An Optimization Model for Short-Term Routing and Scheduling of Offshore Wind Maintenance

Master's Thesis

Aksel Håland Laugaland

University of Bergen

Supervised by Dag Haugland

Department of Informatics

&

Geophysical Institute

May 31, 2023

Abstract

Maintenance costs constitute a significant portion of the total costs for offshore wind investments. Consequently, a substantial amount of research aims to mitigate these costs. Studies targeting short-term decision-making primarily concentrate on finding the most cost-effective routes for the maintenance vessels while scheduling as many maintenance tasks as possible. This thesis suggests an alternative approach where all maintenance tasks are considered optional. Instead of minimizing costs, the optimization model we propose maximizes expected profit. The motivation is to establish a more dynamic relationship between short-term decision-making and long-term strategy. We formulate the problem of selecting routes for maintenance vessels as an integer linear program. Further, we use a mixed integer linear programming sub-problem to generate routes via a column generation algorithm. We have developed several instances for testing the model, which we make available for subsequent research. Our proposed model provides optimal solutions for some of the problem instances where the sub-problem can be solved with exact methods. We also present a meta-heuristic for the sub-problem, capable of finding good solutions to problem instances considering up to 60 maintenance tasks. Lastly, we find that the column generation method outperforms a more straightforward solution algorithm.

Sammendrag

Vedlikeholdskostnader utgjør en betydelig del av de totale kostnadene forbundet med havvind. Mye forskning ser derfor på mulighetene for å redusere disse kostnadene. Studiene som er gjort på kortsiktig beslutningstaking fokuserer hovedsakelig på å finne kostnadseffektive ruter for vedlikeholdsfartøyene, samtidig som man vil få utført mest mulig vedlikehold. Denne masteroppgaven angriper problemet fra en annen vinkel, da den ikke anser noe vedlikehold som obligatorisk. I stedet for å minimere kostnader, maksimerer optimeringsmodellen vi foreslår forventet profitt. Motivasjonen for denne tilnærmingen er å knytte sammen kortsiktig beslutningstaking og langsiktig strategi. Vi formulerer problemet med å velge ruter som et lineært heltallsprogram. Videre bruker vi et blandet lineært heltallsprogram som delproblem for å generere ruter i en søylegenereringsalgoritme. Modellen testes på probleminstanser vi har laget selv, som vi tilgjengeliggjør også for videre arbeid. Den foreslåtte modellen finner optimale løsninger til noen av instansene der delproblemet kan løses eksakt. Vi presenterer også en metaheuristikk for delproblemet som gjør det mulig å finne gode løsninger på instanser med opptil 60 vedlikeholdsoppgaver. Til slutt finner vi ut at søylegenereringsalgoritmen utkonkurrerer en enklere løsningsmetode.

Acknowledgments

I am proud to successfully complete my Master's thesis. It has been a rewarding journey where I have gained valuable knowledge and a deep sense of accomplishment. Reflecting on this experience, I want to acknowledge the contributions of several individuals who have greatly influenced the outcome of this thesis.

First and foremost, I would like to thank my supervisor, Dag Haugland, for excellent guidance throughout the process of composing this thesis. I appreciate your feedback and advice and am grateful for our frequent meetings and discussions.

I would also like to thank Elin Espeland Halvorsen-Weare from SINTEF Ocean. Your insight from the offshore industries from an academic perspective has been essential in shaping and validating the model. Thank you for dedicating time to our regular meetings and always being available to answer my questions.

Finally, thank you to my fellow students in the energy program. You have been an invaluable source of motivation and joy. Thanks to Gunnar, Runar, Erlend, Fredrik, and Benjamin from the optimization group for constructive discussions at the study hall.

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

The global population increases and the growth is expected to continue. Consequently, the demand for energy has never been greater. The war in Ukraine and the following energy crisis in Europe have highlighted the urgent need for diversified energy sources and the importance of more electricity production.

Envisioning a sustainable future, it is essential to take advantage of clean and renewable energy resources. This is established in the Paris Agreement [3], which aims to combat climate change by promoting a transition to renewable energy sources. Moreover, according to the United Nations' Sustainable development goal 7 [4], this energy must be clean and affordable, ensuring that all people have access to sustainable energy. Offshore wind energy is expected to play a significant role in reaching this goal, promising to satisfy a significant part of the growing demand for electricity while keeping the environmental impact at an acceptable level.

The rush to harness wind power has led to an impressive expansion of offshore wind farms (OWFs), especially in North-western Europe. Many of these projects are heavily supported by government funding or innovation funds. However, to accelerate the expansion, it is crucial to reduce costs and increase revenues so that offshore wind becomes more attractive for investors that require profits.

There are many costs associated with an offshore wind investment, and we believe there is room for reductions in many of them. According to National Renewable Energy Laboratory (NREL) [1], operation and maintenance (O&M) costs constitute more than a third of the total cost when calculating the levelized cost of energy for a bottom-fixed OWF in 2019. Figure 1 shows that this makes it the greatest expense in NREL's categorization.

This thesis present an optimization model that aims to optimize short-term maintenance scheduling for offshore wind. The literature already proposes several methods to handle different O&M cost aspects. We come back to some of these studies shortly, but first we provide some background for the specific part of the O&M this thesis aim to optimize.

1.1 Maintenance strategies

Maintenance strategies for OWFs are divided into *proactive* and *corrective* maintenance. The difference between the two is that proactive strategies aim to prevent failures from happening, while corrective strategies are waiting for failures to repair.

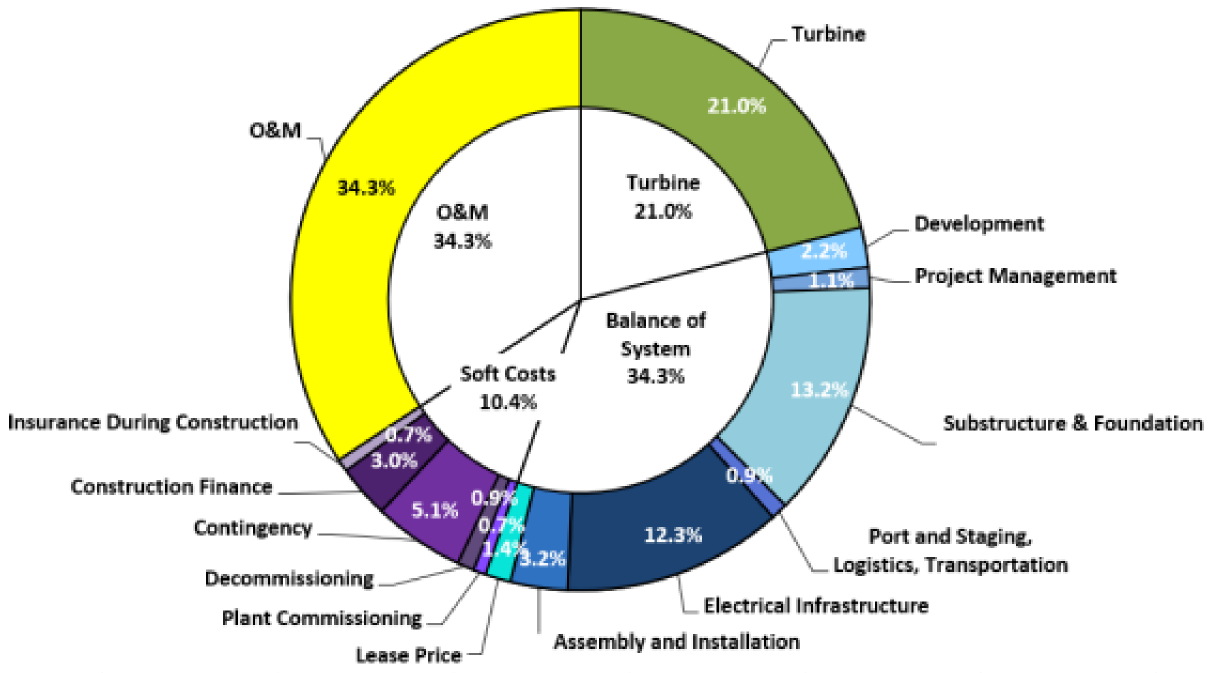


Figure 1: Fixed-bottom offshore wind farm costs for a reference farm in 2019 (reprinted with permission from NREL [1])

The advantage associated with corrective maintenance strategies is reduced maintenance frequency which may lead to reduced maintenance costs. Ren et al. [5] say that this type of strategy can be relevant for smaller wind farms as downtime losses are typically low compared with maintenance costs. However, we know that the number of large-scale offshore wind farms is rapidly increasing, and Karyotakis and Bucknall [6] state that corrective maintenance is impractical for these wind farms. They argue that large-scale wind farms have high failure rates and combined with challenging and unpredictable weather conditions, a corrective maintenance strategy may result in long downtime periods and thus great loss of revenue.

Proactive maintenance strategies are further divided into *preventive*, *condition-based*, and *predictive* maintenance. Preventive strategies fix the frequency of maintenance operations based on experienced component lifetimes. On the other hand, condition-based and predictive maintenance strategies rely on sensors that give information about the current turbine condition [5]. The difference between condition-based and predictive is how the inputs are processed. Condition-based strategies act directly on this input. For instance, if the temperature monitored by a sensor reaches a certain threshold, a corresponding maintenance operation is scheduled immediately. The predictive strategies are more advanced as they are combining multiple sensor inputs to perform a more complex analysis of the turbine condition and further determine the most appropriate time to perform maintenance. The most advanced predictive strategies use digital-twin technology to create detailed condition reports [7].

In addition to corrective and proactive maintenance, *opportunistic* maintenance is also discussed in the literature. Opportunistic maintenance is not consensually defined, but it is characterized as a combination of corrective and proactive maintenance [8]. For instance, if a failure occurs and a technician team has to visit a turbine to fix it, they may take the opportunity to also perform other maintenance tasks on the turbine.

1.2 Maintenance tasks and vessels

The turbines in an OWF necessitates various types of maintenance tasks throughout its lifespan. Together they ensure efficient functioning, maximize energy output, and extend the life of the equipment. Regular inspections, cleaning, and replacement of worn or damaged parts are essential to prevent system failures and downtime. Some of these maintenance tasks, such as replacement of a turbine blade or a gearbox, require specialized vessels and may take several workdays to complete. Due to their complexity, such operations are typically treated as separate projects, i.e, they are not coordinated with simpler tasks.

On the other hand, a significant part of the maintenance tasks, such as replacement of smaller parts, cleaning and inspection, require less personnel, less heavy equipment, and no cranes. These kind of *light maintenance tasks* can be accomplished through the use of *crew transfer vessels (CTVs)* or helicopters. There are also other vessel types that can be used for this purpose, but these are the two most relevant for the problem considered in this thesis. For simplicity we refer to both of them as *vessels*. Both of these vessel types' main function is to transfer *technicians* from an *O&M base* to one or several turbines. A technician in this context, is a worker who has a specific skill, among a set of skills, that might be required to perform a maintenance task. An O&M base is a building, typically at a harbour close to a set of OWFs, that houses a set of vessels and necessary parts and equipment for OWF maintenance.

A CTV (illustrated in Figure 2) is typically 20-30 meters long and able to transfer 12-24 technicians. It has the ability to disembark crews of technicians at offshore wind turbines over the bow, but it is only allowed to do so if the wave heights are under a given safety threshold. A CTV's service speed is typically between 15 and 25 knots, and it can normally travel in up to 12 hours per day if the weather conditions allow it.

A helicopter in this context, is able to carry 6-12 technicians, and it disembarks them at the turbines using a hoist system. With a service speed of 100-150 knots, it is much faster than a CTV, but it has less work hours per day due to fuel limitations.

Figure 2: A crew transfer vessel disembarking technicians at an offshore wind turbine (photo: The Workboat Association [2]).

1.3 Problem statement

The wind farm operator follows a maintenance strategy that determines how often maintenance should be performed and what the thresholds are for a maintenance task to be carried out. This can be seen as the overall plan, typically with rough time resolution, and it does not specify exactly when and how the maintenance tasks should be performed. Based on this strategy, we assume that the wind farm operator at all times can relate to a set of *available maintenance tasks*. We define available maintenance tasks as light maintenance tasks that the operator is interested in completing within a planning horizon of up to two weeks. For a set of available vessels at an O&M base, this thesis addresses the problem of scheduling the available maintenance tasks. By scheduling we mean:

- Select which tasks, among the available tasks, to perform within the planning horizon.
- Decide what day every selected maintenance task shall be carried out.
- Determine which vessel to use for each task.
- Decide the *routes* for the vessels, i.e., the order to visit the turbines and whether or not they should be on standby while technicians perform maintenance.

To optimize the schedule, all these points must be taken into account simultaneously in one optimization problem.

1.4 Short term routing and scheduling models for offshore wind maintenance

Dai, Stålhane and Utne [9] introduce The routing and scheduling problem of a maintenance fleet for offshore wind farms (RSPMFOWF). They consider a vessel fleet of CTVs and helicopters, but also larger vessels with cranes as the RSPMFOWF also accounts for more complex maintenance. The objective of the problem is to minimize costs, where a penalty cost occurs for maintenance that is not scheduled within the desired time window.

A similar problem is addressed by Irawan et al. [10]. They focus on maintenance conducted by transferring technicians by CTVs. They present a more detailed model than [9], where they take available technicians into account, and whether or not they have the correct skills for completing the considered maintenance tasks. The model can be used with multiple O&M bases that service multiple OWFs, and considers a planning horizon of 3 to 7 days.

Stock-Williams and Swamy [11] also propose a detailed model that schedules less complex maintenance, such as minor failure repairs and inspections. The model provides a very short term schedule and is meant to be frequently resolved if any input changes. Their motivation is to provide practical solutions to very detailed situations, for instance if any of the technicians involved gets too sea-seek to work.

Generally in the literature, minimizing costs is a popular option for the objective functions in studies concerning O&M optimization [12, 13, 14, 15]. An obvious reason for this is the economical perspective from owners and investors, but it is also reasonable to believe that minimizing costs indirectly impacts the climate footprint. For instance, reduced fuel consumption for maintenance vessels reduces both costs and emissions. An alternative approach is presented by Yildirim et al. [16] and Zhong et al. [17]. They propose multi-objective optimization models where both costs and reliability are given influence.

Routing and scheduling models often involve *mixed integer linear programs (MILPs)* that can be too hard to solve in reasonable time for commercial solvers. Consequently, a crucial aspect of research in this field involves exploring alternative approaches to find satisfactory solutions. Raknes et al.[12] present a rolling horizon heuristic to solve their proposed routing problem of a joint vessel fleet for multiple OWFs. They discover that dividing the scheduling period into smaller segments and optimizing them individually produces positive outcomes when tested in simulations.

Another possibility to deal with the problem's complexity is to use meta-heuristics. For instance, [11] shows how genetic algorithms can be applied to select transfer plans for technicians among a very large set of alternatives. By utilizing these algorithms, they effectively find transfer plans that provide good simulation results.

1.5 Thesis structure

The rest of the thesis is organized as follows: Chapter 2 gives a detailed definition of the problem we address. In Chapter 3, we briefly introduce mathematical programming and solution methods, focusing on the theory behind the techniques we utilize in this thesis. In Chapter 4, we formulate the considered problem and an exact solution approach mathematically. Further, we propose several heuristic methods that can complement or replace parts of the exact approach in Chapter 5. In Chapter 6, we explain how we build a set of problem instances, which we utilize in experiments on the proposed solution methods in Chapter 7. These experiments aim to verify our solution method's effectiveness and identify trends in good solutions. Finally, Chapter 8 provides some concluding remarks.

2 Problem definition

In this section, we provide the definition of the problem we solve in this thesis, what we consider and not, the assumptions we make, and how we differ from other works on the field.

2.1 An alternative approach

To force the models reviewed in section 1.4 that minimize costs to complete maintenance tasks, they are either constrained to complete all considered maintenance tasks, or they have penalty costs for uncompleted maintenance. We suggest an alternative approach.

Indirectly, an OWF operator profits on maintenance. If a turbine is maintained correctly, it is expected to last longer and to be less exposed to shut-downs. The underlying motivation for this is to produce more electricity that in its turn leads to increased revenues. In other words, maintenance is an investment: The operator pays a price for maintenance and expects that it is profitable, at least if we consider the average of all maintenance tasks performed throughout the lifetime of an OWF.

We believe that it is possible to estimate the expected *revenue* for performing an isolated maintenance task. Exactly how to do this is out of the scope of this thesis, but in the model we propose, we assume that this revenue is known for every available maintenance task. However, some of the works reported in the literature which propose penalty costs for uncompleted maintenance (e.g. [10]) make similar assumptions. They set the penalty cost for a maintenance task based on how important the OWF operator thinks it is to complete it within the considered planning horizon. So, even if our assumption is wrong, we still think our model is useful if we replace revenue with a measure similar to the one used in these penalty costs.

Therefore, the model we present in this thesis maximizes profits instead of minimizing costs. We believe this may contribute to a more dynamic short-term routing and scheduling model that can be more directly connected to long-term planning.

2.2 Inputs and outputs

We structure the inputs to our model by four sets: Turbines, vessels, periods, and technician types. Figure 3 summarizes the relationships between the model parameters and these sets. It is important to note that the yellow boxes represent the actual model inputs, whereas the grey boxes suggest how to determine them.

We formally introduce all parameters in Chapter 4.

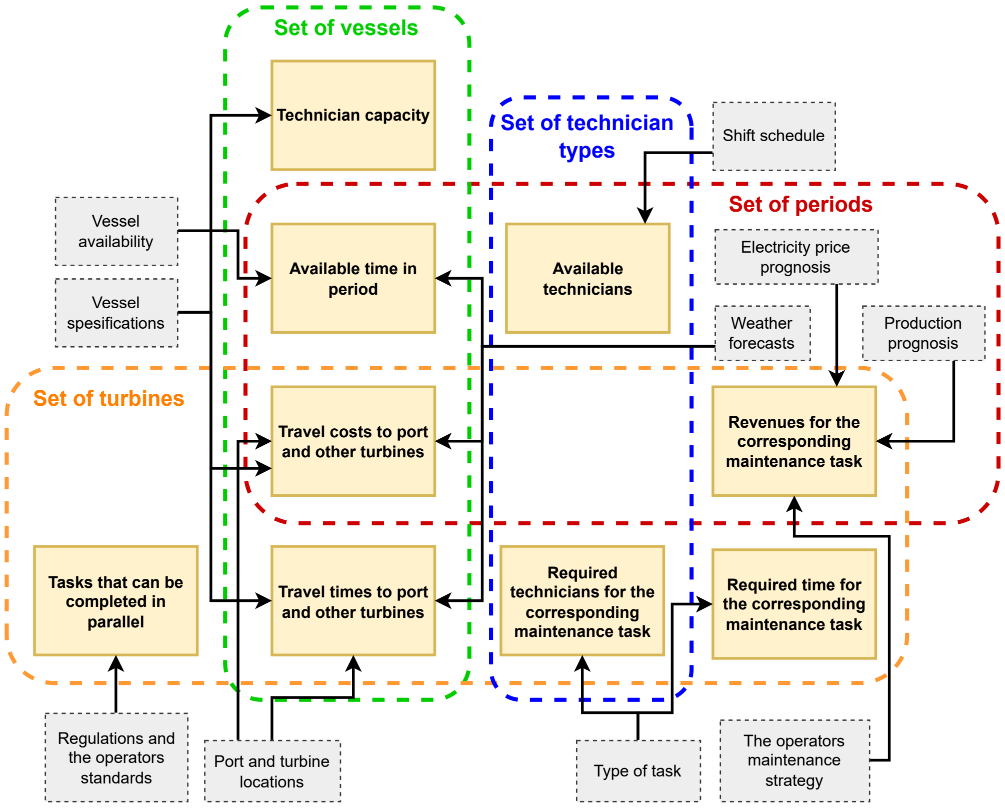

Set of vessels
Technician capacity
Set of technician types
Shift schedule
Vessel availability
Set of periods
Available time in period
Available technicians
Electricity price prognosis
Vessel spesifications
Weather forecasts
Production prognosis
Set of turbines
Travel costs to port and other turbines
Revenues for the corresponding maintenance task
Tasks that can be completed in parallel
Travel times to port and other turbines
Required technicians for the corresponding maintenance task
Required time for the corresponding maintenance task
Regulations and the operators standards
Port and turbine locations
Type of task
The operators maintenance strategy

Figure 3: A comprehensive summary of the model inputs, along with their corresponding index sets, i.e, an input (yellow box) is indexed by all sets (dotted round boxes) that it is drawn inside.

The model we propose considers exactly one O&M base and a set of offshore wind turbines, where each turbine corresponds to exactly one maintenance task. The turbines may belong to one or several OWFs.

The set of periods the model relates to partitions the time in the planning horizon. Each partition typically corresponds to a 12-hour work day.

As all maintenance tasks are considered optional, a possible solution may be not to complete any of them. The model incorporates the required number of technicians of each type and the duration necessary to fulfill a maintenance task. As discussed, every maintenance task is associated with a revenue that reflects how economically favorable it is to complete them in a given period.

The vessel that transfers technicians to a turbine may deliver other technicians at other turbines while the maintenance are conducted, i.e., multiple maintenance tasks can be completed *in parallel*. However, for safety reasons, there is a restriction on how far a vessel can travel away from disembarked technicians. From this limit, we can extract subsets of the maintenance tasks that can be completed in parallel that are given as input to the model.

The vessel fleet is considered prepaid, which means there are no fixed costs. However, the vessels incur operating costs depending on factors such as the type of vessel, distance traveled, and the weather conditions in which the vessel operates. The vessels are also associated with an available time in each period. This availability can vary between vessels and across periods, influenced by various factors. Furthermore, each vessel has a fixed number of technicians it can transport at once and a constant service speed.

The vessels' travel costs are comparable to the expected revenues of completing the maintenance tasks. Suppose the operating cost of a vessel during a period is lower than the total value of the maintenance tasks it is used for. In that case, using the vessel to complete those tasks in that period is economically viable. We also assume that the operator has no incentive to leave available technicians idle.

The goal is to find the most profitable schedule for the planning horizon, i.e., to maximize the total expected revenue from performed maintenance tasks subtracted by the vessels' travel costs. The output is a route for each vessel for each period in the planning horizon.

3 Background

This chapter briefly introduces mathematical programming and the solution methods we utilize in Chapter 4 and 5. The notations from this chapter are used for demonstrating purposes and are independent of the rest of the thesis.

3.1 Mathematical programming

Mathematical programming or mathematical optimization is a framework used to find the best solution to a problem P among a set of alternatives. Such problems can be generalized as

$$Z_P = \max\{f(x) : x \in X\},$$

where x is the variable vector used to describe a solution, f is the *objective function* that measures the quality of a solution, and X is the set of all the solution alternatives. All elements in X are defined as *feasible solutions* to P, that is, they satisfy all the constraints defined by the nature of the problem. If an element $x^* \in X$ satisfies the inequalities $f(x^*) \geq f(x),\ \forall\ x \in X$, it is defined as an *optimal solution* to P.

Optimization problems are further divided into problem categories based on the properties of the function f and the set X. The most important is whether or not the function is linear and whether or not the set X is convex and/or continuous. The two most relevant categories for this thesis are *linear programs* and *integer linear programs.*

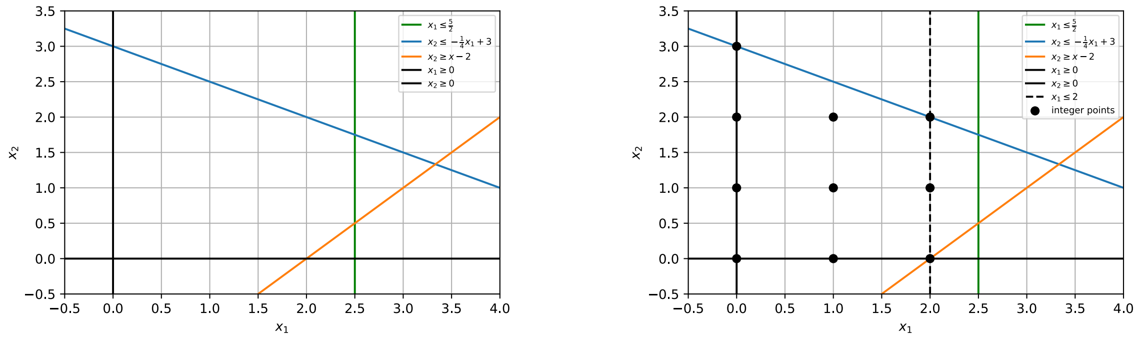
3.1.1 Linear programs

In a linear program (LP), the objective function and the constraints are linear. We can therefore express the objective function $f(x)$ as the scalar product $c'^{\top}x'$, where c' is the coefficient vector. The constraints can be expressed as linear inequalities. We combine the left-hand side of all these inequalities to $A'x'$ and the right-hand side to b'. The LP can then be fully expressed as

$$Z_{LP} = \max\{c'^{\top}x' : A'x' \leq b'\}.$$

However, it is common practice to modify A', b', c', and x' so that we get an equivalent problem where all entries in x only can take non-negative values. This procedure is well described by Vaderbei [18], and the result is the characteristic formulation of an LP in *standard form*,

$$Z_{LP} = \max\{c^{\top}x : Ax \leq b,\ x \in \mathbb{R}^n_+\}.$$



(a) The feasible set of an LP

(b) The feasible set of an ILP. Note that the convex hull is fully described when $x_1 \leq 2$ is added.

Figure 4: A graphical example of an LP (a), and how it can be used to solve an ILP (b) with identical constraints $Ax \leq b$

In a linear program in standard form, it is easy to see that the solution space is delimited by the coordinate axis and the linear inequalities $Ax \leq b$. This is illustrated in Figure 4a. A result of this is that an optimal solution, if one exists, can be found in the intersection of n hyperplanes, i.e., in a *vertex*.

One of the key advantages of LPs is that there exist good algorithms to solve them, e.g., the simplex method. The simplex method was invented in 1947 by George Dantzig, who is known as the father of linear programming [18, 19]. The algorithm is theoretically speaking not *efficient* as it has an exponential worst-case running time, but in many cases, it works well in practice. However, there also exist algorithms that guarantee to solve LPs in polynomial time, such as Karmarkar's algorithm [20].

3.1.2 Integer linear programs

In many optimization problems, we can formulate all the constraints as linear inequalities, but the nature of the problem only allows the entries in x to be binary or integral. Even though an integer linear program (ILP) is expressed very similarly to an LP,

$$Z_{ILP} = \max\{c^\top x : Ax \leq b,\ x \in \mathbb{Z}^n_+\},$$

the integer constraints make ILPs much harder to solve. No known algorithm can guarantee to solve a general ILP in polynomial time, which is classified as NP-hard. However, if such an algorithm exists, which is considered highly unlikely, it would also be able to solve all problems in NP in polynomial time.

Many of the known algorithms to solve ILPs are based on the *linear programming relaxation* (LPR) of the problem. The LPR of an ILP, is identical to the ILP, except that $x \in \mathbb{Z}_+^n$ is substituted by $x \in \mathbb{R}_+^n$. Then, the goal is to add new constraints, that describe the smallest convex set including all $x \in \mathbb{Z}_+^n : Ax \leq b$, i.e. *the convex hull*. In most cases, it is neither realistic nor necessary to fully describe the convex hull. It is enough to describe the convex hull in the neighborhood of an optimal solution. In Figure 4b, we show how an optimal solution of an ILP can be forced into a vertex by adding an extra constraint.

3.1.3 The vehicle routing problem

The vehicle routing problem (VRP) is an example of an ILP, or in some variants, a mixed integer linear program, i.e., an ILP where not all variables are constrained to be integers. The problem was first introduced by Dantzig and Ramser [21] in 1959. They consider the situation where a set of trucks transports gasoline from a bulk terminal to a set of service stations. The objective is to determine the shortest combination of routes for the trucks that meet the gasoline demands from the service stations. They proposes multiple variations of the problems that take different constraints into consideration.

In general, the vehicles in VRP can represent any modes of transportation that have an insensitive for, or are constrained to, visit several locations and then return to the starting point. For instance, maintenance vessels that transfer technicians to offshore wind turbines. Depending on the problem's nature, several constraints may apply.

3.2 Column generation

Column generation is a method used to solve large optimization problems. It was first proposed by Ford and Fulkerson in 1958 [22]. Although they do not use the term column generation, they suggest using it as a technique for solving the multi-commodity network flow problem. Since then, it is used in a wide specter of applications within the field of optimization. For instance, Choi and Tsca [23] use it to solve The vehicle routing problem with a heterogeneous fleet (HVRP), and Dumas et al. [24] use it in their algorithm to solve The pickup and delivery problem with time window (PDPTW). Both HVRP and PDPTW are special cases of VRP, and they are relatable to the considered problem in this thesis.

The literature proposes many different ways to explain column generation. For simplicity, we use the same terms as Wolsey [25], but we avoid the notation caused by the Dantzig-Wolfe reformulation.

We define *the master problem (M)* as the ILP

$$Z_M = \max\{c^\top x : Ax \leq b,\ x \in \mathbb{Z}_+^n\},$$

and *the relaxed problem (LM)* as the LP

$$Z_{LM} = \max\{c^\top x : Ax \leq b,\ x \in \mathbb{R}_+^n\}.$$

Consider the situation where the dimension n is so large that the problem of finding all entries in A and c is itself unsolvable for available algorithms. The first trick in the column generation algorithm is to define *the restricted relaxation of the master problem (RLM)* as

$$Z_{RLM} = \max\{g^\top \lambda : K\lambda \leq b,\ \lambda \in \mathbb{R}_+^p\},$$

where λ and g are the first p entries of x and c, respectively, and K is the sub-matrix of A consisting of all rows, but only the p first columns. We assume that it is relatively easy to find all entries in the RLM, and that it is sufficiently fast to solve. We denote a primal optimal solution to the RLM λ^*, and a dual optimal solution π^*.

In the next step in the column generation algorithm, we search for a column in A that is not in K, and its corresponding entry in c, which can improve the optimal solution of the RLM if we add it. To do that, we take advantage of the RLM's dual problem. The dual RLM is expressed as

$$W_{RLM} = \max\{b^\top \pi : K^\top \pi \geq g,\ \pi \in \mathbb{R}_+^q\}, \tag{1}$$

where π is the dual variable vector, and q is the number of rows in the matrix K. Instead of looking for a column in A directly, we try to construct a new constraint in the dual RLM which, if added, makes the current dual solution π^* infeasible. Let $A_{(:,i)}$ denote column i of A, and c_i denote the i'th entry in c. We want to find the i that violates

$$A_{(:,i)}^\top \pi^* \geq c_i$$

the most. Or equivalently, we can reformulate it as the optimization problem

$$W_{SP} = \max\{c_i - A_{(:,i)}^\top \pi^* : i \in \{1, 2, ..., n\}\}.$$

We define this as the *sub-problem* in the column generation algorithm. The constraint is written this way to maintain generality. However, depending on the problem, this can be derived into multiple, more manageable constraints. Intuitively, we can think of them as rules that $A_{(:,i)}$ and c_i follow, if and only if they are elements in A and c, respectively.

We solve W_{SP} and denote an optimal solution i^*. If $c_{i^*} - A_{(:,i^*)}^\top \pi^* \leq 0$, we know that the current optimal solution to the RLM, λ^*, is also an optimal solution to the LM. This is the stopping condition of the column generation algorithm. On the other hand, if $c_{i^*} - A_{(:,i^*)}^\top \pi^* > 0$, we know that the optimal solution of the RLM may be improved if we add the column $A_{(:,i^*)}$ and the entry c_{i^*} to K and g, respectively. We do that and repeat the steps after solving the RLM until the stop condition is met.

When the stop condition is met, we obtain the optimal solution x^* of the LM. However, there is no guarantee that x^* is feasible for M, e.g. it might have some fractional entries. Therefore, we solve the restricted master problem (RM),

$$Z_{RM} = \max\{\hat{c}^\top \hat{x} : \hat{A}x \leq b,\ \hat{x} \in \mathbb{Z}_+^r\},$$

where $\hat{c}$, $\hat{x}$, $\hat{A}$ and r, represent g, λ, K and p, respectively, from the final RLM, i.e., the version of the RLM that led to the stop condition. If the optimal objective function value Z_{RM}^* of the RM is equal to the optimal value of the final RLM, we know that an optimal solution to the RM is also an optimal solution to M. Then we are done. However, if this is not the case, an optimal solution to RM might be a good enough solution to M in practice.

3.3 Heuristic methods

Some optimization problems turn out to be too complex to solve with exact solution algorithms in a reasonable time. The VRP and the routing part of the problem considered in this thesis are examples of such problems, at least for larger problem instances. A typical way to deal with these kinds of problems is to apply *heuristic methods*.

Heuristic methods are algorithms that take advantage of practical rules or shortcuts to find solutions. The solutions may not be optimal, but in many practical cases, good enough solutions might be satisfactory. Even if an optimal solution is found using a heuristic method, it is typically not possible to prove that it is optimal.

A popular choice of heuristic method for solving VRPs is the *Adaptive large neighborhood search (ALNS)* in combination with the *Simulated annealing (SA)* local search algorithm. We believe it is a good idea to take advantage of this method to solve the problem considered in this thesis as well.

3.3.1 Solution neighborhood

In a heuristic method, a *solution neighborhood* refers to the set of candidate solutions that can be generated by applying a certain modification to a given solution [26, 27]. The idea is that it wraps solutions that share some properties, which turn out to be beneficial in iterative heuristic methods.

For instance, for a set of nodes $\mathcal{N} = \{n_1, n_2, n_3, n_4\}$, let the vector $x_1 = [n_1, n_2, n_3, n_4]$ describe that a vehicle in an instance of VRP visits the nodes in the order specified by the vector. A possible modification is to interchange two nodes' positions in the vector, e.g., n_1 and n_3 so that we obtain the new solution vector $x_2 = [n_3, n_2, n_1, n_4]$. The vector x_2, together with all other combinations that are possible to make by only interchanging two nodes, constitute a neighborhood of x_1.

The interchange method from the example is just one possible way to modify a solution. Multiple terms are used in the literature for a general solution modification method. From now, we consequently refer to such methods as *operators*. A variety of operators are used to access different neighborhoods. These operators can have different properties that serve different purposes. One of these properties is how much they can modify a solution. For instance, one operator can interchange two nodes as in the example above, while another might be able to interchange five. More changes mean that the operator can reach a larger neighborhood of solutions. However, larger is not always better. This problem is addressed by the ALNS, which we return to shortly.

Formally, for an operator o and a random component γ, the neighborhood $\mathcal{H}_{ox}$ of the solution x is defined as the set of all possible outputs $o(x, \gamma)$. The random component enables the operator to return different output solutions in calls with identical values of x.

In some cases, it is appropriate to split the neighborhood concept into a *destroy neighborhood* and a *repair neighborhood*. In this two-step process, a distinction is made between *destroy operators* and *repair operators*. A destroy operator removes parts of a solution or alternatively, we can imagine that it replaces parts of the solution with temporary voids. The set $\mathcal{H}^-_{dx}$ of all the incomplete solutions that can be accessed by calling the destroy operator d on the solution x, is a destroy neighborhood of x. On the other side, a repair operator fills the voids in an incomplete solution $x^- \in \mathcal{H}^-_{dx}$. All possible completions of an incomplete solution x^- that can be accessed by calling a repair operator r constitute a repair neighborhood $\mathcal{H}^+_{rx^-}$. A destroy operator d and a repair operator r can be combined into an operator

$$o(x, \gamma_1, \gamma_2) = r(d(x, \gamma_1), \gamma_2),$$

where the γ_1 and γ_2 are independent random components. This means that for a set of destroy operators $\mathcal{O}^-$ and a set of repair operators $\mathcal{O}^+$, it may be possible to build a set $\mathcal{O}$ of $|\mathcal{O}^- \times \mathcal{O}^+|$ unique operators. However, all pairs (d, r) in $\mathcal{O}^- \times \mathcal{O}^+$ are not necessarily compatible, but the idea is that this split concept of neighborhood facilitates a more dynamic framework.

3.3.2 Simulated annealing

Simulated annealing is an iterative local search algorithm that was first purposed by Kirkpatrick et al. [28] in 1983. The algorithm starts with a feasible solution that can be manually given as input or generated by some construction algorithm. This solution is set as the *current* solution x. Further, the SA-algorithm explores a new feasible solution x' in the neighborhood of x. If f is the objective function, the considered problem is a minimization problem, and $f(x') < f(x)$, x' is accepted, i.e., the current solution is set to x'. Alternatively, if $f(x') \geq f(x)$, x' may still be accepted with a certain probability p. For a *temperature* $T > 0$, this probability typically follows the function

$$p(x, x', T) = e^{-\frac{f(x') - f(x)}{T}}.$$

Note that high temperature and a small difference $f(x') - f(x)$ give a probability close to 1, and that the opposite combination gives a probability close to 0. For each of a given number I_{disp} of iterations, a new neighbor is accepted or declined based on these criteria. In the end, the algorithm returns the best solution observed.

The initial temperature T_0, i.e., the temperature at the first SA-iteration, should be tuned so that the algorithm explores a wide range of solutions in the beginning, but not so wide that it just wastes time and computations on unpromising parts of the solution space. A common way to tune this parameter is to fix the acceptance probability to 0.5 for the first $I_{warmup} < I_{disp}$ iterations and track the deviations $f(x') - f(x)$ for the accepted solutions during these iterations. When the I_{warmup} iterations are completed, the temperature can be initialized as the mean of these deviations. When the initial temperature is set, the temperature is reduced in every iteration in a way that it gets close to zero at the last of the available iterations.

3.3.3 Adaptive large neighborhood search

ALNS is a meta-heuristic framework, i.e, a general approximation algorithm that can be applied to a variety of optimization problems [26]. The ALNS is embedded in a local search framework. In the founding paper of ALNS, Ropke and Pisinger [29] use Simulated annealing.

The main idea in ALNS is to select operators among a set of predefined operators based on their performance during the search [30]. This is done by assigning a weight w_o to each operator $o \in \mathcal{O}$. In every search iteration, the probability of selecting operator o is set to

$$p_o = \frac{w_o}{\sum_{o \in \mathcal{O}} w_o}, \qquad \forall\ o \in \mathcal{O}. \tag{2}$$

This weight is updated every I_{update}-th iteration based on how much the operators contribute. The contribution is measured by a scoring system where every operator o is assigned a score s_o. For a set $\mathcal{C}$ of criteria, this score is increased by a constant Q_c if a solution generated by operator o satisfies criterion $c \in \mathcal{C}$. These criteria may be defined in different ways, but [29] proposes a scoring system where the scores are increased by

Q_1 if the operator generates a new best solution,

Q_2 if the operator generates a previously unseen solution that improves the current solution,

Q_3 if the operator generates a previously unseen solution that is accepted.

Let α_o denote the number of times operator o was called during the last I_{update} iterations. Every I_{update}-th iteration, an updated weight w'_o is calculated by

$$w'_o = (1 - \beta)w_o + \beta\frac{s_o}{\alpha_o}, \tag{3}$$

where $\beta \in [0, 1)$ is a parameter that determines how much the score influences the weight, i.e., how fast the weight can change. After the weight update $w_o \leftarrow w'_o$, all scores are reset to zero. These score system steps, together with the Simulated annealing embedding, are summarized in Algorithm 1.

Algorithm 1 ALNS with Simulated annealing

```
$x \leftarrow$ a constructed feasible solution
$x_{best} \leftarrow x$
$w_o \leftarrow 1,\ s_o \leftarrow 0, \qquad \forall\ o \in \mathcal{O}$
$i \leftarrow 1$
while $i \leq I_{disp}$ do
    if $i$ is divisible by $I_{update}$ then
        update $w_o$ based on (3)
        $s_o \leftarrow 0, \qquad \forall\ o \in \mathcal{O}$
    end if
    select an $o \in \mathcal{O}$ based on probabilities from (2)
    $x' \leftarrow o(x)$
    if $x'$ is accepted according to the Simulated annealing acceptance criteria then
        $x \leftarrow x'$
        if $f(x') < f(x_{best})$ then
            $x_{best} \leftarrow x'$
        end if
    end if
    increment $s_o$ with respect to the scoring system, $\qquad \forall\ o \in \mathcal{O}$
    update the Simulated annealing acceptance criteria
    $i \leftarrow i + 1$
end while
return $x_{best}$
```

4 Mathematical model

In this chapter, we present the mathematical model we use to solve the addressed problem. The section is structured into three parts. Firstly (4.1), we outline the procedure for selecting *routes* from a collection of available options. Secondly (4.2), we describe the process of generating new routes to augment the collection. Finally (4.3), we demonstrate how we combine and resolve these two issues through the application of a column generation algorithm.

The notations we introduce in this chapter are used throughout the thesis. A comprehensive summary of all symbols used in the model can be found in Appendix A.

4.1 Master problem - Route selection

For a given set $\mathcal{V}$ of vessels and a set $\mathcal{T}$ of time periods, we define a set $\mathcal{R}^{ALL}$ that contains all *feasible routes*. We also define an arbitrarily large subset $\mathcal{R} \subseteq \mathcal{R}^{ALL}$. Further, we partition R into the sets R_{vt} of routes that vessel $v \in \mathcal{V}$ can travel in period $t \in \mathcal{T}$. To clarify,

$$\mathcal{R}_{vt} \subseteq \mathcal{R} \subseteq \mathcal{R}^{ALL}, \qquad \forall\, v \in \mathcal{V},\ t \in \mathcal{T},$$

$$\mathcal{R} = \bigcup_{v \in \mathcal{V},\ t \in \mathcal{T}} \mathcal{R}_{vt},$$

and

$$\mathcal{R}_{vt} \cap \mathcal{R}_{v't'} = \emptyset, \qquad \forall\, (v,t),\ (v',t') \in \mathcal{V} \times \mathcal{T},\ (v,t) \neq (v',t').$$

Each route r is associated with a profit P_r that is realized if the route is *selected*. Let us introduce a binary variable x_r, which decides whether route r is selected or not. Our goal is to select the most profitable combination of routes and this is expressed in the objective function

$$\max \sum_{v \in \mathcal{V}} \sum_{t \in \mathcal{T}} \sum_{r \in \mathcal{R}_{vt}} P_r x_r \tag{4}$$

of the master problem.

A vessel can, at most, complete one route per time period. In other words, only one route can be selected per vessel-period combination. This is taken into account by the inequality

$$\sum_{r \in \mathcal{R}_{vt}} x_r \leq 1, \qquad \forall\, v \in \mathcal{V},\ t \in \mathcal{T}. \tag{5}$$

Further, we have to make sure that the combination of selected routes does not require more technicians than what is available at a specific time period. Given a set of technician types $\mathcal{B}$, we define D^{T}_{br} as the demand of technicians of type b for completing route r, and G_{bt} as the number of available technicians of type $b \in \mathcal{B}$ at time period t. Then, the inequality

$$\sum_{v \in \mathcal{V}} \sum_{r \in \mathcal{R}_{vt}} D_{br}^T x_r \leq G_{bt}, \qquad \forall\, b \in \mathcal{B},\ t \in \mathcal{T}, \tag{6}$$

describes the technician limitation.

A maintenance task can, at most, be performed once. We assume that every considered turbine requires exactly one maintenance operation. For a given set of turbines $\mathcal{W}$, we define the parameter

$$I_{wr} = \begin{cases} 1, & \text{if the maintenance task at turbine } w \text{ is completed by route } r \\ 0, & \text{otherwise,} \end{cases}$$

for all $w \in \mathcal{W}$ and all $r \in \mathcal{R}_{vt}$. We can then prevent the possibility of selecting a combination of routes that perform the same maintenance operation more than ones by the inequality

$$\sum_{v \in \mathcal{V}} \sum_{t \in \mathcal{T}} \sum_{r \in \mathcal{R}_{vt}} I_{wr} x_r \leq 1, \qquad \forall\, w \in W. \tag{7}$$

We connect the expressions from this section and formulate our *master problem* as the integer program

$$\begin{aligned} \max \quad & \sum_{v \in \mathcal{V}} \sum_{t \in \mathcal{T}} \sum_{r \in \mathcal{R}_{vt}} P_r x_r \\ \text{s.t.} \quad & \sum_{r \in \mathcal{R}_{vt}} x_r \leq 1, & \forall\, v \in V,\ t \in \mathcal{T} \\ & \sum_{v \in \mathcal{V}} \sum_{r \in \mathcal{R}_{vt}} D_{br}^T x_r \leq G_{bt}, & \forall\, b \in \mathcal{B},\ t \in \mathcal{T} \\ & \sum_{v \in \mathcal{V}} \sum_{t \in \mathcal{T}} \sum_{r \in \mathcal{R}_{vt}} I_{wr} x_r \leq 1, & \forall\, w \in \mathcal{W} \\ & x_r \in \{0,1\}, & \forall\, v \in \mathcal{V},\ t \in \mathcal{T}, r \in \mathcal{R}_{vt}\ . \end{aligned} \tag{8}$$

Now, consider the continuous relaxation of the master problem. It is identical to (8), except that we replace the last group of constraints by

$$x_r \in \mathbb{R} : x_r \geq 0, \qquad \forall\, v \in \mathcal{V},\ t \in \mathcal{T},\ r \in \mathcal{R}_{vt}. \tag{9}$$

Note that the variable x_r is already bound by one above as a result of (5). Since the available technicians G_{bt} are non-negative, the problem has a feasible solution $x_r = 0,\ \forall\ r \in \mathcal{R}$, with objection function value 0. An obvious upper bound $x_r = 1,\ \forall\ \ r \in \mathcal{R}$ exists, and we can therefore conclude that an optimal solution x_r^* also exists and corresponds to an objective function value less than or equal to $\sum_{r \in \mathcal{R}} P_r$.

The dual of the master problem is expressed as

$$
\begin{aligned}
\min \quad & \sum_{v\in\mathcal{V}}\sum_{t\in\mathcal{T}}\lambda_{vt} + \sum_{b\in\mathcal{B}}\sum_{t\in\mathcal{T}} G_{bt}\mu_{bt} + \sum_{w\in\mathcal{W}}\rho_w \\
\text{s.t.} \quad & \lambda_{vt} + \sum_{b\in\mathcal{B}} D^T_{br}\mu_{bt} + \sum_{w\in W} I_{wr}\rho_w \geq P_r, \qquad && \forall\ v\in\mathcal{V},\ t\in\mathcal{T},\ r\in\mathcal{R}_{vt} \\
& \lambda_{vt} \geq 0, && \forall\ v\in\mathcal{V},\ t\in\mathcal{T} \\
& \mu_{bt} \geq 0, && \forall\ b\in\mathcal{B},\ t\in\mathcal{T} \\
& \rho_w \geq 0, && \forall\ w\in\mathcal{W},
\end{aligned}
\tag{10}
$$

where the variables $\lambda_{vt},\ \mu_{bt}$, and ρ_w, correspond to the sets of constraints expressed in (5), (6), and (7), respectively. Let $\lambda^*_{vt},\ \mu^*_{bt}$, and ρ^*_w, be an optimal solution to (10). Since the problem is a linear program, this solution exists by the strong duality theorem. In section 4.3, we show how we take advantage of the dual optimal solution to generate helpful feasible routes.

4.2 Sub problem - Route generation

In section 4.1, we defined a set of feasible routes $\mathcal{R}$. Now we define the requirements of being an element in this set.

4.2.1 Graph representation

Consider the directed graph $G = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N}$ is a set of nodes and $\mathcal{A}$ is a set of arcs. Earlier, we introduced a set of turbines $\mathcal{W}$. Every turbine $w \in \mathcal{W}$ is associated with two nodes in $\mathcal{N}$: One *delivery node* and one *pick-up node*. They correspond to the exact same geographical location (the associated turbine), but they distinguish between delivery and pick-up of technicians. In addition to the two nodes for each turbine, $\mathcal{N}$ also contains an *origin node* and a *destination node*. They also refer to identical locations, namely the port location. To simplify the notations, we represent nodes by integers. For a number $n = |\mathcal{W}|$ of turbines,

$$
\begin{aligned}
\mathcal{W} = &\quad \{w_1,\ w_2,\ ..., w_n\} \\
\mathcal{N}^D = &\quad \{1,\ 2\ ..., n\} \\
\mathcal{N}^P = &\quad \{n+1,\ n+2, ..., 2n\} \\
\mathcal{N}^{OD} = &\quad \{0,\ 2n+1\},
\end{aligned}
$$

where $\mathcal{N}^D$ is the set of delivery nodes, $\mathcal{N}^P$ is the set of pick-up nodes, and $\mathcal{N}^{OD}$ is a set that only contains the origin and destination node. We can now express $\mathcal{N}$ as

$$\mathcal{N} = \mathcal{N}^{OD} \cup \mathcal{N}^D \cup \mathcal{N}^P.$$

For safety reasons, a vessel must remain within a maximum distance of S from the turbine at which it has disembarked technicians. Therefore we define the set $\mathcal{N}_i^S \subseteq \mathcal{N}^D$ for all $i \in \mathcal{N}^D$ that contains all delivery nodes closer than S from i, including i itself.

The set of arcs $\mathcal{A}$ consists of ordered pairs of nodes. It contains the pairs that represent the following arcs:

- From the origin node to all delivery nodes;
  $(0, j),\ \forall\ j \in \mathcal{N}^D$.
- From delivery nodes to other delivery nodes closer than S;
  $(i, j),\ \forall\ i \in \mathcal{N}^D,\ j \in \mathcal{N}_i^S \setminus \{i\}$.
- From delivery nodes to pick-up nodes closer than S;
  $(i, j),\ \forall\ i \in \mathcal{N}^D,\ j \in \mathcal{N}^P : j - n \in \mathcal{N}_i^S$
- From pick-up nodes to the destination node
  $(i, 2n+1),\ \forall\ i \in \mathcal{N}^P$
- From pick-up nodes to all delivery nodes except the corresponding;
  $(i, j),\ \forall\ i \in \mathcal{N}^P,\ j \in \mathcal{N}^D \setminus \{i - n\}$
- From pick-up nodes to all other pick-up nodes closer than S;
  $(i, j)\ \forall\ i \in \mathcal{N}^P,\ j \in \mathcal{N}^P : j - n \in \mathcal{N}_{i-n}^S$.

The set $\mathcal{A}$ is demonstrated for a small problem instance in Figure 5.

A feasible route $r \in \mathcal{R}^{ALL}$ is always a directed path from the origin node to the destination node in the digraph G. To describe such a path, we introduce the variables

$$y_{ij} = \begin{cases} 1 & \text{if the vessel travels directly from node } i \text{ to node } j \\ 0 & \text{otherwise,} \end{cases} \qquad \forall\ (i, j) \in \mathcal{A}.$$

4.2.2 Travel constraints

The equation

$$\sum_{j:(i,j)\in\mathcal{A}} y_{ij} - \sum_{j:(j,i)\in\mathcal{A}} y_{ji} = 0, \qquad \forall\ i \in \mathcal{N} \setminus \mathcal{N}^{OD}, \tag{11}$$

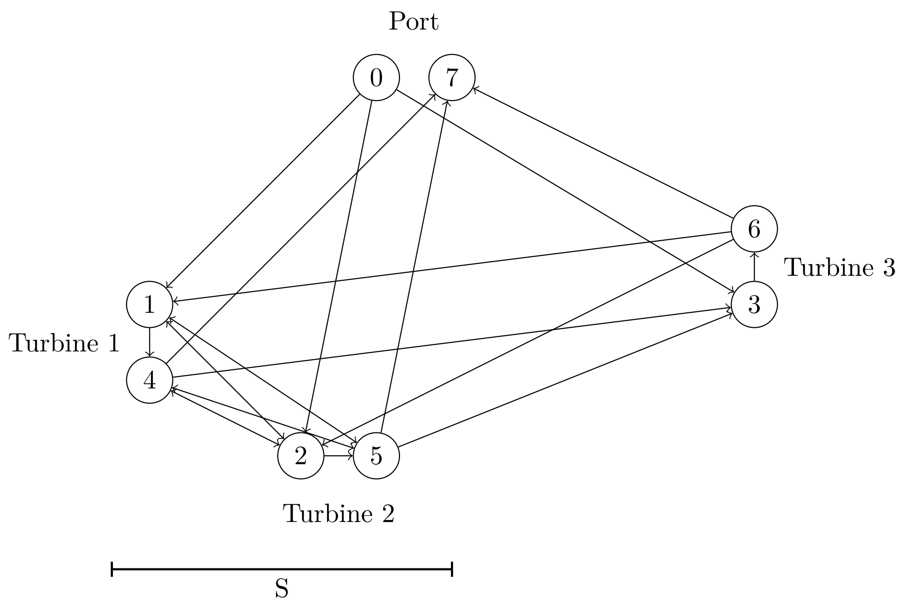


Figure 5: An example of what the graph $G = (\mathcal{N}, \mathcal{A})$ can look like for a simple case with three turbines ($n = 3$). Turbine 1 and Turbine 2 are closer than S.

ensure that the vessel can not visit a delivery or pick-up node without leaving it, and vice versa. The vessel is also forced to leave the origin node exactly once by

$$\sum_{j \in \mathcal{N}^D} y_{0j} = 1, \tag{12}$$

and return to the destination node exactly once by

$$\sum_{i \in \mathcal{N}^P} y_{i(2n+1)} = 1. \tag{13}$$

Another route feasibility criterion is to visit a pick-up node if and only if the corresponding delivery node is visited. This can be expressed as

$$\sum_{i:(i,j) \in \mathcal{A}} y_{ij} - \sum_{i:(i,j) \in \mathcal{A}} y_{i(j+n)} = 0, \qquad \forall\, j \in \mathcal{N}^D. \tag{14}$$

Note that equation (14) does not say anything about the order. It is definitely infeasible to visit a pick-up node before the corresponding delivery node, but this will be taken care of in the following *time constraints*.

4.2.3 Time constraints

The total time vessel v is available in time period t is limited to T^P_{vt}. We define a variable q_i, which equals the time the vessel leaves node i. For a route to be feasible, the vessel has to arrive at the port within the available time. Mathematically, we insist that

$$q_{2n+1} \leq T_{vt}^{P}. \tag{15}$$

A vessel v has a minimum travel time T_{ijv} from node i to node j. We can use this to limit the time at the nodes by

$$q_i + T_{ijv} \leq (T_{vt}^{P} + T_{ijv})(1 - y_{ij}) + q_j, \qquad \forall\ (i,j) \in \mathcal{A} \tag{16}$$

Note that $T_{vt}^{P} + T_{ijv}$ is big enough to make the inequality redundant for routes where a vessel is not traveling directly from i to j. Another parameter $\tilde{T}_i$ reflects the time required for the technicians to complete the task corresponding to node i. Or, more precisely, the minimum time between the visit of a delivery node and its corresponding pick-up node. It is only defined for the delivery nodes $i \in \mathcal{N}^D$. The constraint

$$q_i + \tilde{T}_i \leq q_{i+n}, \qquad \forall\ i \in \mathcal{N}^D, \tag{17}$$

ensures that a pick-up node cannot be visited before the required time after delivery has passed. Note that it also constrains the order. Now, a pick-up node cannot be visited before its corresponding delivery node.

4.2.4 Technician constraints

Recall the set $\mathcal{B}$ of technician types and the number G_{bt} of available technicians of type b from section 4.1. We define the non-negative integer variable z_{bi}, which describes the number of technicians of type b onboard the vessel when it leaves node i. The number of technicians onboard vessel v when it leaves the origin node cannot exceed the number of available technicians in the period. Therefore, the inequality

$$z_{b0} \leq G_{bt}, \qquad \forall\ b \in B \tag{18}$$

must be satisfied. In addition, vessel v is not allowed to have more than K_v technicians onboard. That can be expressed as

$$\sum_{b \in \mathcal{B}} z_{b0} \leq K_v. \tag{19}$$

Every node i, except the origin and destination node, is associated with a demand F_{bi} for technicians of technician type b. The demand at delivery nodes is positive, and the demand at pick-up nodes is defined as the negative of the demand at their corresponding delivery nodes. Then,

$$z_{bi} - F_{bj} \leq z_{bj} + K_v(1 - y_{ij}), \qquad \forall\ (i,j) \in A,\ b \in \mathcal{B} \tag{20}$$

and

$$z_{bi} - F_{bj} \geq z_{bj} - K_v(1 - y_{ij}), \qquad \forall\ (i,j) \in A,\ b \in \mathcal{B} \tag{21}$$

force z_{bi} to be reduced by the technician demand when a node is visited, i.e., $z_{bj} = z_{bi} - F_{cj}$ if $y_{ij} = 1$.

The final constraint in the sub-problem deals with the safety distance between a vessel and disembarked technicians. This issue is partially dealt with by not defining arcs between nodes that can never be visited consecutively. However, we still need to prevent the situation where a vessel travels from delivery node i to delivery node j, via a pick-up node k, where i and k do not correspond to the same turbine and $j \notin \mathcal{N}_i^S$. To achieve this, we formulate the two legal ways i and j can be visited by the same vessel in the same period. Either both i and its corresponding pick-up node $i+n$ are visited before j, or they are both visited after the pick-up node $j+n$ that corresponds to j. We take advantage of the time variable q and state that

$$q_{i+n} \leq q_j \quad \text{or} \quad q_i \geq q_{j+n}, \tag{22}$$

if a vessel visits both i and j, and $j \notin \mathcal{N}_i^S$. Further, we make the constraint (22) redundant if a vessel does not visit both i and j. The total available time in the period, T_{vt}^P, is always a great enough number to achieve this if we rewrite the constraint as

$$\begin{aligned} -T_{vt}^P(1 - \sum_{k:(i,k)\in\mathcal{A}} x_{ik}) + q_{i+n} \leq q_j + T_{vt}^P(1 - \sum_{k\in\mathcal{N}^D} x_{jk}) \\ \text{or} \\ T_{vt}^P(1 - \sum_{k:(i,k)\in\mathcal{A}} x_{ik}) + q_i \geq q_{j+n} - T_{vt}^P(1 - \sum_{k\in\mathcal{N}^D} x_{jk}), \\ \forall\ (i,j) \in \mathcal{N}^D \times \mathcal{N}^D : j \notin \mathcal{N}_i^S. \end{aligned} \tag{23}$$

Finally, to get around the disjunction, we introduce a binary variable m_{ij} for every pair $(i,j) \in \mathcal{N}^D \times \mathcal{N}^D$: $j \notin \mathcal{N}_i^S$. Now the expression can be split into

$$\begin{aligned} -T_{vt}^P(1 - m_{ij}) - T_{vt}^P(1 - \sum_{k:(i,k)\in\mathcal{A}} x_{ik}) + q_{i+n} \leq q_j + T_{vt}^P(1 - \sum_{k\in\mathcal{N}^D} x_{jk}), \\ \forall\ (i,j) \in \mathcal{N}^D \times \mathcal{N}^D : j \notin \mathcal{N}_i^S, \end{aligned} \tag{24}$$

and

$$\begin{aligned} T_{vt}^P m_{ij} + T_{vt}^P(1 - \sum_{k:(i,k)\in\mathcal{A}} x_{ik}) + q_i \geq q_{j+n} - T_{vt}^P(1 - \sum_{k\in\mathcal{N}^D} x_{jk}), \\ \forall\ (i,j) \in \mathcal{N}^D \times \mathcal{N}^D : j \notin \mathcal{N}_i^S. \end{aligned} \tag{25}$$

Note that $m_{ij} = 0$ makes (24) redundant and $m_{ij} = 1$ makes (25) redundant. The result is that at least one side of disjunction (23) must be true if both (24) and (25) are satisfied.

4.2.5 Objective

As mentioned in section 4.1, every route is associated with a profit P_r. This profit can be reformulated as revenues subtracted by costs. The revenue R_{it} is defined for all delivery nodes and reflects the expected revenue associated with completing maintenance at the turbine corresponding to node i at time period t. The cost C^T_{ijvt} describes the cost of traveling from node i to node j by vessel v in time period t. The profit of a route can then be expressed as

$$P_r = \sum_{(i,j)\in\mathcal{A}:j\in\mathcal{N}^D} R_{jt} y_{ij} - \sum_{(i,j)\in\mathcal{A}} C^T_{ijvt} y_{ij}. \tag{26}$$

For every vessel v, we want to find the most profitable route at time period t. The objective is therefore to maximize P_r.

4.3 Column generation

Consider an instance of the continuous relaxation of the master problem (9) where the set of routes $\mathcal{R}$ is a relatively small subset of all feasible routes $\mathcal{R}^{ALL}$, i.e.,

$$\mathcal{R} \subset \mathcal{R}^{ALL},\ |\mathcal{R}| \ll |\mathcal{R}^{ALL}|.$$

As shown in section 4.1, this instance has an optimal dual solution $\lambda^*_{vt},\ \mu^*_{bt}$, and ρ^*_w, which obviously satisfy the dual constraints,

$$\lambda^*_{vt} + \sum_{b\in\mathcal{B}} D^T_{br}\mu^*_{bt} + \sum_{w\in W} I_{wr}\rho^*_w \geq P_r, \qquad \forall\ v\in\mathcal{V},\ t\in\mathcal{T},\ r\in\mathcal{R}_{vt}.$$

We substitute P_r by the right-hand side of equation (26). The technician demand D^T_{br} can be interchanged with the route variable z_{b0}, i.e., the number of technicians of type b onboard vessel v when it leaves the port. These modifications result in the inequality,

$$\lambda^*_{vt} + \sum_{b\in\mathcal{B}} z_{b0}\mu^*_{bt} + \sum_{w\in W} I_{wr}\rho^*_w \geq \sum_{(i,j)\in\mathcal{A}:j\in\mathcal{N}^D} R_{jt} y_{ij} - \sum_{(i,j)\in\mathcal{A}} C^T_{ijv} y_{ij},\ \ \forall\ v\in\mathcal{V},\ t\in\mathcal{T},\ r\in\mathcal{R}_{vt}. \tag{27}$$

Further, we want to express the master parameter I_{wr} by the route variable y_{ij}. To achieve this, we allow ourselves to abuse the notation a bit by indexing ρ with $j\in\mathcal{N}^D$, instead of its original $w\in\mathcal{W}$. This makes no practical difference as there is a one-to-one relationship between turbines and delivery nodes. Thus, we apply the substitution

$$\sum_{w\in W} I_{wr}\rho^*_w \to \sum_{(i,j)\in\mathcal{A}:j\in\mathcal{N}^D} y_{ij}\rho^*_j, \tag{28}$$

to inequality (27). After some relocation and contraction, we are left with

$$\sum_{(i,j)\in\mathcal{A}:j\in\mathcal{N}^D} y_{ij}(R_{jt}-\rho_j^*)-\lambda_{vt}^*-\sum_{(i,j)\in\mathcal{A}} C_{ijv}^T y_{ij}-\sum_{b\in\mathcal{B}} z_{b0}\mu_{bt}^*\leq 0, \qquad \forall\, v\in\mathcal{V},\ t\in\mathcal{T}. \tag{29}$$

Note that we went from representing route $r \in \mathcal{R}$ by D_{br}^T and I_{wr}, to representing it by y_{ij} and z_{b0}. We know that an arbitrary route $r \in \mathcal{R}$, corresponds to some values y'_{ij} and z'_{b0}, and that it satisfies inequality (29) when v and t equals the route's corresponding vessel and period. We are interested in finding new routes $r \in \overline{\mathcal{R}} \cap \mathcal{R}^{ALL}$. If we can find a route r' that satisfies all the constraints discussed in section 4.2, and at the same time violates (29), we know that $r' \in \overline{\mathcal{R}} \cap \mathcal{R}^{ALL}$. Now, we connect all constraints presented in this chapter with an objective function that aims to maximize the violation of (29). For all vessels $v \in \mathcal{V}$ and all time periods $t \in \mathcal{T}$, we solve the mixed integer linear program,

$$
\begin{aligned}
\max \quad & -\lambda^*_{vt} + \sum_{(i,j)\in\mathcal{A}:j\in\mathcal{N}^D} y_{ij}(R_{jt} - \rho^*_j) - \sum_{(i,j)\in\mathcal{A}} C^T_{ijvt} y_{ij} - \sum_{b\in\mathcal{B}} z_{b0}\mu^*_{bt} \\
\text{s.t.} \quad & \sum_{j:(i,j)\in\mathcal{A}} y_{ij} - \sum_{j:(j,i)\in\mathcal{A}} y_{ji} = 0, && \forall\, i \in \mathcal{N} \setminus \mathcal{N}^{OD} \\
& \sum_{j\in\mathcal{N}^D} y_{0j} = 1 \\
& \sum_{i\in\mathcal{N}^P} y_{i(2n+1)} = 1 \\
& \sum_{i:(i,j)\in\mathcal{A}} y_{ij} - \sum_{i:(i,j)\in\mathcal{A}} y_{i(j+n)} = 0, && \forall\, j \in \mathcal{N}^D \\
& q_{2n+1} \le T^P_{vt} \\
& q_i + T_{ijv} \le (T^P_{vt} + T_{ijv})(1 - y_{ij}) + q_j, && \forall\, (i,j) \in \mathcal{A} \\
& q_i + \tilde{T}_i \le q_{(n+i)}, && \forall\, i \in \mathcal{N}^D \\
& -T^P_{vt}(1 - m_{ij}) - T^P_{vt}(1 - \sum_{k\in\mathcal{N}^D} x_{ik}) + q_{i+n} \\
& \qquad \le q_j + T^P_{vt}(1 - \sum_{k\in\mathcal{N}^D} x_{jk}), && \forall\, (i,j) \in \mathcal{N}^D \times \mathcal{N}^D : j \notin \mathcal{N}^S_i \\
& T^P_{vt} + m_{ij} T^P_{vt}(1 - \sum_{k\in\mathcal{N}^D} x_{ik}) + q_i \\
& \qquad \ge q_{j+n} - T^P_{vt}(1 - \sum_{k\in\mathcal{N}^D} x_{jk}), && \forall\, (i,j) \in \mathcal{N}^D \times \mathcal{N}^D : j \notin \mathcal{N}^S_i \\
& z_{b0} \le G_{bt}, && \forall\, b \in \mathcal{B} \\
& \sum_{b\in\mathcal{B}} z_{b0} \le K_v \\
& z_{bi} - F_{bj} \le z_{bj} + K_v(1 - y_{ij}), && \forall\, (i,j) \in \mathcal{A},\ b \in \mathcal{B} \\
& z_{bi} - F_{bj} \ge z_{bj} - K_v(1 - y_{ij}), && \forall\, (i,j) \in \mathcal{A},\ b \in \mathcal{B} \\
& y_{ij} \in \{0,1\}, && \forall\, (i,j) \in \mathbf{A} \\
& q_i \in \mathbb{R}^+_0, && \forall\, i \in \mathcal{N} \\
& z_{bi} \in \mathbb{Z}^+_0, && \forall\, b \in \mathcal{B},\ i \in \mathcal{N}.
\end{aligned}
\tag{30}
$$

If the optimal objective function value is positive, the route corresponding to the solution extends the set $\mathcal{R}$ to $\mathcal{R}' = \mathcal{R} \cup \{r'\}$. Otherwise, the route is ignored. When the sub-problem (30) is solved for all vessels and time periods, the relaxed master problem is solved again, but for $\mathcal{R}'$ this time. The new optimal dual solution is then used to create a new set of sub-problem instances. This process is repeated until none of the sub-problem instances have positive optimal objective function values. When that happens, we solve the integer master problem for the current set of feasible routes. The whole procedure is expressed in Algorithm 2. Note that even if the last relaxed optimal solution is optimal for all feasible routes $\mathcal{R}^{ALL}$, the integral optimal solution does not have to be. We can only be sure of that if the optimal objective function value is equal for the relaxed and the integer master problem. However, the gap between the two is an indication of how good the integer solution is.

Algorithm 2 Column generation

```
$\mathcal{R} \leftarrow \emptyset$
stopping criterion $\leftarrow$ **False**
**while** stopping criterion $=$ **False do**
    $\lambda^*,\ \mu^*,\ \rho^* \leftarrow$ solve the dual master problem (10) for $\mathcal{R}$
    stopping criterion $\leftarrow$ **True**
    **for** $(v, t) \in \mathcal{V} \times \mathcal{T}$ **do**
        $r \leftarrow$ solve the sub problem (30) with $v$, $t$, $\lambda^*$, $\mu^*$ and $\rho^*$ as input
        **if** $r$ violates the dual constraint (29) **then**
            $\mathcal{R} \leftarrow \mathcal{R} \cup \{r\}$
            stopping criterion $\leftarrow$ **False**
        **end if**
    **end for**
**end while**
best-found integer solution $\leftarrow$ solve the integer master problem (8) for $\mathcal{R}$
```

5 Heuristic approach

We encounter two main challenges when we implement Algorithm 2 and test it on various input data. The first one is related to the complexity of the sub-problem (30). We use Gurobi 10.0 [31] to solve the master problem and the sub-problem. While Gurobi efficiently solves the master problem, its performance on the sub-problem proves to be highly sensitive to the number of turbines n. Already for $n = 10$, Gurobi may use an unreasonable amount of time to find an optimal solution to the sub-problem. Considering that we may need to solve the sub-problem many times to meet the stop condition in Algorithm 2, we should be able to solve it relatively quickly.

The second challenge is that some input data result in the case where the optimal objective function value of the final master problem is smaller than the objective function value of the corresponding continuous relaxation. In some cases, we are even able to manually create feasible routes that can improve the final integer solution.

In this chapter, we propose two heuristic algorithms that answer these challenges. The chapter is structured as follows: First, we present the ALNS framework both algorithms utilize. Then, we explain each algorithm's properties. Finally, we discuss the consequences of using heuristic methods.

We advise the reader to recall definitions and explanations from Section 3.3.

5.1 Route generation with ALNS

The heuristic algorithms produce routes, meaning they both provide solutions to the sub-problem (30). However, instead of representing the solution with all variables in the sub-problem, we express it by the two vectors u and $\overline{u}$. The vector u consists of two instances of every turbine visited in the solution, while $\overline{u}$ contains one instance of each turbine not visited. For instance, consider the sub-problem instance with four turbines which, for simplicity, are represented by positive integers. Then,

$$u = [1, 1, 4, 2, 4, 2] \qquad \text{and} \qquad \overline{u} = [3], \tag{31}$$

express the solution where a vessel visits three of the turbines. First, it travels from the port to Turbine 1 and is on standby at the turbine while the transferred technicians conduct maintenance. When they are done, it embarks them again and travels to Turbine 4. At Turbine 4, the vessel disembarks the required technicians before it immediately travels to Turbine 2. Further, the vessel travels back to Turbine 4 and picks up the technicians when they are done, then back to Turbine 2 and picks up the technicians when they are done there. Finally, it travels from Turbine 2 back to the port. Further, $\overline{u}$ reflects that Turbine 3 is not visited in the example solution.

It is easy to extract the variable y_{ij} from u as the first occurrence of a number in u corresponds to a turbine's delivery node, and the second occurrence corresponds to a turbine's pick-up node. However, in some cases, the variable we try to extract may not be defined because of the safety distance S. For example, consider Turbine 2 and Turbine 4 from (31). If they are located at a distance greater than S from each other, the arc between the delivery nodes of Turbine 4 and Turbine 2 does not exist. Hence, no y_{ij} corresponds to that arc, and the route is infeasible.

On the other hand, if all variables y_{ij} extracted from u exist, it is impossible to express a route that violates the constraints (11) to (14) introduced in Section 4.2.1. Further, by using the travel times T_{ijv} and the required times $\tilde{T}_i$ for the technicians to complete the maintenance tasks, we can extract all time variables q_i and check whether the vessel can complete the route within the available time T^P_{vt}. We also determine the minimal numbers of technicians, z_{b0}, the vessel must bring from the port to complete the route and ensure that they do not exceed the vessel's capacity or the number of available technicians of each type. Finally, we verify that none of the turbines in u that are further than S from each other are maintained in parallel.

We implement an ALNS framework that follows the principals of Algorithm 1 from Section 3.3. However, in the subsequent discussion, solutions are identified by the vector pair $(u, \overline{u})$ as explained above, rather than the symbol x from Algorithm 1. Further, we initialize the temperature according to the warm-up procedure described at the end of Section 3.3.2.

5.1.1 Operators

We create two classes of destroy operators: *k-remove* and *p-percent-remove*. The k-remove operator class selects k random turbines from u. By removing a turbine, we mean removing both instances of it in u and append one instance of it to $\overline{u}$. If the current u contains less than k turbines, it removes all visited turbines from u. We instantiate two destroy operators from this class where k is set to 1 and 2, respectively. The p-percent-remove class removes p percent of the turbines in u. If p percent of the visited turbines are non-integer, p is replaced by $\lceil p \rceil$. We instantiate operators for $p \in \{20,\ 50\}$ and add them to the collection of destroy operators.

For the repair operators, we create three procedures: *k-random-insert*, *k-greedy-insert*, and *k-greedy-insert-regret*. The k-random insert class is the simplest one, as it picks k random turbines from $\overline{u}$ and, in random order, inserts them one by one into the position in u that results in the best objective function value. We test all position combinations for both instances of each turbine. We instantiate instances of the k-random-insert for $k = 1$, $k = 2$, and $k = 3$.

The k-greedy-insert selects k turbines from $\overline{u}$ based on a probability that depends on the turbines' revenue. Recall the revenue R_{wt} earned by maintaining turbine w in period t. We set the probability of selecting turbine $w \in \overline{u}$ to

$$\frac{R_{wt}}{\sum_{w \in \overline{u}} R_{wt}}, \qquad \forall\, w \in \overline{u}.$$

For $k > 1$, we repeat this selection process until we have selected k turbines. Afterward, we reinsert them in the same procedure as in k-random-insert. For k-greedy-insert, we instantiate operators for $k \in \{1, 2\}$.

The k-greedy-insert-regret class builds upon the k-greedy-insert by considering all possible insertion orders for the selected turbines. Once all the turbines are selected, it evaluates the objective function value for each possible insertion order and selects the order that results in the best objective function value. This approach aims to account for the potential interactions between turbines and find a more efficient arrangement of the selected turbines within the solution. Finally, we instantiate operators for $k = 2$ and $k = 3$ and add them to the collection of repair operators.

We create eight operator pairs from the introduced destroy- and repair-operators and use them in Algorithm 1. From now on, when we refer to *the ALNS*, we mean this algorithm with these operators and the other specifications introduced in this chapter. Table 1 gives an overview of all the operator pairs.

Table 1: Overview of the pairs of destroy and repair operators

	Destroy operator	**Repair operator**
Operator 1	1-remove	1-random-insert
Operator 2	1-remove	2-random-insert
Operator 3	1-remove	1-greedy-insert
Operator 4	1-remove	2-greedy-insert-regret
Operator 5	2-remove	2-greedy-insert-regret
Operator 6	3-remove	3-random-insert
Operator 7	20-percent-remove	2-greedy-insert
Operator 8	50-percent-remove	3-greedy-insert-regret

5.2 Heuristic method 1: Sub-problem solver

The first heuristic algorithm we introduce is a straightforward solution algorithm for the sub-problem (30). The motivation for using the ALNS to solve the sup-problems is that an exact solver cannot provide optimal (or even good) solutions in a reasonable time for more challenging problem instances.

To solve the sub-problem, we set the max iterations I_{disp} to 5 000 and designate the first 500 iterations to the warm-up procedure.

5.3 Heuristic method 2: Warm start

Instead of initializing $\mathcal{R}$ with the empty set, as we first proposed with Algorithm 1, pre-generating some routes might be a good idea to kick-start the column generation. The idea is to create as many good routes as possible in a given number of seconds T^S. However, to avoid producing many similar routes, we replace the set of turbines $\mathcal{W}$ by a unique subset $\mathcal{W}' \subseteq \mathcal{W}$ each time.

For each vessel in each period, we generate all the feasible routes where the vessel visits only one turbine. Further, we produce the best routes for all subsets $\mathcal{W}' \subset \mathcal{W}$ where $|\mathcal{W}'| = 2$. For the problem sizes we consider in this thesis, this is possible to achieve with an *exhaustive search*, i.e., we check all possible solutions.

As there may be an unmanageable number of combinations when considering subsets of cardinality greater than two, we can only produce and check some possible solutions. In all iterations during the restoring time, we select a random vessel, a random period, and a random subset of cardinality k and solve it with the ALNS. The cardinality is given as a function of the current time a. Let a_0 be the time we start this last part of the warm start algorithm, and let a_1 be the time we terminate it. Then, the function

$$k(a) = \left\lceil \frac{a - a_1}{a_1 - a_0} \cdot (|\mathcal{W}| - 2) \right\rceil + 2 \tag{32}$$

determine the cardinality k. That is, $|\mathcal{W}'|$ increases linearly with the elapsed time, and converges to $|\mathcal{W}|$.

In the warm-start method, the number of iterations I_{disp} for each ALNS run is set to 2000, allowing more runs to be completed within the designated time. The designated time T^S is determined based on the problem instance size and the chosen algorithm. We save details on this for the experiments chapter, but Algorithm 3 summarizes a general overview of the procedure.

The motivation for the warm start method can be divided into three. Firstly, it can reduce the number of column generation iterations necessary to meet the stopping criteria in Algorithm 2, reducing the total run

Algorithm 3 Warm start

```
$\mathcal{R} \leftarrow \emptyset$
$a \leftarrow$ the time we start the algorithm
$a_1 \leftarrow a + T^S$
create all feasible routes for subsets $\mathcal{W}' : |\mathcal{W}| \leq 2$
$a \leftarrow$the time at the moment
$a_0 \leftarrow a$
while $a < a_1$ do
    $\mathcal{W}' \leftarrow$ a random subset of $\mathcal{W}$ with cardinality $k(a)$ (32)
    $v \leftarrow$ a random vessel, $t \leftarrow$ a random period
    $r' \leftarrow$ generate a route for $W'$, $v$, and $t$ using the ALNS
    $\mathcal{R} \leftarrow \mathcal{R} \cup \{r'\}$
    $a \leftarrow$the time at the moment
end while
```

time. Secondly, we believe that routes just servicing a few turbines (typically the ones generated early in this method) can reduce the potential gap between the optimal objective function value of the final master problem and the final master problem relaxation. The third reason is more for experimental purposes, as we use the warm start to solve the problem without column generation. We return to this with more details in Chapter 7.

5.4 Consequences

We cannot guarantee optimal solutions when we solve the sub-problem with the ALNS. Algorithm 2 may therefore terminate too early. A consequence is that we cannot be sure that the solution to the final master problem relaxation is an upper bound for the problem. However, suppose we can solve small instances to optimality by using the ALNS to solve the sub-problem. In that case, we believe it can also be used to create close-to-optimal solutions for more complex instances.

6 Instance generation

In our work, we have defined the problem differently from other studies in the field. As a result, we find that none of the benchmark problem instances in the literature are suitable for testing our proposed solution method. Consequently, one contribution of this thesis is the generation of the new instances we use for experiments in our study, which can be used in future research in the field. We have published all instances and a guide on how to read them on GitHub [32].

Despite making multiple attempts to obtain operation and maintenance data from owners and operators of existing offshore wind farms, we have been unsuccessful. Therefore, we generate test instances using open-source information. While we cannot guarantee these instances' accuracy or realism, they provide some guidelines for the potential problem sizes and implementation performance.

We create eight problem instances that mainly differ in the cardinality of three of the input sets: turbines, periods, and technician types. Table 2 gives a complete overview of these cardinalities.

Table 2: The cardinalities of the input sets in the generated problem instances

	Turbines	Vessels	Periods	Technician Types
Instance 1	4	2	2	1
Instance 2	6	2	2	2
Instance 3	8	2	2	2
Instance 4	10	2	3	2
Instance 5	15	2	5	2
Instance 6	25	2	6	2
Instance 7	45	2	8	3
Instance 8	60	2	14	4

6.1 Turbines

For turbine locations, we use the coordinates of the turbines in the Triton Knoll wind farm [33]. Triton Knoll consists of 90 bottom fixed turbines located approximately 40 kilometers off the east coast of England [34]. It is co-owned and operated by the German energy company RWE Renewables. They have an operation and maintenance base at Grimsby Port, one of the closest ports to the wind farm. We have not been in contact with RWE, but we believe they perform most of the light maintenance on the Triton Knoll turbines by using

CTVs that travel from and return to this base within a workday. All instances therefore use the location of this base and the locations of a subset of the Triton Knoll turbines as positional inputs. Figure 6 shows all turbine locations and the base location.

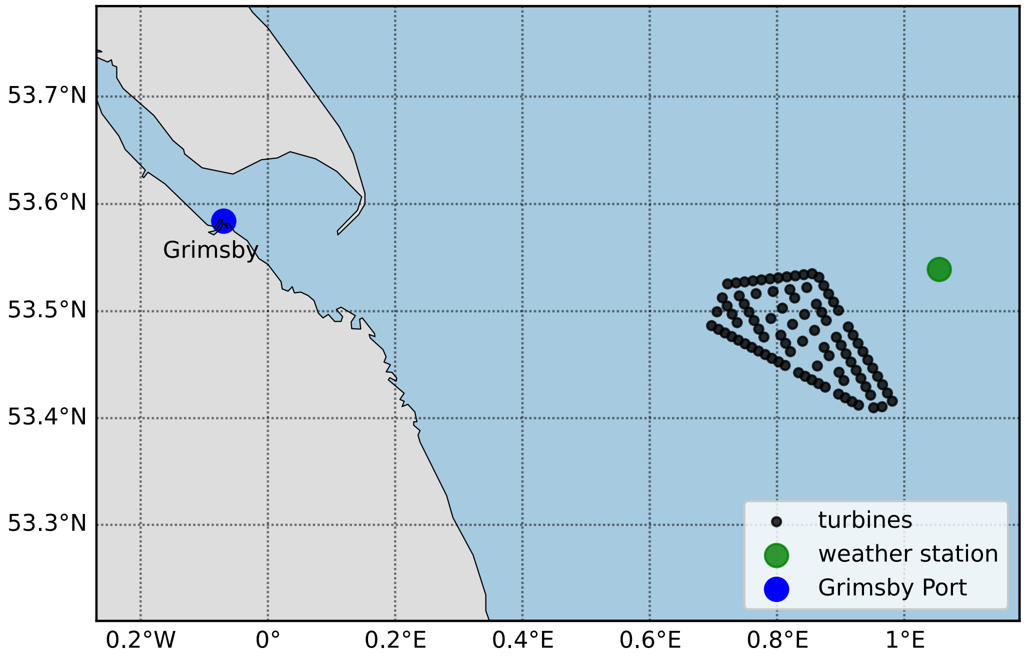


Figure 6: Map illustrating the specific locations of the turbines in Triton Knoll, the RWE maintenance base in Grimsby, UK, and the weather station supplying the wave data for the problem instances.

6.2 Vessels

We have repeatedly observed two vessels, the M/V Maker and the MSC Swath 2, near the Triton Knoll turbines and in the area between the wind farm and the base. Our observations were facilitated by tracking the AIS signals of these vessels on MarineTraffic [35]. In all instances, we utilize the speed and technician capacity of the two vessels to generate realistic input parameters. This information is reported in their specification sheets [36, 37]. However, these sheets do not provide information about fuel consumption. Therefore, we rely on the fuel consumption data in another CTV's specification sheet [38]. This particular CTV offers both the M/V Marker's and the MSC Swath 2's engine systems as options, and its sheet includes fuel consumption data for both.

Table 3: Specifications of the vessels that are used in the instances. The Min cost and the Max cost entries are the approximated costs when the significant wave height is less than 0.5m and equal to 1.5, respectively.

Name	**Speed**	**Tech. Cap.**	**Min Cost**	**Max Cost**
MV Marker [36]	27kn	24	€21/nm	€25/nm
MSC Swath 2 [37]	22kn	24	€39/nm	€47/nm

6.3 Weather conditions

To increase the realism of the instances regarding weather conditions, we use historical wave data (2017-01-01 to 2022-12-31) from a weather station close to Triton Knoll [39]. The exact position of the weather station is shown in Figure 6. We assign a date range to each instance, where the duration of the range is the same as the number of periods in that particular instance. The wave data in the assigned period affect the instance data in two ways; Firstly, it restricts the available time in the period. We set the available time for a vessel in a period to the maximum time interval between 08:00 and 20:00, in which the significant wave height does not exceed the vessels' transfer limitation. We use a transfer limitation of 1.5 meters significant wave height for both vessels.

Secondly, we assume that a vessel's service speed is constant regardless of weather conditions. However, we adjust its fuel consumption based on the average significant wave height in the period. For a significant wave height h and a vessels consumption D^{FUEL} from its specification sheet, we adjust the fuel consumption to

$$D^{ADJ} = \begin{cases} D^{FUEL}(1 + \frac{0.1h}{1.5}), & \text{if } h > 0.5 \\ D^{FUEL}, & \text{otherwise.} \end{cases}$$

Further, we use this adjusted fuel consumption and a fuel price of €1.5/L to extract each vessel's cost per distance in each period. The min and max costs for each vessel are given in Table 3.

6.4 Technicians

The required time to complete the maintenance task for delivery node i, denoted as $\tilde{T}_i$, is assigned a random duration between 1 and 6 hours, with a precision of half an hour.

For all instances, we set the number of required technicians F_{bi} of type b to complete the maintenance task corresponding to delivery node i, to a random integer between 0 and 4, where

$$\sum_{b\in\mathcal{B}} F_{bi} \in \{3,4,6\}, \qquad \forall\ i \in \mathcal{N}^D. \tag{33}$$

To achieve this, we generate a set of $|\mathcal{B}|$ random integers between 0 and 4. We then check if these numbers satisfy the condition defined in (33). If the condition is not met, we discard the set and repeat the process until it is. This procedure is used to set the technician demands for all maintenance tasks.

Each vessel's technician capacity K_v is given from its specification sheet. For both of the vessels we use in all instances, the corresponding sheet says $K_v = 24$.

Regarding the number of available technicians G_{bt} of type b in period p, we want the sum over types to be close to (within 20%), but never above, the sum of the vessels' technician capacities. We also want the available technicians to be distributed almost evenly (max deviation 20%) between the technician types. We therefore set G_{bt} to a random integer where,

$$\frac{4}{5}\sum_{v\in\mathcal{V}} K_v \leq \sum_{b\in\mathcal{B}} G_{bt} \leq \sum_{v\in\mathcal{V}} K_v, \qquad \forall\ t \in \mathcal{T},$$

and

$$\frac{4}{5}\cdot\frac{\sum_{v\in\mathcal{V}} K_v}{|\mathcal{B}|} \leq G_{bt} \leq \frac{6}{5}\cdot\frac{\sum_{v\in\mathcal{V}} K_v}{|\mathcal{B}|}, \qquad \forall\ b \in \mathcal{B},\ t \in \mathcal{T}$$

in all instances.

We set the safety distance S to 2 nautical miles (nm) in all instances.

6.5 Revenues

Before we get into how we set the revenues for each maintenance task, let us first develop an understanding of the cost levels involved in traveling from Grimsby Port to the Triton Knoll turbines. These turbines are located at different distances from the port, with the closest one about 28.5 nm away and the farthest one around 39.5 nm away.

Under conditions of less than 0.5m significant wave height, traveling to the nearest turbine and back to the port using the most cost-effective vessel would take approximately 2.1 hours and cost €1200. On the other hand, for the furthest turbine, using the most expensive vessel for the round trip would take about 3.6 hours and cost €3713.

Each of the considered turbines has precisely one corresponding available maintenance task. For all instances, we partition the turbines' delivery nodes in two; 10% of them (rounding upwards) correspond to *failure* *repairs*, while the restoring correspond *routine jobs.* We denote the set of delivery nodes that corresponds to failure repairs $\mathcal{N}^{DF}$.

Failure repairs involve restoring a turbine from a non-functioning state to its full energy harvesting potential. Therefore, the revenue for such tasks should reflect the earnings from the energy markets when a turbine is operative. We approximate the expected earnings E_t to €11400 for all periods $t \in \mathcal{T}$, but we also add a random component $\omega_t \in [-5000, 5000]$ to simulate variations in wind and electricity prices. Let us define the set $\mathcal{T}_t^A \subseteq \mathcal{T}$ for all $t \in \mathcal{T}$, that consists of t itself and all periods in the planning horizon that come after t. In each instance, we set the revenues for failure repairs to

$$R_{it} = \sum_{t \in \mathcal{T}^A} (E_t + \omega_t), \qquad i \in \mathcal{N}^{DF},\ t \in \mathcal{T}.$$

We set the revenues for the routine checks relatively low compared to the failure repairs. A task corresponding to delivery node $i \in \mathcal{N}^D \backslash \mathcal{N}^{DF}$ is given a random revenue between €500 and €20000 in the first period. Then, we decrease the revenue by 5% for each day after that. The decreasing revenue is intended to encourage early completion when costs are similar, as weather forecast uncertainties increase with time. Let each period be represented by its order in the planning horizon, i.e., $\mathcal{T} = \{1, 2, ..., |\mathcal{T}|\}$. We can then summarize how we set the revenues for the routine checks as

$$R_{i,1} \in [500, 20000], \qquad \forall\ i \in \mathcal{N}^D \setminus \mathcal{N}^{DF}$$

and

$$R_{it} = 0.95 R_{i,t-1}, \qquad \forall\ i \in \mathcal{N}^D \setminus \mathcal{N}^{DF},\ t \in \mathcal{T} : t > 1.$$

7 Experiments

The main goal of the experiments we present in this chapter is to test the proposed solution method. To achieve this, we address the two questions:

- Is the proposed ALNS sup-problem solver good enough?
- Is the column generation working well to solve the considered problem?

We explain our two-phase methodology to answer these questions in Section 7.1. Further, we present and discuss the results in Section 7.2. A secondary goal of the experiments is to get an intuition of what characterizes good solutions to the generated problem instances. Consequently, we finish Section 7.2 by examining the factors that influence the solutions' quality and see if we can determine any repeating trends across the instances.

7.1 Method

We define abbreviations for the three solution algorithms that we use in the experiments:

- CGA-H: The column generation algorithm (Algorithm 2) with warm start and heuristically solved sub-problem,
- CGA-E: The column generation algorithm (Algorithm 2) with warm start and sub-problem solved to optimality using Gurobi, and
- CA: *The challenger algorithm*, which is an extended version of the warm start heuristic (we introduce it in Section 7.1.2).

The designated time for the warm start in CGA-H and CGA-E is summarized in Table 4. Note that the time is set to 0 in CGA-E for Instances 1-3. This is because CGA-E finds equally good solutions in a shorter time without warm start for these problem instances.

7.1.1 Phase 1

CGA-H and CGA-E are already well-defined in Chapters 4 and 5. In the first part of the experiment, we solve the eight instances presented in Chapter 6, using CGA-H. To determine robustness, we solve each

Table 4: The designated time for the warm-start heuristic in CGA-H and CGA-E

	CGA-H	CGA-E
Instance 1	60	0
Instance 2	60	0
Instance 3	60	0
Instance 4	60	60
Instance 5	100	100
Instance 6	100	100
Instance 7	200	200
Instance 8	300	300

instance five times, calculate the average solution, and compare it to the best among the five. Further, we also try to solve all instances using CGA-E. However, the process is terminated if the stopping criteria are not met within 15,000 seconds.

The motivation for this phase is to determine if CGA-H consistently finds optimal, or at least as good solutions as the CGA-E, to the instances CGA-E is able to solve in a reasonable time. Additionally, we aim to get an indication of the instance sizes that CGA-E can successfully solve and assess the performance of CGA-H on instances larger than this potential threshold.

7.1.2 Phase 2

The Challenger Algorithm is derived from the warm-start method (Algorithm 3). In CA, we set the time limit T^S equal to the average time CGA-H takes to reach the stopping condition for each problem instance in Phase 1. When the allocated time is over, CA solves the master problem (8) for the generated routes. Like with CGA-H, we also run CA five times for each instance to determine the robustness and get a better basis of comparison with the CGA-H. To be clear, we summarize CA in Algorithm 4.

Algorithm 4 Challenger algorithm

```
$T^S \leftarrow$ the average time used by CGA-H to solve the same instance
$\mathcal{R} \leftarrow$ the output from Algorithm 3 with time limit $T^S$
Solve the master problem 8 for $\mathcal{R}$
```

The motivation for Phase 2 is to investigate if column generation is a good idea. We want a better understanding of whether or not the dual weights efficiently force CGA-H/CGA-E to generate routes that contribute in better solutions.

7.1.3 Implementation choices and hardware specifications

We implement all algorithms in Python 3.9. To significantly improve speed, we use Numba [40] to compile most of the functions called by the heuristic methods. We solve the master problem, including its continuous relaxation, with Gurobi 10.0 [31]. IN CGA-E, we also solve the sub-problem with Gurobi 10.0.

We conduct all experiment runs on a PC equipped with an 11th Gen Intel Core i7-1165G7 processor with a base frequency of 2.80 GHz, 16.0 GB of installed RAM (15.7 GB usable), and a 64-bit Windows 10 operating system.

7.2 Results

Table 5 presents the results from Phase 1 of the experiments. In the table, “Average” refers to the average solution quality, measured as profit in euros, among the five runs for CGA-H. The “Best” column signifies the best solution found by both considered solution algorithms for each instance, while the “Run time” columns display the average running time for CGA-H and the single running time for each instance using CGA-E, respectively.

From the results, it is evident that CGA-E successfully solves the first four instances. However, it cannot reach the stopping criteria within 15,000 seconds for the four most complex instances, indicating that CGA-E has limitations in solving larger problem instances within a reasonable time. On the other hand, CGA-H consistently produces solutions for all instances.

Notably, CGA-E outperforms CGA-H in terms of speed for the first three instances, where both algorithms generate solutions of equal quality. This is because Gurobi can prove optimality and stop the search at some point. Conversely, the ALNS solver uses its 5000 available iterations even though an optimal solution might be found early in the search.

Also for Instance 4, the two algorithms find solutions of equal quality. However, going from Instance 3 to Instance 4, the run time for CGA-E increases with a factor of 100. In contrast, the run time of CGA-H does not even double. Even though we expected this to happen at some point, it is interesting that the jump happens going from an instance with eight turbines and two periods to an instance with ten turbines and three periods.

Table 5: Comparison of results from five runs of CGA-H and one run of CGA-E across all instances.

	CGA-H (5 runs)			CGA-E (1 run)	
	Average [€]	Best [€]	Run time [s]	Best [€]	Run time [s]
Instance 1	36109	36109	53	36109	2
Instance 2	44474	44474	130	44474	12
Instance 3	56829	56829	155	56829	51
Instance 4	79514	79514	229	79514	5156
Instance 5	174075	174245	443	-	-
Instance 6	299568	300014	657	-	-
Instance 7	612839	612860	2225	-	-
Instance 8	1022974	1025825	10661	-	-

Another observation from Table 5 is the robustness of CGA-H. For the first four instances, all five runs find the same solution or at least solutions of equal quality. Further, for Instance 4 to Instance 7, we see that the differences between the objective function value of the best-found solution and the average objective function value across the five runs are less than €500. Relative to the objective function values, this is less than 0.15%.

Using our solution methods, the only way to confirm if a solution from Table 5 is optimal is by comparing it to an optimal solution of the continuous relaxation of the master problem found by CGA-E. If both solutions have equal objective function values, we can guarantee optimality (as explained in Section 4.3). Therefore, we provide the solutions to the relaxed master problem for CGA-H and CGA-E in Table 6. The "Gap" column in this table presents differences in objective function values between the best-found solutions to the master problem and the best-found solutions to the relaxed master problem for all instances. For the best-found objective function value z_1^* to the master problem and the best-found objective function value z_2^* to the relaxed master problem, we define the gap as

$$\frac{2(z_2^* - z_1^*)}{z_2^* + z_1^*} \cdot 100\%.$$

In the first two instances, Table 6 reveals no difference in the solution quality between the master problem and its continuous relaxation when we use CGA-E. Hence, CGA-E and CGA-H find optimal solutions to Instance 1 and Instance 2.

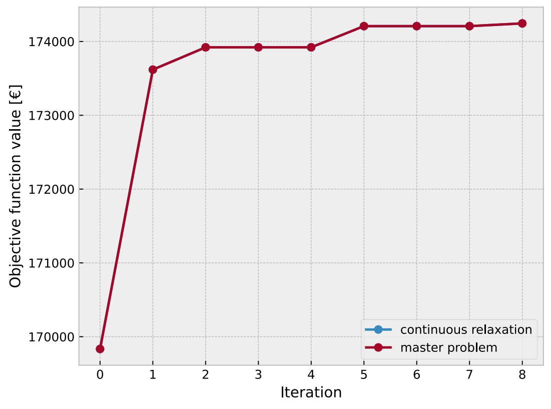
Table 6: Best observed solutions to the continuous relaxation of the master problem after meeting the column generation stopping criteria for both solution algorithms.

	CGA-H [€]	CGA-E [€]	Gap [%]
Instance 1	36109	36109	0.00
Instance 2	44474	44474	0.00
Instance 3	57071	57071	0.42
Instance 4	80215	80215	0.88
Instance 5	174245	-	0.00
Instance 6	300017	-	0.00
Instance 7	615526	-	0.43
Instance 8	1034616	-	0.85

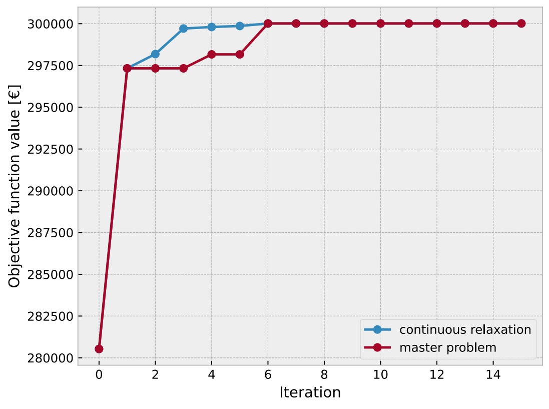
For Instance 3 and Instance 4, we are not able to prove optimality. However, since the heuristic methods make many random decisions and we still get the same solution quality across all runs using both algorithms, it is likely that the solutions are optimal.

Table 6 indicates a 0% gap for Instance 5 and Instance 6. However, these gaps depend on CGA-H's master problem relaxation, which is not proven to be upper bounds as the sub-problem are solved using ALNS. Consequently, we cannot guarantee that the best-found solutions by CGA-H to Instance 5 and Instance 6 are optimal. Nevertheless, the robustness of CGA-H across the first six instances gives a reason to believe that these solutions are at least close to optimal. Moreover, Figure 7a-b supports this belief. It shows minimal or no increase in objective function value during the last couple of iterations in the runs that led to the best-found solutions to Instance 5 and Intance 6.

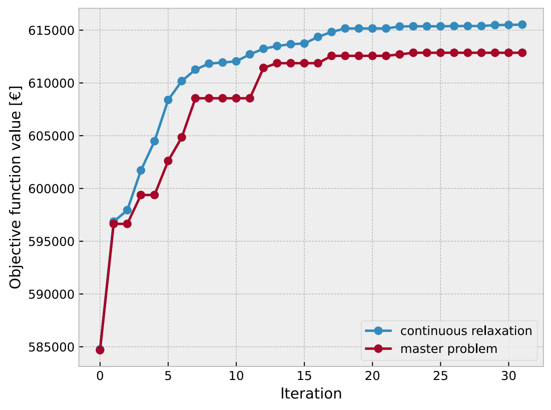
For the final two instances, it is less likely, but not impossible, that CGA-H's best-found solutions are optimal. The fact that the instances have 45 and 60 turbines, respectively, mean that there are an enormous number of feasible routes. There is a substantial chance that certain routes not generated by CGA-H could improve the solution. However, Figure 7c-d shows a strong indication of convergence in the objective function value of the relaxed master problem in both Instance 7 and Instance 8. Consequently, we consider the probability that they represent upper bounds for the master problem to be high. As indicated in Table 6, the gaps from these potential upper bounds are less than 1% for both instances. Therefore, we consider it likely that the best-found solutions to Instance 7 and Instance 8 have less than 1% lower objective function value than optimal solutions.



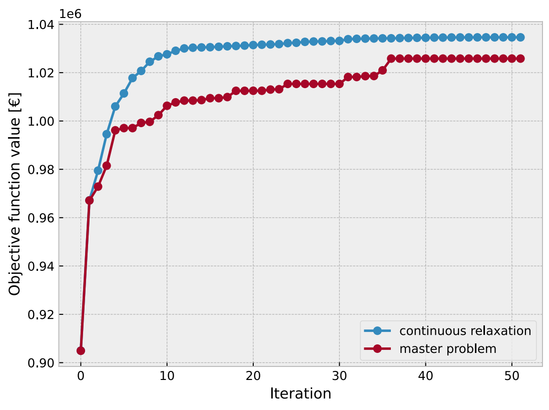
(a) Instance 5



(b) Instance 6



(c) Instance 7



(d) Instance 8

Figure 7: Depiction of the evolution of the objective function value for both the master problem and its relaxation, as obtained in the runs leading to the best-fund solutions by CGA-H for Instances 5-8

Based on the results from phase 1, we argue that the proposed ALNS manages to find good enough solutions to the sub-problem. The robustness of CGA-H indicates that the ALNS-framework does not make it terminate prematurely. This is further supported by the converging trends in Figure 7 and the fact that CGA-H finds equally good solutions as CGA-E to all Instances CGA-E is able to solve.

Table 7 presents the results from phase two of the experiment. CA finds equally good solutions as CGA-H to the two first instances. For the restoring instances, CGA-H outperforms CA both in terms of solution quality and robustness. These results tend to confirm that the column generation is operating as expected. The dual weights from the master problem play an effective role in generating useful routes in CGA-H.

Table 7: Comparison of five runs of CGA-H and CA across all instances.

	CGA-H (5 runs)		CA (5 runs)		
	Average	Best	Average	Best	Run time
Instance 1	36109	36109	36109	36109	53
Instance 2	44474	44474	43946	44474	130
Instance 3	56829	56829	56421	56829	155
Instance 4	79514	79514	78984	79118	229
Instance 5	174075	174245	171397	172132	443
Instance 6	299568	300014	290553	292917	657
Instance 7	612839	612860	591884	595167	2225
Instance 8	1022974	1025825	986233	987793	10661

Figure 8 presents the evolution of four aggregated measures in the CGA-H runs that resulted in the best-found solutions to the four most complex instances. The measures are defined as follows:

- *Vessel utilization* evaluates the degree to which the vessels are utilized. In scenarios where all vessels are in operation during all periods, the vessel utilization would be 100%. Conversely, if none of the vessels are utilized in any period, the utilization would drop to 0%.
- *Capacity utilization* quantifies the number of occupied seats in the vessels, summed over all selected routes, as a percentage of available seats.
- *Technician utilization* represents the percentage of available technicians that are deployed.
- Lastly, *turbines serviced* indicates the proportion of available maintenance tasks that are scheduled in the solution.

A brief examination of Figure 8 suggests that the number of turbines serviced and the capacity utilization positively correlate with the solution's quality. On the other hand, vessel utilization appears to decrease across all instances as the solution quality improves. In other words, effective solutions schedule many tasks on a minimal number of routes. Intuitively, this makes sense as the distances between the turbines are very short compared to the distance between the wind farm and the maintenance base.

The technician utilization seems to have little influence on the solution quality. However, the fact that both the vessel utilization and the technician utilization are low in all solutions suggests an imbalance in the

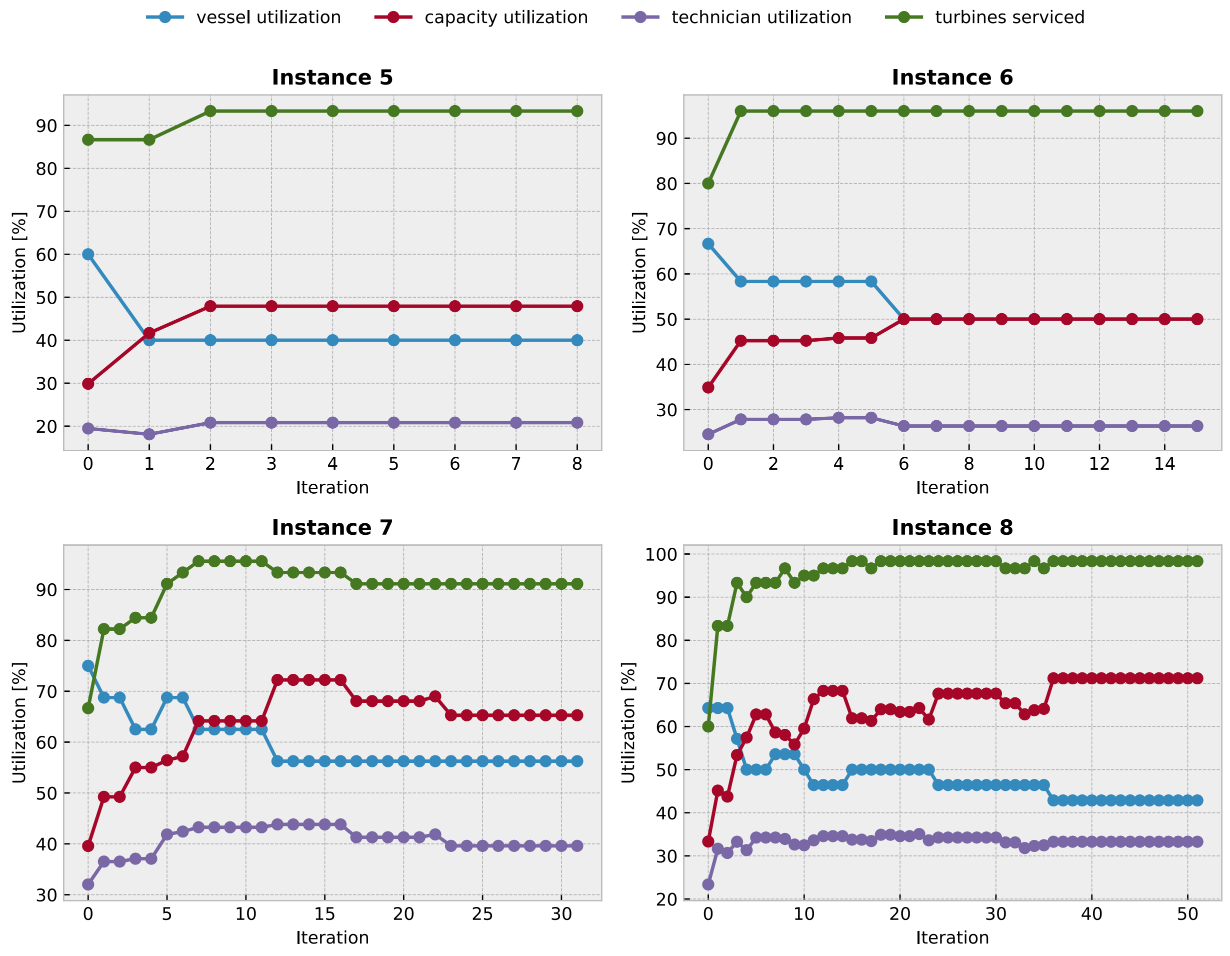


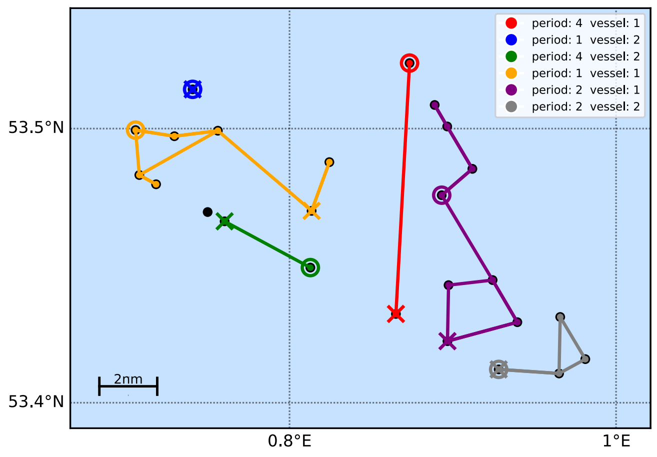
Figure 8: The evolution of the utilization measures in the CGA-H runs that found the best solution for Instance 5 to Instance 8

problem instances. There are unnecessarily many vessels and technicians available relative to the number of available maintenance tasks.

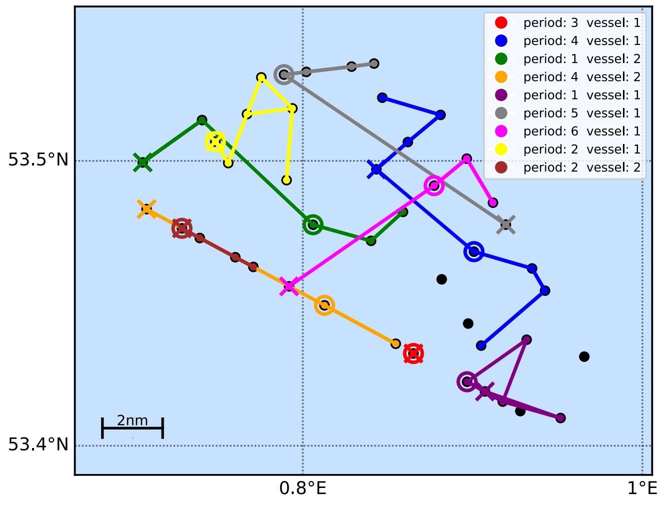
Another interesting observation in Figure 8 is that there exist solutions to Instance 7 that schedule more maintenance tasks than the best-found solution. This implies that the best-found solutions either prioritize maintenance tasks associated with higher expected revenue or they manage to execute the maintenance with significantly lower travel costs, making it economically viable to leave some maintenance tasks uncompleted.

Figure 9 offers a graphical representation of the best-found solutions for Instance 6 and Instance 7. It should be noted that it is impossible to completely deduce the routes from the figure, as some edges are traversed multiple times. Nonetheless, the figure highlights which turbines are serviced on the same route, and it provides insight into the movement patterns of the vessels.

Comparing the two solutions visualized in Figure 9, we observe that turbines located in close geographical proximity often appear to be serviced on the same routes. This is particularly evident with turbines grouped within a relatively small radius, such as the orange and grey routes in (a) and the pink and yellow routes in (b). Intuitively, this arrangement is logical as it enables parallel completion of more tasks.



(a) Instance 6



(b) Instance 7

Figure 9: A graphical visualization of the best-found solutions to Instance 6 and Instance 7. Cross symbols indicate the first turbine visited in a route, and the circles signify the last turbine that is visited before the vessel heads back to the base.

8 Concluding remarks

The number of operational offshore wind farms is expected to grow significantly in the forthcoming decades. Therefore, the importance of effective logistics for maintaining the turbines will become increasingly prominent. Improving the strategies and methods for maintenance planning will not only favor owners and investors economically but also contribute to producing more green electricity and lowering the electricity prices for consumers.

This thesis has considered the short-term routing and scheduling problem for offshore wind maintenance. We reviewed existing works addressing similar problems and proposed a new mathematical model that maximizes expected profit rather than minimizes costs. Furthermore, we have presented two variants of a solution method based on column generation. One solves the sub-problem using an exact approach, while the other is based on an adaptive large neighborhood search.

To test the proposed model and solution method, we have developed eight problem instances based on the Triton Knoll wind farm, which is situated on the east coast of the UK. A significant effort was put into creating realistic input data by tracking maintenance vessels and using historical wave data. However, due to little information, numerous assumptions were made to generate input for the maintenance tasks under consideration. As such, we invite future researchers with access to more comprehensive details about the maintenance tasks to expand upon these published instances.

In the experiments, we found that the proposed solution method can guarantee optimal solutions to the two simplest instances, and we have reason to believe that the best-found solutions to the other six instances are optimal or near-optimal. The experiments also revealed that the proposed ALNS framework efficiently provides good enough solutions to the sub-problem and that the column generation operates as expected.

For future work, we suggest challenging the column generation approach with an adaptive large neighborhood search that searches for complete solutions rather than solving the sub-problem. However, such a heuristic method probably needs more complex operators that take advantage of problem-specific information to compensate for the dual weights in the column generation.

Another possibility for further research is determining the expected revenues, which we have assumed as known in this thesis. It could be interesting to systematically combine information about the turbines' current conditions, the expected electricity production, and price forecasts, to appropriate values for these. In the developed instances, we approximated revenue inputs with some of these factors in mind. However, there is unquestionably a need for a more systematic approach for our model to be practically applicable.

In other words, there is potential for further improvements and additions to the proposed model and solution method. With that being said, this thesis has investigated an alternative approach to the existing works on the field. Exploring a wide range of models for offshore wind maintenance is essential in finding the most effective logistics procedures. One of the most significant contributions from this thesis is to participate in this exploration, which hopefully is a step towards more reliable green electricity for the world's increasing population.

References

[1] W. Musial, P. Beiter, P. Spitsen, J. Nunemaker, and V. Gevorgian, "2020 Offshore Wind Technology Data Update," Tech. Rep. NREL/TP-5000-78471, National Renewable Energy Lab (NREL), 2021.

[2] Workboat Association, "The workboat association homepage," 2023. Accessed: 2023-05-19.

[3] United Nations Framework Convention on Climate Change, "Paris agreement." `https://unfccc.int/documents/184656`, 2015.

[4] United Nations, "Sustainable development goal 7: Affordable and clean energy." `https://sdgs.un.org/goals/goal7`, 2015.

[5] Z. Ren, A. S. Verma, Y. Li, J. J. Teuwen, and Z. Jiang, "Offshore wind turbine operations and maintenance: A state-of-the-art review," *Renewable and Sustainable Energy Reviews*, vol. 144, p. 110886, 2021.

[6] A. Karyotakis and R. Bucknall, "Planned intervention as a maintenance and repair strategy for offshore wind turbines," *Journal of Marine Engineering & Technology*, vol. 9, no. 1, pp. 27–35, 2010.

[7] K. Sivalingam, M. Sepulveda, M. Spring, and P. Davies, "A Review and Methodology Development for Remaining Useful Life Prediction of Offshore Fixed and Floating Wind turbine Power Converter with Digital Twin Technology Perspective," in *2018 2nd International Conference on Green Energy and Applications (ICGEA)*, pp. 197–204, 2018.

[8] É. Thomas, É. Levrat, and B. Iung, "Overview on opportunistic maintenance," *IFAC Proceedings Volumes*, vol. 41, no. 3, pp. 245–250, 2008. 9th IFAC Workshop on Intelligent Manufacturing Systems.

[9] L. Dai, M. Stålhane, and I. B. Utne, "Routing and scheduling of maintenance fleet for offshore wind farms," *Wind Engineering*, vol. 39, no. 1, pp. 15–30, 2015.

[10] C. A. Irawan, D. Ouelhadj, D. Jones, M. Stålhane, and I. B. Sperstad, "Optimisation of maintenance routing and scheduling for offshore wind farms," *European Journal of Operational Research*, vol. 256, no. 1, pp. 76–89, 2017.

[11] C. Stock-Williams and S. K. Swamy, "Automated daily maintenance planning for offshore wind farms," *Renewable Energy*, vol. 133, pp. 1393–1403, 2019.

[12] N. T. Raknes, K. Ødeskaug, M. Stålhane, and L. M. Hvattum, "Scheduling of Maintenance Tasks and Routing of a Joint Vessel Fleet for Multiple Offshore Wind Farms," *Journal of Marine Science and Engineering*, vol. 5, no. 1, 2017.

[13] T. A. T. Nguyen and S.-Y. Chou, “Maintenance strategy selection for improving cost-effectiveness of offshore wind systems,” *Energy Conversion and Management*, vol. 157, pp. 86–95, 2018.

[14] C. Stock-Williams and S. K. Swamy, “Automated daily maintenance planning for offshore wind farms,” *Renewable Energy*, vol. 133, pp. 1393–1403, 2019.

[15] A. Gutierrez-Alcoba, E. Hendrix, G. Ortega, E. Halvorsen-Weare, and D. Haugland, “On offshore wind farm maintenance scheduling for decision support on vessel fleet composition,” *European Journal of Operational Research*, vol. 279, no. 1, pp. 124–131, 2019.

[16] M. Yildirim, N. Z. Gebraeel, and X. A. Sun, “Integrated Predictive Analytics and Optimization for Opportunistic Maintenance and Operations in Wind Farms,” *IEEE Transactions on Power Systems*, vol. 32, no. 6, pp. 4319–4328, 2017.

[17] S. Zhong, A. A. Pantelous, M. Beer, and J. Zhou, “Constrained non-linear multi-objective optimisation of preventive maintenance scheduling for offshore wind farms,” *Mechanical Systems and Signal Processing*, vol. 104, pp. 347–369, 2018.

[18] R. J. Vanderbei *et al.*, *Linear programming.* Springer, 2020.

[19] G. B. Dantzig, “Application of the simplex method to a transportation problem,” *Activity analysis and production and allocation*, 1951.

[20] N. Karmarkar, “A new polynomial-time algorithm for linear programming,” in *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pp. 302–311, 1984.

[21] G. B. Dantzig and J. H. Ramser, “The Truck Dispatching Problem,” *Management Science*, vol. 6, no. 1, pp. 80–91, 1959.

[22] L. R. Ford Jr and D. R. Fulkerson, “A suggested computation for maximal multi-commodity network flows,” *Management Science*, vol. 5, no. 1, pp. 97–101, 1958.

[23] E. Choi and D.-W. Tcha, “A column generation approach to the heterogeneous fleet vehicle routing problem,” *Computers & Operations Research*, vol. 34, no. 7, pp. 2080–2095, 2007.

[24] Y. Dumas, J. Desrosiers, and F. Soumis, “The pickup and delivery problem with time windows,” *European journal of operational research*, vol. 54, no. 1, pp. 7–22, 1991.

[25] L. A. Wolsey, *Integer Programming.* John Wiley & Sons, 2 ed., 2021.

[26] E.-G. Talbi, *Metaheuristics: from design to implementation.* John Wiley & Sons, 2009.

[27] J.-L. Ambite, "Local search," 2001. https://www.cs.cmu.edu/afs/cs/project/jair/pub/volume15/ambite01a-html/node9.html Accessed: 2023-02-19.

[28] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[29] S. Ropke and D. Pisinger, "An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows," *Transportation Science*, vol. 40, pp. 455–472, 11 2006.

[30] S. Ropke and D. Pisinger, "A unified heuristic for a large class of Vehicle Routing Problems with Backhauls," *European Journal of Operational Research*, vol. 171, no. 3, pp. 750–775, 2006.

[31] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2023. https://www.gurobi.com Accessed: 2023-04-19.

[32] A. Laugaland, "Problem instances for an optimization model for short-term routing and scheduling of offshore wind maintenance." https://github.com/aksellaug/OWF-maintenance-problem-instances, 2023.

[33] C-MAP, "C-map: Nautical charts," 2023. https://appchart.c-map.com/redirect-route.html?id=spbsDV5Bl&key=8D4RQyEuJlmSUSmT4eLVEhQdPRvm13c3 Accessed: 2023-03-29.

[34] RWE Renewables, "About triton knoll," 2023. https://www.tritonknoll.co.uk/about-triton-knoll Accessed: 2023-03-13.

[35] Northern Offshore Services, "Live map," 2023. https://www.marinetraffic.com/en/ais/home/centerx:1.5/centery:53.4/zoom:9 Accessed: 2023-03-20 to 2023-03-28.

[36] Northern Offshore Services, "M/v marker," 2023. https://n-o-s.eu/the-fleet/m-v-maker/ Accessed: 2023-03-16.

[37] Maritime Craft Services, "Mcs swat 2," 2023. https://www.maritimecraft.co.uk/files/MCS-SWATH-2.pdf Accessed: 2023-03-16.

[38] Strategic Marine, "Stratcat27," 2023. https://www.strategicmarine.com/vessel/stratcat-27-crew-transfer-vessel/ Accessed: 2023-03-16.

[39] CEFAS, "Cefas wavenet," 2023. https://wavenet.cefas.co.uk/Map Accessed: 2023-01-29.

[40] Numba Development Team, "Numba: A high performance python compiler," 2021. https://numba.pydata.org/ Accessed: 2023-04-19.

Appendix A Symbols from the mathematical model

Sets

$\mathcal{V}$	Set of vessels
$\mathcal{T}$	Set of time periods
$\mathcal{W}$	Set of turbines
$\mathcal{R}$	Set of feasible routes
$\mathcal{R}^{ALL}$	Set of all feasible routes
$\mathcal{R}_{vt}$	Set of a feasible routes for vessel v at time period t
$\mathcal{N}$	Set of all nodes
$\mathcal{N}^D$	Set of delivery nodes
$\mathcal{N}^P$	Set of pickup nodes
$\mathcal{N}^{OD}$	Set containing the pick-up node and the delivery node
$\mathcal{A}$	Set of arcs
$\mathcal{B}$	Set of technician types

Parameters

$\tilde{T}_i$	Time required for completing the task corresponding to delivery node i
C^T_{ijv}	Cost of traveling from node i to node j by vessel v
D^T_{br}	The required number of technicians of type b for completing route r
F_{bi}	Demand for technicians of type b at node i
G_{bt}	Available technicians of type b in time period t
I_{wr}	If turbine w is maintained in route r or not
K_v	Maximal number of technicians that can be onboard vessel v
P_r	Profit for completing route r

R_{it}	Revenue for completing maintenance at the turbine corresponding to delivery node i in time period t
S	Safety distance, i.e., how far a vessel is allowed to travel away from technicians that are left on a turbine
T^P_{vt}	Total time available for vessel v in time period t
T_{ijv}	Minimum travel time from node i to node j using vessel v

Variables

x_r	Decision variable for selecting route r or not
λ_{vt}	Dual variable corresponding to the primal constraint (5)
μ_{bt}	Dual variable corresponding to primal constraints (6)
ρ_w	Dual variable corresponding to primal constraint (7)
y_{ij}	Decision variable for traveling directly from node i to node j or not
z_{bi}	Integer variable describing the the number of technicians of type b onboard when leaving node i
q_i	Continuous variable describing the time when leaving node i
m_{ij}	Binary variable for pairs of tasks that cannot be completed in parallel