assumption that no installation require more visits than there are days in the planning period, these constraints are created such that there is never more than one departure to an installation on any day. Note that the formulation of

constraints (A.2) and (A.3) have been modified from those in Shyshou et al. (2012) by utilizing the sets \mathcal{R}_{vi} and \mathcal{R}_{vl} . Finally, there are binary constraints on the variables:

$$\delta_v \in \{0, 1\}, \quad v \in \mathcal{V}, \quad (\text{A.7})$$

$$x_{vrt} \in \{0, 1\}, \quad v \in \mathcal{V}, \quad r \in \mathcal{R}_v, \quad t \in \mathcal{T}. \quad (\text{A.8})$$

A.1.3 Measuring persistence

The following section describes the extensions needed in order to minimize both cost and the number of changes from the existing solution. The objective is to minimize the number of changes from the existing solution. The existing solution is referred to as the *baseline solution*. The most appropriate method for measuring changes depends on the preferences of decision makers. In the following model, only the changes in departures to an installation are used to measure change. This is based on information from Statoil that the most important factor to be persistent is the supply ordering routines, that is, at what times the installations need to report their demand. This is determined by the time the PSVs depart from the supply depot. Let \mathcal{N}_B be the set of installations in the baseline solution B . Change from an existing (baseline) solution B to a new solution is then measured as:

$$\sum_{t \in \mathcal{T}} \sum_{i \in \mathcal{N} \cap \mathcal{N}_B} |\sigma_{it} - \sigma_{it}^B|, \quad (\text{A.9})$$

where σ_{it} is 1 if a PSV which services installation i leaves the supply depot on day t in the current solution and 0 otherwise. σ_{it}^B has the corresponding values for the baseline solution B . Note that σ_{it} is a variable, while σ_{it}^B is a parameter. Also note that σ_{it} is binary due to constraints (A.6), which limit the number of departures to a platform to one per day. The sum is only including the installations which are both in the current and the baseline problem to avoid counting the changes which inevitably arise when adding or removing installations. σ_{it} can be calculated using the following formula:

$$\sigma_{it} = \sum_{v \in \mathcal{V}} \sum_{r \in \mathcal{R}_{vi}} x_{vrt}, \quad i \in \mathcal{N} \cap \mathcal{N}_B, \quad t \in \mathcal{T}. \quad (\text{A.10})$$

Expression (A.9) uses the absolute value operator, which is non-linear. In order for us to use it in a mixed integer problem, it needs to be linearized. First, introduce a new variable

$$\gamma_{it} = \begin{cases} 1, & \text{if there is a change in departures to installation } i \text{ on day } t \\ 0, & \text{otherwise} \end{cases}$$

which is defined for all $i \in \mathcal{N} \cap \mathcal{N}_B, t \in \mathcal{T}$. In other words, $\gamma_{it} = |\sigma_{it} - \sigma_{it}^B|$. Thus expression (A.9) can be replaced with the following:

$$\sum_{i \in \mathcal{N} \cap \mathcal{N}_B} \sum_{t \in \mathcal{T}} \gamma_{it}. \quad (\text{A.11})$$

In order to link γ_{it} with σ_{it} and linearize the absolute value operator, constraints (A.12) - (A.14) are added to the mathematical model:

$$\gamma_{it} \geq \sigma_{it} - \sigma_{it}^B, \quad i \in \mathcal{N} \cap \mathcal{N}_B, t \in \mathcal{T}, \quad (\text{A.12})$$

$$\gamma_{it} \geq \sigma_{it}^B - \sigma_{it}, \quad i \in \mathcal{N} \cap \mathcal{N}_B, t \in \mathcal{T}, \quad (\text{A.13})$$

$$\gamma_{it} \in \{0, 1\}, \quad i \in \mathcal{N} \cap \mathcal{N}_B, t \in \mathcal{T}. \quad (\text{A.14})$$

By adding constraints (A.12) - (A.14) to the voyage-based model the problem is redefined to include two objectives, minimizing cost and minimizing the number of changes from an existing solution, as follows:

$$\min \sum_{v \in \mathcal{V}} C_v^{TC} \delta_v + \sum_{v \in \mathcal{V}} \sum_{r \in \mathcal{R}_v} \sum_{t \in \mathcal{T}} C_{vr}^S x_{vrt}, \quad (\text{A.1 revisited})$$

$$\min \sum_{i \in \mathcal{N} \cap \mathcal{N}_B} \sum_{t \in \mathcal{T}} \gamma_{it}, \quad (\text{A.15})$$

A.1.4 Measuring robustness

The slack of a voyage is measured as the number of hours from the vessel arrives at the supply depot until the next time the supply depot *opens*. This is based on the assumption that a vessel that arrives within the opening hours of the supply depot is not serviced until the next day. Let \mathcal{S} be the set of all possible slack durations (in hours), indexed by s . \mathcal{R}_{vls} is a subset of \mathcal{R}_{vl} , and contains the voyages with a duration

of $l \in \mathcal{L}$ that PSV $v \in \mathcal{V}$ may sail that have a slack of $s \in \mathcal{S}$ hours. Robustness is measured as the *share of voyages with a slack above a minimum threshold*, $\alpha_l, l \in \mathcal{L}$. The number of robust voyages in the solution can be found using the formula:

$$\sum_{v \in \mathcal{V}} \sum_{l \in \mathcal{L}} \sum_{s \in \mathcal{S} | s \geq \alpha_l} \sum_{r \in \mathcal{R}_{vls}} \sum_{t \in \mathcal{T}} x_{vrt}, \quad (\text{A.16})$$

where α_l is the minimum hours of slack for voyages of length $l \in \mathcal{L}$. This expression can be normalized by dividing by the total number of voyages used in the solution, such that the robustness value is the *share of voyages used that have the required amount of slack*. This value would be calculated as follows:

$$\frac{\sum_{v \in \mathcal{V}} \sum_{l \in \mathcal{L}} \sum_{s \in \mathcal{S} | s \geq \alpha_l} \sum_{r \in \mathcal{R}_{vls}} \sum_{t \in \mathcal{T}} x_{vrt}}{\sum_{v \in \mathcal{V}} \sum_{r \in \mathcal{R}_v} \sum_{t \in \mathcal{T}} x_{vrt}}. \quad (\text{A.17})$$

Appendix B

Voyage generation using dynamic programming

This section describes the dynamic programming approach used to generate the set of candidate voyages, \mathcal{R} , for the VBM presented in Appendix A. The text is taken from Borthen and Loennechen (2015). First, the voyage generation procedure is described, and an introduction to the dynamic programming approach is given. Second, label data, label extension and label domination is explained. Finally, the dynamic programming algorithm is described.

B.1 The voyage generation procedure

The voyage generation procedure generates the set of all candidate voyages, \mathcal{R} . Algorithm 10 shows pseudocode describing the voyage generation procedure. As in Halvorsen-Weare et al. (2012), the candidate voyages are generated for all PSVs with similar properties at the same time. If two PSVs have the same sailing speed and fuel consumption rates, a voyage will have the same duration and cost for the two PSVs. Therefore, the candidate voyages of the PSV with lowest capacity of the two will be a subset of the candidate voyages of the PSV with highest capacity. This relationship is utilized in Algorithm 10. The first step of the algorithm is to generate all subsets of PSVs, $\mathcal{V}^S \subseteq \mathcal{V}$, that have the same sailing speed and fuel consumption rates. Let W be the set containing all these subsets of PSVs. Then the algorithm loops through all sets of PSVs $\mathcal{V}^S \in W$, and for each \mathcal{V}^S , the PSV with the highest capacity, v^{MX} , is identified. The candidate voyages of v^{MX} , $\mathcal{R}_{v^{MX}}$, are generated. For the rest of the PSVs in the set, the set of candidate voyages is the subset of $\mathcal{R}_{v^{MX}}$ that uses a capacity lower than or equal to the PSV capacity, Q_v .

Algorithm 10 is very similar to what is used by Halvorsen-Weare et al. (2012). The difference is how the candidate voyages of v^{MX} are generated. In order to generate the candidate voyages of v^{MX} , Halvorsen-Weare et al. (2012) generate all feasible subsets of installations. For each subset, all possible orderings of visits are enumerated, and

Algorithm 10 Pseudocode describing the voyage generation. The function GENERATEVOYAGES is described in Algorithm 11.

```

1: procedure GENERATEVOYAGES(Maximum number of installations  $\overline{M}$ )
2:    $W \leftarrow$  set of subsets of PSVs with equal sailing speed and fuel consumption
   rates
3:   for all subsets of PSVs  $\mathcal{V}^S \in W$  do
4:      $v^{MX} \leftarrow$  PSV with highest capacity in  $\mathcal{V}^S$ 
5:      $\mathcal{R}_{v^{MX}} \leftarrow$  GENERATEVOYAGES( $v^{MX}, \overline{M}$ )  $\triangleright$  Candidate voyages of  $v^{MX}$ 
6:     for all PSVs  $v \in \mathcal{V}^S | v \neq v^{MX}$  do
7:       for all voyages  $r \in \mathcal{R}_{v^{MX}}$  do
8:         if capacity used by voyage  $r \leq Q_v$  then  $\triangleright Q_v$  is the capacity of  $v$ 
9:            $\mathcal{R}_v \leftarrow \mathcal{R}_v \cup \{r\}$ 
10:        end if
11:      end for
12:    end for
13:  end for
14: end procedure

```

the feasible voyage with the shortest duration is added to the set of candidate voyages. This is a simple and straightforward approach, but it is computationally heavy. A dynamic programming approach is likely to be much more efficient. Therefore, a labelling algorithm utilizing dynamic programming is implemented to generate the candidate voyages. The output of the implemented algorithm is the same as that of the total enumeration approach, but the algorithm removes infeasible and dominated voyages, and utilizes already computed partial voyages to reduce the computational time of the voyage generation.

Candidate voyages are generated for a given set of PSVs. The voyage generation is equivalent to finding the shortest paths through a connected, directed graph, where the supply depot and each installation is a node, and all nodes are connected. Let N be the number of installations. A destination node with index $N + 1$ is added to the graph because all voyages have to return to the supply depot. The problem is solved by generating all feasible, undominated paths from the start node, with index 0, to the destination node. One such path represents a feasible voyage that has either the lowest cost or the shortest duration for the set of visited installations.

Using the terminology of dynamic programming presented by Lundgren et al. (2010), each subproblem is called a *stage*. In each stage, *states* represent partial voyages, and each state is represented by a *label*.

B.2 Label data

A label contains the following information:

$$\begin{aligned}
 n & - \text{Current installation} \\
 t & - \text{Departure time from the current installation} \\
 c & - \text{Cost of the voyage so far} \\
 q & - \text{Quantity of supplies delivered to installations so far} \\
 v & - \text{Integer number representing the visited installations} \\
 m & - \text{Number of installations visited, including the supply depot} \\
 p & - \text{The predecessor label}
 \end{aligned} \tag{B.1}$$

A complete label is written

$$L = (n, t, c, q, v, m, p).$$

The current installation of a label L is denoted by $n(L)$ and the departure time from the current installation is denoted by $t(L)$. The quantity of supplies delivered, denoted by $q(L)$, is the sum of supplies delivered to visited installations. The cost of a label is the sum of sailing, waiting and service costs, and is denoted by $c(L)$. To speed up the algorithm, the set of installations already visited is represented by an integer, denoted by $v(L)$. The number 2^i is added to v when installation i is visited. Accordingly, a node is visited if

$$v \bmod 2^{i+1} \geq 2^i.$$

The number of installations visited is saved in order to determine whether the voyage has visited the maximum number of installations, and is denoted by $m(L)$. To be able to find the sequence of visits, a label contains information about its predecessor, denoted by $p(L)$. The initial stage corresponds to all partial candidate voyages of size one, and contains only the initial label

$$L_0 = (0, 0, 0, 0, 2^0, 1, \emptyset).$$

The initial label represents a partial voyage starting at the supply depot. It has no predecessor label, denoted by \emptyset (the empty set), and has visited one installation.

B.3 Label extension

Using the terminology of dynamical programming, a resource corresponds to a quantity that varies along a path according to functions called resource extension functions. When a label L_A is extended along the arc (i, j) from its current installation, $n(L_A) = i$, to another installation, j , a new label L_B is created. The resources of L_B

are set according to the following resource extension functions:

$$n(L_B) = j, \quad (B.2)$$

j is the current installation of L_B .

$$t(L_B) = t(L_A) + T_{ij}^S + h_j^W + T_j^{SER}, \quad (B.3)$$

where T_{ij}^S is the sailing time from i to j , T_j^{SER} is the service time at j and h_j^W is the waiting time at installation j . If $j \neq N + 1$, the waiting time is defined by

$$h_j^W = \begin{cases} T_{jd}^O - t_j^A, & \text{if } t_j^A < T_{jd}^O \\ T_{j,d+1}^O - t_j^A, & \text{if } t_j^A \geq T_{jd}^C \\ T_{j,d+1}^O - T_{jd}^C, & \text{if } T_{jd}^O \leq t_j^A < T_{jd}^C \text{ and } t_j^A + T_j^{SER} > T_{jd}^C \\ 0, & \text{otherwise} \end{cases} \quad (B.4)$$

and if $j = N + 1$, the waiting time is defined by

$$h_j^W = \begin{cases} T_{jd}^O - t_j^A, & \text{if } t_j^A \leq T_{jd}^O \\ T_{j,d+1}^O - t_j^A, & \text{if } t_j^A > T_{jd}^O \end{cases} \quad (B.5)$$

where $t_j^A = t(L_A) + T_{ij}^S$ is the arrival time at j , d is the day of arrival, T_{jd}^O is the opening hour of j on d and T_{jd}^C is the closing hour of j on d .

If a PSV arrives before the installation opens, it has to wait until the opening hour. If a PSV arrives after the installation closes, it has to wait until the installation opens the next day. If a PSV arrives within the opening hours, but does not finish the service before the closing hour, it has to wait until the installation opens again the next day and continue the service. Otherwise, the PSV arrives within the opening hours and finishes the service before the closing hours, hence it does not have to wait at all. As mentioned in the assumptions of the mathematical model in Section 4.2, it is assumed that the service time of an installation is always shorter than the opening hours, i.e. if a PSV arrives before the installation opens it will always finish service before the installation closes again. The assumption simplifies the waiting time calculation.

If $j = N + 1$, the voyage is finished, and $t(L_B)$ corresponds to the duration of a finished voyage. Note that in this context the supply depot has a service time of 0, since the service is captured in the departure time when starting the voyage.

$$c(L_B) = c(L_A) + FC_{ij}^S + FC^I(t(L_B) - (t(L_A) + T_{ij}^S)), \quad (B.6)$$

where FC_{ij}^S is the fuel cost of sailing from i to j and FC^I is the fuel cost per hour at

an installation.

$$q(L_B) = \begin{cases} q(L_A), & \text{if } j = N + 1 \\ q(L_A) + \frac{D_j}{F_j}, & \text{otherwise} \end{cases} \quad (\text{B.7})$$

where D_j is the weekly demand of installation j and F_j is the number of weekly visits demanded by installation j .

$$v(L_B) = v(L_A) + 2^j, \quad (\text{B.8})$$

installation j is marked as visited in L_B by adding 2^j to $v(L_A)$, the integer representing visited installations.

$$m(L_B) = \begin{cases} m(L_A), & \text{if } j = N + 1 \\ m(L_A) + 1, & \text{otherwise.} \end{cases} \quad (\text{B.9})$$

The number of visited installations is increased by 1 if j is not the supply depot.

$$p(L_B) = L_A \quad (\text{B.10})$$

L_A is the predecessor of L_B .

The above-mentioned extension from L_A to L_B along (i, j) is called a resource feasible extension if and only if:

$$n(L_A) \neq N + 1 \quad (\text{B.11})$$

If the current installation of L_A is $N + 1$, it means that the PSV has already returned to the supply depot, hence the label should not be extended.

$$t(L_B) \leq \bar{T}^{VOY} \quad (\text{B.12})$$

The departure time of a label must be less than the maximum duration of a voyage, \bar{T}^{VOY} .

$$t(L_B) \geq \underline{T}^{VOY}, \text{ if } j = n + 1 \quad (\text{B.13})$$

The duration of a finished voyage must be greater than the minimum duration of a voyage, \underline{T}^{VOY} .

$$q(L_B) \leq Q \quad (\text{B.14})$$

The quantity of supplies delivered must be less than the load capacity of the PSV, Q .

$$v(L_A) \bmod 2^j + 1 < 2^j \quad (\text{B.15})$$

The current installation of the new label cannot be already visited by the voyage.

$$m(L_B) \leq \overline{M} \quad (\text{B.16})$$

The number of visited installations must be less than or equal to the maximum number of installations in a voyage.

B.4 Label domination

Label extension ensures that all generated paths are feasible, but by itself, it results in a total enumeration of all feasible voyages. By using label dominance, the number of generated paths is significantly reduced. If label L_A dominates L_B , denoted by $L_A \prec L_B$, L_B is not extended, and hence none of the successors of L_B are generated. The general dominance criteria are:

$L_A \prec L_B$ if and only if

- The labels are in the same node
- Every resource feasible extension of L_B is resource feasible for L_A
- For every resource feasible extension of L_B , the cost is greater than or equal to the cost of the same extension of L_A

All of the resource extension functions of the defined label are non-decreasing and separable. A resource extension function is separable if there exists no interdependencies between the resource it is describing and the other resources. For a resource with a non-decreasing and separable resource extension function, it is true that if two labels have consumed the same amount of the resource, they have the same resource feasible extensions with regards to that resource. It is also true that if the resource is a positive resource, a higher consumption is better, and if it is negative a lower consumption is better. This results in the following dominance criteria for the defined label:

$L_A \prec L_B$ if and only if

$$n(L_A) = n(L_B) \quad (\text{B.17})$$

$$v(L_A) = v(L_B) \quad (\text{B.18})$$

$$c(L_A) \leq c(L_B) \quad (\text{B.19})$$

$$t(L_A) \leq t(L_B) \quad (\text{B.20})$$

Criterion (B.17) ensures that the first general dominance criterion is satisfied, namely that the labels are in the same node. Criterion (B.18) ensures that the labels have visited the same installations, meaning that $v(L_A) = v(L_B)$ and $m(L_A) = m(L_B)$. Since the demand of supplies of an installation is the same regardless of when it is visited, $q(L_A) = q(L_B)$ holds if (B.18) is satisfied. Criterion (B.19) ensures that L_A has a lower or equal cost and criterion (B.20) ensures the same for departure time. In other words, criterion (B.17) - (B.20) ensures that every resource feasible extensions of L_B is resource feasible for L_A , satisfying the second general dominance criterion. Since both the cheapest voyage and the voyage with the shortest duration are needed for the candidate voyages, every resource feasible extension of L_B must have an equal or higher cost *and* an equal or longer duration than the same extension from L_A for L_B to be dominated. Criterion (B.19) and (B.20) ensures this, hence the third and final general dominance criterion is satisfied.

Note that the order in which the dominance criterion are checked affects the number of labels checked for dominance. The more labels a criterion affects, the earlier it should be checked. As most labels will have visited different installations, criterion (B.18) is ideally checked first. Second, the current installation of a label should be checked. The sequence of checking the cost and departure time is assumed to not differ much in the number of affected labels.

B.5 The labelling algorithm

The problem of finding the candidate voyages for a set of installations can be decomposed into \overline{M} subproblems, where \overline{M} is the maximum number of installations visited in a voyage. A subproblem is to find the set of feasible, undominated voyages, partial or finished, of size k , $k \in \{1, 2, \dots, |\overline{M}| + 1\}$. This set is called M_k and represents one stage. Note that the candidate voyages of length k are the feasible, undominated labels of subproblem k that satisfy $n(L) = N + 1$, in other words, the voyages that are finished. Dominance criteria (B.17) - (B.20) ensure that each feasible, undominated label of length k has a feasible, undominated label found in subproblem $k - 1$ as a predecessor. In other words, for every candidate voyage of length k , the voyage visiting the $k - 1$ first installations is a feasible, undominated label in subproblem $k - 1$. To utilize the optimal substructure of the problem, the labelling algorithm extends all labels in a stage before extending the labels of the next stage, starting at stage 1.

The pseudocode presented in Algorithm 11 shows the generation of candidate voyages for a PSV v and a maximum number of installations in a voyage, \overline{M} . The initial label L_0 is created and added to the initial stage M_1 . Then all stages are looped through, starting at M_1 , and all unfinished labels in a stage are extended. A label is unfinished if the current installation of the label is not the supply depot. Algorithm 12 shows the pseudocode for the extension of a label. After all unfinished labels have been extended, each stage M_k contains the set of undominated, feasible labels corresponding to a partial or finished voyage of length k . The labels with the supply depot as current node are added to the set of candidate voyages.

The algorithm described in Algorithm 12 loops through all installations $i \in \mathcal{N}$ for the given label, L_A . If L_A has visited i , there should be no extension from L_A to i , so the algorithm continues to the next installation. If i has not been visited, there is an extension from L_A to i , resulting in label L_B . The extension is defined by the resource extension functions (B.2) - (B.10). L_B is a feasible label if it satisfies (B.11) - (B.16). If L_B is feasible, it is added to the next stage by Algorithm 13. Algorithm 13 goes through all the existing labels in the next stage, and if any of the existing labels dominate L_B , the algorithm terminates. If none of the existing labels dominates L_B , the algorithm goes through the labels again. If L_B dominates any of the existing labels, they are removed from the stage. Finally, L_B is added to the next stage.

When the labelling algorithm has extended the labels of stage one through \overline{M} , all candidate voyages have been generated. The output of the algorithm is the set of all candidate voyages for PSV v , \mathcal{R}_v . The output is used in Algorithm 10, as described earlier.

Algorithm 11 Algorithm for generating the candidate voyages for a given PSV. Used in Algorithm 10

```

1: procedure GENERATEVOYAGESSET(PSV  $v$ , Maximum number of installations
    $\overline{M}$ )
2:    $\mathcal{R}_v \leftarrow \emptyset$  ▷ Set of candidate voyages for PSV  $v$ 
3:    $L_0 = (0, 16, 0, 0, 2^0, 1, \emptyset)$ 
4:    $M_1 \leftarrow M_1 \cup \{L_0\}$ 
5:   for ( $k=1$  to  $\overline{M}$ ) do
6:     for all unfinished labels  $L^U \in M_k \mid n(L^U) \neq n+1$  do
7:       EXTEND( $L^U, k$ )
8:     end for
9:     for all finished labels  $L^F \in M_k \mid n(L^F) = n+1$  do
10:       $R_v \leftarrow R_v \cup \{L^F\}$ 
11:    end for
12:  end for
13: end procedure

```

Algorithm 12 Algorithm for extending a label in a given stage. Used in Algorithm 11

```

1: procedure EXTEND(label  $L_A$ , stage  $k$ )
2:   for all installations  $i \in \mathcal{N}$  do
3:     if  $v(L_A) \bmod 2^{i+1} \geq 2^i$  then                                 $\triangleright$  has already visited  $i$ 
4:       continue
5:     else
6:        $L_B \leftarrow$  the label created when  $L_A$  is extended from  $L_A$  to  $i$ 
7:       if  $L_B$  is feasible then
8:         ADDTOSTAGE( $L_B$ ,  $k + 1$ )
9:       end if
10:    end if
11:  end for
12: end procedure

```

Algorithm 13 Algorithm for adding a label to a stage. Used in Algorithm 12

```

1: procedure ADDTOSTAGE(label  $L_X$ , stage  $k$ )
2:   for all existing label  $L_Y \in M_k$  do
3:     if  $L_X \succ L_Y$  then
4:       stop procedure
5:     end if
6:   end for
7:   for all existing label  $L_Y \in M_k$  do
8:     if  $L_X \prec L_Y$  then
9:        $M_k \leftarrow M_k \setminus \{L_X\}$ 
10:    end if
11:  end for
12:   $M_k \leftarrow M_k \cup \{L_Y\}$ 
13: end procedure

```

Appendix C

Code and test instances

All programmed code, test instances and relevant output files are provided in a zipped archive file. The archive contains two folders: (1) Exact methods, containing the implementation and data for the VBM and (2) HGSADC, containing the implementation and data for the HGSADC.

C.1 Exact methods

The folder named "Exact methods" contains all files related to the voyage-generation, the VBM and the ϵ -constraint method, described in Appendices A and B. It contains the folders "Voyage generation" and "VBM".

C.1.1 Voyage generation

The folder named "Voyage generation" contains the code and input files for the voyage generator. The source code can be found in the folder "src", and the input found is an excel file found in "data/input". The parameter values for the voyage generation is set in the first sheet of the excel file, "Parameters", and the test instances in this thesis is found in the sheet named "April 2016". Voyages can be generated by running the file called "voyageGeneration.jar", which uses the excel file as input. The output is found in "data/output".

C.1.2 VBM

The folder named "VBM" contains the code, input files and output data for the voyage-based model and the ϵ -constraint method. The source code is found in the file "VoyageBasedModel.mos" in the folder "src". The settings of the model can be set by changing the parameters at the beginning of the file. Here one can define what input files to use, the timeout limit, the output file folder, the accepted optimality gap and whether to use the ϵ -constraint method. The input files are in the folder

"data/input/april 2016". The subfolder "baseline" contains the baseline solutions for each problem size, the subfolders "fixed fleet" and "variable fleet" contain the input files for the fixed fleet and variable fleet problem, respectively. The output files used are in the folder "data/output". The subfolder "epsilon" contains the optimal pareto fronts generated by the ϵ -constraint method, while the subfolders "fixed fleet" and "variable fleet" contain the output files for the fixed fleet and variable fleet problem, respectively. The folder "src/VBMSolver" contains a program for solving multiple problem instances. This program is currently not adapted for general use.

C.2 HGSADC

The folder named "HGSADC" contains the input files, all of the programmed code and the output data of the HGSADC. The input files can be found in the folder "hgs/input". The parameters used in the HGSADC and all test instances are found in the excel file called "input data hgs.xls". The parameter values for the voyage generation is set in the first sheet of the excel file, "Parameters", and the test instances in this thesis is found in the sheet named "April 2016". The output files of all data described in the master's thesis can be found in the folder named "data/hgs/output". All results for the different objectives of the MSVPP can be found in the folder named "MSVPP", and all results of the SVPP, including the parameter testing, can be found in the folder named "SVPP".

All of the source code can be found in the folder named "src", and python procedures used to plot the results are found in the folder named "plotting".

Execute the file called "HGSADC.jar" found in the HGSADC-folder in order to run the HGSADC. The file uses "input data hgs.xls" as input, and generates output in the folder named "data/hgs/output".