Joachim Klungseth

A Simulation-Based Optimization Model for Fleet Sizing and Fleet Composition of a Well-Boat Fleet

December 2020

Master's thesis

Master's thesis

2020

Joachim Klungseth

NTNU
Norwegian University of
Science and Technology
Faculty of Engineering
Department of Marine Technology

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

dedication (optional)

Summary

The Norwegian aquaculture industry has had a phenomenal growth over last decades and is expected to continue this growth during the next couple of decades. This will lead to a huge demand of new well-boats and increased need to streamline well-boat operations.

Different optimization techniques could be of huge help when making decisions regarding size and capabilities of the future well-boat fleet. By combining simulation models with optimization, it is possible to obtain a close to optimal solution with the minimal amount of computational time. Simulation-based optimization also offers a great way of dealing with uncertainties and stochastic processes.

Meta-heuristics and stochastic adaptive search techniques are reliable algorithms for simulation-based optimization. Stochastic adaptive search techniques will always converge towards the global optimum and given enough time, they always find a good estimation of the optimal solution. Meta-heuristics is not guaranteed to converge towards the global optimum, but in practice they have shown to be efficient tools for finding good solutions.

For this problem the genetic algorithm was proven to outperform the others. In general, the techniques that were quickest to narrow the search in on a promising area, gave the best results. A reason for this could be that the solution space has one dominant global optimum. The algorithms that finds this area relatively fast and stays there are also those who gives the best result. The genetic algorithm and the LAST does this and are also those who delivers the best solutions.

Performing Pareto Front analysis, the variance of the solution showed no sign of being correlated with the strength of the solution. This is a clear advantage for any stakeholder, as they could choose the best solution without compromising on risk tolerance.

The completage percentage for missions for a given fleet was one the other hand strongly correlated with the strength of the solution. Which again works in the favor of the stakeholder and reduces the amount of compromises needed.

The best solution found, came from the genetic algorithm, with these values for

the decision variables: Fleet size 8, vessel types 1, 1, 1, 2, 2, 3, 5 and 5.

The other algorithms also favored vessel type 1, 2 and 5, and a fleet size of 7 or 8. That the smaller vessels are favored is not what the current market trend would suggest. Then again new builts are mainly projected for long term contracts while this model is aimed at the spot market.

Sammendrag

Den norske oppdrettsindustrien har hatt en fabelaktig vekst de siste 20 årene og denne veksten er forventet å fortsett i flere tiår framover. Dette vil medføre et stort behov for nye brønnbåter, samt verktøy for å effektivisere brønnbåtoperasjoner.

Forskjellige optimeringsmetoder kan være til stor hjelp når flåtestørrelse og sammensetning skal bestemmes. Ved å kombinere simuleringsmodeller med optimeringsalgoritmer er det mulig å finne gode løsninger med minimal bruk av datakraft. Simuleringsbasert optimering tilbyr også en god løsning for å håndtere usikkerhet og stokastiske prosesser.

Meta heuristikker og stochastic adaptive search teknikker er pålitelige algoritmer til bruk i simuleringsbasert optimering. Stochastic adaptive search teknikker har beviselige convergens egenskaper og vil alltid nærme seg det globele optimum. meta heuristikker har ikke disse konvergens egenskapene men har i praksis vist seg å kunne fungere like godt.

For dette problemet har den genetiske algoritmen vist seg å kunne utkonkurrere de andre. På generell basis har de teknikkene som raskets kunne peile seg inn på et lovende område, også levert de beste resultatene. En grunn til dette kan være at løsningsområdet er dominert av ett globalt optimum. Derfor var det de algoritmene som først fant dette området og konsentrerte søket her, som også gjorde det best. LAST kom også godt ut.

Ved å gjennomføre en Pareto Front analyse og sette gode løsninger opp mot variansen, ble det vist liten korrelasjon mellom gode løsninger og høy varians. Dette er et godt resultat for alle stakeholderene, da de kan gå for beste løsning uten å gå på kompromiss med risiko villigheten.

Fullføringsraten på oppdrag for en gitt flåte viste seg å ha en sterk korrelasjon med styrken på løsningen. Dette virker igjen i favør av stakeholderene, da beste flåte kan velges uten å gå på kompromiss med andre interesser.

Den beste løsningen ble funnet av den genetiske algoritmen, med disse beslut-

ningsvariablene: Flåtestørrelse 8, båttyper 1, 1, 1, 2, 2, 3, 5 og 5.

De andre algoritmene favoriserte også båttypene 1, 2 og 5, samt en flåtestørrelse på 7 eller 8 brønnbåter. Det at de mindre båtene ble foretrukket er ikke på linje med den nåværende trenden i markedet. Denne modellen er laget for spot markedet mens de fleste ny bygg blir prosjektert for lengere kontrakter. Dette kan forklare noe av forskjellen.

Preface

This report is the final delivery in the course TMR4930 - Marine Technology, Master's Thesis, at the institute of Marine Technology at NTNU in Trondheim. The aim of this thesis is to construct a simulation-based optimization model for use in the Norwegian aquaculture industry.

I'm very thankful for my five years of studying Marine Technology at NTNU. I would like to thank everyone that have helped me through these years, both professors and my fellow students. It has been a great experience, and now this thesis will try to sum up some of what I have learned.

Through previous courses at NTNU I have learned about the use of optimization as a decision support method. It was interesting to combine this with what I previously have learned about the maritime industry and programming.

Hans Tobias Slette has been a huge resource for me during this Master's Thesis. The simulation model would not have been possible without him.

At last I will thank Bjørn Egil Asbjørnslett for guidance along the way.

Table of Contents

Summary **i**

Sammendrag **iii**

Preface **v**

Table of Contents **ix**

List of Tables **xi**

List of Figures **xv**

Abbreviations **xvi**

1 Introduction **1**

1.1 The fish farming Industry 1

1.2 Fleet sizing and fleet composition 3

2 System Description **5**

2.1 The aquaculture industry 6

2.1.1 The fish farming value chain 7

2.2 Well-boats 9

2.2.1 Transportation of smolt from hatcheries to fish farms 10

2.2.2 Transportation of fully grown fish from fish farms to slaughter houses 10
2.2.3 Performing delousing at the fish farms 11

3 Problem Description **13**
3.1 Defining the optimization problem 13
3.2 Creating the simulation model 15

4 Literature Review **17**
4.1 Maritime fleet sizing and mix problems 17
4.2 Simulation-based optimization 18
4.3 Discrete parametric optimization 21
4.3.1 Meta-heuristics 22
4.3.2 Stochastic adaptive search 25

5 The Simulation-Based Optimization model **29**
5.1 The simulation model 29
5.1.1 Input and assumptions 32
5.2 The optimization model 35

6 Resultes **39**
6.1 Pure random search 39
6.2 Genetic algorithm 42
6.3 Tabu search 44
6.4 Simulated annealing 46
6.5 Learning automata search technique 48

7 Discussion **53**

8 Conclusion and further work **57**
8.1 Conclusion 57
8.2 Further work 59

Bibliography **61**

Appendix **67**

List of Tables

5.1 Capabilities of each well-boat type 32

5.2 Work capability rates for each type of well-boat presented in tons/hour . . 33

5.3 Mission types and structure 33

6.1 Performance of the pure random search 39

6.2 Performance of the genetic algorithm 42

6.3 Performance of the tabu search 44

6.4 Performance of the simulated annealing 46

6.5 Performance of the LAST 48

7.1 Performance of the meta-heuristics 54

List of Figures

1.1 Possible increase in production volume 2

2.1 Production areas by traffic light categories 7

2.2 Life cycle of farmed fish 8

3.1 Well-boat operations 15

4.1 Simulation-based optimization loop 19

4.2 Genetic Algorithm 23

4.3 Tabu Search 24

4.4 Learning Automata Search Technique 26

4.5 Simulated Annealing 27

5.1 Simulation model 31

5.2 Mission generation 34

6.1 Revenue from the pure random search 40

6.2 Shows highest revenue over the regression line 40

6.3 Shows the percentage completed compared to how much it costed 41

6.4 Revenue from the pure random search 42

6.5 Shows highest revenue over the regression line 43

6.6 Shows the percentage completed compared to how much it costed 43
6.7 Revenue from the tabu search 44
6.8 Shows highest revenue over the regression line 45
6.9 Shows the percentage completed compared to how much it costed 45
6.10 Revenue from the simulated annealing 46
6.11 Shows highest revenue over the regression line 47
6.12 Shows the percentage completed compared to how much it costed 47
6.13 Revenue from the LAST 48
6.14 Shows highest revenue over the regression line 49
6.15 Shows the percentage completed compared to how much it costed 49
6.16 Probability of choosing each fleet size 50
6.17 Probability of choosing the different vessel 50
6.18 Variance compared to revenue 51

8.1 Simulated Annealing Values 94
8.2 LAST Values 94
8.3 Genetic Algorithm Values 94
8.4 Tabu Search Values 94
8.5 Pure Random Search Values 94
8.6 Simulated Annealing Regression 95
8.7 LAST Regression 95
8.8 Genetic Algorithm Regression 95
8.9 Tabu Search Regression 95
8.10 Pure Random Search Regression 95
8.11 Simulated Annealing Variance 96
8.12 LAST Variance 96
8.13 Genetic Algorithm Variance 96
8.14 Tabu Search Variance 96
8.15 Pure Random Search Variance 96
8.16 Simulated Annealing Pareto Front 97
8.17 LAST Pareto Front 97

8.18 Genetic Algorithm Pareto Front 97
8.19 Tabu Search Pareto Front 97
8.20 Pure Random Search Pareto Front 97
8.21 Simulation-based optimization loop 98
8.22 Simulation-based optimization loop 98
8.23 Simulation-based optimization loop 99
8.24 Simulation-based optimization loop 99
8.25 Simulation-based optimization loop 100
8.26 Simulation-based optimization loop 100

Abbreviations

SBO	=	Simulation-Based Optimization
MCS	=	Monte Carlo Simulation
MFSMP	=	Maritime Fleet Size and Mix Problems
MAB	=	Maximum Allowed Biomass
CO_2	=	Carbon Dioxide
NH_3	=	Ammonium
O_2	=	Oxygen
TAN	=	Total Ammonia Nitrogen
LAST	=	the Learning Automata Search Technique
SAS	=	Stochastic Adaptive Search

Chapter 1

Introduction

1.1 The fish farming Industry

Between 1992 and 2012 the Norwegian fish farming industry had a formidable growth. Over these 20 years the production volumes grew every year and in total it increased more than ten times over the whole period, according to SSB (1). In 2012 Olafsen (2) published a report about value creation in the ocean. This report states that the Norwegian fish farming industry, has a potential of growth up to five times the 2010 levels within 2050. In production volumes this is an increase from 1 million ton farmed fish in 2010 to 5 million tons in 2050.

Despite the potential presented by Olafsen, since 2012 the production volumes has stabilized at 2012 levels, SSB (1). Even without any growth over the resent years, there is still a lot of optimism within and on behalf of the industry. In 2015 a message from the Ministry of trade, industry and fisheries to the parliament, stated the five fold increase of fish farming volumes as a goal for the industry, Stortingsmelding 16 (2015) (3). Although

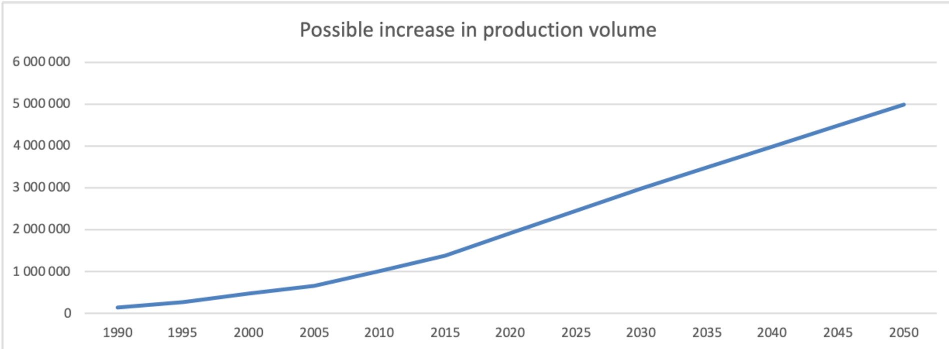

Figure 1.1: Possible increase in production volume

the production volumes has stabilized, the export value has more than doubled between 2012 and 2018, SSB (1). The demand for Norwegian fish is high, Tekna (2018) (4) and could be one of the key drivers for this increase.

To reach a five fold increase in production volume by 2050, Olafsen (2012) (2) and Ytreberg (2018) (5) addresses several different issues that needs to be solved. Four of these are listed below:

- Environmental impact
- Lice and diseases
- Lack of new farming areas
- Supply of fish fodder

Being able to control these problems are a precondition for the industry to grow again. Stein Lier-Hansen in Norsk Industri says the fish farming industry needs to be more innovative, and look to other industries for new technological developments Aarre (2018) (6).

Fish farming has up until now, only been done in coastal and weather protected areas. The high density of fish farms along the cost gives good conditions for fish lice and other diseases to spread. Environmentalists are raising concerns about the natural wildlife in the fjords. The three first issues listed above are all correlated, and these also needs to be solved before lack of fish fodder becomes a problem. One way of reducing these issues are to move fish farms further out in the sea. The possibilities of offshore fish farming has been investigated by the industry and several companies like SalMar (7), Nordlaks (8) and Norway Royal Salmon (9), have ongoing projects. While some farmers are moving offshore, others are aiming for a better and more efficient delousing strategies. Higher demand for Delousing and fish cages further out from the cost, causes new demands for the well-boats and their services. Several industry players have over the last years pointed out this lack of supply of well-boats services, Furuset (2019) (10), Nodland (2015) (11) and Solem (2017) (12).

This combined with a fivefold increase in salmon production could also mean a five times or higher demand of well-boat services. As well-boats delivers smolt to the fish farms and transports fully grown salmon from fish farms to slaughterhouses, the demand for them can grow proportionally with the industry. The trend of fish farmers investigating the possibility of moving farming locations further out in more exposed sea, could also change the operational requirements for well-boats, and ultimately have a huge impact on the composition of the future fleet. We already see the shift towards bigger and more technologically advanced well-boats, Kvile (2019) (13). With new fish farms in even more exposed locations, this trend of new builds is likely to continue.

1.2 Fleet sizing and fleet composition

To ensure new vessels are able to complete assigned missions in a cost effective way, different optimization methods could be an efficient tool. Optimization can be used in the decision making process to decide which capabilities a single wellboat or fleet needs, to fulfill contracted tasks. Vehicle routing, fleet sizing, fleet composition and a mix of these types of problems can often be very complex and close to impossible to describe mathematically. Even if the problem can be described mathematically it could still be to complex and require too much computational time to be solved. In such scenarios the problem or the required accuracy of the solution can be modified to make the problem solvable. One way is to simplify the problem and then solve it mathematically. Thus finding the exact solution to a simplified version of the problem. Another way is to only search through a small part of the whole solution space. This way you are not guaranteed to find the optimal solution, but you can within a short amount of time find a close to optimal solution. There exists numerous search algorithms for doing this. These techniques are often referred to as meta-heuristics and are meant for finding a solution that can be considered good enough, without actually searching through the whole solution space. These meta-heuristics are often used in simulation-based optimization.

With simulation, complex and stochastic systems can be modeled and estimated. Simulation is an imitation of the a real-world system or process over time, Banks et al. (2011) (14). Instead of writing an exact mathematical equation, a simulation model could give a far more realistic representation of the real-world problem. Compared to mathematical equations, simulation models can handle uncertainties and stochastic processes in a more realistic and uncompromised way. By utilizing Monte Carlo Simulation (MCS) a system with uncertainty can be analysed without increasing the size of the optimization problem, Ioannis (2003) (15).

This way of combining simulation with optimization is called simulation-based optimization (SBO), and could be a powerful tool within fleet sizing, fleet composition and mix of these problems. These types of problems are called maritime fleet size and mix problems (MFSMP).

Later in this thesis it will be discussed how simulation-based optimization (SBO) could be used to solve a maritime fleet size and mix problems (MFSMP). Different search algorithms, meta-heuristics, will be explained and tested against each other.

Chapter 2

System Description

This thesis will explore the possibility of utilizing simulation-based optimization (SBO) in a maritime fleet sizing problem. Several different methods and meta-heuristics can be used in SBO. By discussing strengths and weaknesses of the different methods, this thesis will try to answer why the different meta-heuristics are chosen, how they work and how they interact with the simulation model.

The simulation model needs to give an accurate description of the reel life problem, and still be neath and well adapted to the optimization algorithms. An operational analysis of well-boat operations will be the baseline for the simulation model. It is important for the model to give a good representation of the different tasks a well-boat performs, so it can give a trust worthy answer to the profitability of different fleet compositions.

2.1 The aquaculture industry

The aquaculture industry is one of Norway's biggest exporting industries and also one of the biggest aquaculture producers in the world. In 2018 the farming industry produced approximately 1,35 million tons of fish, with a worth of almost 68 billion NOK (16). The industry has grown significantly over the last decades in both volumes produced and created revenue. Although the growth in volumes produced has stopped over the last five years, revenues in the industry has continued to grow (17). Government regulations and permits for maximum allowed biomass (MAB) at fish farming locations, could be a reason for the latest years stagnation in produced volumes.

The Norwegian coastline is divided into 13 different production areas and then these areas are categorized after how much problems they have with diseases and fish lice. Production areas are divided into categories which decides how much increase in production or MAB you can apply for (18). The production areas are categorized with a color where red indicates problems with diseases and lice, yellow indicates some problems and green indicates few problems (19), see figure 2.1 below. Fish farms located in green areas are offered a 2 % annual growth in MAB (20). Besides this all fish farmers are also allowed to apply for up to 6 % growth in MAB, if some extra criteria are met.

Despite the latest regulations and the stagnation of production volumes, the government still has a goal of a fivefold increase in production volumes. To reach this goal new technologies and methods needs to be implemented in the industry, to deal with problems related to diseases, lice and escaped fish. Closed fish cages, exposed fish cages further away from shore and better lice treatment are some suggested solution to the problem. The two last solutions could also mean an increased demand for well-boat services. Exposed fish cages means smolt and fully grown fish needs to be transported over longer distances and better lice treatment could result in more delousing missions for the well-boat fleet.

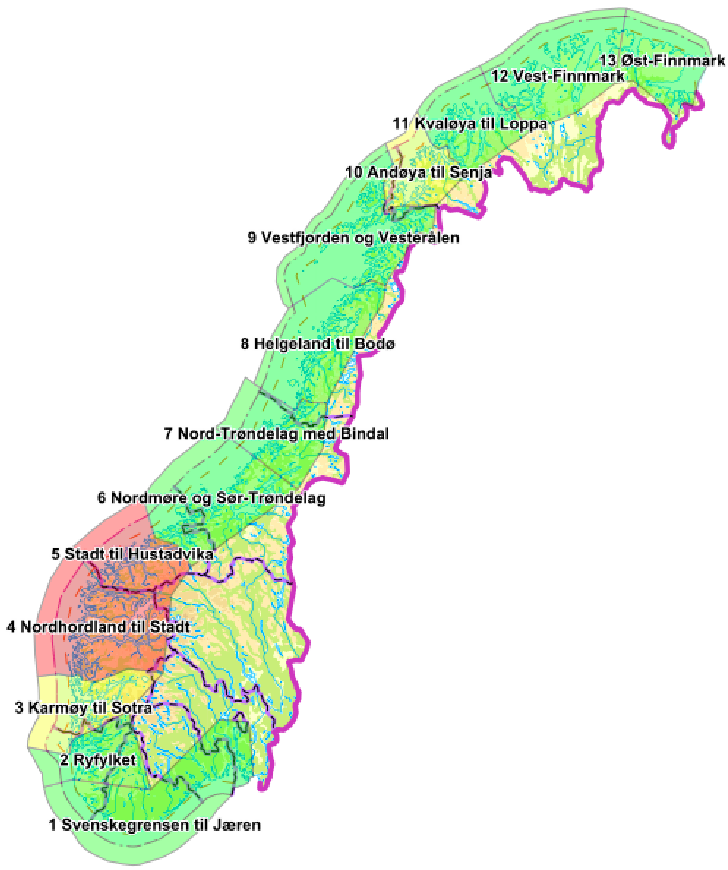

Figure 2.1: Production areas by traffic light categories

2.1.1 The fish farming value chain

The fish farming value chain is an almost 3 year long process from fish roe to the main course on the dinner table, laks.no (2020) (21). The whole process starts onshore at the smolt facilities. Here fish roe are hatched into fry in fresh water tanks. Then the fry grows into smolt, and after 10 to 16 months the smolt are ready to be put out in the sea. Before the smolt could go on a well-boat and out in the sea, it has to be prepared for the salty sea water. In the last period in the smolt facility the salinity in the water tanks are gradually increased. This way the smolt get used to life in the sea.

At this point the well-boat are introduced into the value chain. The smolt are pumped into the wells of the well-boat, and carried out to the fish cages. Here their again pumped out off the wells and into the cage. The transportation stage onboard the well-boat are a very vulnerable and stressful for time the smolt. Thus there are strict regulations for the welfare of the smolt.

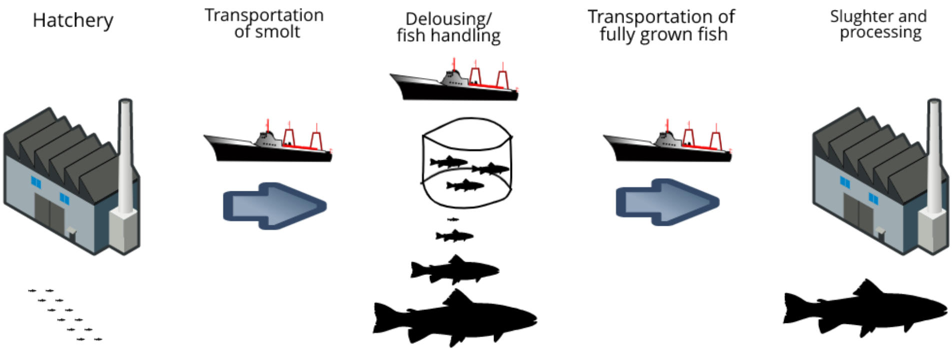

Figure 2.2: Life cycle of farmed fish

After the smolt has been put in the sea, it will stay in the fish cage until it reaches its slaughter weight of around 4-6 kilograms. This usually takes anything from 16 - 22 months, Nofia (2020) (22). During the growth period in the sea, well-boats will often assist with delousing. On average the fish will go through delousing 2 - 3 times during its lifetime, NTS ASA (2019) (23).

After the fish are fully grown to optimal slaughter weight. They will again be transported by a well-boat, to the slaughter facility. Here they are either pumped over to holding cages or they are pumped straight into the slaughter house. Although the fully grow fish are more robust than the smolt, transportation by a well-boat is still stressful for the fish. Fish welfare has to be one of the highest priorities to prevent any losses.

When the fish first goes into the slaughter house, the whole process from a living fish to a finished product in the stores, has to be streamlined and efficient. Fresh fish has a higher market value than frozen fish. Therefor slaughter, meat processing and distribution needs to be one big operation. As Norwegian fish are distributed to more than 100 countries, Chramer (2017) (24), it is required a high level of efficiency in slaughtering, processing and distribution.

2.2 Well-boats

The main task of a well-boat is to transport live fish in a safe and efficient way. Always maintaining the welfare of the fish and keep stress to a minimum during the transport, is important to minimize fish mortality and to ensure the quality of the meet. Government regulations are strict and requires constantly surveillance of the water in the wells. The regulations regarding water quality are also dependent many parameters, like the size of the fish the well-boats are carrying, and whether the well-boat are sailing with closed or open wells.

Most well-boats has the choice whether they are sailing with closed or open wells. Sailing with open wells means that seawater are circulating through the wells. The well-boat is pumping fresh sea water into the wells and the old water out. This way the water quality is always kept at a high level. The constant change of water in the wells, keeps carbon dioxide CO_2 and ammonium NH_3 at low levels, and oxygen O_2 at a high level. When sailing with closed wells, the water in the wells are not renewed only recycled. Instead of taking in fresh water, the water already in the well is constantly cleaned and filtered. There are strict regulations for when well-boats are allowed to run with open wells. Especially when going out of areas that has problems with lice and diseases, like the PD-zone (zone with high occurence of Pancreas disease) Forskrift om transport av akvakulturdyr (2008) (25).

When sailing with closed wells NH_3 and CO_2 needs to be closely monitored as these are toxic for the fish, Rosten (2010) (26). NH_3 and CO_2 are wastes form the fish's own metabolism. Other parameters critical for the fish's welfare are Total Ammonia Nitrogen (TAN), level of oxygen, temperature and pH-level.

Well-boats are designed to be able to carry out the three main well-boat missions. These missions are:

- Transportation of smolt from hatcheries to fish farms
- Transportation of fully grown fish from fish farms to slaughterhouses
- Performing delousing at the fish farms

2.2.1 Transportation of smolt from hatcheries to fish farms

Transportation of smolt from hatcheries to fish farms is the mission that requires the most delicate treatment of the fish. Usually the density of fish in the wells is not more than 35-50 kg/m^3, Heen (2015) (27). The fish is loaded onboard the vessels at the fish farm, then they are transported out to the fish farm and pumped over to a fish cage. Smolt transportation have some extra hygienic regulations, before transporting smolt the wells needs to be disinfected and wait for 48 hours. If the same well-boat is transporting more smolt from the same hatchery to the same fish farm it does not have to wait for the 48 hour quarantine time. Because of this quarantine and other regulations it is usually the same ships that always carries smolt.

2.2.2 Transportation of fully grown fish from fish farms to slaughter houses

Transportation of fully grown fish from fish farms to slaughter houses is the most commonly task a well-boat can do. These missions stands for about 60 % of all performed well-boat missions, Nodland (2015) (28). Before the fish can be transported from the fish farms they have not been fed in a few days. This starvation calms the fish and gives less contamination of the water inside the wells. When pumped down into the wells the fish are counted and measures are taken to keep track of the biomass pumped into the wells. Arriving at the slaughterhouse the fish could either be pumped into a waiting cage or directly into the slaughterhouse. This last method is quite a bit slower, but it eliminates the risk of lice and other diseases. After every transportation the wells are disinfected.

2.2.3 Performing delousing at the fish farms

Performing delousing at the fish farms is something most fish experiences during their lifetime, on average 2-3 times. There are three main methods for lice treatment, chemical, mechanical and biological. Chemical treatment is usually done with hydrogen peroxide, but lately a new technique using heated water has been introduced. But now this method has been proven to be painful for the fish, Mattilsynet (2019) (29) Mechanical delousing is done by pumping the fish onboard and sending them through brushes to brush away the lice. With different sized fish this method could harm bigger fish, and will not be very effective on smaller fish. Biological delousing is done by introducing wrasse or lumpfish into the cages. Wrasse and lumpfish feed on fish lice and they eat the lice right off the fish.

Chapter 3

Problem Description

This chapter will describe the problem that will be solved later on in chapter 6. To give a proper description of the problem, this chapter will try to explain the structure of the problem, and which assumptions that had to be made. How uncertainties are dealt with and how demand frequencies for different well-boat services are distributed throughout the year.

3.1 Defining the optimization problem

The full problem is to find an optimal well-boat fleet to solve a number of different tasks in the way that creates the most revenue for the stakeholders. This is a huge problem that will be close to impossible to solve to optimality. In a problem like this with lots of uncertainties and a solution space too big to calculate, simulation-based optimization is often the preferred optimization method.

In this thesis the generation of missions for the well-boat fleet is assumed to be stochastic. That means missions are generated without any warnings. In a real life situation up coming demand for well-boat services will be warned some time in advance. To adjust for this the vessels are given more time to start and complete the mission. Therefor mission generation is treated as stochastic occurrences, although it's not completely random. Smolt are usually put out in the sea either in the spring or in the autumn, and most of the fish are harvested roughly 1.5 years later. This makes spring and autumn high season for transportation of live fish. These variations throughout the year will be accounted for in the simulation model, and is describe further in chapter 5.

Now the problem is reduced to a fleet sizing and fleet composition problem. One way to simplify it even further would be to see it only as a question of capabilities for the fleet. How much is needed of each service each year, and then solve it mathematically. To better account for the stochastic nature of the problem, a more accurate problem imitation, could be found with SBO. In a simulation model the stochastic behavior of services demanded will be handled in a more realistic way. When a demand occurs a ship could fulfill it although the demand is smaller than the potential capacity the well-boat could supply.

The next step in solving the problem is to decide on a set of decision variables. The decision variables will be a set of different well-boat types and how many of each type. How to organize this in the optimization model will be discussed later on while describing the different meta-heuristics used. One way is to set each well-boat type as a variable and let it denote how many well-boats of that type there are in the fleet. Another method is to set the fleet size to be one variable and then create as many new variables as there are well-boats in the fleet, then each of these variables takes a value that are associated with one type of well-boat. The structure of the decision variables could effect the way the optimization model searches through the solution space, and hereby effect the end result.

3.2 Creating the simulation model

The simulation model can be seen as an imitation of the real life operational problem. Its input is a predefined fleet and the output are some sort of comparable performance measure, like total cost, total income, mission completion percentage, total revenue etc. See figure 3.1 for a schematic representation of well-boat operations.

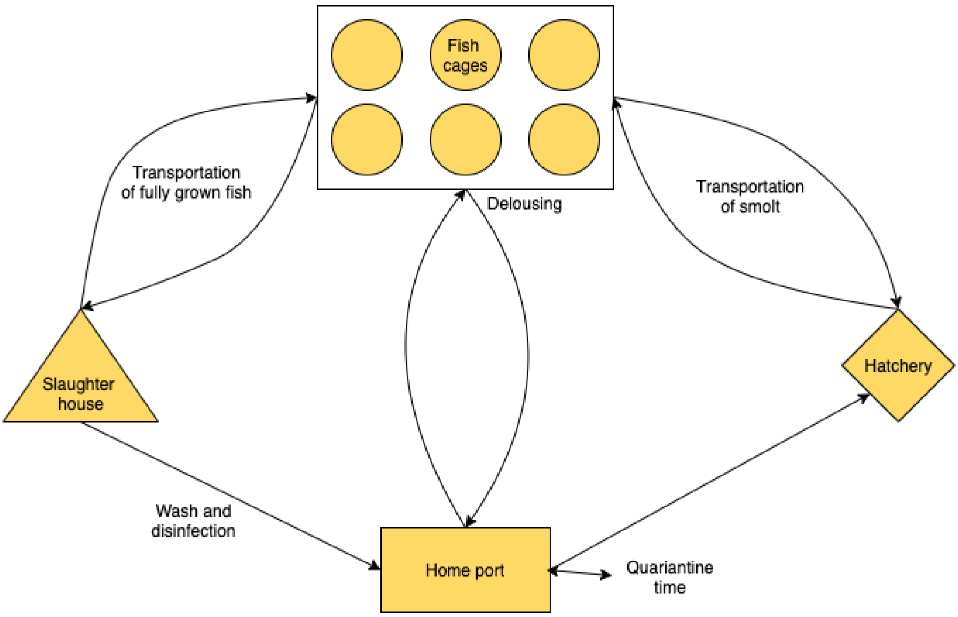

Figure 3.1: Well-boat operations

When describing the real life problem and representing it in a simulation model, several assumptions had to be made. First of all the model presented is a generic model and does not represent an actual problem. Locations for home port, hatcheries, slaughter houses and fish cages are chosen based on normal sailing distances for well-boats. These distances where estimated by analysing AIS data for well-boats. Areas with extra restrictions regarding disinfection and quarantine time where not taken into account in the model. Neither does it differentiate between sailing with open or closed wells.

Although the optimization model dose not consider ship routing, the simulation model still needs a set of rules to define how different missions are assigned to different vessels. When a mission is generated and there are several ships at quay to choose from. Which ship gets chosen? When a vessel gets free and there are several missions to choose from. Which mission gets chosen? By applying different heuristics for these choices into the simulation model, we could let the optimization model find the best heuristic by treating them as a decision variable. This greatly increase the complexity of the model and has not been done in this thesis, more about it in chapter 5.

Chapter 4

Literature Review

4.1 Maritime fleet sizing and mix problems

Imai and Rivera (2001) (30) describes the problem of fleet sizing as, finding the optimal number of vehicles to satisfy a given demand for loaded trips. With this formulation the goal is to find the exact optimal solution to the problem. By formulating an exact mathematical optimization model this could be done, but with the complexity of a maritime fleet sizing and/or scheduling problem, the problem can often be to big to calculate.

Alvares et al. (2011) (31) proposes a mathematical solution to a bulk shipping fleet sizing problem. They formulated a mixed integer programming model. They simplified the real life problem by idealizing some aspects of the real business case. For instance they constrained the problem in time and geographical location, to make the model solvable within reasonable time. They also solved the the problem for several different scenarios representing different levels of risk tolerance. This way they could compare the results and evaluated the trad off between stability and profitability.

Fagerholt et al. (2009) (32) proposes a simulation-based optimization method for solving strategic planning problems in maritime tramp and industrial shipping. They combine Monte Carlo simulation with an optimization framework to make a decision support system for short term routing and scheduling. By implementing different decision heuristics in the simulation this methodology is able to evaluate and compare how these heuristics handles different scenarios. The strength of this method is that it is able to handle stochastic variables in the routing problem. It is also a flexible algorithm that easily can be configured to support a wide range of problems, like fleet sizing or analysis of long term contracts.

Shyshou et al. (2010) (33) developed a simulation-based model to evaluate what effect the change of future spot prices on the number of long term ATHS hires. Much effort where put into the development of a realistic simulation model, to get a detailed representation of the real life situation. They aimed at finding the optimal fleet size for a range of different future spot rates. In conclusion of the study they found that the the fleet size where quite sensitive to lower spot rats and on the other hand more insensitive to higher spot prices.

4.2 Simulation-based optimization

In simulation-based optimization, simulations are used to evaluate the value of a feasible solution. The simulation model takes input parameters and creates an output or a value for the given input parameters or solution. Then the optimization program processes this output to create a new solution, which again is run through the simulation model. Simulation-based optimization (SBO) can be seen as an automated process based on numerical simulations and mathematical optimizations algorithms, Attia (2012) (34). A SBO model can be thought of as a loop, see figure 4.1. It runs through the same algorithm, every time changing the input parameters trying to find a better solution. The model will always store the best solution so far, and run until it gets stopped by a predetermined stopping criteria. This could be a certain amount of time or number of loops, or it can be set to stop by a convergence criteria or time without improving the best solution.

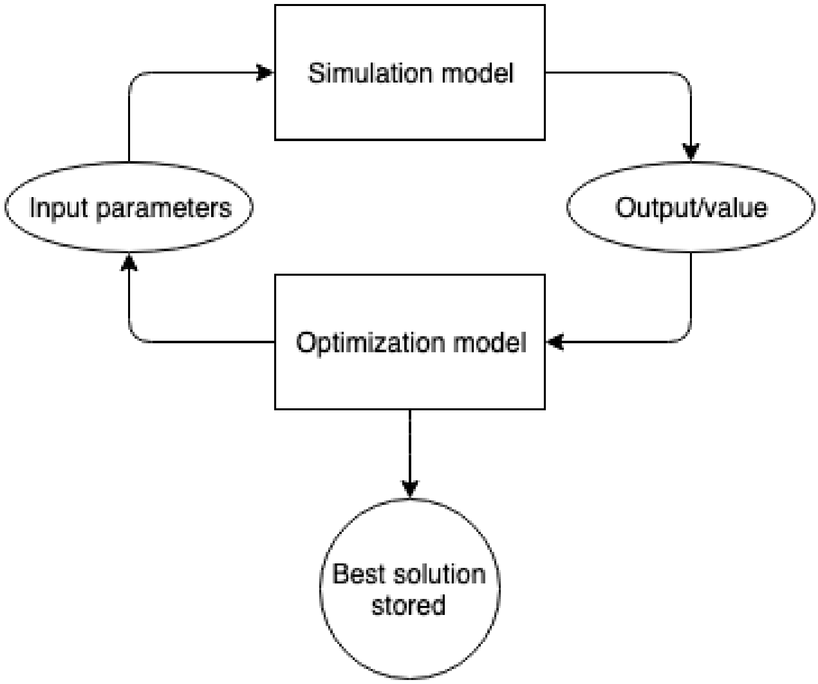

Figure 4.1: Simulation-based optimization loop

One of the strengths of simulation-based optimization (SBO) is that it can reduce the computational time, which again leads to cost and time savings. The idea of SBO is that you can find a good enough solution in an short amount of time, without having to compute all feasible solutions. Unlike many other optimization methods SBO will not find the mathematically best solution, but can rather be seen as a cost effective way of finding a close to optimal solution. Another strength of using simulations is its flexibility in modeling real life situations. Utilizing simulation models allows for an accurate description of the real life problem

In a review on simulation-based optimization methods applied to building performance analysis Nguyen et al. (2014) (35) Divides SBO into three phases:

- Preprocessing

- Running optimization

- Post processing

The preprocessing phase consists of building a SBO model, setting constraints and objective function, defining decision variables, selection of optimization algorithm and coupling the algorithm with the simulation model. In this phase it is important to understand the real life system to be able to model it and choose the best suited optimization technique. Especially to be able to find the balance between over simplification and over detailing the model. To make the right assumptions and screening out non significant variables is key in this phase.

While running the optimization the main task is to monitor the convergence of the solution and to detect errors. This is to avoid unnecessary computational time. In a SBO model convergence does not mean that the final solution is reach but rather that the algorithm are getting problems finding better solutions. Calculating the speed of convergence or the time the algorithm uses to converge could be close to impossible. There are done some studies trying to calculate the speed of convergence. One study worth mentioning is Wright and Alajmi (2005) (36). They investigated the robustness of a genetic algorithm. They tested the speed of convergence with population sizes of 5, 15 and 30, and found that the smaller population sizes with a high possibility of crossover and mutation found the best solution.

The post processing phase is where data is collected and analyzed. Data is organized in different charts and diagrams, commonly used methods are Pareto front analysis, convergence plots, variance plots and sensitivity analysis.

Evins et al. (2012) (37) proposed a slightly different approach to the phases of SBO. Evins 4 phase approach looks a little like the one described above but has some differences. The main difference comes in the optimization phase, where Evins has divided it into to phases, initial optimization and detailed optimization. In the initial optimization the 21 variables are only allowed to vary between low, medium and high. By analysing the Pareto front of the initial result some variable could be ruled out. These variables did not change along the Pareto front and could thereby be locked as constants. To the remaining variables additional steps in the variables ranges where implemented and a new round of optimization was run. This optimization round was called detailed optimization. The more variables and the wider the range of these variables are, the more suited this technique will be.

4.3 Discrete parametric optimization

Gosavi (2015) p. 29 (38) describes parametric optimization as follows: *Parametric optimization is the problem of finding the values of decision variables (parameters) that maximize or minimize some function of the decision variables*.

A typical parametric problem is to maximize or minimize an objective function, f(x(1),x(2),...,x(N)), with a set of N decision variables (x(1),x(2),...,x(N). If the closed form of the objective function is known and linear, the problem can be solved with linear programming techniques, like simplex. If the objective function f(.) on the other hand is unknown or too time consuming to calculate, the use of linear techniques will not be possible. Sometimes f(.) consists of stochastic elements like a probability or density function, that are to hard to obtain in closed form. In these cases simulations can be a powerful tool to estimate the objective function f(.).

Discrete parametric optimizations denotes a non continues function, as the function may has gaps. With these types of function derivatives may be of little use even knowing the closed form of the function. By assuming a finite solution space, it will be possible to use simulations. With simulations the function can be estimated at any given point.

For problems with a smaller solution space brute-force or exhaustive search can be used. This mean to make an estimation of all possible solutions to obtain the optimal solution. Sometimes the solution space gets to big to calculate an estimation of all possible solution. Then there will be a need for a technique that can quickly search through parts of the solution space, and still find a close to optimal solution. Meta-heuristic and stochastic adaptive search techniques that can do this. These techniques work well when solving discrete combinatorial optimization problems. Stochastic adaptive search techniques often have well understood convergence properties while these properties are unknown for meta-heuristics. Although the convergence properties of meta-heuristics are unknown, these techniques still work well in practice.

4.3.1 Meta-heuristics

Meta-heuristic methods can be seen as a way of guiding the search process through the solution space. They are not aimed at finding the optimal solution but rather a good solution within a reasonable amount of time. In problems with a large solution space these techniques work very well. Meta-heuristics are an extension of the local search method. Where a local search method can get stuck in a local optimum meta-heuristics can be used instead to avoid this, Lundgren (2010) (39).

Two commonly used meta heuristics are the genetic algorithm and tabu search. They both starts with a feasible solution or a population of feasible solutions and then tries to improve the best solution(s) available.

The genetic algorithm is based upon evolutionary theories where only the fittest survives. The algorithm lets the fittest (best) solution in the population to reproduce and the new solution will replace the worst solution in the population. There are several different ways this reproduction can be done and has to be fitted to the actual problem at hand. One way of doing this will be to make small changes in one or more decision variables in the fittest solution. Another way, could be to pair the the two best solutions and take half of the decision variables from each.

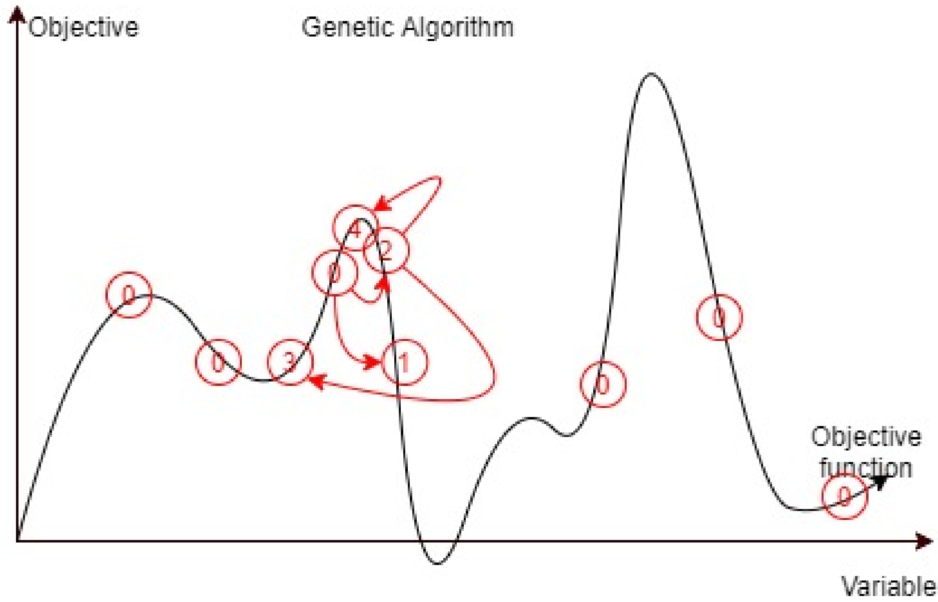

Figure 4.2: Genetic Algorithm

Figure 4.2 above, is a visual representation of how the genetic algorithm work. First it randomly picks out a initial population, denoted 0. Then the best solution is used to produce a new solution. As the new solution tends to be close to the solution used to produce it, the algorithm struggles with breaking out of local optimums. Thus often several of the best initial solutions are used for reproduction. This way the probability of getting stuck in at local optimum is lowered and the probability of finding the global optimum is increased.

The tabu search method was introduced by Glover in 1986 (40) as a meta-heuristic to avoid cycling in the search. The tabu search separates it self form other meta-heuristics by storing a list of previous moves of the variables in the solution. This list can be made in many different ways. It can store single moves or combinations of moves, or even the opposite move. But it does not store moves that are already on the list.

The moves in the tabu list is considered forbidden. In this way the tabu search stops itself form cycling. Thereby allowing the algorithm to efficiently search through a larger part of the solution space. The length of the tabu list has to be adjusted after the size of the problem. The bigger the problem the bigger the tabu list needs to be. A too short tabu list may not prevent cycling and a too long one can cause the algorithm to wonder too much in the solution space.

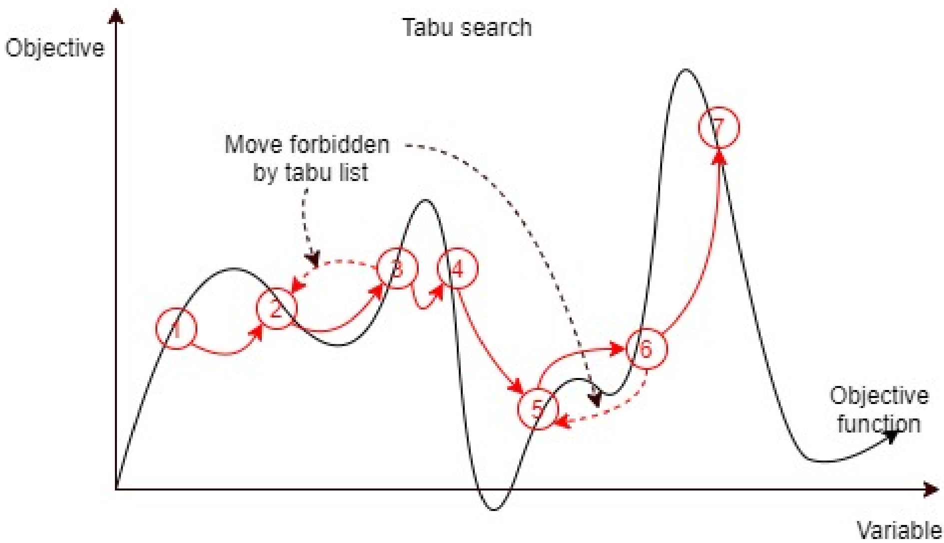

Figure 4.3: Tabu Search

In figure 4.3 it is shown how the algorithm wanders around in the solution space. The tabu list prevents the search from visiting solutions or areas it has visited before.

The tabu search is also very useful for tightly constrained problems, as it can temporary move through infeasible solutions, Cordeau and Laporte (2003) (41). A general tabu search stores the best solution but uses the current solution to make a new solution. This way allows for more wondering of the algorithm.

4.3.2 Stochastic adaptive search

Stochastic adaptive search (SAS) techniques are actually meta-heuristics with proven convergence properties. This means that these techniques guarantees to converge towards the global optimum.

The pure random search is categorised as SAS although it does not adapt. But the technique is stochastic and guarantees convergence. It is completely random and are often used as a baseline to test the effectiveness of other techniques. Pure random search starts with setting probabilities for selecting any value in the decision variables, the then randomly search until it meets the selected stopping criteria.

The learning automata search technique (LAST) is quite similar to pure random search. It starts like the random search but adapts with every iteration. For every iteration the technique updates the possibilities for choosing a specific value of the decision variables. The values that earlier have shown to give good solutions gets favored, and becomes more likely to be chosen again.

All probabilities are stored in a probability matrix which changes with every simulation. To change the probability matrix in effective way, some knowledge about the expected solutions could be required. The expected range of the solution values and the expected amount of iterations, will influence the gain, or the factor used to adjust the probabilities after every iteration.

This is shown in figure 4.4. In the figure, darker areas shows areas where the next solution has a higher probability of ending up. As seen, the probabilities changes with every solution. A good solution gives a slightly higher probability of the next solution ending up in the same area. A very good solution increases the probabilities even more, and a bad solution decreases the probabilities.

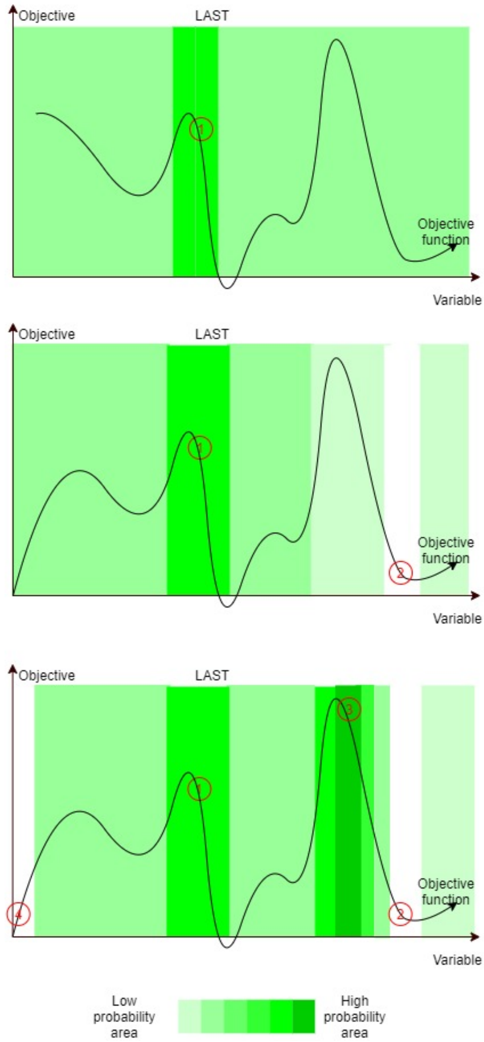

Figure 4.4: Learning Automata Search Technique

Simulated annealing starts at a randomly selected solution and then moves to a neighbour. If the neighbour is better or equal to the current solution it moves. If the neighbour is worse than the current solution there is still a low possibility for it to move. This type of move is called exploration and is the reason this technique can break out of a local optimum. As the search goes, the possibility of moving to a worse neighbour is lowered. The best solution is always stored along the way so it does not get lost. Simulated annealing was a little breakthrough when it was discovered, and has shown remarkable results in solving both small and large problems.

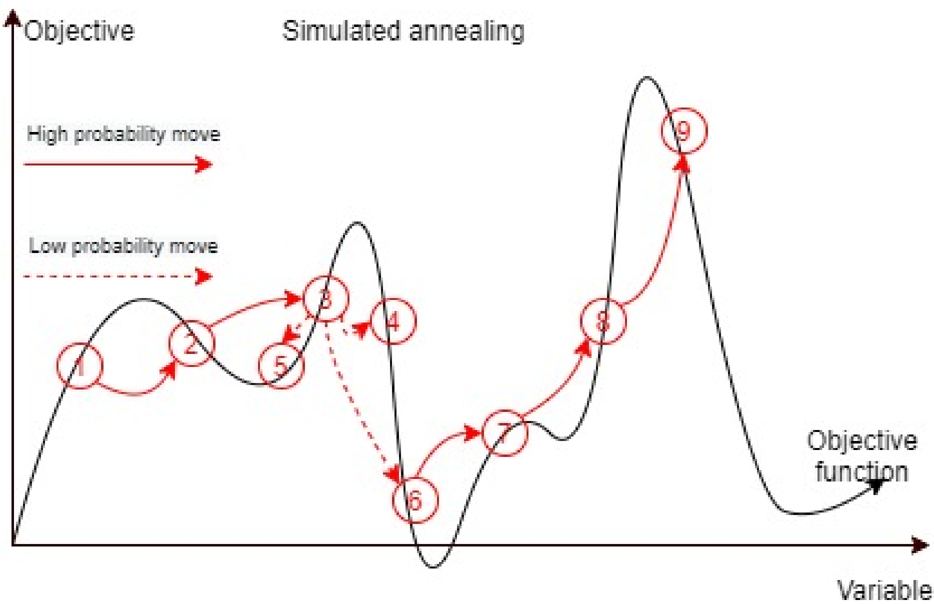

Figure 4.5: Simulated Annealing

The simulated annealing algorithm moves from the current best solution to the next solution, just like one variant of the genetic algorithm. But it always has a low probability of using the latest solution instead of the best solution to produce a new solution. In figure 4.5, this is shown when the sixth solution is used instead of the third to produce the seventh solution. After it has made this move it does not come back to the third solution, but continues from the sixth and the third solution is stored.

This was just a short description of some meta-heuristics and SAS techniques. This field is constantly evolving because of its ability to solve problems that are too big for analytical methods.

Chapter 5

The Simulation-Based Optimization model

5.1 The simulation model

This simulation model is an imitation of real life well-boat fleet operations. It is incorporated in an optimization algorithm, as described in the previous chapter. The model is a discrete-event simulation as it is modeling a system that changes over time in a way where state variables change instantaneously at separate points in time, Law (2000) (42). This means that its is a non continues model which changes its state every time a well-boat starts or finishes a mission.

This simulation model has been made in cooperation with Hans Tobias Slette, see figure 5.1 for an overview of the model. At the start Slette was supposed to use the model for his own article, but in the end he didn't use this model at all. The signatory has done all the model design, sketching up the model, deciding how it should react to different input and scenarios (weather, mission generation, mission/vessel matching) Slette has implemented this in simulink and written most of the code in the simulation model. The signatory has coded how missions are generated and and how missions and vessels are matched in the model.

The input for the simulation model is a selection of preset parameters, and a set of decision variables given by the optimization model. The model starts with generating a mission which then gets matched with a vessel. If no vessel are free the mission is stored until a vessel gets ready. After a mission is matched with a vessel, weather conditions has to be within set boundaries before the vessel can leave port. Here the model splits in four roads, one for each mission type. Dependent on mission type the well-boat sails straight to the fish cages or stops by the hatchery first. When at the fish cages, a new weather check is done, before operations at the cages can start. For transportation of fully grown fish, the well-boats now sails to the slaughter house for unloading. Before sailing back to port, the model check if there are any outstanding volumes left in the mission. If it is, the vessel goes back in the loop.

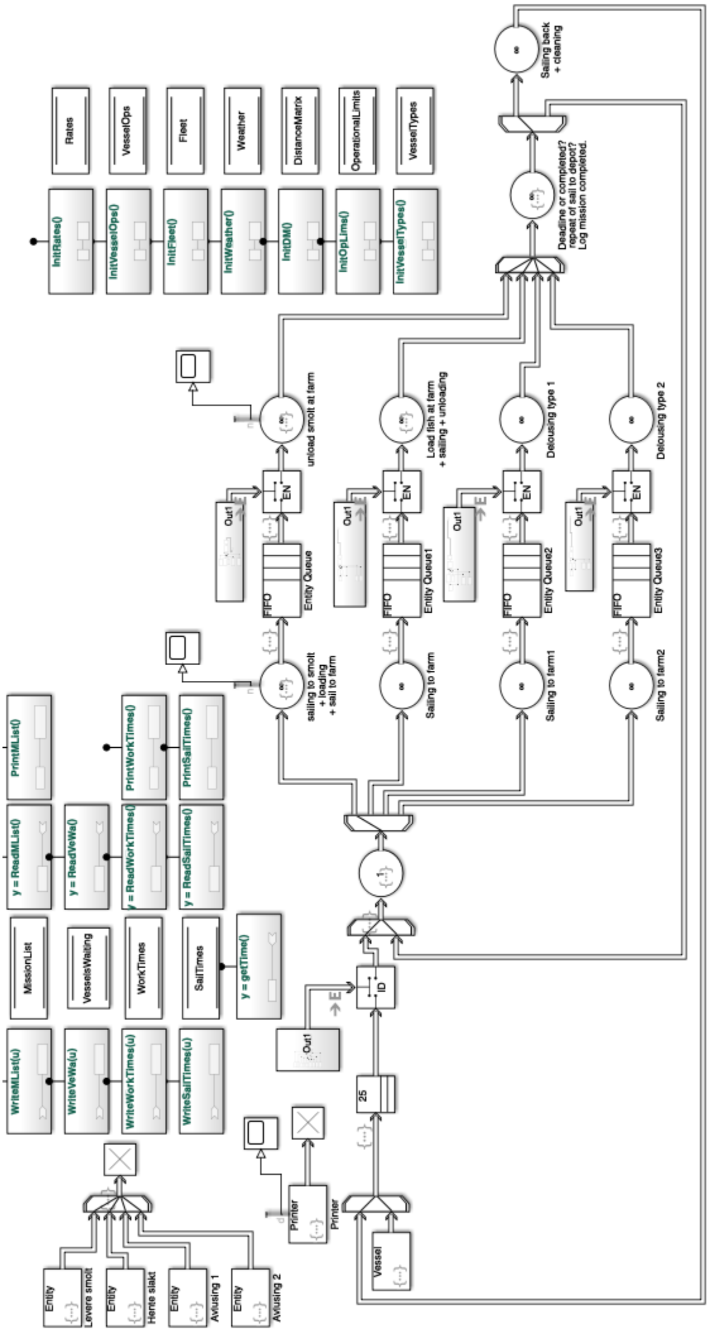

Figure 5.1: Simulation model

5.1.1 Input and assumptions

The simulation model simulates how a well-boat fleet solves generated missions during the course of a year. The model takes in the number of each vessel type in the fleet, and calculates the the profits this fleet makes in the course of a year. The capabilities and costs for each well-boat can be read in table 5.1 and 5.2.

Vessel type	Capacity in ton salmon/ smolt	Speed knots	Capex mNOK	Sailing cost mNOK/ hour	Work cost mNOK/ hour	Crew cost mNOK/ hour	Delousing 1 capability	Delousing 2 capability
1	480/160	12	15	0,0019	0,00072	0,0022	yes	yes
2	480/160	12	12,5	0,0019	0,00072	0,0022	yes	no
3	225/75	10	10	0,0013	0,00062	0,0019	yes	no
4	180/60	11	10	0,0013	0,00062	0,0019	yes	no
5	150/50	11	7,5	0,0010	0,00045	0,0016	yes	no

Table 5.1: Capabilities of each well-boat type

The 5 vessel types are based on vessels in the fleet from Norsk fisketransport AS (46). Values for sailing- and work cost are estimated from power use in the different scenarios (23; 43; 44; 45). In appendix figure 8.21, 8.22, 8.23, 8.24, 8.25 and 8.26 power use for to different vessels can be seen. Extrapolation has been used to adjust for different vessel sizes and speeds. Fuel consumption is assumed to 200 grams/kWt and fuel prize to 4000 NOK/ton.

The work capabilities for each well-boat are presented in table 5.2

Work type	Smolt loading	Somlt unloading	Salmon loading	Unloading waiting cage	Direct unloading	Delousing 1	Delousing 2
boat type 1	80	100	150	200	80	100	350
boat type 2	80	100	150	200	80	100	0.001
boat type 3	60	75	150	150	80	75	0.001
boat type 4	60	75	150	150	80	75	0.001
boat type 5	50	50	100	100	80	50	0.001

Table 5.2: Work capability rates for each type of well-boat presented in tons/hour

The different missions that could be generated are shown in table 5.3

Mission type	Mission #	Loc 1	Loc 2	Work loc 1	Work loc 2	Mission size in ton	Rates in mNOK/ ton fish
Smolt	1	2	5-15	1	2	50-500	0,0021
Fully grown fish	2	5-15	3-4	3	4-5	200-1200	0,0007
Delousing 1	3	1	5-15	0	6	200-1200	0,00063
Delousing 2	4	1	5-15	0	7	200-1200	0,00063

Table 5.3: Mission types and structure

Most wellboats are working on long contracts while some are available on the spot market. In today's spot market most well-boats operates with day rates, but still some operate with rates per kilo treated or transported fish (43). The shift towards day rates is a way to move the short term risk from well-boat owners to fish farmers. As the farmers needs to take risk of unexpected events (weather, minor malfunctions, etc.). In the long well-boat owners bears the cost of this risk. To be able to account for weather and other uncertainties, rates per ton treated or transported fish has been chosen as the way to reward the vessels when completing a mission.

The time between two missions of similar type, are generated following a normal distribution times a sinus function. Smolt missions are mainly in the spring and fall and the corresponding sinus function goes almost to zero in the winter and summer. Transportation of fully grown fish comes on average 1.5 years after the smolt has been set out. Thus this function has a straighter shape but also peaks in the spring and fall. The probability of diseases and fish lice grows With the time the fish spends in the cages. Therefore most delousing missions are generated in between when the smolt is set out in the cages and the harvesting. In figure 5.2 the three different sinus functions are sketched up. The actual mission generation density will not necessarily look like this as it is a stochastic process. But during the course of the Monte Carlo simulations the average will look something like this.

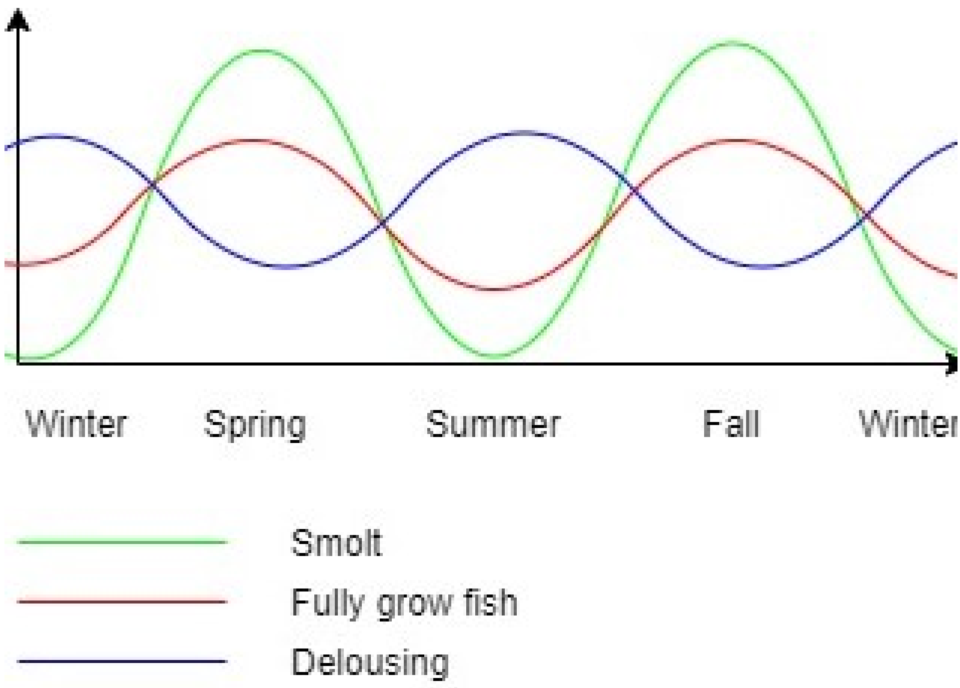

Figure 5.2: Mission generation

Since it is easier to have all different missions structured in the same matrix, vessels without delousing of type 2 capabilities, are here said to have a very small capacity. The model needs a value above 0, but a very low rate will prevent the vessel from being chosen to perform the mission. Time for crew change, loading of bunkers and maintenance have not been implemented in the model. Weather conditions are assumed to have an effect on the missions and each boat has a different tolerance for bad weather depending on its size. The weather is checked before the boats sails from depot and again before it stars working at a location. Different heuristics for what each boat should do if the weather is not good enough could be implemented in the model, and solved by the optimization model.

The vessle - mission matching is done by making a priority matrix including every possible match. First the expected time each ship will use to complete each mission is calculated. Then their priority will increased by 1 if the matching can fulfill some extra criteria. For smolt missions boats who comes fro a smolt mission or has already waited 48 hours has higher priority. For all missions the priority is increased if the vessel is able to complete the whole mission alone. If weather is well within the operational boundaries for the vessel, priority is increased. In the end the vessels who can perform the mission at the lowest expected cost will be given a higher priority. How much priority a match should be given in each of these instances, is whole optimization problem in its own. It is not considered in this thesis as it will add to much complexity compared to the available time.

The rest of the input for the simulation model are listed in the appendix.

5.2 The optimization model

In general the optimization model chooses the decision variables which then again are sent to the the simulation model. The simulation model sends back the output value. Based on this output the optimization model changes the decision variables, and sends it back in the loop. As described in chapter 4 there are many different algorithms for how the optimization model chooses these decision variables. In this thesis five of these algorithms have been tested up against each other, two meta-heuristics and three SAS techniques. All five techniques are coded in Matlab and simulations have been run in Simulink. All Matlab codes can be found in Appendix, and an overview of the simulation model is found earlier in this chapter.

The main measures for the performance of the different algorithms are: Time used and value of the best solution. The best value found so far is stored in a vector after every iteration. This vector is again used to make a regression line. When the derivative of this nonlinear regression line sinks below a certain level the optimization is stopped. The time it takes to find a slightly better solution, is from this point expected not to be worth it. The best solution so far is considered good enough. There are numerous other ways of stopping the optimization, maximum number of iterations, maximum number of iterations after best solution is found, convergence criteria etc. The method explained above, combined with a minimum number of iterations is a way stopping the optimization which also give feedback on how fast the algorithm searches through the solution space.

Since the computer running the simulations also were used for other tasks, the actual time used is fluctuating and not comparable. Instead the number of iterations used could be a better measure for how fast the algorithm finds this solution. By using number of iterations as a measure of time, the complexity and time it takes to run the optimization code is not accounted for. But in a case like this it is much smaller than the time used to run the simulations, and can therefor be neglected.

In the optimization models fleet size is treated as one decision variable with a range from 3 to 15. Each vessel is also considered as a decision variable and could take numbers in the rage from 1 to 5. This way the fleet size variable decides how many variables that is in the model at any given time. Structuring the decision variables this way is thought to be better suited for the genetic algorithm and the simulated annealing. Setting the vessel types as decision variables, will require additional restrictions to avoid possible ”neighbours” to be too far apart.

Pure Random Search

The pure random search technique is the simplest of all techniques and works as the baseline to measure the performance of the others. It picks the decision variables by random every time, without considering any output from previous iterations.

Genetic Algorithm

The genetic algorithm takes the best or a selection of the best solutions and tries to build further on these. In this thesis it has been chosen to always build on the two best solutions so far. This is a compromise between quickly moving towards a optimum and the risk of getting stuck in a local optimum. The initial population was set to 8. Which again is the same compromise.

Tabu Search

The tabu search is set up to quickly search through the whole solution space. Instead of only searching in a small area, like the genetic algorithm tends to do, the tabu search always moves to the next solution. To prevent cycling or visiting previously visited solutions, the tabu list forbids the algorithm from doing certain moves. The tabu list could be made and used in several different ways. the use of the tabu list has to be suited for your decision variables. When writing the tabu list the actual move or the opposite move for every decision variable could be put in the list. A move from one solution to another could be forbidden only if the whole move is in the tabu list, or if only one decision variable does a move that's in the tabu list. The problem as it is solved in this thesis has a lot of decision variables but not that many values each decision variable can take. Therefor making the whole move forbidden will not be an effective way of leading the algorithm around the whole solution space. While making the move forbidden if one decision variable makes a forbidden move, will make everything forbidden. Here a mix between these two has been used. If 2/3 or more of the decision variables makes a forbidden move, the whole move is considered forbidden. Without doing it like this, the tabu list would either be too long or to short to be considered expedient for the number of iterations. Also a move is forbidden if the solution has been visited before.

LAST

The LAST randomly picks the next solution, but the probabilities for picking a specific value for a specific decision variable changes. For every iterations these probabilities changes depending on whether the solution was good or not. This way the algorithm spends more time searching in promising areas, without getting stuck in a local optimum. The difficult part of this method is to choose the My/Mu constants. These constants decides how much the probabilities changes in every iteration. Setting these too high and the algorithm will quickly narrow in on one area without truly exploring the whole solution space. Too low and it will use unnecessary much time before narrowing down in a promising area. Here these were set by running small scale tests of the algorithm.

Simulated Annealing

The simulated annealing algorithm works a little like the genetic algorithm. The next solution is found by picking a neighbor of the current solution. If this is better than the current it becomes the new current solution. But if it was not better it would still have a probability of becoming the new current solution. This way the algorithm can jump out of a local optimum. The probability for this jump is gradually lowered throughout the search. How to lower this probability could be difficult to decide. Here there has also been used some small scale testing to find a reasonable function.

Chapter 6

Resultes

6.1 Pure random search

Best revenue	Iterations	Fleet Size	Fleet
207	163	7	1,1,1,2,2,5,5

Table 6.1: Performance of the pure random search

The performance of the pure random search can be seen in table 6.1 above. After searching through 163 solutions the algorithm stopped itself after the derivative of the regression line came close to zero.

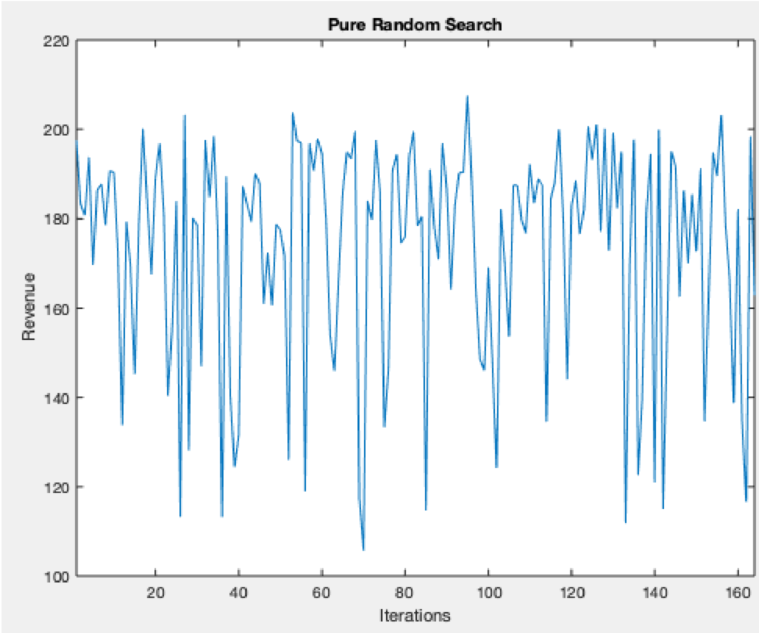

Figure 6.1: Revenue from the pure random search

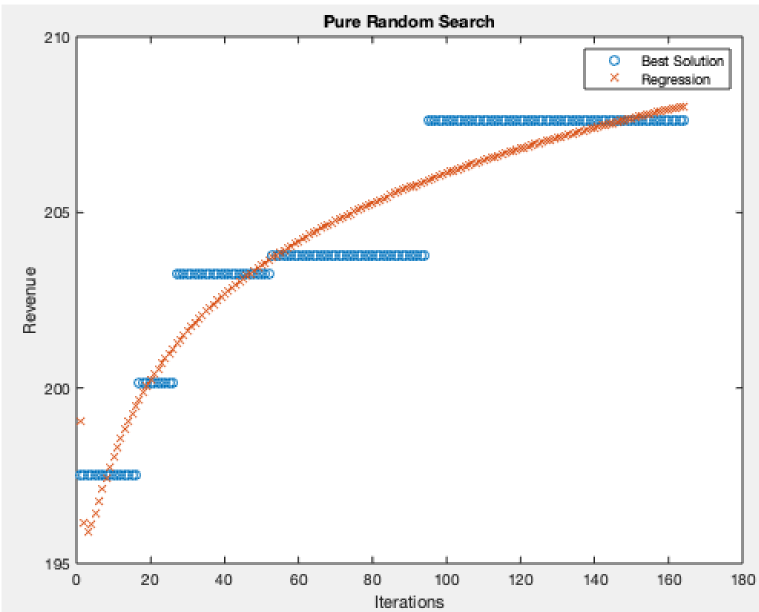

Figure 6.2: Shows highest revenue over the regression line

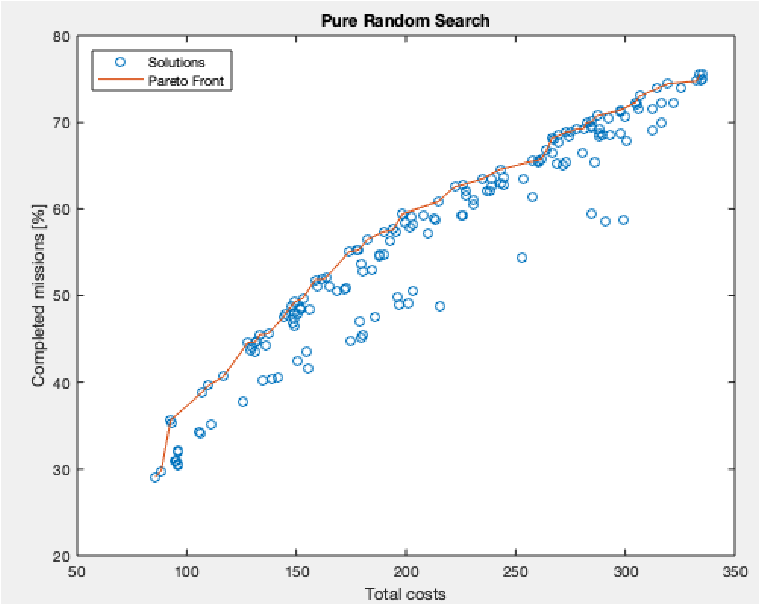

Figure 6.3: Shows the percentage completed compared to how much it costed

Figure 6.1 shows revenue from all solutions obtained by the algorithm. It shows no change with time and is as scattered in the end as in the beginning, which would be expected from the pure random algorithm. The best solution is found after about 100 iterations and the algorithm stops itself as the regression line flattens out, as seen in figure 6.2. From figure 6.3 we can see how many of the generated missions the fleet is able to complete. The even distribution along the diagonal is a clear sign of the algorithm picking solution from all over the solution pool. With changes in rates the Pareto front in this figure could serve as a decision tool for fleet sizing.

6.2 Genetic algorithm

Best revenue	Iterations	Fleet Size	Fleet
211	110	8	1,1,1,2,2,3,5,5

Table 6.2: Performance of the genetic algorithm

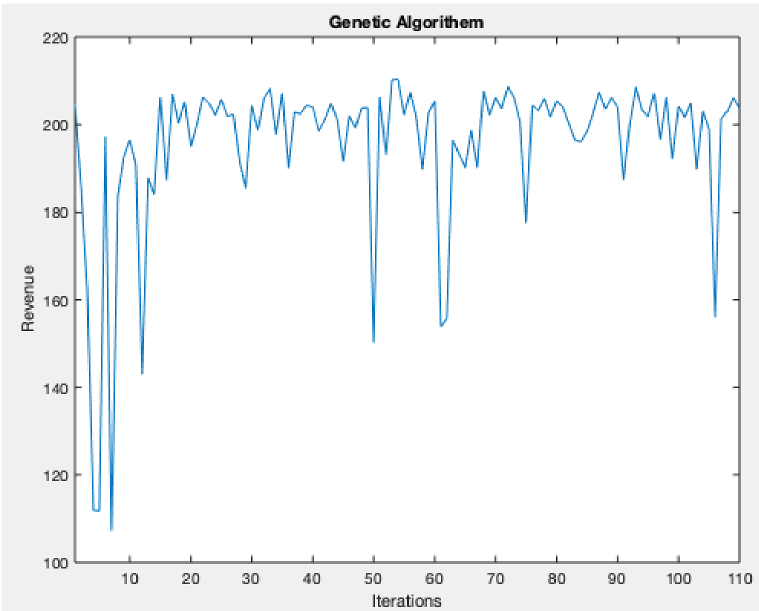

Figure 6.4: Revenue from the pure random search

Figure 6.4 shows revenue from all solutions obtained by the algorithm. It shows a clear change with time, and after only 15 iteration it only searches in the higher part of the solution space. This is an expected behaviour from the genetic algorithm. It quickly finds good solutions, but runs the risk of getting stuck in a local optimum. The best solution is found after about 55 iterations and the algorithm stops itself as the regression line flattens out, as seen in figure 6.5. From figure 6.6 we can see how many of the generated missions the fleet is able to complete. The solutions are highly concentrated around 60 - 65 percent of the missions completed.

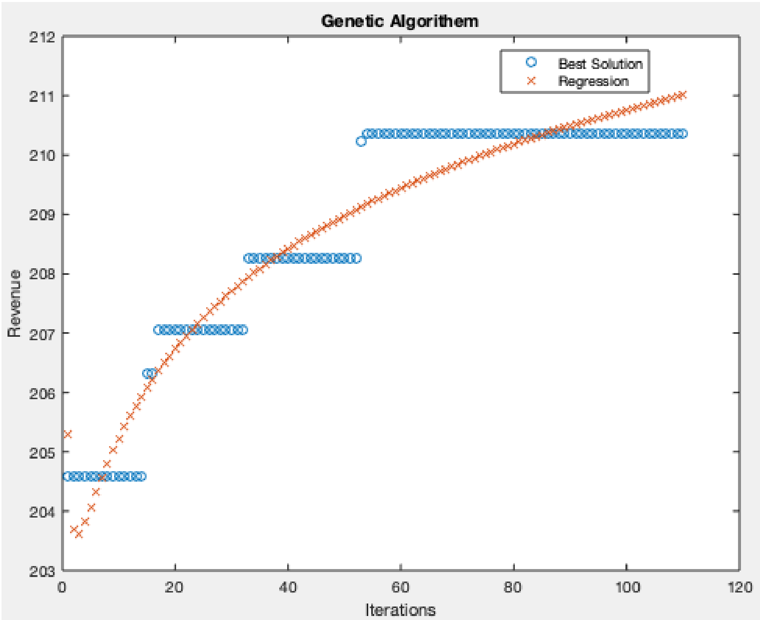

Figure 6.5: Shows highest revenue over the regression line

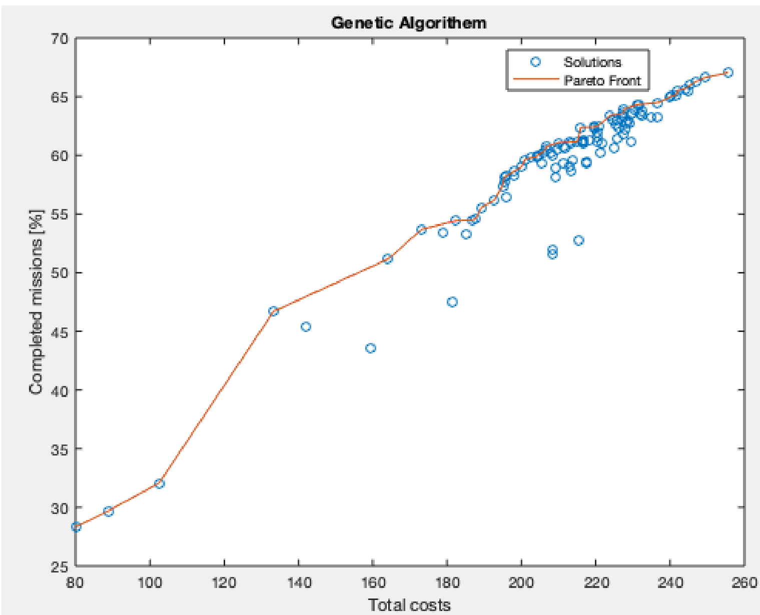

Figure 6.6: Shows the percentage completed compared to how much it costed

6.3 Tabu search

Best revenue	Iterations	Fleet Size	Fleet
206	60	7	1,1,1,2,2,2,4

Table 6.3: Performance of the tabu search

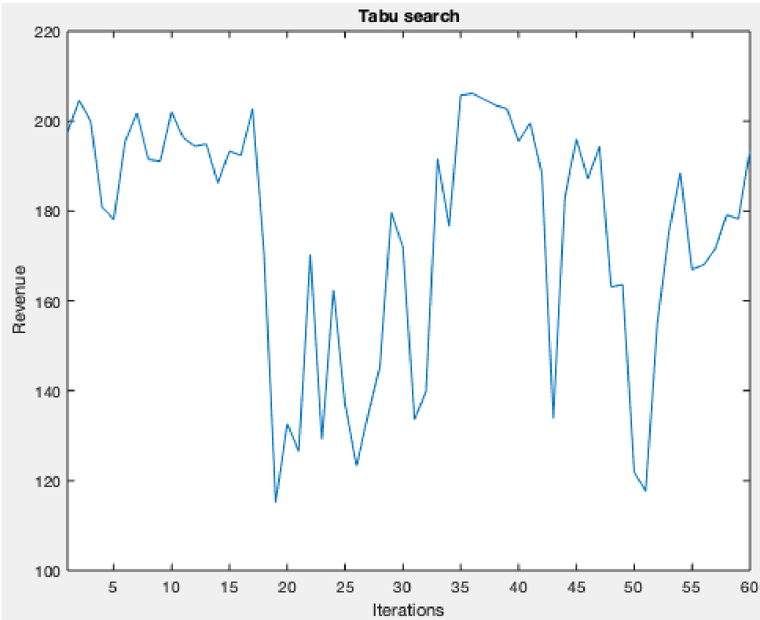

Figure 6.7: Revenue from the tabu search

Figure 6.7 shows revenue from all solutions obtained by the algorithm. As expected there are few rapid changes in revenue value, as the search is guided from one neighbour to another. The best solution is found after about 35 iterations and the algorithm stops itself as the regression line flattens out, as seen in figure 6.8. It seems like the search was stopped too early. With a solution space of this size it needs more iterations to be able to search through all parts of the solution space. The reason why it stopped too early, is because the first few solutions found where quite good and it flattened out the regression line. This is a clear disadvantage of using this criteria to stop the algorithm. From figure 6.9 we can see how many of the generated missions the fleet is able to complete. The solutions are evenly distributed along the diagonal. a clear sign that the algorithm has

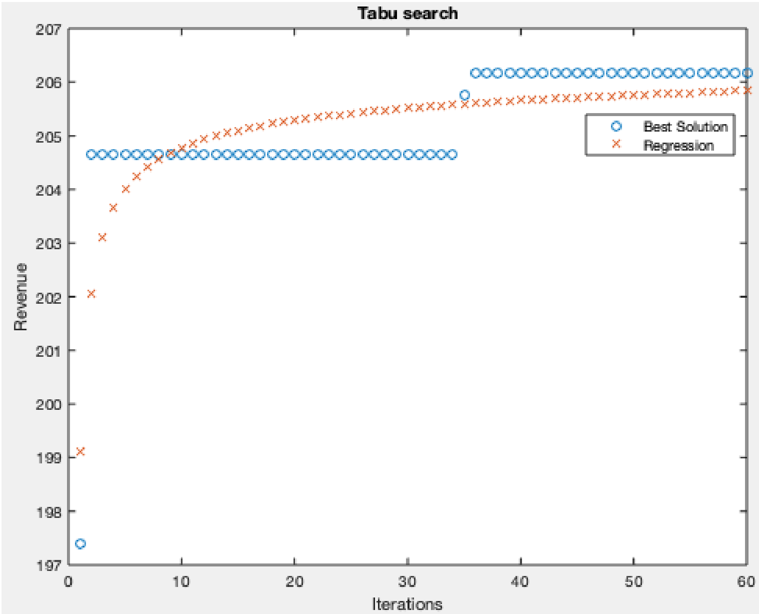

Figure 6.8: Shows highest revenue over the regression line

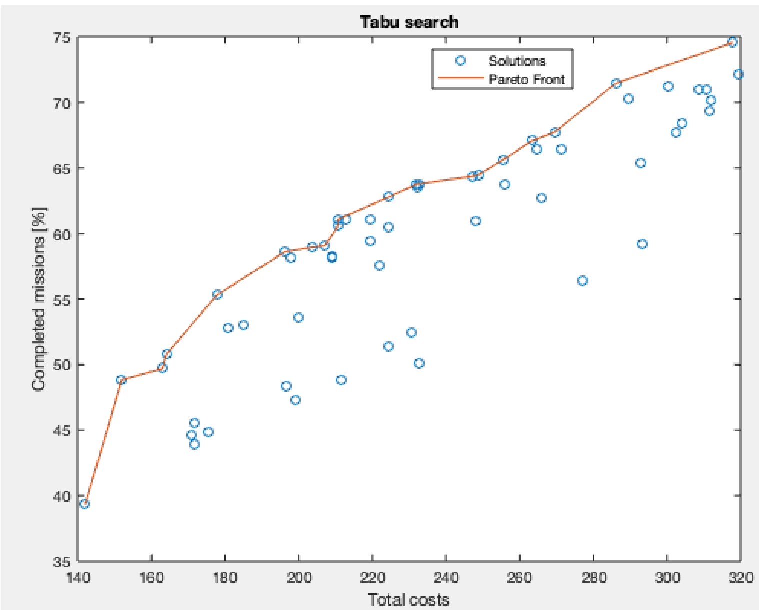

Figure 6.9: Shows the percentage completed compared to how much it costed

been through most of the possible fleet sizes.

6.4 Simulated annealing

Best revenue	Iterations	Fleet Size	Fleet
207	151	11	1,1,1,3,3,3,3,4,5,5,5

Table 6.4: Performance of the simulated annealing

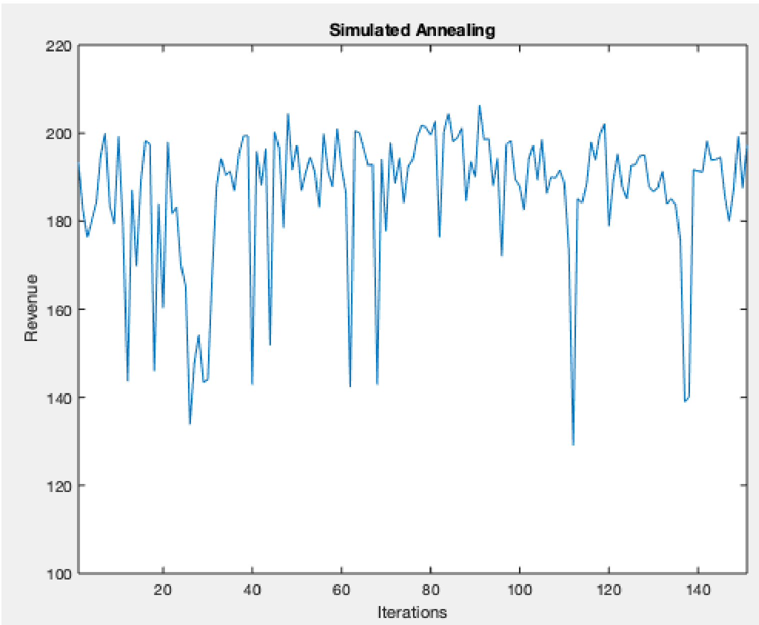

Figure 6.10: Revenue from the simulated annealing

Figure 6.10 shows revenue from all solutions obtained by the algorithm. It clearly shows signs of having similarities with the genetic algorithm. Each the the solution comes out significantly worse than the previous solution, it quickly comes back up. Especially this happens later in the search. Early in the search it seems like the stays low for a few iterations. This could be the exploration moves as explained in chapter 4. The best solution is found after about 90 iterations and the algorithm stops itself as the regression line flattens out, as seen in figure 6.11. From figure 6.12 we can see how many of the

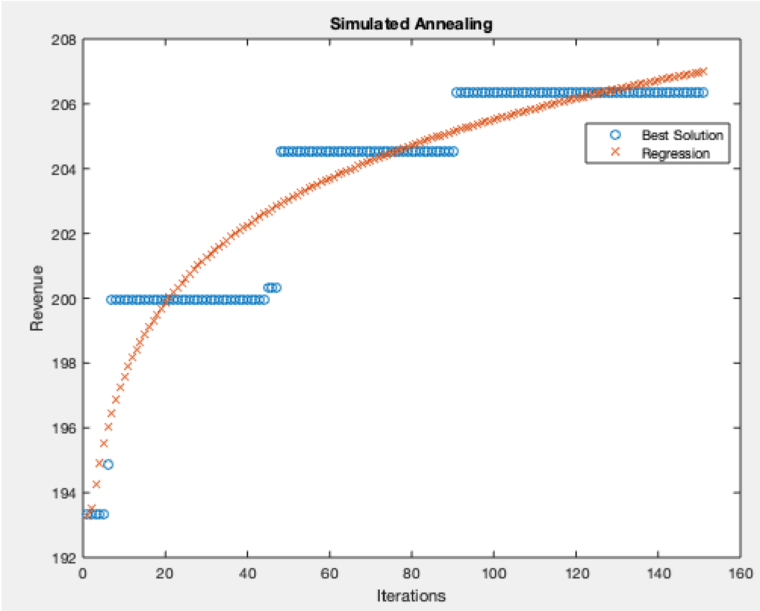

Figure 6.11: Shows highest revenue over the regression line

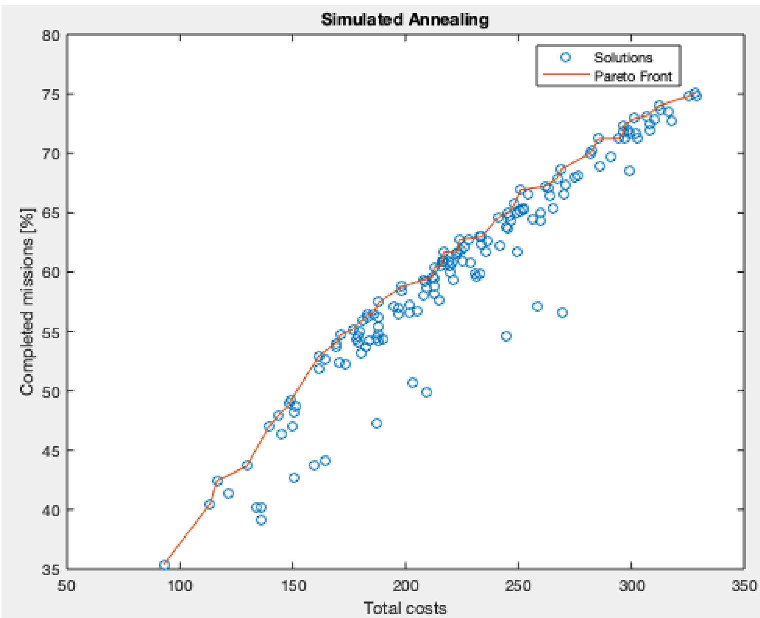

Figure 6.12: Shows the percentage completed compared to how much it costed

generated missions the fleet is able to complete. The solutions are concentrated between 55 and 75 percent of the missions completed. It could seem like the equivalent graph for the genetic algorithm, only with a slightly wider search area.

6.5 Learning automata search technique

Best revenue	Iterations	Fleet Size	Fleet
209	190	8	1,1,2,2,2,4,5,5

Table 6.5: Performance of the LAST

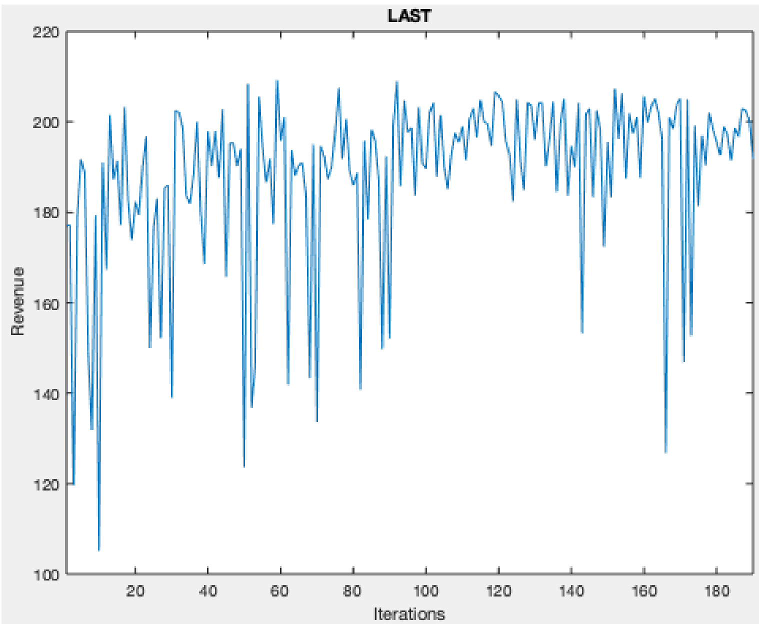

Figure 6.13: Revenue from the LAST

Figure 6.13 shows revenue from all solutions obtained by the algorithm. It starts just as the pure random search, and then it gradually evolves. The solution get better and better with time, and the algorithm almost stops creating bad solutions at all. The best solution is found after only 60 iterations and the algorithm stops itself as the regression line flattens out, as seen in figure 6.14. Although the best solution is found rather quickly

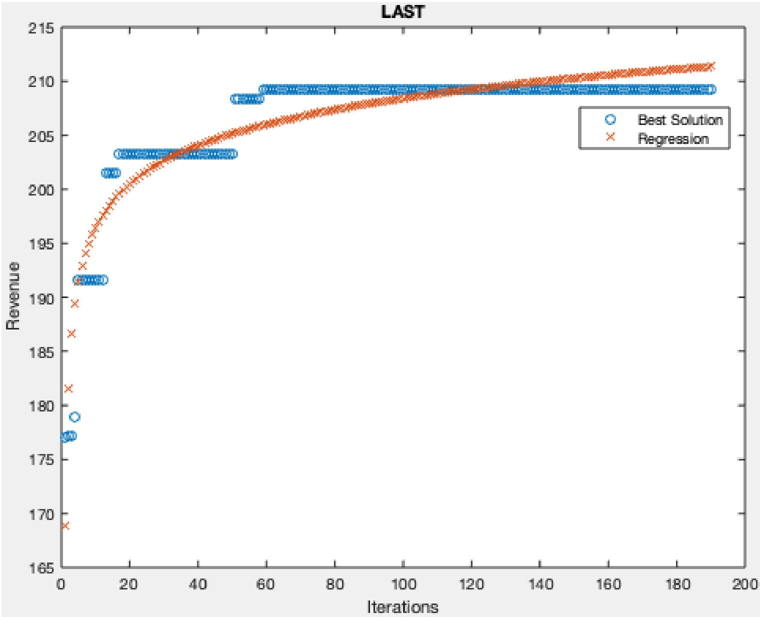

Figure 6.14: Shows highest revenue over the regression line

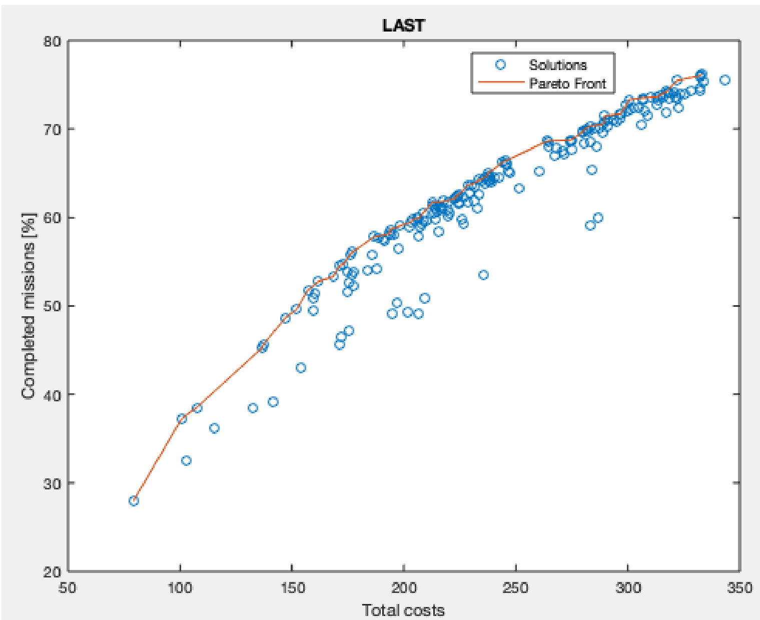

Figure 6.15: Shows the percentage completed compared to how much it costed

it still runs for a long time due to all the bad solution found at the start. From figure 6.15 we can see how many of the generated missions the fleet is able to complete. There are two different places with a high density of solutions. The strength of this algorithm is that it does not need to go through a worse solution or a set of worse solutions to move from one local optimum to another. Thus effectively searching through several promising areas at the same time.

Fleet size	Probability
3	0,000089
4	0,000087
5	0,008132
6	0,044427
7	0,149960
8	0,134008
9	0,321281
10	0,065850
11	0,000087
12	0,092261
13	0,104157
14	0,036239
15	0,043423

Figure 6.16: Probability of choosing each fleet size

Boat #	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**Sum**
Prob. boat 1	0,36	0,11	0,58	0,29	0,12	0,3	0,22	0,39	0,34	0,16	0,18	0,18	0,23	0,19	0,27	**3,92**
Prob. boat 2	0,29	0,24	0,12	0,4	0,42	0,47	0,14	0,01	0,25	0,18	0,16	0,24	0,13	0,2	0,18	**3,42**
Prob. boat 3	0,07	0,06	0,07	0,04	0,05	0,01	0,23	0,21	0,12	0,09	0,26	0,11	0,07	0,2	0,15	**1,71**
Prob. boat 4	0	0,02	0,04	0,03	0,03	0	0,32	0,15	0,12	0,27	0,16	0,22	0,13	0,2	0,22	**1,92**
Prob. boat 5	0,29	0,58	0,2	0,25	0,38	0,23	0,09	0,23	0,17	0,3	0,24	0,26	0,44	0,21	0,17	**4,03**

Figure 6.17: Probability of choosing the different vessel

As seen from figure 6.16 the algorithm clearly favors a vessel size between 7 and 9, which also is the area where most of the other algorithm also found their best solution. Figure 6.17 shows which boats who got favored. Vesseltype 1, 2 and 5 is way more likely to be chosen than 3 and 4. This because these have given good results throughout the search.

Figure 6.18 shows a low degree of variance when running MCS. This shows that most solutions perform well considering the stochastic nature of the mission generation and weather. There is evidence for correlation between a solutions ability to perform in all situations and its expected revenue.

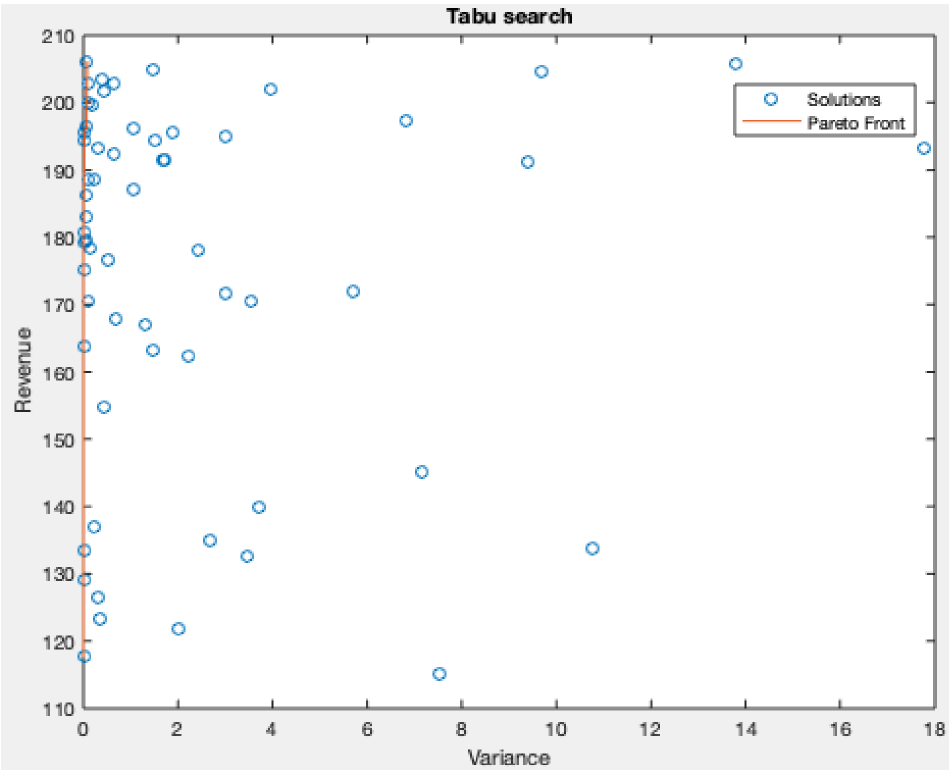

Figure 6.18: Variance compared to revenue

Chapter 7

Discussion

4 out of the 5 optimization algorithm found a fleet size of 7 - 8 vessels to be optimal. Although none of them where able to complete more than 75 % of the mission. The reason for this could be the stochastic mission generation, seasonal variations or missions aborted due to high weather. Seasonal variations causing the algorithm to choose between maximizing revenue in the high season and loosing money in the low season, could be an explanation. Making rates swing along with the seasons could have been considered.

It seems like the vessels 1 is favored bye all algorithms and vessel 2 and 5 are favored by 4 out of 5. This does not fit with the trend in the current market situation. The later years the trend has been shifting towards bigger and more better quipped vessels. Small changes in cost rates could make a huge impact on the fleet composition. Additional cost due to onshore personnel and administration is not considered in this thesis. If considered it could have made the algorithms slightly shift towards bigger well-boats.

Many small vessels gives the fleet a possibility serving more missions at the same time. Since the model does not allow vessels to jump between missions without going through the depot, a fleet with fewer and bigger vessels misses that opportunity to reduce sailing times. More and smaller vessels also gives more flexibility and might deal with uncertainties in a better way. That this model is aimed at the spot market while most well-boats are designed for long term contracts could also be a likely reason all algorithm deviates from the real world trend.

	Pure Random Search	Genetic Algorithm	Tabu Search	Simulated Annealing	LAST
Best revenue	207	210	206	207	209
Iterations	163	110	60	151	190

Table 7.1: Performance of the meta-heuristics

From table 7.1 we can see that there is almost no difference between the best solutions found by each algorithm. The best and the worst only differed by less than 2 %. Despite such a small difference it does not seem to be random which algorithm who actually finds the best solution. From figure 6.4 and figure 6.13 it seems like the best solution obtained comes from continuously searching in a promising area. Both the genetic algorithm and the LAST delivers a high percentage good solutions close to the best solution. The genetic algorithm is the first to narrow the search in a promising region, and thus finding the best solution. This could be because the solution space is dominated by on global optimum. Thereby allowing this algorithm to search here for a longer time. It could also be a result of the optimization model stopping too early, not giving the other algorithms enough time to explore the whole solution space.

The tabu search is an algorithm that would need more time. It is designed to wonder around in the solution space, obtaining the best solution by minimizing the risk of not having explored any promising regions. This algorithm fell victim to the stopping criteria put up for the optimization model. By finding really good solutions early on, the algorithm misread it to be a sign that it would not be worth the time looking for a better solution.

Simulated annealing is an algorithm that also by nature will need more time than the genetic algorithm. The whole theory behind it is to start out as the genetic algorithm (though without the initial population). Then jumping from one promising region to another after searching in each for some time. Setting the probabilities for the algorithm to explore could be quite tricky and needs more research. From figure 6.10 it seems like the algorithm does some exploration. But it is hard to say if it explores too much or just do not get enough time to search in each region.

With the learning automata search technique the running time needed is determined by the number of decision variables and their range. In this case the technique clearly favors some ranges of the decision variables. These are also the same ranges where the other algorithms found their best solutions.

Chapter 8

Conclusion and further work

8.1 Conclusion

The Norwegian aquaculture industry has a huge potential for growth over the next decades, as both the industry and politicians sees the opportunity for big revenues. This will set high standards for the future fleet of well-boats. Optimization methods could be a powerful tool to ensure efficient transportation and delousing of fish, both close to shore and further out in the sea.

With an increased demand for well-boats the pressure on utilizing these boats to the fullest of their capacities will also increase. Different optimization techniques could be a valuable tool when trying to utilize your fleet to the fullest. Finding the exact solution to a maritime fleet sizing problem will often be too complex and time consuming. One approach to this problem is to simplify the problem enough so that exact mathematical algorithms can be used. By obtaining the closed form of the objective function linear programming can often be used. If the problem is to big the algorithm could use years obtaining the optimal solution from the solution space. Simplifying these problems without leaving out significant decision variables can be a complex task.

Another approach is to develop an algorithm that can quickly search through the solution space to find a solution considered good enough. With this approach the real life problem could be modeled more accurately. By running simulations a probability function of the expected outcome can be derived. This way simulation can be used instead of finding the closed form of the objective function. The mean and variance of these probability functions can be a measure for the solution.

Simulation-based optimization methods could be well suited for solving these types of real life problems. Compared to the more conventional linear optimization, simulation-based optimization can handle uncertainties in a better way. By running Monte Carlo simulations the effect of stochastic and correlating variables could be estimated. Also by analysing the variance of the solution, uncertainties can be mitigated and the risk for stakeholders could be lowered.

The simulation model presented in this thesis can handle stochastic variables, like weather and mission generation, in a way linear optimization would struggle.

In Simulation-based optimization many different heuristics for searching through the solution space can be used. Meta-heuristics and stochastic adaptive search techniques are two different classifications of these heuristics. Meta-heuristics techniques can not guarantee that the algorithm convergences towards the global optimum like all stochastic adaptive search techniques do. Still they are widely used and they have been shown to work very well in practice.

The different search algorithms tested gave a little unexpected result, but this could be explained by the nature of the problem, and the set up of the decision variables. In the end the genetic algorithm proved to be the best, although it does not handle local optimums in a good way. Its superiority could result from a dominant global optimum in the solution space, or too little time for the other algorithms to search for the best solution. The fact that most algorithms came out with quite similar solutions could support the theory of a dominant global optimum.

The criteria for stopping the optimization model could have been chosen in another way. Some algorithms, like the tabu search could have been stopped too early. But using the derivative of the regression line too decide when too stop also seemed too work very well for both the Genetic algorithm the LAST which both outperformed the pure random search.

All algorithms also favored the smaller well-boats. This is a clear difference form the real world trend. Projected and newly built well-boats are usually built to serve longer contracts, while this model is aimed at the spot market. For this reason the model could favor the smaller vessels because of the flexibility they provide. The biggest and most advanced well-boat is also preferred by the algorithms. This is mainly because it can perform delousing missions the other vessels cant.

8.2 Further work

It would be recommended to work closer with a industry partner to be able make an even more realistic simulation model. Also parameters like cost rates and income could be more accurately calculated. With more insight into the indusrty it could be possible at add differnet heuristics for the choice of matching well-boats with missions in the simulation model.

Heuristics for dealing with bad weather could also be made for the simulation model. The decisions whether to wait or cancel a mission and how long each vessel should wait, could also the topic for a whole new optimization problem. By implementing these heuristics as decision variables in the optimization model, the solution space will be greatly enlarged. This would again required more computational time than expedient for this thesis.

Another thing to look deeper into is the actual results obtained from the optimization algorithms. Could Pareto front analysis or variance analysis say something about the robustness of the solution? Or how risky they are compared to other solutions?

Bibliography

[1] *Statistikkbanken, Akvakultur* (2017) Available at: https://www.ssb.no/statbank/table/07326/tableViewLayout1/ (accessed: 3. February 2020).

[2] Olafsen, T., Winther U., Olsen Y.og Skjermo J. (2012) *Verdiskapning basert på produktive hav i 2050* Trondheim: Det Kongelige Norske Videnskabers Selskab og Norges Tekniske Vitenskapsakademi. Available at: https://www.sintef.no/globalassets/upload/fiskeri_og_havbruk/publikasjoner/verdiskaping-basert-pa-produktive-hav-i-2050.pdf (accessed: 8. May 2019).

[3] Meld. St. 16 (2014–2015) (2015) *Forutsigbar og miljømessig bærekraftig vekst i norsk lakse- og ørretoppdrett*. Oslo: Ministry of Trade, Industry and Fisheries.

[4] Tekna (2018) *Norsk oppdrett i endring* Tekna. Available at: https://www.tekna.no/kurs/innhold/norsk-oppdrett-i-endring/ (accessed: 28. Oktober 2019).

[5] Ytreberg, R. (2018) *To av tre sjefer i havbruksnæringen tror ikke på regjeringens vekstmål* Dagens Naeringsliv. Available at: https://www.dn.no/havbruk/sjomat/oppdrettslaks/pwc/to-av-tre-sjefer-i-

havbruksnaringen-tror-ikke-pa-regjeringens-vekstmal/2-1-210676 (accessed: 10. january 2020).

[6] Aarre, E. (2018) *Miljøproblemer hindrer vekst i oppdrettsnæringen* Aftenposten. Available at: https://www.aftenposten.no/okonomi/i/gMBMA/miljoeproblemer-hindrer-vekst-i-oppdrettsnaeringen (accessed: 29. Oktober 2019).

[7] SalMar (2020) *HAVBASERT FISKEOPPDRETT*. Available at: https://www.salmar.no/havbasert-fiskeoppdrett-en-ny-aera/ (accessed: 10. January 2020).

[8] Nordlaks (2020) *Havfarm*. Available at: https://www.nordlaks.no/havfarm/om-havfarm-prosjektet (accessed: 10. November 2019).

[9] Norway Royal Salmon (2020) *ARCTIC OFFSHORE FARMING*. Available at: https://www.arcticoffshorefarming.no/ (accessed: 10. January 2020).

[10] Furuset, A. (2019) *Disse nye brønnbåtene leveres i løpet av året*. Fiskeribladet. Available at: https://fiskeribladet.no/tekfisk/nyheter/?artikkel=68440 (accessed: 10. January 2020).

[11] Nodland, E. (2015) *Det må bygges flere brønnbåter*. iLaks. Available at: https://ilaks.no/det-ma-bygges-flere-bronnbater/ (accessed: 10. January 2020).

[12] Solem , L. K. (2017) *Bruker 70 mill. på ny båt - håper den blir liggende ved kai*. Dagens Naeringsliv. Available at: https://www.dn.no/havbruk/bruker-70-mill-pa-ny-bat-haper-den-blir-liggende-ved-kai/2-1-165950 (accessed: 10. January 2020).

[13] Kvile, K. (2019) *Ingen kunne sett for seg veksten brønnbåtene har hatt*. Fiskeribladet. Available at: https://fiskeribladet.no/tekfisk/nyheter/?artikkel=65995 (accessed: 21. January 2020).

[14] Banks, J., Carson, J.S., Nelson, B.L. and Nicol D.M. (2001) *Discrete-Event System Simulation*. Third edition. New Jersey: Prentice Hall

[15] Ioannis, K.K. (2003) Optimal Operation of Batch Processes under Uncertainty: A Monte Carlo Simulation-Deterministic Optimization Approach, *Industrial & Engineering Chemistry Research*, 42(26), P. 6815-6822. DOI: 10.1021/ie034001m

[16] *Statistikkbanken, Akvakultur* (2017) Available at: https://www.ssb.no/statbank/table/07326/tableViewLayout1/ (accessed: 8. May 2019).

[17] Moe, E. (2017) *The Norwegian aquaculture analysis 2017* Bergen: EY. Available at: https://www.ey.com/Publication/vwLUAssets/EY_-_The_Norwegian_Aquaculture_Analysis_2017/$FILE/EY-Norwegian-Aquaculture-Analysis-2017.pdf (accessed: 8. May 2019).

[18] Fiskeridirektoratet (2019) *Kapasitetsjustering / Trafikklyssystemet* Available at: https://www.fiskeridir.no/Akvakultur/Tildeling-og-tillatelser/Kapasitetsjustering-Trafikklyssystemet (accessed: 10. May 2019).

[19] Fiskeridirektoratet (2020) *Yggdrasil aquakultur* Available at: https://kart.fiskeridir.no/share/5d8a92f44301 (accessed: 09. february 2020).

[20] Fiskeridirektoratet (2020) *Kapasitetsjustering / Trafikklyssystemet* Available at: https://www.fiskeridir.no/Akvakultur/Tildeling-og-tillatelser/Kapasitetsjustering-trafikklyssystemet/Kapasitetsjustering-trafikklyssystem-2017-2018 (accessed: 09. february 2020).

[21] Laks.no (2020) *Norsk laks fra fjord til bord* Available at: https://laks.no/lakseproduksjon/ (accessed: 09. february 2020).

[22] Nofima (2020) *Optimalisert postsmoltproduksjon (OPP)* Available at: https://nofima.no/prosjekt/optimalisert-postsmoltproduksjon-opp/ (accessed: 27. October 2019)

[23] NTS ASA (2019) Contact with managment department, 07 May 2019.

[24] Chramer, C. (2017) *Med norsk laks i kofferten* Norges Sjømatråd. Available at:

https://seafood.no/aktuelt/Fisketanker/med-norsk-laks-i-kofferten/ (accessed: 27. October 2019)

[25] Forskrift om transport av akvakulturdyr (2008) *Forskrift om transport av akvakulturdyr* Available at: https://lovdata.no/dokument/SF/forskrift/2008-06-17-820 (accessed: 28. October 2019)

[26] Rosten, T. (2010) *Forutsetninger for optimalisering av systemer for transport av levende fisk i brønnbåt* Norsk Institutt for Vannforskning. Available at: https://vannforeningen.no/wp-content/uploads/2015/06/2010_808461.pdf (accessed: 28. October 2019)

[27] Heen, V. (2015) *Transport av Smolt fra Settefiskanlegg* LFFH. Available at: http://lffh.no/transport-av-smolt-fra-settefiskanlegg/ (accessed: 27. October 2019)

[28] Nodland, E. (2015) *Antall brønnbåter minker for hver dag*. iLaks. Available at: https://ilaks.no/antall-bronnbater-minker-for-hver-dag/ (accessed: 10. January 2020).

[29] Mattilsynet (2019) *Termisk avlusing: Fiskevelferd, forskning og avklaring fra Mattilsynet*. Available at: mattilsynet.no/fisk_og_akvakultur/fiskevelferd/termisk_avlusing_fiskevelferd_forskning_og_avklaring_fra_matt (accessed: 10. January 2020).

[30] Imai, A. and Rivera, F. (2001) Strategic fleet size planning for maritime refrigerated containers, *Maritime Policy and Management*, 28(4), 361-374, DOI: 10.1080/03088830010020629

[31] Alvarez, J.F., Tsilingiris, P., Engebrethsen E.S., and Kakalis N.M.P. (2011) Robust Fleet Sizing and Deployment for Industrial and Independent Bulk Ocean Shipping Companies, *INFOR: Information Systems and Operational Research*, 49(2), p. 93-107, DOI: 10.3138/infor.49.2.093

[32] Fagerholt, K., Christiansen, M., Hvattum, L.M., Johansen, T.A.V. and Vabø, T.J. *A decision support methodology for strategic planning in maritime transportation* Trondheim: MARINTEK doi:10.1016/j.omega.2009.12.003

[33] Shyshou, A., Gribkovskaia, I., and Barceló, J. (2010) A simulation study of the fleet sizing problem arising in offshore anchor handling operations. *European Journal of Operational Research*, 203(1), p. 230–240.

[34] Attia, S. (2012) *Computational optimization zero energy building design: interviews with 28 international experts, International energy agency (IEA) task 40: towards net zero energy buildings subtask B.* Louvain La Neuve: Architecture et climat. Avalable at: https://orbi.uliege.be/bitstream/2268/168345/1/Attia_Optimisation%20Interviews_2012.pdf (accessed: 10. May 2019)

[35] Nguyen, A.T., Reiter, S. and Rigo, P. (2014) A review on simulation-based optimization methods applied to building performance analysis *Applied energy* 113, p. 1043-1058. doi:10.1016/j.ejor.2013.04.058

[36] Wright, J.A. and Alajmi, A. (2005) The robustness of genetic algorithms in solving unconstrained building optimization problems. *Proceedings of the 9th international building performance simulation association conference*. Montreal: IBPSA. p. 1361–1368.

[37] Evins, R., Pointera, P., Vaidyanathanb, R. and Burgess, S. (2012) A case study exploring regulated energy use in domestic buildings using design-of-experiments and multi-objective optimisation *Building and Environment* 54, P. 126-136.

[38] Gosavi, A. (2015) *Simulation-Based Optimization, Parametric Optimization Techniques and Reinforcement Learning*, Operations Research/Computer Science Interfaces Series, volume 55, Boston: Springer US. DOI 10.1007/978-1-4899-7491-4_3

[39] Lundgren, J., Rönnqvist, M. and Värbrand, P. (2010) *Optimization*. Lund: Studentlitteratur AB

[40] Glover, F. (1986) Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*. 13(5), P. 533-549. doi: 10.1016/0305-0548(86)90048-1

[41] Cordeau, J.F. and Laporte, G. (2003) A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological*. 37(6), P. 579-594. doi:10.1016/S0191-2615(02)00045-0

[42] Law, A.M. and Kelton, W.D. (2000) *Simulation Modeling and Analysis*. Third edition. The McGraw-Hill Companies.

[43] Solvtrans AS (2020) Contact with logistics department, 17 November 2020.

[44] Froy rederi AS (2020) Contact with Technical department, 11 November 2020.

[45] Napier AS (2020) Contact with Quality manager, 11 November 2020.

[46] Norsk fisketransport AS (2020) *Fartoy* Available at: https://norskfisketransport.no/fartoy/ (accessed: 8. May 2019)

Appendix

Matlab koder

Code 8.1: Plotting

```

% Reads input from file
VesselTypes = ...
    readmatrix('JoachimSimInput.xlsx','Sheet','Baattyper','Range','A2:k6');
Rates = ...
    readmatrix('JoachimSimInput.xlsx','Sheet','Rater','Range','A2:H6');
OperationalLimits = ...
    readmatrix('JoachimSimInput.xlsx','Sheet','Operasjonsgrenser','Range','A2:G6');
MissionTypes = ...
    readmatrix('JoachimSimInput.xlsx','Sheet','Oppdragstyper','Range','B2:O5');
VesselOps = ...
    readmatrix('JoachimSimInput.xlsx','Sheet','Baat-oppdrag','Range','B2:E6');
DistanceMatrix = ...
    readmatrix('JoachimSimInput.xlsx','Sheet','Distansematrise','Range','B2:P16');

Hs = ncread('NordsjoWeather.nc','Weather');

SimTime = 8760;

VOr = length(VesselOps(:,1));
```

```
VOc = length(VesselOps(1,:));
VTr = length(VesselTypes(:,1));
VTc = length(VesselTypes(1,:));
Rr = length(Rates(:,1));
Rc = length(Rates(1,:));
DMr = length(DistanceMatrix(:,1));
DMc = length(DistanceMatrix(1,:));
OLr = length(OperationalLimits(:,1));
OLc = length(OperationalLimits(1,:));
```

Code 8.2: Plotting

```
% Create vectors and set parameters
SimRevenueVector = [];
TotSimRevenue = 0;
SimCostVector = [];
TotSimCost = 0;
PercentageVector = [];
Percentage = 0;
for j=1:40
    run('MainJoachim.m'); %running simulation
    SimRevenueVector = [SimRevenueVector Revenue];
    TotSimRevenue = TotSimRevenue + SimRevenueVector(j);
    SimCostVector = [SimCostVector TotalCost];
    TotSimCost = TotSimCost + SimCostVector(j);
    PercentageVector = [PercentageVector MissionPercentage];
    Percentage = Percentage + PercentageVector(j);
end
AvarageTotalRevenue = TotSimRevenue/j;
AvarageTotalCost = TotSimCost/j;
AvarageTotalPercentage = Percentage/j;

% Update vectors
Variance = var(SimRevenueVector);
VarianceVector = [VarianceVector Variance];
SolutionVector = [SolutionVector; Variance AvarageTotalRevenue fleet];
RevenueVector = [RevenueVector AvarageTotalRevenue];
```

```
CostVector = [CostVector AvarageTotalCost];
MissionPercentageVector = [MissionPercentageVector ...
    AvarageTotalPercentage];
if BestValue ≤ AvarageTotalRevenue
    BestValue = AvarageTotalRevenue;
    BestValueVector = [BestValueVector AvarageTotalRevenue];
    BestSolution = [fleetsize fleet];
    bestfleet = fleet;
    bestfleetsize = fleetsize;
else
    BestValueVector = [BestValueVector BestValue];
end

Counter = Counter + 1
XVector = [XVector Counter];
xm = [XVector'];
ym = [BestValueVector'];
CurveFit = table(xm, ym);

% Make a regression curve to the BestValueVector
if Counter ≥ 60
    writetable(CurveFit,'BestSolutions.csv');
    run('regression.m');

    % Convergens criteria
    Derivative = -b/Counter^2 + c/Counter
    if Derivative ≤ 0.024
        breaking = 1;
    end
end
```

Code 8.3: Plotting

```
% Reads input from file
Fleet = ...
    readmatrix('JoachimSimInput.xlsx','Sheet','Flaate','Range','A2:B16');

```

```
for k = 1:15
    if Fleet(k,2) == 80
        Fleet = Fleet([1:k-1],1:2);
        break
    end
end



SimTime = 8760; %Hours. Total simulation time. one year. (time-step)


WhetherRand = randi([1 length(Hs(1,1,:))-SimTime]);
Weather = squeeze(Hs(1,1,WhetherRand:WhetherRand+SimTime-1));

MaxWeather = max(Weather);
Weather = Weather/(MaxWeather/9);
Weather = round(Weather);

Fr = length(Fleet(:,1));
Fc = length(Fleet(1,:));
Wr = length(Weather(:,1));
Wc = length(Weather(1,:));
% Run simulation
sim('JoachimBronnbatModell.slx');
% Write output data
MissionList = ans.MissionListRes.Data;
SailTimes = ans.SailTimesRes.Data;
WorkTimes = ans.WorkTimesRes.Data;
% calculating cost
SailCost = 0;
WorkCost = 0;
CrewCost = 0;
VesselCost = 0;
for i=1:Fr
    SailCost = SailCost + SailTimes(i)*VesselTypes(Fleet(i,2),7);
    WorkCost = WorkCost + WorkTimes(i)*VesselTypes(Fleet(i,2),8);
    CrewCost = CrewCost + (SailTimes(i)+WorkTimes(i)) * ...
```

```
        VesselTypes(Fleet(i,2),9);
    VesselCost = VesselCost + VesselTypes(Fleet(i,2),6);
end

TotalCost = CrewCost + SailCost + WorkCost + VesselCost;

% Completedet missions and revenue calculation
CompletedSmolt = 0;
CompletedSlaught = 0;
CompletedDel1 = 0;
CompletedDel2 = 0;
for i = 1: length(MissionList(:,1))
    if MissionList(i,2)==1 %If smolt-oppdrag
        CompletedSmolt = CompletedSmolt + MissionList(i,5) - ...
            MissionList(i,7);
    elseif MissionList(i,2)==2 %If slakt-oppdrag
        CompletedSlaught = CompletedSlaught + MissionList(i,5) - ...
            MissionList(i,7);
    elseif MissionList(i,2)==3 %If avlusing 1-oppdrag
        CompletedDel1 = CompletedDel1 + MissionList(i,5) - ...
            MissionList(i,7);
    else %If avlusing 1-oppdrag
        CompletedDel2 = CompletedDel2 + MissionList(i,5) - ...
            MissionList(i,7);
    end
    if MissionList(i,1)==0
       break
    end
end
MissionPercentage = (sum(MissionList(:,5)')-sum(MissionList(:,7)')) ...
    / sum(MissionList(:,5)') * 100;
Income = (CompletedSmolt*MissionTypes(1,11) + ...
    CompletedSlaught*MissionTypes(2,11) + ...
    CompletedDel1*MissionTypes(3,11) + ...
    CompletedDel2*MissionTypes(4,11));
Revenue = Income - TotalCost;
```

Code 8.4: Plotting

```
figure(1)
plot(RevenueVector);
ylabel('Revenue')
xlabel('Iterations')
title(Title)
axis([1 Counter 100 220])

figure(2)
plot(BestValueVector);
hold on
fplot(@(x) a+b/x+c*log(x))
hold off
ylabel('Revenue')
xlabel('Iterations')
title(Title)
legend('Best Solution','Regression')
axis([1 Counter 160 220])

figure(3)
plot(z.xm,z.ym,'o');
hold on
plot(z.xm,z.y,'x');
ylabel('Revenue')
xlabel('Iterations')
title(Title)
legend('Best Solution','Regression')
hold off

SolutionVector = sortrows(SolutionVector);
ParetoFront = [SolutionVector(1,:)];
w = 1;
for i = 2:Counter
    if ParetoFront(w,2) < SolutionVector(i,2)
        ParetoFront = [ParetoFront; SolutionVector(i,:)];
        w = w + 1;
    end
end
```

```
figure(4)
plot(VarianceVector, RevenueVector, 'o');
hold on
plot(ParetoFront(:,1)', ParetoFront(:,2)')
hold off
ylabel('Revenue')
xlabel('Variance')
title(Title)
legend('Solutions','Pareto Front')


MissionVector = sortrows([CostVector' MissionPercentageVector']);
ParetoFront2 = [MissionVector(1,:)];
w = 1;
for i = 2:Counter
    if ParetoFront2(w,2) < MissionVector(i,2)
        ParetoFront2 = [ParetoFront2; MissionVector(i,:)];
        w = w + 1;
    end
end

figure(5)
plot(MissionVector(:,1)', MissionVector(:,2)', 'o');
hold on
plot(ParetoFront2(:,1)', ParetoFront2(:,2)')
hold off
ylabel('Completed missions [%]')
xlabel('Total costs')
title(Title)
legend('Solutions','Pareto Front')
```

Code 8.5: Plotting

```
clear;
clc;

```

```
Title = 'Pure Random Search';
tic; % Starts timer

run('ReadData.m'); % Reads input from file

% Create vectors and set parameters
RevenueVector = [];
CostVector = [];
MissionPercentageVector = [];
BestValueVector = [];
VarianceVector = [];
SolutionVector = [];
XVector = [];
BestValue = 0;
Counter = 0;
breaking = 0;
Mmax = 500; %number of iterations

filename = 'JoachimSimInput.xlsx';

% Creating a new random fleet before every simulation
for i = 1:Mmax
    vesselnumber = zeros(1,15);
    fleet = [80 80 80 80 80 80 80 80 80 80 80 80 80 80 80];
    fleetsize = randi([3 15]); %Diciding on number of vessels in ... the fleet
    vesselnumber(1:fleetsize) = [1:fleetsize];
    BtNummer = vesselnumber';
    fleet(1:fleetsize) = randi([1 5],1,fleetsize); %Choosing type ... of vessels in the fleet
    BtType = fleet';
    data(1) = randi(3);


    writematrix(BtNummer,filename,'Range','A2:A16');
    writematrix(BtType,filename,'Range','B2:B16');

    run('SimRun.m');
```

```

    if breaking == 1
        break
    end
end

toc; %Stops timer

run('Plot.m'); % Plot output
```

Code 8.6: Plotting

```
clear;
clc;

Title = 'Genetic Algorithem';
tic; % Starts timer

run('ReadData.m'); % Reads input from file

% Create vectors and set parameters
RevenueVector = [];
CostVector = [];
MissionPercentageVector = [];
BestValueVector = [];
VarianceVector = [];
SolutionVector = [];
Population = [];
XVector = [];
BestValue = 0;
Counter = 0;
breaking = 0;
Mmax = 96; %number of iterations
filename = 'JoachimSimInput.xlsx';

% Creating a population of 8 random fleets
for j = 1:8
```

```
    fleet = [80 80 80 80 80 80 80 80 80 80 80 80 80 80 80];
    fleetsize = randi([3 15]); %Diciding on number of vessels in ...
        the fleet
    vesselnumber = zeros(1,15);
    vesselnumber(1:fleetsize) = [1:fleetsize];
    fleet(1:fleetsize) = randi([1 5],1,fleetsize); %Choosing type ...
        of vessels in the fleet

    % Writing to input file befor simulations
    BaatNummer = vesselnumber';
    fleet = sort(fleet); %sorting the fleet vector
    BaatType = fleet';
    writematrix(BaatNummer,filename,'Range','A2:A16');
    writematrix(BaatType,filename,'Range','B2:B16');

    run('SimRun.m');

    Population = [Population; AvarageTotalRevenue fleetsize fleet];
end
Population = sortrows(Population,'descend');

% Running the whole algorithem
for k = 1:Mmax
    % Find a neighbor for the two best solutions so far and running
    % simulations for each of them
    for l = 1:2
        if Population(l,2) == 3
            Changefleetsize = randi([0 1]); %Diciding on change in ...
                fleetsize
        elseif Population(l,2) == 15
            Changefleetsize = randi([-1 0]); %Diciding on change in ...
                fleetsize
        else
            Changefleetsize = randi([-1 1]); %Diciding on change in ...
                fleetsize
        end
        fleetsize = Population(l,2) + Changefleetsize; %Diciding on ...
            number of vessels in the fleet
```

```
        fleet = Population(1,3:17);
        if Changefleetsize == -1
            fleet(randi([1 fleetsize])) = 80; %Removing one vessel ...
                from fleet
        elseif Changefleetsize == 1
            fleet(fleetsize) = randi([1 5]); %Adding one vessel to ...
                fleet
        end
        fleet = sort(fleet);

        for m=1:15
            if m ≤ fleetsize
                if fleet(m) == 1
                    ChangeVessel = randi([0 1]); %Diciding on ...
                        change in fleet
                elseif fleet(m) == 5
                    ChangeVessel = randi([-1 0]); %Diciding on ...
                        change in fleet
                else
                    ChangeVessel = randi([-1 1]); %Diciding on ...
                        change in fleet
                end
                fleet(m) = fleet(m) + ChangeVessel; %Changing fleet
            else
                fleet(m) = 80;
            end
        end
        fleet = sort(fleet);

        % Writing to input file before simulations
        vesselnumber = zeros(1,15);
        vesselnumber(1:fleetsize) = [1:fleetsize];
        BaatNummer = vesselnumber';
        fleet = sort(fleet); %Sorting the fleet vector
        BaatType = fleet';
        writematrix(BaatNummer,filename,'Range','A2:A16');
        writematrix(BaatType,filename,'Range','B2:B16');

```

```
        run('SimRun.m');
        % Putting latest fleet into the population
        Population(6+l,:) = [AvarageTotalRevenue fleetsize fleet];

    end
    Population = sortrows(Population,'descend'); % Sorting population
    if breaking ==1
        break
    end
end

toc; %stops timer

run('Plot.m'); % Plot output
```

Code 8.7: Plotting

```
clear;
clc;

Title = 'Tabu search';
tic; % Starts timer

run('ReadData.m');% Reads input from file

% Create vectors and set parameters
RevenueVector = [];
BestValueVector = [];
CostVector = [];
MissionPercentageVector = [];
VarianceVector = [];
SolutionVector = [];
XVector = [];
SolutionList = [];
TabuList = [];
BestValue = 0;
Counter = 0;
```

```
breaking = 0;
Mmax = 500; %number of iterations
filename = 'JoachimSimInput.xlsx';

vesselnumber = zeros(1,15);
fleet = [80 80 80 80 80 80 80 80 80 80 80 80 80 80 80];
fleetsize = randi([3 15]); %Diciding on number of vessels in the fleet
vesselnumber(1:fleetsize) = [1:fleetsize];
fleet(1:fleetsize) = randi([1 5],1,fleetsize); %Choosing type of ...
    vessels in the fleet

% Writing to input file befor simulations
BaatNummer = vesselnumber';
fleet = sort(fleet); %Sorting the fleet vector
BaatType = fleet';
writematrix(BaatNummer,filename,'Range','A2:A16');
writematrix(BaatType,filename,'Range','B2:B16');

run('SimRun.m');

SolutionList = [SolutionList fleet];
BestSolution = [fleet];
TabuList = [fleetsize fleet fleetsize fleet];

% Running the algorithem
for h=1:Mmax
    % While-loop makes sure that no moves towards the next fleet ...
        are tabu
    vesselnumber = zeros(1,15);
    Forbidden = 0;
    while Forbidden < 2
        Forbidden = 2;
        Move = [fleetsize fleet];

        %Diciding on change in fleet size
        if fleetsize == 3
            Changefleetsize = randi([0 1]); %Diciding on change in ...
                fleetsize
```

```
        elseif fleetsize == 15
            Changefleetsize = randi([-1 0]); %Diciding on change in ...
                fleetsize
        else
            Changefleetsize = randi([-1 1]); %Diciding on change in ...
                fleetsize
        end
        newfleetsize = fleetsize + Changefleetsize; %Diciding on ...
            number of vessels in the fleet
        Move = [Move newfleetsize];

        newfleet = fleet;
        if Changefleetsize == -1
            newfleet(randi([1 newfleetsize])) = 80; %Removing one ...
                vessel from fleet
            newfleet = sort(newfleet);
        elseif Changefleetsize == 1
            newfleet(newfleetsize) = randi([1 5]); %Adding one ...
                vessel to fleet
            newfleet = sort(newfleet);
        else
        end

        %Diciding on change in fleet
        for i=1:15
            if i≤newfleetsize
                if newfleet(i) == 1
                    ChangeVessel = randi([0 1]); %Diciding on ...
                        change in fleet
                elseif newfleet(i) == 5
                    ChangeVessel = randi([-1 0]); %Diciding on ...
                        change in fleet
                else
                    ChangeVessel = randi([-1 1]); %Diciding on ...
                        change in fleet
                end
                newfleet(i) = newfleet(i) + ChangeVessel; %Changing ...
                    fleet
```

```
            else
                newfleet(i) = 80;
            end
        end
        newfleet = sort(newfleet);
        Move = [Move newfleet];

        % Checking if the move is forbidden by the tabu list
        for i=1:max(length(TabuList(:,1)),1)
            Tabu = 0;
            for j=1:newfleetsize+1
                if Move(j) == TabuList(i,j) && Move(j+16) == ...
                    TabuList(i,j+16)
                    if j == 1 && j > length(TabuList(:,1))-5
                        Tabu = Tabu + 1;
                    end
                    Tabu = Tabu + 1;
                end
            end
            if round((newfleetsize+1)/1.5) ≤ Tabu
                Forbidden = 1
                break
            end
        end

        %Checking if the solution already has been visited before
        for i=1:max(length(SolutionList(:,1)),1)
            if newfleet == SolutionList(i,:)
                Forbidden = 0
                break
            end
        end

    end

    % Checking if Move is in the TabuList and updating it
    test = 0;
```

```
    for d = 1:length(TabuList(:,1))
        if Move == TabuList(d,:)
            if d == length(TabuList(:,1))
                test = 1;
                break
            end
            TabuList(d:length(TabuList(:,1))-1,:) = ...
                TabuList(d+1:length(TabuList(:,1)),:);
            TabuList(length(TabuList(:,1)),:) = Move;
            test = 1;
            break
        else
        end
    end

    % Updating it TabuList
    if test == 0;
        if length(TabuList(:,1)) ≥ 10
            TabuList = TabuList(2:length(TabuList(:,1)),:);
            TabuList = [TabuList; Move];
        else
            TabuList = [TabuList; Move];
        end
    end

    fleetsize = newfleetsize;
    fleet = newfleet;

    % Writing to input file befor simulations
    vesselnumber(1:newfleetsize) = [1:newfleetsize];
    BaatNummer = vesselnumber';
    BaatType = newfleet';
    writematrix(BaatNummer,filename,'Range','A2:A16');
    writematrix(BaatType,filename,'Range','B2:B16');

    run('SimRun.m');

    SolutionList = [SolutionList; newfleet];
```

```
    if breaking == 1
        break
    end
end

toc; %Stops timer

run('Plot.m');
```

Code 8.8: Plotting

```
clear;
clc;

Title = 'Simulated Annealing';
tic; % Starts timer

run('ReadData.m'); % Reads input from file

% Create vectors and set parameters
RevenueVector = [];
CostVector = [];
MissionPercentageVector = [];
BestValueVector = [];
VarianceVector = [];
SolutionVector = [];
XVector = [];
BestValue = 0;
breaking = 0;
Phases = 10; %number of iterations
Iterations = 20;
Counter = 0;
filename = 'JoachimSimInput.xlsx';

vesselnumber = zeros(1,15);
```

```
fleet = [80 80 80 80 80 80 80 80 80 80 80 80 80 80 80];
fleetsize = randi([3 15]); %Diciding on number of vessels in the fleet
vesselnumber(1:fleetsize) = [1:fleetsize];
fleet(1:fleetsize) = randi([1 5],1,fleetsize); %Choosing type of ...
    vessels in the fleet
BtNummer = vesselnumber';
BtType = fleet';

writematrix(BtNummer,filename,'Range','A2:A16');
writematrix(BtType,filename,'Range','B2:B16');

%run('MainJoachim.m');
run('SimRun.m');

CurrentValue = Revenue;

for l = 1:Phases
    for j = 1:Iterations
        %Deciding on change in fleetsize
        if fleetsize == 3
            Changefleetsize = randi([0 1]);
        elseif fleetsize == 15
            Changefleetsize = randi([-1 0]);
        else
            Changefleetsize = randi([0 1]);
            if Changefleetsize == 0
                Changefleetsize = -1;
            end
        end
        newfleetsize = fleetsize + Changefleetsize; %Changing ...
            number of vessels in the fleet
        newfleet = fleet;

        if Changefleetsize == -1
            newfleet(fleetsize) = 80; %Removing one vessel from fleet
        elseif Changefleetsize == 1
            newfleet(newfleetsize) = randi([1 5]); %Adding one ...
                vessel to fleet
```

```
        else
        end

        % Deciding on change in fleet
        for m=1:15
            if m ≤ newfleetsize
                if newfleet(m) == 1
                    ChangeVessel = randi([0 1]);
                elseif newfleet(m) == 5
                    ChangeVessel = randi([-1 0]);
                else
                    ChangeVessel = randi([0 1]);
                    if ChangeVessel == 0
                        ChangeVessel = -1;
                    end
                end
                newfleet(m) = newfleet(m) + ChangeVessel; %Changing ...
                    fleet
            else
                newfleet(m) = 80;
            end
        end

        % Writing to input file
        vesselnumber = zeros(1,15);
        vesselnumber(1:newfleetsize) = [1:newfleetsize];
        BtNummer = vesselnumber';
        BtType = newfleet';
        writematrix(BtNummer,filename,'Range','A2:A16');
        writematrix(BtType,filename,'Range','B2:B16');

        run('SimRun.m');

        if breaking == 1
            break
        end

        Delta = CurrentValue - Revenue;
```

```
        T = 2*Phases/l - 0.1*l + 5;

        if Delta ≤ 0
            fleet = newfleet;
            fleetsize = newfleetsize;
        else
            U = rand;
            if U ≤ exp(-(Delta/T))
                fleet = newfleet;
                fleetsize = newfleetsize;
            end
        end

        CurrentValue = Revenue;

    end
        if breaking == 1
            break
        end
end

toc; %Stops timer

run('Plot.m');
```

Code 8.9: Plotting

```
clear;
clc;

Title = 'LAST';
tic;

run('ReadData.m');% Reads input from file

% Create vectors and set parameters
RevenueVector = [];
```

```
CostVector = [];
MissionPercentageVector = [];
BestValueVector = [];
VarianceVector = [];
SolutionVector = [];
XVector = [];
BestValue = 0;
breaking = 0;
Mmax = 500; %number of iterations
Counter = 0;
filename = 'JoachimSimInput.xlsx';
M = 1;
Fbest = 0;
Rmax = 230;
Rmin = 150;
My = 0.1;
Mu = 0.06;
Solution = zeros(1,16);
sol = 0;
vesselnumber = zeros(1,15);
fleet = [80 80 80 80 80 80 80 80 80 80 80 80 80 80 80];

Bfs = zeros(1,13);
B = zeros(5,15);

% Setting probabilities for the first fleet
Pf = zeros(1,13);
for k = 1:13
    Pf(k) =1/13; %Set the probability of each choice.
end
Pm = zeros(5,15);
for i = 1:5
    for j = 1:15
        Pm(i,j) =1/5; %Set the probability of each choice.
    end
end


```

```
while M < Mmax + 2 %Starting iteration loop

    vesselnumber = zeros(1,15);
    fleet = [80 80 80 80 80 80 80 80 80 80 80 80 80 80 80];

    Random = rand;
    Pfcum = 0;
    for i = 1:13
        Pfcum = Pfcum + Pf(i);
        if Random ≤ Pfcum
            fleetsize = i+2; %choosing fleetsize
            break
        end
    end

    for i = 1:fleetsize
        Random = rand;
        Pmcum = 0;
        for k = 1:5
            Pmcum = Pmcum + Pm(k,i);
            if Random ≤ Pmcum
                fleet(i) = k; %choosing fleet
                break
            end
        end
    end

    for g = 1:M
        if Solution(g) == [fleetsize sort(fleet)]
            sol = 1;
            break
        end
    end
    if sol == 1
        continue
    end

    vesselnumber(1:fleetsize) = [1:fleetsize];
```

```
    BtNummer = vesselnumber';
    BtType = fleet';

    writematrix(BtNummer,filename,'Range','A2:A16');
    writematrix(BtType,filename,'Range','B2:B16');

    run('SimRun.m');

    R = Revenue;
    F = (R-Rmin)/(Rmax-Rmin);
    Fbest = F;
    xfs = fleetsize - 2;
    x = fleet(1:fleetsize);

    % Changing probabilities for choosing a decision variable
    accumulatorfs = 0;
    for d = 1:13
        if Bfs(d) < Bfs(xfs)
            Pf(d) = Pf(d) - Mu*(Bfs(xfs)-Bfs(d))*Pf(d); %changing ...
                the probability of each choice.
        elseif Bfs(d) > Bfs(xfs)
            Pf(d) = Pf(d) + ...
                Mu*(Bfs(d)-Bfs(xfs))*(1-Pf(d))*Pf(xfs)/12; ...
                %changing the probability of each choice.
        end

        if d == xfs
        else
            accumulatorfs = accumulatorfs + Pf(d); %changing the ...
                probability of each choice.
        end
    end
    Pf(xfs) = 1 - accumulatorfs; %changing the probability of each ...
        choice.

    % Changing probabilities for choosing a decision variable
    for i = 1:fleetsize
        accumulator = 0;
```

```
            for d = 1:5
                if B(d,i) < B(x(i),i)
                    Pm(d,i) = Pm(d,i) - My*(B(x(i),i)-B(d,i))*Pm(d,i); ...
                        %changing the probability of each choice.
                elseif B(d,i) > B(x(i),i)
                    Pm(d,i) = Pm(d,i) + ...
                        My*(B((d),i)-B(x(i),i))*((1-Pm(d,i))*Pm(x(i),i)/4); ...
                        %changing the probability of each choice.
                end
                if d == x(i)
                else
                    accumulator = accumulator + Pm(d,i); %changing the ...
                        probability of each choice.
                end
            end
            Pm(x(i),i) = 1 - accumulator; %changing the probability of ...
                each choice.
        end

        % Updating B matrix
        for i = 1:fleetsize
            if F > B(x(i),i)
                B(x(i),i) = F;
            end
        end
        if F > Bfs(xfs)
            Bfs(xfs) = F;
        end

        Solution = [Solution; fleetsize sort(fleet)];
        if breaking == 1
            break
        end
        M = M + 1;
    end

    toc;
```

```
run('Plot.m');
```

Input data

Boat type	Length [m]	Capacity slaughter [ton]	Capacity smolt [ton]	Speed [kn]	CAPEX [mNOK]	Sail cost [mNOK/hour]	Work Cost [mNOK/hour]	Crew Cost [mNOK/hour]	Delousing 1	Delousing 2
1	85	480	160	12	15	0.0019	0.00072	0.0022	1	1
2	85	480	160	12	12.5	0.0019	0.00072	0.0022	1	0
3	63	225	75	10	10	0.0013	0.00062	0.0019	1	0
4	63	180	60	11	10	0.0013	0.00062	0.0019	1	0
5	51	150	50	11	7.5	0.0010	0.00045	0.0016	1	0

Boat type	Smolt loading	Smolt unloading	Slaughter loading	Slaughter unloading	Slaughter direct unloading	Delousing 1	Delousing 2
1	80	100	150	200	80	100	350
2	80	100	150	200	80	100	0.0001
3	60	75	150	150	80	75	0.0001
4	60	75	150	150	80	75	0.0001
5	50	50	100	100	80	50	0.0001

Boat type	Smolt loading	Smolt unloading	Slaughter loading	Slaughter unloading	Delousing	Sailing
1	8	6	6	7	6	7
2	8	6	6	7	6	7
3	8	5	5	6	5	6
4	8	5	5	6	5	6
5	7	4	4	5	4	6

Mission type	Mission number	Loc 1 low	Loc 1 high	Loc 2 low	Loc 2 high	Work 1	Work 2 low	Work 2 high	Ton/mission low	Ton/mission high	Time between missions
Smolt	1	2	2	5	15	1	2	2	50	500	75
Slaughter	2	5	15	3	4	3	4	5	200	1200	12
Delousing 1	3	1	1	5	15	0	6	6	200	1200	25
Delousing 2	4	1	1	5	15	0	7	7	200	1200	50

Boat type	Smolt	Slaughter	Delousing 1	Delousing 2
1	1	1	1	1
2	1	1	1	0
3	1	1	1	0
4	1	1	1	0
5	1	1	1	0

Location number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	20	15	24	26	57	49	63	60	64	44	44	50	43	26
2	20	0	14	7	36	55	30	48	70	48	36	24	66	56	27
3	15	14	0	8	41	67	44	61	74	51	49	33	65	57	35
4	24	7	8	0	47	69	35	55	81	40	47	22	75	66	40
5	26	36	41	47	0	37	52	59	36	83	35	55	34	20	12
6	57	55	67	69	37	0	52	45	35	95	26	65	58	40	32
7	49	30	44	35	52	52	0	20	79	44	27	18	86	72	40
8	63	48	61	55	59	45	20	0	77	61	25	40	93	74	47
9	60	70	74	81	36	35	79	77	0	115	55	85	32	17	43
10	64	48	51	40	83	95	44	61	115	0	70	28	114	104	73
11	44	36	49	47	35	26	27	25	55	70	0	39	68	50	24
12	44	24	33	22	55	65	18	40	85	28	39	0	89	76	45
13	50	66	65	75	34	58	86	93	32	114	68	89	0	19	46
14	43	56	57	66	20	40	72	74	17	104	50	76	19	0	31
15	26	27	35	40	12	32	40	47	43	73	24	45	46	31	0

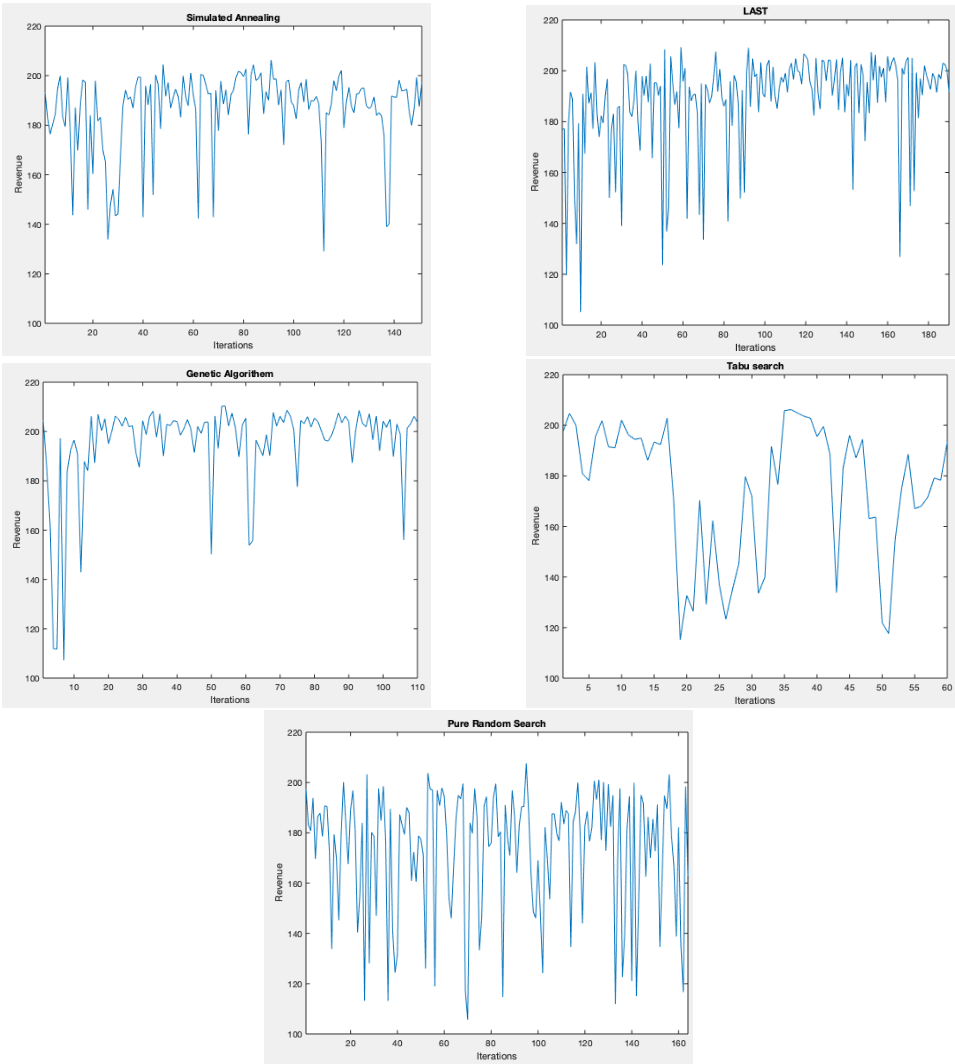

Figure 8.5: Pure Random Search Values

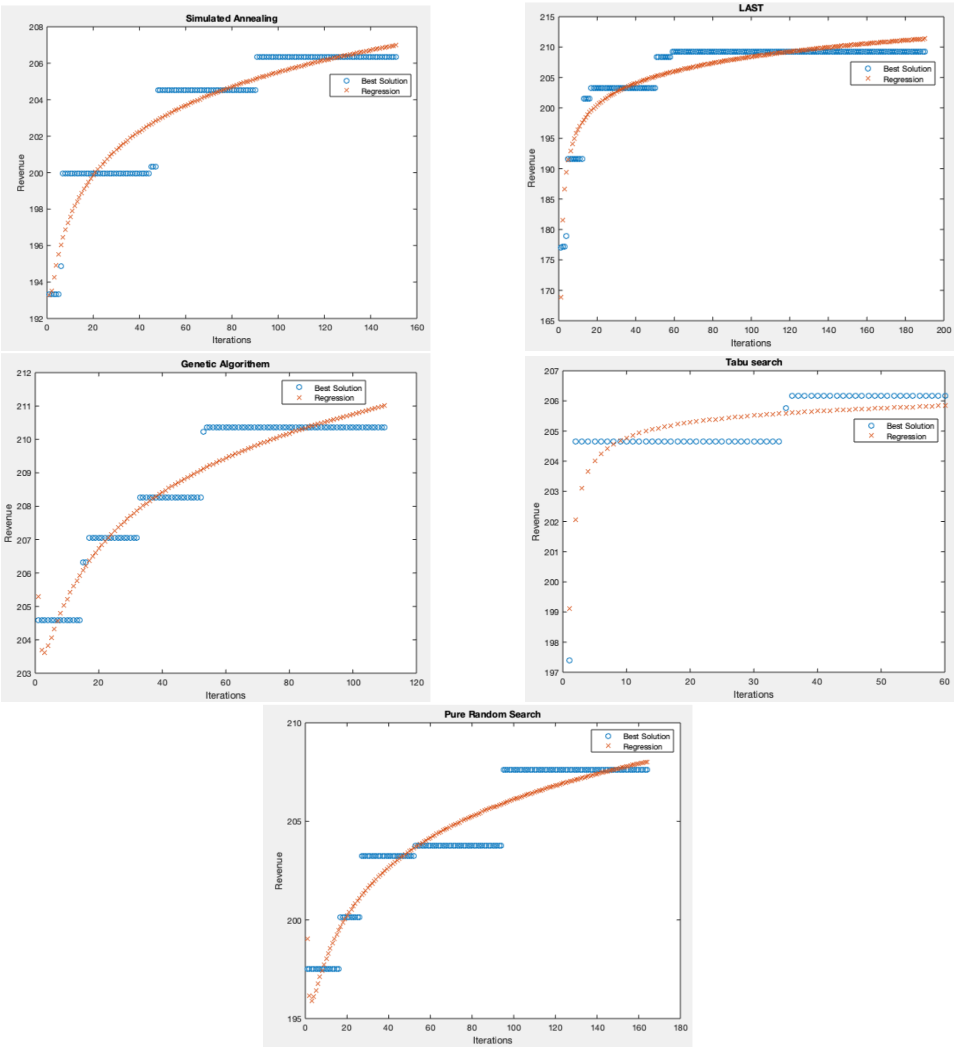

Figure 8.10: Pure Random Search Regression

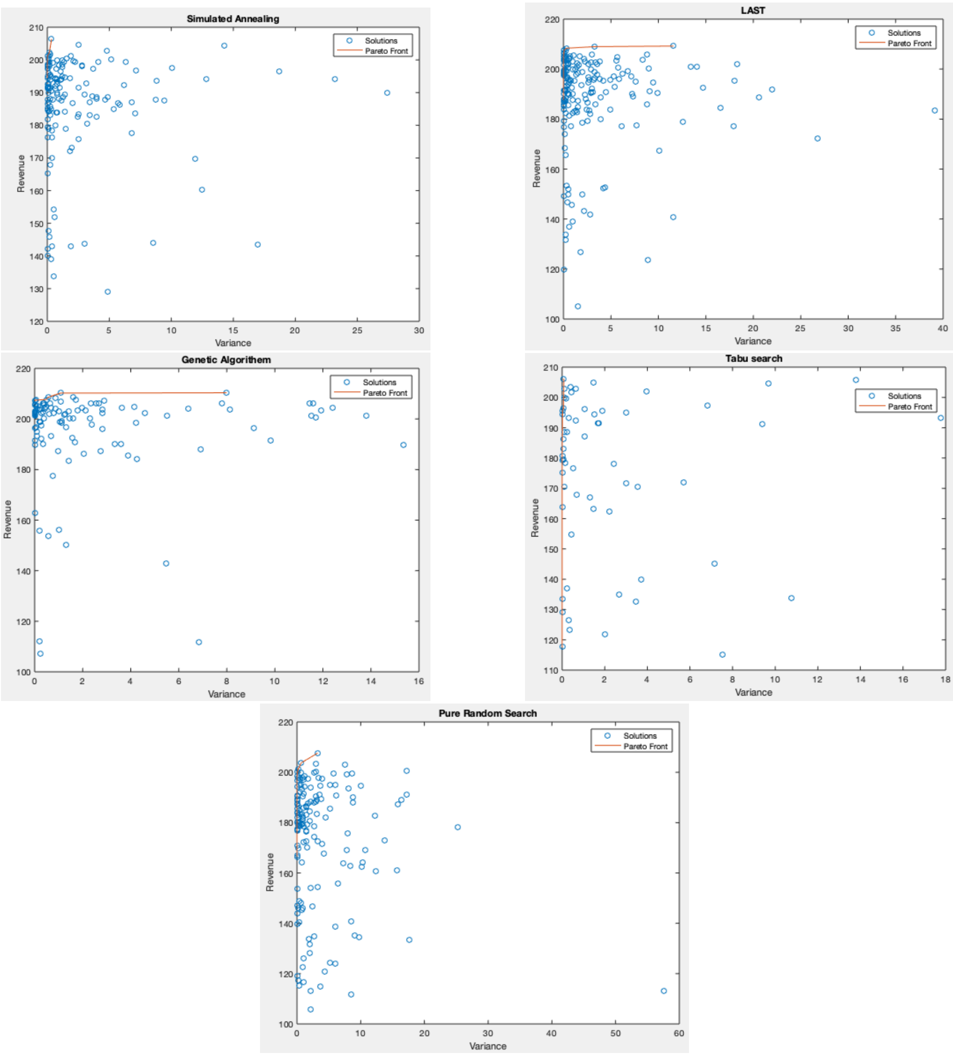

Figure 8.15: Pure Random Search Variance

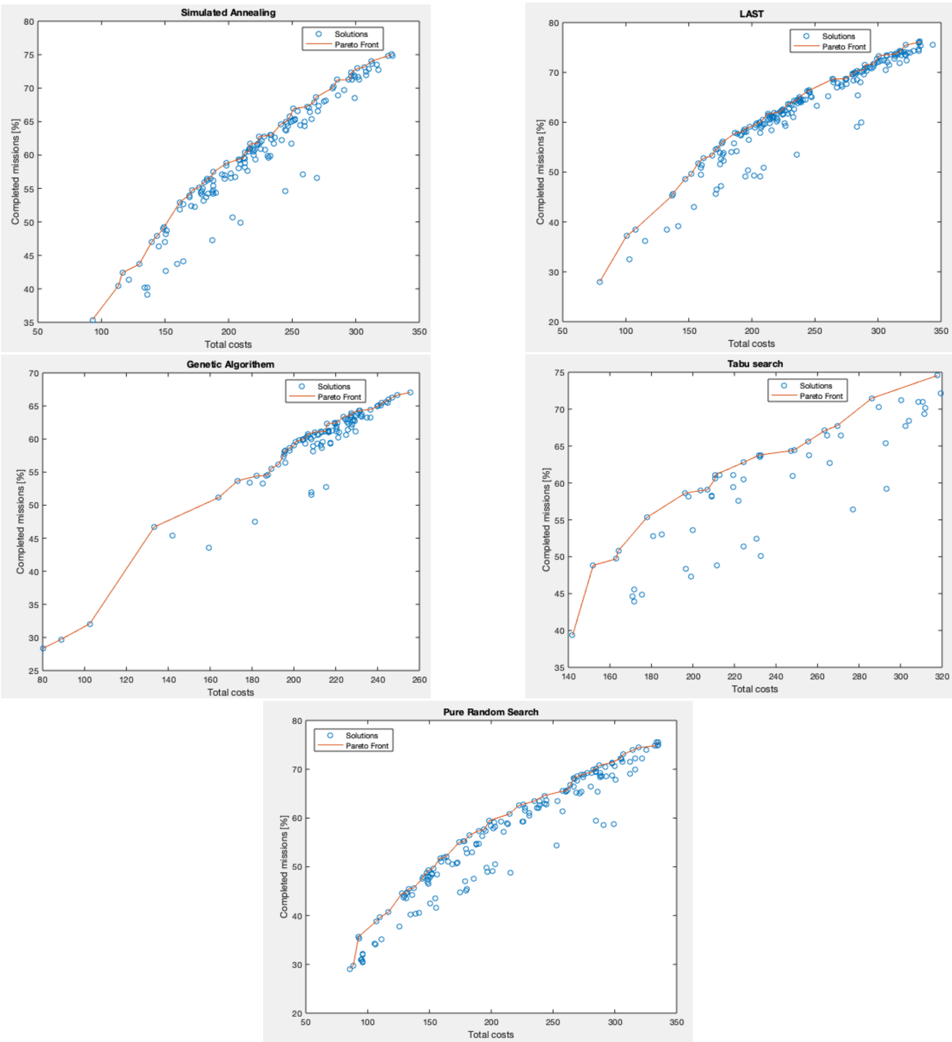

Figure 8.20: Pure Random Search Pareto Front

GÅSØ
FREYJA

Tidsrom	Aktivitet	kW behov/til gode.	Totalt behov kW x h	Totalt til gode kW x h	Totalt på tavle x h	Antall gen.på tavle/kW/% belastning.
Dag 1						
00:00-13:30	St.by Sistranda	180-220kW/1180kW/84%	2970	15930	18900	1/1400kW/16%
13:30-14:30	manøver/ avgang.	650kW/2150kW/77%	325	2150	2800	2/2800kW/23%
14:00-16:30	Transit tom båt, 10kn	1100kW/300kW/22%	2750	750	3500	1/1400kW/78%
16:30-17:00	Manøver ankomst mær.	700kW/2100kW/75%	350	1050	1400	2/2800kW/25%
17:00-19:30	Lasting ved mær	800-900kW/500kW/36%	2250	1250	3500	1/1400kW/64%
19:30-20:00	Forhaling ved mær.	1300kW/1500kW/54%	650	750	1400	2/2800kW/46%
20:00-22:00	lasting ved mær.	800-900kW/500kW/36%	1800	1000	2800	1/1400kW/64%
22:00-22:30	Manøver avgang mær.	1600kW/1200kW/43%	800	600	1400	2/2800kW/57%
22:30-02:30	Transit Lastet m/ UV	2400kW/400kW/14%	9600	1600	11200	2/2800kW/86%
Dag2						
02:30-03:30	Manøver ank.slakteri	1600kW/2600kW/62%	1600	2600	4200	3/4200kW/38%
03:30-04:30	Avventer lossing med UV	900KW/500kW/36%	900	500	1400	1/1400kW/64%
04:30-06:45	Lossing til Ventemerd	600-650 KW/750kW/54%	163	2063	3150	1/1400kW/46%
06:45-07:30	manøver avg.slakteri	650kW/2150kW/77%	488	1613	2100	2/2800kW/23%
07:30-09:00	Transit tom båt, 12,5 kn	2100kW/700kW/25%	3150	1050	4200	2/2800kW/75%
09:00-10:00	Manøver	700kW/2100kW/75%	700	2100	2800	2/2800kW/25%
10:00-21:00	Lasting/Lossing ved mær	800-900 kW/500kW/36%	9900	5500	15400	1/1400kW/64%
21:00-21:30	Manøver	700kW/2100kW/75%	350	1050	1400	2/2800kW/25%
21:30-24:00	Transit lastet m/UV	2400kW/400kW/14%	6000	1000	7000	2/2800kW/86%
Dag 3						
00:00-01:00	Manøver ank.slakteri	1600kW/2600kW/62%	1600	2600	4200	3/4200kW/38%
0100-07:00	Avventer lossing Ulvan	550kW/850kW/61%	3300	5100	8400	1/1400kW/39%
07:00-10:00	Lossing til Ventemerd	600-650KW/750kW/54%	1950	2250	4200	1/1400kW/46%
10:00-11:00	manøver avg.slakteri	650kW/2150kW/77%	650	2150	2800	2/2800kW/23%
11:00-13:00	Lenser og vasker	230-270kW/1130kW/81%	540	2260	2800	1/1400kW/19%
13:00-14:00	Manøver ank.kai.	650kW/2150kW/77%	650	2150	2800	2/2800kW/23%
14:00-16:00	Ozonering ved kai	700kW/700kW/50%	1400	1400	2800	1/1400kW/50%
16:00-17:00	forhaling.	650kW/2150kW/77%	650	2150	2800	2/2800kW/23%
17:00-24:00	St.by Sistranda	180-220kW/1180kW/84%	1540	8260	1400	1/1400kW/16%
			57026	70876	120750	1,57/2200/44%

Figure 8.21: Simulation-based optimization loop

EL. BALANCE - TAUPIRI - FITJAR MEKANISKE VERKSTED

CIRCUIT NO.	CONSUMERS		EFFEKT KW	U.F.	L.F.	MANØVERING TIL /FRA KAI KW	L.F.	GANGE KW	L.F.	LASTING KW	L.F.	LAND STRØM KW	L.F.	NØD STRØM KW
278	**ICCP/ICAF**													
	ICCP		3,2	0,8	0,8	2,5	0,8	2,5	0,8	2,5	0,8	2,5		
	ICAF		0,3	0,8	0,8	0,2	0,8	0,2	0,8	0,2	0,8	0,2		
350	**PROSESSANLEGG**													
	VAKUM PUMPE 1	Softst.	65,0	0,8					0,8	52,0				
	VAKUM PUMPE 2	Softst.	65,0	0,8					0,8	52,0				
	VAKUM PUMPE 3	Softst.	65,0	0,8					0,8	52,0				
	ARBEIDSLUFTKOMPRESSOR 1		50,0	0,8					0,5	25,0				
	ARBEIDSLUFTKOMPRESSOR 2		10,6	0,8					0,5	5,3				
	Prosessanlegg div. Uttak		15,0	0,8					0,8	12,0				
	Prosessanlegg Trafo/E-Cells		26,0	0,8					0,8	20,8				
	Prosessanlegg Trafo/E-Cells		26,0	0,8					0,8	20,8				
362	**RSW ANLEGG**													
	RSW KOMPRESSOR 1	Freq.	122,0	0,8			0,8	97,6	0,8	97,6				
	RSW KOMPRESSOR 2	Freq.	122,0	0,8			0,8	97,6	0,8	97,6				
	RSW SIRKULASJONSPUMPE 1	Freq.	44,4	0,8			0,8	35,5	0,8	35,5				
	RSW SIRKULASJONSPUMPE 2	Freq.	44,4	0,8										
	RSW FYLLEPUMPE	Freq.	30,0	0,8			0,8	24,0	0,1	3,0				
	RSW KONDENSERPUMPE 1	Freq.	13,2	0,8			0,8	10,6	0,8	10,6				
	RSW KONDENSERPUMPE 2	Freq.	13,2	0,8										
363	**BLODVANN**													
363.001.010	BLODVANN PUMPE		9,2	0,8					0,8	7,4				
363.026.010	FLUSHING SCHUTES PUMP		12,6	0,8					0,8	10,1				
	Sjøvann til E celler		7,5	0,8					0,8	6,0				
	Blodvann Renseanlegg		19,0	0,8					0,8	15,2				
364	**OKSYGEN OG OZONANLEGG**													
	OKSYGEN Generator		0,2	0,8			0,8	0,2	0,1	0,0				
	OZONE Generator		8,6	0,8			0,8	6,9	0,1	0,9				
	UV-Reactor Ozone System		26,4	0,8			0,8	21,1	0,1	2,6				
371	**VANNSIRKULASJON**													
	PROSESSANLEGG - VANNSIRKULASJONSPUMPE 1	Freq.	45,0	0,8					0,8	36,0				
	PROSESSANLEGG - VANNSIRKULASJONSPUMPE 2	Freq.	45,0	0,8					0,8	36,0				
	UV- ANLEGG		39,0	0,8					0,8	31,2				
381	**TANKVASKEANLEGG**													
	TANKVASKEPUMPE Booster Pump		23,0	0,8					0,1	2,3				
	SW. Pump 1 - Tankwashing System		7,5	0,8					0,1	0,8				
	SW. Pump 2 - Tankwashing System		7,5	0,8					0,1	0,8				
403	**STYRE MASKIN**													
	STYREMASKINS PUMPE 1	Freq.	5,5	0,8	0,8	4,4	0,3	1,7	0,1	0,6				
	STYREMASKINS PUMPE 2	Freq.	5,5	0,8	0,8	4,4	0,3	1,7	0,1	0,6			1,0	5,5
404	**TUNNEL THRUSTERS**													
	BAUG THRUSTER	Freq.	250,0	0,8	0,6	150,0								
	HYDRAULIKK FOR BAUG THRUSTER		2,2	0,8	0,6	1,3								
	AKTER THRUSTER	Freq.	250,0	0,8	0,6	150,0								
	HYDRAULIKK FOR AKTRE THRUSTER		2,2	0,8	0,6	1,3								

Page 1 E-0431-100-0

Figure 8.22: Simulation-based optimization loop

	EL. BALANCE - TAUPIRI - FITJAR MEKANISKE VERKSTED													
CIRCUIT NO.	CONSUMERS		EFFEKT KW	U.F.	L.F.	MANØVERING TIL /FRA KAI KW	L.F.	GANGE KW	L.F.	LASTING KW	L.F.	LAND STRØM KW	L.F.	NØD STRØM KW
41	**NAVIGATION**													
	NAVIGATIONAL EQUIPMENT		3,0	0,8	0,8	2,4	0,8	2,4	0,8	2,4	0,8	2,4	1,0	3,0
42	**COMMUNICATION**													
	COMMUNICATION EQUIPMENT		1,5	0,8	0,8	1,2	0,8	1,2	0,8	1,2	0,8	1,2	1,0	1,5
460	**HYDRAULIC DECK MACHINERY**													
	HYDRAULIKK PUMPE 1		86,0	0,8					0,8	68,8				
	HYDRAULIKK PUMPE 2		86,0	0,8					0,8	68,8				
501	**MOB BÅT**													
	MOB BÅT DAVIT		18,0	0,8										
551	**GALLEY MACHINERY**													
	GALLEY MACHINERY		18,0	0,8	0,2	3,6	0,2	3,6	0,2	3,6	0,2	3,6		
558	**LAUNDRY EQUIPMENT**													
	LAUNDRY MACHINERY		2,0	0,8	0,2	0,4	0,2	0,4	0,2	0,4	0,2	0,4		
570	**VENTILATION / AIR CONDITION**													
	AC1 - Accomodation and Wheelhouse	Freq.	6,4	0,8	0,7	4,5	0,7	4,5	0,7	4,5	0,3	1,9		
	AC1 - Accomodation and Wheelhouse Heating		47,0	0,8	0,7	32,9	0,7	32,9	0,7	32,9	0,3	14,1		
	Aircondition System		13,8	0,8										
574.001.010	S-7 - Em. Generator Room Supply Fan		0,2	0,8	0,7	0,1	0,7	0,1	0,7	0,1	0,1	0,0		
574.002.010	E-6 - Exhaust Fan Workshop Aft.		0,1	0,8	0,7	0,1	0,7	0,1	0,7	0,1	0,1	0,0		
574.005.010	S-5 - Supply Fan No. 1 Engine Room	Freq.	3,5	0,8	0,7	2,4	0,7	2,4	0,7	2,4	0,1	0,3		
574.006.010	S-6 - Supply Fan No. 2 Engine Room	Freq.	3,5	0,8	0,7	2,4	0,7	2,4	0,7	2,4	0,1	0,3		
574.007.011	S-4 - Factory Supply Fan		1,3	0,8	0,7	0,9	0,7	0,9	0,7	0,9	0,1	0,1		
574.008.010	E-12 - Exhaust Fan Wardrobe		0,1	0,8	0,7	0,1	0,7	0,1	0,7	0,1	0,1	0,0		
574.010.010	E-5 - Exhaust Fan RSW Room		0,2	0,8	0,7	0,1	0,7	0,1	0,7	0,1	0,1	0,0		
574.010A.010	E-4 - Exhaust Ex-Fan RSW Room		0,8	0,8	0,7	0,5	0,7	0,5	0,7	0,5	0,1	0,1		
574.019.010	E-3 - Exhaust Fan Wardrobe		0,1	0,8	0,7	0,0	0,7	0,0	0,7	0,0	0,1	0,0		
574.021.010	S-8 - Suppy Fan Technical Room Fwd.		1,3	0,8	0,7	0,9	0,7	0,9	0,7	0,9	0,1	0,1		
574.022.010	S-9 - Supply Fan Corridor Fishtanks		0,2	0,8	0,7	0,1	0,7	0,1	0,7	0,1	0,1	0,0		
574.024.010	E-8 - Exhaust Fan Workshop Fwd.		0,0	0,8	0,7	0,0	0,7	0,0	0,7	0,0	0,1	0,0		
574.027.010	S-2 - Supply Fan ECR		0,1	0,8	0,7	0,1	0,7	0,1	0,7	0,1	0,1	0,0		
574.032.010	S-10 - Supply Fan Vacuum Pump Room		1,3	0,8	0,7	0,9	0,7	0,9	0,7	0,9	0,1	0,1		
574.033.010	S-11 - Supply Fan Steering Gear Room		0,1	0,8	0,7	0,0	0,7	0,0	0,7	0,0	0,1	0,0		
574.039.010	S- - Supply Fan Hydraulic Room		0,0	0,8	0,7	0,0	0,7	0,0	0,7	0,0	0,1	0,0		
574.045.010	S- - Supply Fan Technical Room		1,3	0,8	0,7	0,9	0,7	0,9	0,7	0,9	0,1	0,1		
581	**SANITARY SUPPLY SYSTEM**													
581.002.010	HYDROPHORE PUMP NO. 1		2,6	0,8	0,5	1,3	0,5	1,3	0,5	1,3	0,2	0,5		
581.002.020	HYDROPHORE PUMP NO. 2		2,6	0,8	0,5	1,3	0,5	1,3	0,5	1,3	0,2	0,5		
581.003.010	FW. PUMP TO CLEANING SYSTEM		6,3	0,8	0,5	3,2	0,5	3,2	0,5	3,2	0,2	1,3		
581.023.010	VARMTVANNS SIRKULASJONSPUMPE 1		0,2	0,8	0,5	0,1	0,5	0,1	0,5	0,1	0,2	0,0		
581.023.020	VARMTVANNS SIRKULASJONSPUMPE 2		0,2	0,8	0,5	0,1	0,5	0,1	0,5	0,1	0,2	0,0		

Page 2 E-0431-100-0

Figure 8.23: Simulation-based optimization loop

	EL. BALANCE - TAUPIRI - FITJAR MEKANISKE VERKSTED													
CIRCUIT NO.	CONSUMERS		EFFEKT KW	U.F.	L.F.	MANØVERING TIL /FRA KAI KW	L.F.	GANGE KW	L.F.	LASTING KW	L.F.	LAND STRØM KW	L.F.	NØD STRØM KW
582	**SANITARY DISCHARGE SYSTEM**													
	Jetz Vakumtoalett		14,8	0,8	0,3	4,4	0,3	4,4	0,3	4,4	0,1	1,5		
582.001.010	Sewage Discharge Pump		1,5	0,8	0,3	0,5	0,3	0,5	0,3	0,5	0,1	0,2		
601	**DIESEL ENGINES**													
	TØRNE GEAR		1,8	0,8							0,7	1,3		
	Jacket Water Heater ME		6,0	0,8							0,5	3,0		
701	**FUEL OIL TRANSFER SYSTEM**													
701.001.010	FO Transfer pump		2,5	0,8	0,1	0,3	0,1	0,3	0,1	0,3				
703	**FUEL OIL**													
703.001.050	FO St.By Pump ME		1,3	0,8										
711	**LUBE OIL TRANSFER SYSTEM**													
711.010.010	LO St.by Pump ME		6,3	0,8										
	LO St.By Pump Gearbox		6,4	0,8										
	LO St.By Pump Gear		15,0	0,8										
712	**LUBE OIL PURIFICATION PLANTS**													
712.035.010	Sludge Pump		1,1	0,8	0,1	0,1	0,1	0,1	0,1	0,1				
721	**SEAWATER COOLING SYSTEM**													
721.010.010	SJØVANNS KJØLEPUMPE 1 ME		4,6	0,8	0,8	3,7	0,8	3,7	0,8	3,7				
721.010.020	SJØVANNS KJØLEPUMPE 2 ME		4,6	0,8										
721.010.030	SJØVANNS KJØLE PUMPE FOR HYDRAULIKK		2,6	0,8					0,8	2,0				
721.010.040	SJØVANNS KJØLE PUMPE 1 AUX ENGINE		3,4	0,8	0,8	2,7	0,8	2,7	0,8	2,7				
721.010.050	SJØVANNS KJØLE PUMPE 2 AUX ENGINE		3,4	0,8										
721.010.060	SJØVANNS KJØLE PUMPE AC COOLER		2,6	0,8	0,8	2,0	0,8	2,0	0,8	2,0	0,2	0,5		
721.010.080	SJØVANNS KJØLE PUMPE VAKUM KOMPRESSOR		2,6	0,8	0,8	2,0	0,8	2,0	0,8	2,0				
722	**FRESHWATER COOLING SYSTEM**													
722.003.010	FW. GEARBOX CIRCULATION PUMP 1		2,6	0,8	0,8	2,0	0,8	2,0	0,8	2,0				
722.003.020	FW. GEARBOX CIRCULATION PUMP 2		2,6	0,8										
722.005.010	FW. AC UNIT COOLING PUMP		2,6	0,8	0,8	2,0	0,8	2,0	0,8	2,0	0,2	0,5		
722.006.020	FW. LT St.By Pump ME		2,6	0,8										
722.006.030	FW. HT St.By Pump ME		6,3	0,8										
801	**BALLAST SYSTEM**													
801.001.010	Ballast Pump		6,3	0,8	0,2	1,3	0,2	1,3	0,5	3,2				
	Ballast Separator		20,0	0,8	0,2	4,0	0,2	4,0	0,5	10,0				
803	**BILGE SYSTEM**													
803.001.010	Bilge Pump		5,5	0,8	0,1	0,6	0,1	0,6	0,1	0,6	0,1	0,3		
	Bilge Water Separator		3,0	0,8	0,1	0,3	0,1	0,3	0,1	0,3	0,1	0,3		
813	**FIRE FIGHTING SYSTEM**													
813.001.010	Fire Pump		17,3	0,8										
813.031.010	Emergency Fire Pump		12,6	0,8									1,0	12,6
885	**LIGHT & SMALL POWER**													
	FLOM LYS		3,5	0,9	0,3	1,1	0,3	1,1	0,6	2,1	0,1	0,4	0,1	0,4
L3-F19/20	SØKE LYS		2,0	0,9	0,1	0,2								
LP	LYS INNREDNING		3,0	0,9	0,6	1,8	0,6	1,8	0,6	1,8	0,6	1,8	0,2	0,6
LP	LYS UNDER SKIP		2,0	0,9	0,6	1,2	0,6	1,2	0,6	1,2	0,6	1,2	0,2	0,4

Page 3 E-0431-100-0

Figure 8.24: Simulation-based optimization loop

EL. BALANCE - TAUPIRI - FITJAR MEKANISKE VERKSTED

CIRCUIT NO.	CONSUMERS	EFFEKT KW	U.F.	L.F.	MANØVERING TIL /FRA KAI KW	L.F.	GANGE KW	L.F.	LASTING KW	L.F.	LAND STRØM KW	L.F.	NØD STRØM KW
LP	LYS UTE	2,0	0,9	0,3	0,6	0,3	0,6	0,6	1,2	0,3	0,6	0,1	0,2

Page 4 E-0431-100-0

Figure 8.25: Simulation-based optimization loop

EL. BALANCE - TAUPIRI - FITJAR MEKANISKE VERKSTED

CIRCUIT NO.	CONSUMERS	EFFEKT KW	U.F.	L.F.	MANØVERING TIL /FRA KAI KW	L.F.	GANGE KW	L.F.	LASTING KW	L.F.	LAND STRØM KW	L.F.	NØD STRØM KW
885	**HEATING**												
LP	OPPVARMING INNREDNING	35,0	0,9	0,5	17,5	0,5	17,5	0,5	17,5	0,5	17,5		
LP	OPPVARMING UNDER SKIP	42,0	0,9	0,5	21,0	0,5	21,0	0,5	21,0	0,5	21,0		
	TOTAL LAST				444		429		916		80		24

	MANØVERING TIL /FRA KAI	GANGE	LASTING	LAND STRØM	NØD STRØM
LAST TILKOBLET BUS A	206	225	519		
TILGJENGELIG EFFEKT AKSELGENERATOR	750	750	750		
LAST FAKTOR	27 %	30 %	69 %		
LAST TILKOBLET BUS B	239	205	398		
TILGJENGELIG EFFEKT AUX GEN. 1	565	565	565		
LAST FAKTOR	42 %	36 %	70 %		
LAST TILKOBLET MAIN BUS	444	429	916	80	24
TILGJENGELIG EFFEKT	1315	1315	1315	95	60
LAST FAKTOR	34 %	33 %	70 %	84 %	40 %

LAST LANDSTRØM	80	kW
TILGJENGELIG LANDSTRØM	95	kW
LAST FAKTOR	84 %	

LAST NØDSTRØM	24	kW
TILGJENGELIG NØDGENERATOR	95	kW
LAST FAKTOR	25 %	

Page 5 E-0431-100-0

Figure 8.26: Simulation-based optimization loop