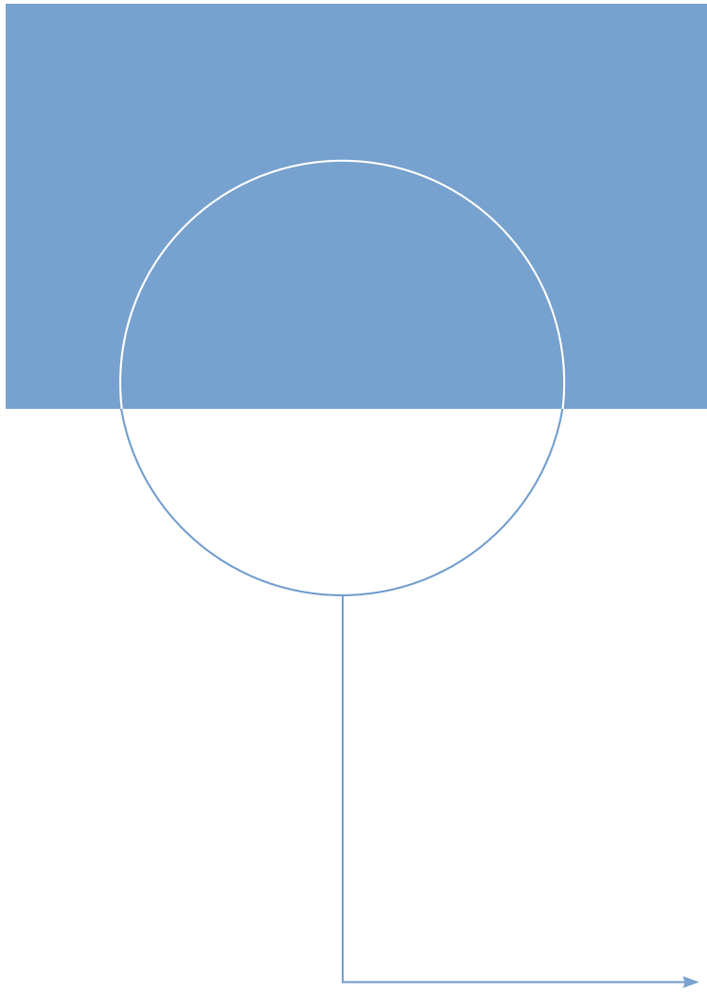


Master's thesis

2020

Eivind Oppedal, Kristian Løkka Thorstein

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

Eivind Oppedal
Kristian Løkka Thorstein

An Extended Model and a New Matheuristic for the Offshore Helicopter Routing Problem with Split Deliveries

July 2020

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

NTNU

Norwegian University of
Science and Technology

An Extended Model and a New Matheuristic for the Offshore Helicopter Routing Problem with Split Deliveries

Eivind Oppedal
Kristian Løkka Thorstein

Industrial Economics and Technology Management
Submission date: July 2020
Supervisor: Magnus Stålhane

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

Preface

This master's thesis concludes our Master of Science degree in Managerial Economics and Operations Research at the Norwegian University of Science and Technology (NTNU). This work is a continuation of our project report, Oppedal and Thorstein (2019), for the subject "TIØ4500 - Managerial Economics and Operations Research, Specialization Project".

This thesis has been carried out in collaboration with Tieto Oil & Gas, and we would like to thank Gaute Messel Nafstad from Tieto for providing data and insight on the problem. We would also like to give special thanks to our supervisor, Professor Magnus Stålhane, for his excellent feedback and guidance throughout the semester.

Trondheim, July 1, 2020
Eivind Oppedal, Kristian Løkka Thorstein

Summary

The oil and gas industry is the biggest industry in Norway and is expected to remain important for several decades. The extraction of petroleum takes place on the Norwegian continental shelf, and offshore installations are vital for the production. In this thesis, the problem of transporting working personnel to and from these installations with the use of helicopters is studied.

The requests for transportation are aggregated into orders, which consist of personnel going to or from the same installation at the same time. A heterogeneous fleet of helicopters stationed at multiple heliports is used for the transportation. The sequence of heliports and installations visited by a helicopter during a day is called a flight sequence. Each helicopter is allowed to use multiple heliports, but must start and end its flight sequence at the same heliport. Which heliports to use for pickup/delivery of each order is optional. When replacing the installations in a flight sequence with the orders handled at each installation, we get what is defined as a route. An order can be split between multiple helicopters. The objective is to minimise the total cost of chartering and using the helicopters, while completing all orders and satisfying all regulations and operational restrictions.

A literature review on the general pickup and delivery problem and the offshore helicopter routing problem is presented, with focus on relevant extensions and heuristic solution methods. Allowing helicopters to use multiple heliports and making it optional which heliports to pickup/deliver each order at, are new extensions that are not found in the literature. A complete mathematical formulation of the problem is provided by an arc-flow model. In addition, a new matheuristic is proposed as a solution method for the problem. A decomposition approach is used for the matheuristic, and the problem is decomposed into three parts. In the first, changes are made to flight sequences from the current solution, and in the second a labeling algorithm is used to generate routes from the changed flight sequences. Finally, a mixed integer programming model is solved to find the best combination of routes and passengers. These three parts make up one iteration, and the matheuristic performs as many iterations as possible until the termination criterion is met.

The arc-flow model is only able to find the optimal solution for instances with up to eight orders included, and the matheuristic solves almost all of the same instances to optimality. Tests show that the matheurisitic consistently finds approximately the same solutions for each run of instances with up to 40 orders. Including the new extensions reduces, on average, the objective value by approximately 9 % and the number of helicopters by 0.70. Splitting of orders, on the other hand, is found to have little impact on the total cost. Based on the results, the new extensions could be beneficial additions to the offshore helicopter routing problem.

Sammendrag

Olje- og gassindustrien er den største industrien i Norge, og det er forventet at den forblir viktig i flere tiår fremover. Utvinningen av petroleum foregår på den norske kontinentalsokkelen, og offshore-installasjoner er sentrale for produksjonen. I denne masteroppgaven blir transport av personell til og fra disse installasjonene ved hjelp av helikopter studert.

Forespørsler om transport av personell er samlet i ordrer, som er definert som personell som skal til eller fra samme installasjon på samme tidspunkt. En heterogen flåte med helikoptre stasjonert ved flere heliporter brukes til transporten. Sekvensen av heliporter og installasjoner som besøkes av et helikopter i løpet av en dag kalles for en flyvesekvens. Hvert helikopter kan benytte flere heliporter, men må starte og slutte sin flyvesekvens på den samme heliporten. Hvilke heliporter som brukes til å hente/levere ordre er valgfritt. Når installasjonene i en flyvesekvens byttes ut med ordrene som håndteres på hver installasjon, får vi det som kalles en rute. En ordre kan splittes mellom flere helikoptre. Målet er å minimere de totale kostnadene av å leie og bruke helikoptrene, samtidig som alle ordre fullføres og alle reguleringer og restriksjoner overholdes.

En litteraturstudie for det generelle pickup and delivery-problemet og offshore helikoptertransport blir presentert, med fokus på relevante utvidelser og heuristiske løsningsmetoder. Det å tillate at helikoptre benytter flere heliporter og gjøre det valgfritt hvilke heliporter som skal brukes til å hente/levere hver ordre, er nye utvidelser som ikke er funnet i litteraturen. En komplett matematisk beskrivelse av problemet er gitt av en arc-flow-modell. I tillegg er en ny matheuristikk foreslått som en løsningsmetode for problemet. En dekomponeringsmetode er brukt for matheuristikken, og problemet er dekomponert i tre deler. I den første endres flyvesekvensene fra den nåværende løsningen, og i den andre blir en labeling-algoritme brukt til å generere ruter fra de endrede flyvesekvensene. Til slutt løses et blandet heltallsproblem for å finne den beste kombinasjonen av ruter og passasjerer. Disse tre delene utgjør en iterasjon, og matheuristikken utfører så mange iterasjoner som mulig før termineringskriteriet er nådd.

Arc-flow-modellen er kun i stand til å finne optimal løsning for instanser med opptil åtte ordre inkludert, og matheuristikken løser nesten alle de samme instansene til optimalitet. Tester viser at matheuristikken konsekvent finner tilnærmet de samme løsningene for hver kjøring av instanser med opptil 40 ordre. Inkludering av de nye utvidelsene reduserer gjennomsnittlig objektivverdi med omtrent 9 % og det totale antallet helikoptre som trengs med 0,70. Splitting av ordre viser seg derimot å ha liten innvirkning på objektivverdien. Basert på resultatene virker det som de nye utvidelsene kan være nyttige tilskudd ved planlegging av offshore helikoptertransport.

Contents

List of Figures

List of Tables

1 Introduction

The oil and gas industry is the biggest and most important industry in Norway, and is the largest sector in terms of export value, government revenues, investments, and value added (Norsk Petroleum, 2020d). In 2017, around 225 000 people in Norway were directly or indirectly employed by the oil and gas sector (Norsk Petroleum, 2020a). This made up approximately 8.5 % of the total workforce (Statistics Norway, 2020). Although the production of oil and gas in Norway has declined slightly since the peak in 2004, there is still a high level of activity on the Norwegian continental shelf. At the end of 2019, there were 87 fields in production and another 13 under development (Norsk Petroleum, 2020b). Only 48 % of the estimated total recoverable resources have been produced and sold, and the activity on the Norwegian continental shelf is expected to remain high for the next 50 years, with an increase in production in the first five years (Norsk Petroleum, 2020c).

The petroleum production takes place offshore, mostly in the North Sea. Offshore installations such as oil platforms, rigs, and boats are used for well drilling, extraction, and storage of the petroleum, and are consequentially a vital part of the production. These installations are in constant operation, and unexpected stops in production are extremely costly. All installations are therefore in need of the right personnel present at all times. The personnel normally work offshore for two to four weeks, followed by two to four weeks of time off. Because of this, there is a demand for transportation of personnel between the offshore installations and the mainland. This transportation is mainly conducted by helicopters. On the mainland, heliports located at airports on the coast of Norway are used as the starting and ending point of the helicopter trips. In 2018, more than 380 000 people were transported offshore, according to data from Tieto. This is more than 1000 people per day and corresponds to more than 50 fully loaded helicopters.

The chartering and use of the helicopters are expensive, and the difference between good and poor planning can have a large impact on transportation cost. A reduction in distance flown would also be favourable from an environmental point of view as it leads to lower emissions. Planning the transportation is a complex task, both because of the great number of people transported every day and all the rules and regulations regarding offshore helicopter transportation. It is therefore expected that the use of optimisation and mathematical programming models can be helpful to achieve more effective planning.

Transportation to offshore installations with the use of helicopters, called the offshore helicopter routing problem (HRP), has been studied in the literature to some extent. Some studies have a tactical and strategical objective, in the form of finding optimal locations for airfields or the optimal composition of the helicopter

fleet, but the majority of studies have been performed from an operational point of view. Within the literature on operational helicopter planning, the focus is mostly on cost-effective planning or safety, and mostly related to either the North Sea or Brazil's offshore oil fields. Common restrictions in the operational HRP include seat and fuel capacity and a set time for when the transportation has to be done, but there are many other possible restrictions as well. Even though several aspects of the operational helicopter routing problem are well studied, there are still aspects that are of great interest to study further.

The main purpose and contribution of this thesis is to study the benefit of including two new extensions to the HRP that have not previously been discussed in the literature. The first is the option to choose which heliports personnel should be picked up at, or delivered to, and the second allows helicopters to visit any heliport. By adding these extensions we make the problem more complex and harder to solve, and the overall goal is therefore to see whether these extensions decrease the operational costs sufficiently to warrant solving a more complex problem. An arc-flow model with the new extensions is presented as a complete mathematical formulation of the problem. To solve the problem with a realistic number of transportation requests, we have developed a new matheuristic. Results from testing of the matheuristic, both with and without the new extensions, can be used to gain insight to which aspects of the problem that are beneficial to include and how well the matheuristic performs.

The rest of the thesis is outlined as follows: a description of the problem is given in Chapter 2. In Chapter 3 we present a review of the relevant literature on the general pickup and delivery problem, the offshore helicopter routing problem, and matheuristics. The arc-flow model for the problem, together with possible preprocessing and symmetry breaking, is presented in Chapter 4. In Chapter 5 we propose a new matheuristic that can be used to solve the problem both faster and with more transportation requests than with the arc-flow model. A computational study is performed in Chapter 6 to test the performance of the matheuristic, and study the benefit of adding the new extensions to the helicopter routing problem. Our concluding remarks are found in Chapter 7, and in Chapter 8 we present suggestions for future research.

2 | Problem Description

The problem studied in this thesis is the planning of transportation of personnel between the mainland of Norway and offshore installations, with the use of helicopters. The objective is to minimise the total cost of transportation for a given day while satisfying both the demand for transportation and all regulations.

An installation is defined as any platform, rig, ship, or other offshore installation with a helipad and a need for transportation of personnel with the use of helicopters. On the mainland, the personnel are transported to or from heliports located at airports on the coast of Norway. The personnel are organised into orders, which are defined as groups of personnel to be transported either to or from the same installation at the same time. Each order can be completed by more than one helicopter, and if so, it is called splitting an order.

A heterogeneous fleet of helicopters is used for the transportation. Each helicopter has a maximum number of seats which limits the number of passengers. For every helicopter there is a set cruising speed and an associated fuel consumption, which together with the fuel tank capacity gives a maximum range before having to refuel. Fuel levels must never drop below a predefined safety margin. Costs related to the use of a helicopter are divided in a fixed chartering cost and a variable cost dependent on the distance flown.

A trip is defined as a sequence of orders and heliports, starting with a heliport, continuing with one or more orders located at one or more installations, and ending at a heliport. Trips can start and end at two different heliports. Throughout a day a helicopter can perform several trips, and the sequence of trips in a day is defined as a helicopter's route. A route must start and end at the same heliport, the helicopter's home heliport. All routes have a related flight sequence, which is defined as the sequence of heliports and installations in a route. A flight sequence is therefore similar to a route, but consists of installations instead of orders.

The orders must be served within specified time windows, defined by an earliest and latest starting time of service at the installation. The offshore installations can only be visited by one helicopter at any given time, whereas the heliports are assumed to have an unlimited capacity for helicopters. A helicopter visiting an installation has to stay for a fixed amount of time and must leave once this time is up. Visiting an installation without performing either a pickup or a delivery is not allowed. Helicopters must travel directly to the next destination and are not allowed to wait during a trip by hovering next to an installation. Personnel can only be transported by a helicopter for a limited number of legs, and must remain with their assigned helicopter throughout the whole transportation. After finishing a trip, the helicopter must be empty. This means that all orders in a trip must be completed before, or when, the helicopter arrives at a heliport. Before starting

a new trip the helicopters must wait at the heliport for a certain amount of time, which includes time for safety inspections and refuelling.

The objective is to minimise the daily total cost associated with completing all orders to and from offshore installations, in accordance with all regulations. Information about the helicopters, offshore installations and heliports is given. The personnel use conventional means of transportation to get to the heliports, and how they get there is not considered a part of this problem. Which heliports to use for pickup or delivery of an order is not given, and has to be decided. Furthermore, decisions have to be made regarding which helicopters to use and the distribution of passengers in an order for the different helicopters. The route of each helicopter, and thereby the time schedule for departures and arrivals, has to be decided as well.

3 | Literature Review

In this chapter we present a literature review for the problem studied in this thesis. We start by studying different variants, extensions, and solution methods for the general pickup and delivery problem in Section 3.1. In Section 3.2 we present literature on helicopter transport planning, with focus on solution methods and the use of the extensions from Section 3.1. Section 3.3 gives a brief overview of the literature on matheuristics, the solution method used in this thesis. Finally, in Section 3.4 we discuss how this thesis is positioned with regards to the literature, and its contributions to the field of study. The general pickup and delivery problem is a well-studied problem, and it is therefore necessary to limit our literature review on this problem type. For the extensions, the review is largely based on surveys to find the most relevant papers. When it comes to the offshore helicopter routing problem, we include all literature that is considered relevant for the problem studied in this thesis, as this is a less studied field.

3.1 The General Pickup and Delivery Problem

The problem in this thesis, described in the Chapter 2, falls under the category of the general pickup and delivery problem (GPDP), which is described by Savelsbergh and Sol (1995). The GPDP is a generalisation of three closely related problems, the vehicle routing problem (VRP), the pickup and delivery problem (PDP), and the dial-a-ride problem (DARP). The purpose of the GPDP is to construct routes for vehicles to satisfy transportation requests. The transportation requests generally consist of:

- A load of some size to be transported.
- One or more origins where the load is to be picked up.
- One or more destinations where the load is to be delivered.

Transportation requests can be static, which means that all requests are known before the routes are constructed, or dynamic, where some requests are known before the routes are constructed, and new requests become available during the execution of the routes (Savelsbergh and Sol, 1995). In this thesis, only static problems are considered.

This section presents the PDP, VRP, and DARP, and some of their extensions, as well as solution methods used. For the problem in this thesis, relevant extensions include multiple depots, a heterogeneous fleet, multiple trips for each vehicle, split deliveries, and the use of time windows. Literature on the extensions is gathered from papers on all three problem types, but as the VRP is the most studied of the three, the majority of the papers are about the VRP.

3.1.1 The pickup and delivery problem

In the PDP, all transportation requests specify a single origin and a single destination (Savelsbergh and Sol, 1995). It is defined by Berbeglia et al. (2007) as a "class of vehicle routing problems in which objects or people have to be collected and distributed". They further classify three PDP structure types: many-to-many (M-M) problems, one-to-many-to-one (1-M-1) problems and one-to-one (1-1) problems. In the M-M problem, any node can be both the pickup node and delivery node for any commodity transported. In 1-M-1 problem, the commodities from the depot are destined to the customers while the commodities from the customer are destined to the depot. In the last of the three structures, 1-1 problems, there is only one origin and one destination for each commodity. In addition to the classification of structures, Berbeglia et al. (2007) provide relevant literature on multiple problem types within each structure.

3.1.2 The vehicle routing problem

Laporte (1992) describes the VRP as "the problem of designing optimal delivery or collection routes from one or several depots to a number of geographically scattered cities or customers, subject to side constraints". It differs from the PDP in that all origins or destinations are located at the depot. In its basic version, some commodity is transported from a central depot to customers by a homogeneous fleet of vehicles, and every customer is served by exactly one vehicle. All vehicles must return to the depot at the end of the route.

The VRP with pickups and deliveries (VRPPD) can be described by the structure classification scheme from Berbeglia et al. (2007) as a 1-1 problem with one pickup and one delivery. Another variant of this is the VRP with simultaneous pickup and delivery (VRPSPD), in which customers require both the pickup and delivery of commodities at the same visit. The VRPSPD is defined as a "multi-vehicle Hamiltonian 1-M-1 PDP with combined demands" (Berbeglia et al., 2007), where "Hamiltonian" means that all customers are visited once, with both pickup and delivery performed during the visit.

3.1.3 The dial-a-ride problem

The DARP is in many ways equal to the VRPPD, with the main difference between them being the transportation of people instead of objects. According to Cordeau and Laporte (2007), vehicle capacity is usually more constraining in the DARP than in the VRPPD because of this. When transporting people, it is also desirable to reduce user inconvenience, and that has to be weighted against minimising the routing cost. Cordeau and Laporte (2007) provide several formulations of the DARP as well as literature on algorithms for both the single-vehicle and multi-vehicle DARP. These formulations want to minimise the cost of satisfying transportation requests, while Parragh et al. (2014) study the DARP with profits, where trips are chosen to maximise profit.

3.1.4 Multiple depots

With multiple depots in the GPDP, the depot that serves a customer is chosen among multiple possible depots. In some instances, when the customers are clustered around the depots such that they realistically only can be served by the closest depot, this problem can be solved as several single depot problems, but otherwise we need a model that takes into account multiple depots. Montoya-Torres et al. (2015) provide a survey on the multi-depot VRP (MDVRP) where both single-objective and multi-objective MDVRPs are covered.

A formal definition of the MDVRP is given by Renaud et al. (1996). Two different formulations are presented in Contardo and Martinelli (2014), a vehicle-flow formulation and a set-partitioning formulation. An exact algorithm with column generation and several strengthening inequalities is used to solve the problem. A third formulation of the MDVRP is provided by Baldacci et al. (2011), who formulates it as a period VRP and solves it to optimality.

The formal definition from Renaud et al. (1996) states that all vehicle routes must start and end at the same depot, but not all MDVRPs have this requirement. Kek et al. (2008) describe a variant of the problem in which vehicles have to start and end at a depot, but the start and end depots can be different. A similar approach is taken by J. Li et al. (2016), who studied the MDVRP with time windows and shared depot resources, where the vehicles end their routes in the depot that is the closest to their last customer.

3.1.5 Heterogeneous fleet

In the traditional VRP, PDP, and DARP, the fleet of vehicles is homogeneous, which means that all vehicles are identical. This is not the case in many real world problems, where the vehicles available can differ in terms of capacity, range, speed, and cost of use. Koç et al. (2016) present a review of the literature on the heterogeneous VRP (HVRP), and discuss mainly two different types of the HVRP: the Fleet Size and Mix (FSM) VRP and the Heterogeneous Fixed Fleet (HF) VRP. The objective in both the FSM and the HF is to minimise the sum of fixed and variable vehicle costs. Golden et al. (1984) present the FSM as the problem of deciding the fleet of vehicles and the routing of the fleet, while in the HF the fleet is predetermined, as described by F. Li et al. (2007). In the literature review, Koç et al. (2016) also suggest three formulations for the HVRP, a three-index formulation from Baldacci et al. (2008), a two-index formulation from Baldacci et al. (2009), and a set partitioning formulation from Baldacci and Mingozzi (2008).

Salhi et al. (2014) propose a four-index formulation of the heterogeneous fleet VRP with multiple depots (MDHFVRP), and add several constraints to tighten the formulation, as well as possible modifications for slightly different problems. Among these modifications is the possibility of removing one index if the vehicle does not have to return to the depot it originated from. Xu et al. (2012) describe the same type of prob-

lem. Both papers use a variable neighbourhood search (VNS) algorithm as the solution method. A similar problem is solved by Dayarian et al. (2015), a MDHFVRP with time windows derived from real-life milk collection, but it is solved using a branch-and-price algorithm.

3.1.6 Multiple trips

In some cases of routing problems there is only a single vehicle available, and it may be necessary to allow multiple trips in order to satisfy demand. Allowing multiple trips can also provide improved results in problems with several vehicles. Because of this, problems that include multiple trips for vehicles have been studied. Cattaruzza et al. (2016) present an extensive survey on the multi-trip VRP (MTVRP).

The MTVRP is similar to the standard VRP, with the exception being that it allows for several trips. Similar to routes in the standard VRP, every trip must start and end at the depot. The actual formulations of the MTVRP varies in different papers, with the number of indices used ranging from 2 to 4. A common approach is a 4-index formulation (Cattaruzza et al., 2016), with two indices for arcs, one for vehicles and one for trips. Both Gribkovskaia et al. (2006) and Alonso et al. (2008) make use of this type of formulation, but they extend the planning horizon from one to several days.

Although the approach described above is common, other ways of formulating the MTVRP are used as well (Cattaruzza et al., 2016). Aghezzaf et al. (2006) present a formulation of what they call the inventory routing problem, in which the use of multiple trips is handled without a trip index. They use a heuristic to find trips that can be merged into routes, and then use column generation to find a solution. Other 3-index formulations highlighted in Cattaruzza et al. (2016) are from Azi et al. (2010) and Hernandez et al. (2014), who are able to remove the vehicle index by adding new variables that handle time windows and the vehicle fleet size. Like Aghezzaf et al. (2006), they use column generation as a part of their solution method, but both Azi et al. (2010) and Hernandez et al. (2014) combine it with a branch-and-price scheme to find the exact solution to the problem.

A paper that does not provide a mathematical formulation, but instead focuses on a heuristic solution approach for the MTVRP, is François et al. (2016). It describes two adaptive large neighbourhood search (ALNS) algorithms. The first ALNS algorithm is combined with bin packing, which is a common approach for solving MTVRPs. The other algorithm includes local search operators specifically designed for the inclusion of multiple trips to the VRP. When using the multi-trip operators, the authors consider trips as part of a "giant tour" in which trips are no longer treated as separate, but viewed as part of a route. The depots separating trips are treated as if they were customers. They find that both the ALNS with bin packing and the ALNS with multi-trip operators yield good results, although the one with bin packing is slightly better than the one with multi-trip operators. Nevertheless, they believe that because the multi-trip operators modify the route more slowly, they might be extra useful "in more constrained problems, when the feasibility

is highly dependent on customer sequences" (François et al., 2016).

3.1.7 Split delivery

It is often sufficient and convenient for the customers to be served by only one vehicle, but sometimes this is not possible due to capacity constraints of the vehicles, or better solutions can be found when using more than one. To allow for more than one vehicle to serve the customers, a model that allows for split deliveries has to be used. Dror et al. (1994) define the split delivery VRP (SDVRP) as a VRP in which the constraint that each customer must be served by exactly one vehicle is relaxed. They also provide a mathematical formulation for the problem, together with valid inequalities and an exact solution method. A survey by Archetti and Speranza (2012) further provides a problem formulation of the SDVRP, as well as known properties of the problem and a review of literature on the SDVRP.

Desaulniers (2009) discusses the SDVRP with time windows and provides a mathematical model that is solved using a Dantzig-Wolfe decomposition and branch-and-price-and-cut. Another variant, the VRP with split deliveries and pickups, is studied in Mitra (2008). The vehicles in this problem are not allowed more than one trip and the pickup and delivery demands may exceed the vehicle capacity, thus requiring split pickup and delivery. A heuristic based on clustering of customers, where each cluster is served by one vehicle, is used to provide a good initial solution to the problem. Parragh et al. (2014) present a DARP with split deliveries and time windows, solved by using a branch-and-price algorithm and VNS. The VNS algorithm consists of a shaking phase with three different types of neighbourhoods, created by removing and inserting order pairs, and local searches. Split deliveries are incorporated when inserting pickup and delivery pairs by allowing orders to be inserted and handled partially in several routes. They find that the splitting is most useful when customer locations are clustered, and that when the customer locations are randomly spread out, the split request property of the problem rarely leads to improved solutions.

The PDP with split loads is discussed by Nowak et al. (2008), with extra emphasis on the benefit of using split loads. They provide a theoretical view on the benefit of the split load aspect, and se this to create a heuristic, which is then used on a real-world industry problem. The benefit of split loads is found to be most closely tied to the load sizes in the problem, the cost associated with pickups and deliveries, and how often loads are going from the same origin or to the same destination. Their work is continued in Nowak et al. (2009), who find that the benefit is the greatest with loads just over half of the vehicle capacity, as two of these loads cannot be transported by a single vehicle without splitting. The same result is found in Korsvik et al. (2011), who study a ship routing and scheduling problem with split loads, solved by a large neighbourhood search (LNS). The LNS algorithm uses a combination of a descent local search and a destroy and repair algorithm. The splitting of loads is incorporated into the LNS through two of the four local search operators. In these two operators, either one or two loads are removed from all routes they are present in. Then, the best combination of insertion points in all routes is found for each load, which can include a split

of the load between two or more routes. Another ship routing and scheduling problem with split loads is discussed by Stålhane et al. (2012), who introduce a new path-flow formulation for the problem and solve it using branch-and-price-and-cut.

Another PDP, a one-to-one PDP with split loads and multiple vehicles, is solved by Haddad et al. (2018) using a large neighbourhood-based metaheuristic and a branch-and-price algorithm. In the metaheuristic a two-phase method is used, in which the first phase consists of a randomised variable neighbourhood descent. The second phase utilises dynamic programming in the form of a shortest path problem for the insertion of each pair of order nodes. First every occurrence of an order is removed, and then all possible insertion variants is considered, with the possibility of splitting an order between several insertion positions. This is performed by solving a shortest path problem dynamically for each order with the use of labels with distance and load quantity as the resources. After finding all non-dominated labels, a knapsack problem is solved to find the combination of labels which gives the lowest distance increase while still fulfilling the orders.

3.1.8 Time windows

Time windows represent an upper and lower time limit for when service must start at each customer, or for when pickups and deliveries of people and commodities must start. Early arrival is usually allowed, which results in the vehicle having to wait until the time window is open before it can start to serve the customer or do a pickup or delivery. The depot can also have a time window for earliest departure and latest arrival. Time windows are generally included by having time variables in the formulation. Constraints are then added to handle these variables, limit them according to time windows at each customer, and ensure that the vehicles return to the depot before the closing of the time window (Cattaruzza et al., 2016).

Hernandez et al. (2014) study a VRP with time windows (VRPTW) and limited trip duration, with the inclusion of time windows at both the customers and the depot. An exact two-phase algorithm is used to find the set of trips with the lowest associated cost while using no more than a fixed number of vehicles. The first phase is to generate all possible trips that satisfy the limited trip duration. Using column generation and a branch-and-price approach, the second phase then selects the set of trips that results in the lowest cost and visits all customers.

A branch-and-price approach is also used in the exact algorithm presented by Azi et al. (2010), who also study a MTVRP with time windows and limited trip duration. In this paper the authors include revenues associated with each customer and the possibility that a fleet of fixed size might not be able to serve all customers. The same problem and mathematical model is studied further in Azi et al. (2014), in which an ALNS algorithm is used to solve the problem.

The DARP studied by Parragh et al. (2014) include time windows, in addition to the split requests. The

time windows are included at both the pickup and delivery location. In DARPs there is usually a given time either for pickup or delivery of a person, and Parragh et al. (2014) compute the delivery time window from the given pickup time window and the maximum ride duration. To evaluate solutions in their VNS algorithm, an evaluation function with penalties for violating certain constraints is used. Time windows are handled by this evaluation function, adding an associated penalty when time windows are violated. A different way of handling time windows is used in Korsvik et al. (2011), in which the authors only consider solutions which are feasible with regards to time windows.

3.2 Helicopter transport planning

The offshore HRP can be considered a rich variant of the GPDP. Sierksma and Tijssen (1998) present a way to apply the terminology commonly used for VRPs to HRPs. By using the same concept our problem has the following VRP characteristics:

- The depots are the heliports.
- The customers are the installations.
- The demand is the demand for transportation of personnel.
- The vehicles are the helicopters.
- The limited driving time is the limited range of each helicopter.
- Split deliveries are present.
- The objective is the minimisation of total costs, both for chartering and using the helicopters.

The offshore HRP literature can be divided into three parts based on the time scope: strategic, tactical and operational planning. The problem studied in this thesis is operational, as is the majority of the problems found in the literature. We do, however, present a short overview of the strategic and tactical planning literature in addition to the literature on operational planning.

3.2.1 Strategic and tactical planning

Most of the helicopter transport planning have a day to day planning horizon, but problems regarding strategic and tactical planning have been studied as well. These problems can have a planning horizon of months, years, or even decades. Hermeto et al. (2014) present a brief overview of the literature on different levels of planning. Furthermore, the authors provide a model for deciding the airfield locations for helicopter transport to offshore units. The objective is to minimise the total cost of opening and operating airfields and the use of helicopters.

Strategic planning is also found in Fernández-Cuesta et al. (2017), who aim to find the optimal fleet composition for helicopters and the optimal locations for bases and refuelling hubs. A combination of operational and tactical planning is used in Barbarosoğlu et al. (2002). The authors present a two-level model for helicopter mission planning during a disaster relief operation. The top level decisions are tactical, while the low level decisions are operational.

3.2.2 Operational helicopter planning

Operational helicopter planning usually considers planning of helicopter routes throughout a day. In this subsection we present studies performed on the operational offshore HRP by relating them to the topics and extensions from Section 3.1, and present the methods used to solve the problems.

The objective in operational offshore HRPs is often to minimise either total costs or total duration. This the case for most of the papers presented in this literature review, but there are some studies with other objectives. Minimisation of total distance, which is closely related to costs and duration, is the objective in Sierksma and Tijssen (1998). Menezes et al. (2010) includes minimisation of the number of landings together with total costs. Safety concerns are also the focus in several studies. In Qian et al. (2011), Qian et al. (2012), and Gribkovskaia et al. (2015) the objective is to minimise the risk of accidents and fatalities for both the pilot and passengers.

Helicopter routes must satisfy a set of operational constraints, which in most cases include a limit on seat capacity, safety levels for fuel, safety inspections between trips and having the start and finish of each trip for a helicopter at the same mainland heliport. Depending on the problem studied, there may be several other operational constraints included. Total weight limit is included in several studies like Abbasi-Pooya and Husseinzadeh Kashan (2017), Husseinzadeh Kashan et al. (2019) and Rosa et al. (2016). These also include a restricted flight time, either for trips, routes or both. This is also considered in Moreno et al. (2005), Moreno et al. (2006), Sierksma and Tijssen (1998) and Velasco et al. (2009).

Multiple depots

The fleet of helicopters is stationed at heliports, which are the depots in the offshore HRP. The heliports are often located at airports, and consequently there may be a significant distance between heliports. This reduces the number of solutions where an installation can be served by helicopters from two different heliports. In both Rosa et al. (2016) and Moreno et al. (2005), there are two heliports, but the routing problem is solved separately for the two heliports as they are not close enough to serve the same installations. As Moreno et al. (2006) is a continuation of the work in Moreno et al. (2005), this is also the case in this study.

Menezes et al. (2010) describe a problem that is similar to the traditional MDVRP where multiple heliports can be used to serve the same installations, but all helicopters have to return to the same heliport they

started at. Haugseth et al. (2019) also present a problem with multiple heliports where each trip must start and end at the same heliport.

Heterogeneous fleet

The fleet of helicopters is mainly defined in three different ways in the relevant literature: with one helicopter, a fleet of homogeneous helicopters, or a fleet of heterogeneous helicopters. Abbasi-Pooya and Husseinzadeh Kashan (2017) and Husseinzadeh Kashan et al. (2019) describe a problem with only one helicopter. This is also the case in Velasco et al. (2009). In Sierksma and Tijssen (1998) multiple helicopters are being used, but the fleet is homogeneous.

In real problems the fleet of helicopters is often heterogeneous. The seat capacity, range, and cruising speed can vary significantly for different helicopters. A greater seat capacity and range are often accompanied by a higher cost of use. More flexibility in choice of helicopters expand the solution space and can result in better solutions. Menezes et al. (2010) use a fleet of helicopters with varying seat capacity and flight time cost, but any other differences are ignored. Rosa et al. (2016), Moreno et al. (2005), and Moreno et al. (2006) all include the weight limit, fuel capacity, and cruising speed as parameters that depend on the helicopter, as well as seat capacity and flight time cost. Rosa et al. (2016) further includes a fixed cost for the use of a helicopter, as well as an earliest take-off time and a latest return time for each helicopter. Many of the same parameters are found in Haugseth et al. (2019), where costs for chartering and using the helicopters, fuel capacity, cruising speed, and seat capacity vary for the helicopters in the fleet.

Multiple trips

For offshore HRPs, allowing multiple trips for the helicopters can prove beneficial, especially in problems with minimisation of costs as an objective. Chartering and using helicopters are costly, and limiting the number of helicopters in use can therefore reduce the total costs by a substantial amount. The number of personnel to be transported is often high, which further emphasises the possible need for multiple trips.

Allowing multiple trips is almost a necessity in problems with only a single helicopter available for transportation, in order to transport as many passengers as possible throughout the day. In Abbasi-Pooya and Husseinzadeh Kashan (2017) and Husseinzadeh Kashan et al. (2019) there is only a single helicopter available and this helicopter must therefore perform multiple trips in order to satisfy the transportation demand. This is also the case in Velasco et al. (2009), in which a single helicopter performs several trips.

Haugseth et al. (2019) considers a problem with multiple trips allowed for the helicopters in the fleet. The study also includes transportation between two installations, in addition to the usual transportation between a heliport and an installation, during a trip. The same applies for Rosa et al. (2016).

The need for multiple trips can be reduced when not requiring all passengers to be transported, and by this not satisfying the complete transportation demand. Moreno et al. (2005) consider a case study for Petrobras in Brazil. They include a high penalty associated with not completing a transportation request. The penalty is set high because stops in activities on the platform results in high cost and loss of revenue. In this study, the helicopters may perform multiple trips each day, but the total number of daily trips has an upper limit. Findings in the study are used further in Moreno et al. (2006) and multiple trips with a daily upper limit is therefore included in this paper as well. Petrobras is also used for the case study in Menezes et al. (2010), which has a high number of transportation requests daily. This study also includes the possibility to not fulfil all requests, but with a penalty in the objective function for the unfulfilled demand. The helicopters may perform multiple trips during a day in this study as well, and the same upper limit on total trips per helicopter throughout the day is used.

Split deliveries

For offshore helicopter routing, the option to split orders can be very beneficial, or even necessary. An installation could need pickup or delivery services several times a day, and without split pickup and deliveries there could be no feasible solutions, or the solutions found could easily be improved. Split deliveries for offshore helicopter routing was first discussed by Sierksma and Tijssen (1998). They study the transportation of crew from an airport in the Netherlands to 51 installations in the North Sea. In addition to solving the problem with a heuristic, Sierksma and Tijssen (1998) discuss properties of the optimal solution of the SDVRP and use the properties to improve the design of the heuristic.

Menezes et al. (2010) solve a HRP with multiple heliports serving several installations. Although split deliveries are allowed, they are not discussed in the paper. The split delivery aspect of the problem is emphasised more by Moreno et al. (2005) and Moreno et al. (2006), who develop two models and a heuristic.

Some problems allow split delivery of passengers to installations, but split deliveries are not part of the model. In Haugseth et al. (2019) and Rosa et al. (2016) every order or single passenger is assigned two nodes, one for pickup and one for delivery. The nodes can only be visited by one helicopter, but there could be multiple nodes at a single installation. Hence there can be split deliveries for the installations, but since the model is based on the use of nodes for each order, the model itself does not include split deliveries.

Time windows

Moreno et al. (2005) and Moreno et al. (2006) consider problems with a set of specific departure times and then assign passengers based on requested departure time from the customers, and thereby do not have the need for further time windows. The same use of set departure times is also applied in Menezes et al. (2010). The study performed in Velasco et al. (2009) does not include time windows in the formulation, because passengers are assumed to be available for pickup and delivery at respective nodes at any time, which is the

assumption commonly used in HRPs without time windows.

A time window is included for the heliport in Rosa et al. (2016), so the helicopters have an earliest departure time and a latest arrival time at the heliport. Both Abbasi-Pooya and Husseinzadeh Kashan (2017) and Husseinzadeh Kashan et al. (2019) include a parameter for the time when passengers travelling from the heliport is available for transportation, which creates a time window with an earliest departure time from the heliport for each of these passengers. Time windows are included for all nodes in Haugseth et al. (2019) with the use of an earliest and latest arrival time for each node. This results in time windows for all customer nodes and heliports.

Safety in helicopter transportation

Using helicopters for offshore transportation can provide convenient, fast and reliable transportation, but does involve risk and potential for accidents. Safety during helicopter transportation is therefore the focus and objective in several studies.

Qian et al. (2011) study safety in the Norwegian oil industry by looking at the expected number of fatalities due to helicopter transportation. Safety is considered as risk during take-off, cruising and landing. A set of helicopters is used for transportation between several offshore installations and a heliport, and the objective is to minimise the expected number of fatalities. Passenger and pilot safety is also the focus in Qian et al. (2012). Gribkovskaia et al. (2015) also study safety associated with helicopter transportation, by trying to minimise the total passenger risk during takeoff and landing. Different routing policies for helicopter transportation to offshore installations in a hub and spoke fashion is studied in Halskau (2014), with the objective of minimising expected number of fatalities.

Menezes et al. (2010) includes safety concerns by having a mixed objective of minimising both operating costs and the total number of offshore landings. Offshore landings are given a cost in the objective function, hence making it beneficial to reduce the total number of landings at offshore installations. By reducing the number of offshore landings the risk of accidents is also reduced. Emergency logistics is studied in Caballero-Morales and Martinez-Flores (2019), in which the authors consider the problem of evacuating multiple installations in the case of emergencies.

Solution methods in operational helicopter planning

In the literature regarding HRPs, a variety of solution methods are used. Most of them have in common that some sort of heuristic approach is used. As explained by Moreno et al. (2005), unscheduled events during the day, like bad weather, could lead to a demand for changes in flight schedules. The new flight schedules need to be available quickly, and algorithms that can find a good solution quickly are therefore needed. For problems of realistic size in helicopter routing to offshore installations, most exact methods used have been

found too slow. Heuristics have therefore been the most studied solution method.

An exact solution method is presented in Sierksma and Tijssen (1998), which utilises column generation and the solving of travelling salesman problems (TSP). To achieve integer solutions to the problem a rounding procedure is performed. As commented by the authors the exact approach is not viable for short-term planning with limited computational time available. A cluster heuristic is therefore provided, which exploits the fact that installations visited in a specific route often are located near each other. The heuristic in Sierksma and Tijssen (1998) differs from other cluster heuristics in the sense that the clusters of installations and the associated routes are constructed simultaneously, instead of first constructing all clusters and then creating the routes. Additionally, the authors provide multiple improvement heuristics. Rosa et al. (2016) propose a clustering search metaheuristic. A simulated annealing metaheuristic is used to generate new solutions. The authors use a penalised objective function to penalise violation of some constraints, as obtaining feasible solutions with regards to all constraints requires long computational time.

Haugseth et al. (2019) present both exact and heuristic solution methods. In the first exact solution method a labeling algorithm is used to generate all possible trips for the HRP, and then branch-and-check and aggregated branching variables are used to solve the problem. Additionally, the authors present an exact solution approach which utilises column generation and branch-and-price. Two heuristic solution methods are also provided, a column generation matheuristic and an ALNS.

A column generation based heuristic is also used in Moreno et al. (2006). This study improves the heuristic algorithm presented in Moreno et al. (2005), in which the problem is divided into a construction phase where trips are constructed sequentially and an assembly phase which selects trips to use for each helicopter. Moreno et al. (2006) improve the algorithm by adding a column generation procedure that exploits the problem as much as possible. The authors exploit the fact that for different departure times only a limited number of installations are served, and a helicopter can serve a limited number of installations during a trip. For few visits during a trip all possible routes are generated, and a heuristic approach in the form of a neighbourhood search is used when there are more installation visits during trips. The same kind of column generation based algorithm is used in Menezes et al. (2010) as they study the same problem.

Genetic algorithms (GAs) are also being used for the HRP. This is the case for Romero et al. (2007) in which a two-stage algorithm is proposed. The first stage is a planning problem where the best sequence of orders for each helicopter is decided by a heuristic, and the second stage is an allocation problem where a GA is used to find the best allocation of orders for the helicopters. A GA is also used in the memetic algorithm in Velasco et al. (2009), but here with the inclusion of a local search procedure for an offspring when replacing solutions. Abbasi-Pooya and Husseinzadeh Kashan (2017) use Grouping Evolution Strategy as their solution method. A penalised objective function is used in this study as well, which includes penalties for violating

time limits, helicopter capacity and weight limit. Husseinzadeh Kashan et al. (2019) study a similar problem, and propose a "league championship algorithm" to solve the problem.

3.3 Matheuristics

Matheuristics is a class of solution methods that combine the use of mathematical programming models and heuristic schemes. The mathematical programming model can be combined with the heuristic in many different ways. Archetti and Speranza (2014) provide a survey on matheuristics for routing problems, and classify the matheuristics into three categories:

- Decomposition approaches, in which the problem is divided into smaller subproblems, and one or more of these subproblems are solved using mathematical programming models.
- Improvement heuristics, in which mathematical programming models are used to improve solutions found by a heuristic approach.
- Branch-and-price/column generation-based approaches, in which the exact method is sped up by modifying the convergence process such that optimality is no longer guaranteed.

The decomposition approaches are described by Archetti and Speranza (2014) as "particularly suitable for the solution of complex and integrated problems", like the VRP. Routing problems can often be divided into two decisions: which customers that should be served by the same vehicle, and what the routes for the vehicles should be. This is the basis for the cluster first-route second approach, where customers first are clustered and assigned to vehicles, and then the sequencing of customers is done. The first matheuristic of this kind was proposed by Fisher and Jaikumar (1981), who choose a cluster of customers heuristically, and solve a TSP to build routes. These types of heuristics can be used iteratively as well to allow for longer time periods. Other approaches that consist of two phases, one heuristic and one exact, but do not fit into the cluster first-route second approach, are called two-phase approaches. Flisberg et al. (2009) decide the daily routes of logging trucks by first solving a linear programming problem to find transportation nodes and then combine the nodes into routes through a tabu search. A different two-phase problem was studied by Halvorsen-Weare and Fagerholt (2013), who assign cargo and routes to maritime vessels and then schedule when vessels should be serviced such that the solution is feasible. In order to solve the problem for real life instance sizes, a local search heuristic is used to for the routing and cargo assignment, while the scheduling is done by exact methods.

Improvement heuristics combine the use of a heuristic with an exact method, with the aim that the exact method should improve the solution from the heuristic. The exact method can be used before the heuristic to get a good starting point for the heuristic, it can be integrated into the heuristic to improve the searching phase, or it can be used at the end of the heuristic to improve the final solution. This means that

they can be split into two groups: heuristics where the exact method is used only once, called "one-shot approaches", and heuristics where the exact method is used repeatedly as a part of the search process. A one-shot approach is used by Archetti et al. (2014), who use tabu search to find good and potential good arcs for a routing problem with profit and travelling cost. A tour is initially made up by only good arcs, before a mixed integer programming (MIP) model is used to insert potentially good arcs into the tour.

The third category, branch-and-price/column generation-based approaches, make use of branch-and-price and/or column generation. Branch-and-price and column generation algorithms are used to find the exact solution of many types of routing problems, including the PDP, VRP and DARP, and "are at the moment the exact leading methodology" (Archetti and Speranza, 2014). Generating all columns or branching through all possibilities can be time consuming for large problems, and thus heuristic methods are included in the algorithm to speed up the process. In Danna and Le Pape (2005), a general cooperation scheme between branch-and-price and local search is proposed, and it is showed that the local search helps the branch-and-price algorithm to find good solutions faster. As shown in Subsection 3.2.2, column generation based approaches has been used for HRPs by Moreno et al. (2006), Menezes et al. (2010), and Haugseth et al. (2019) as well.

3.4 Contributions

In this section we position this thesis with regards to the literature review performed in the previous sections. In order to do this we have summarised features and solution methods for the most relevant studies from Subsection 3.2.2 and compared them to our own problem. The comparison is shown in Table 3.1.

Table 3.1: A comparison of the most relevant papers on the HRP, with regards to the different extensions and solutions methods.

Paper	Objective	Multiple heliports	Heterogeneous fleet	Multiple trips	Split delivery	Time windows	Multiple heliports during a route	Optional heliport for order	Solution method
This thesis	**Min total costs**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Decomposition matheuristic**
Haugseth et al. (2019)	Min total costs	Yes	Yes	Yes	No	Yes	No	No	Branch-and-price, matheuristic, ALNS
Husseinzadeh-Kashan et al. (2019)	Min total duration	No	No	Yes	No	No	No	No	League championship algorithm
Abbasi-Pooya and Husseinzadeh-Kashan (2017)	Min total duration	No	No	Yes	No	Yes	No	No	Grouping evolution strategy
de Alvarenga Rosa et al. (2016)	Min total costs	Yes	Yes	Yes	No	Yes	No	No	Clustering search heuristic
Menezes et al. (2010)	Min total costs and number of landings	Yes	Yes	Yes	Yes	No	No	No	Column generation heuristic
Velasco et al. (2009)	Min total duration	No	No	Yes	No	No	No	No	Memetic algorithm
Moreno et al. (2006)	Min total costs	Yes	Yes	Yes	Yes	No	No	No	Column generation heuristic
Moreno et al. (2005)	Min total costs	Yes	Yes	Yes	Yes	No	No	No	Construcion and assembly heuristic
Sierksma and Tijssen (1998)	Min total distance	No	No	No	Yes	No	No	No	Exact column generation, cluster heuristic

Since all papers listed in Table 3.1 concern HRPs, it is expected that the problem studied in this thesis includes some of the same problem aspects as the papers in the table. We see that this is indeed the case, as all papers include at least one of the extensions from our problem. Some of the papers, such as Haugseth et al. (2019) and Rosa et al. (2016), study a similar problem, but neither include splitting of orders. Menezes et al. (2010), Moreno et al. (2006), and Moreno et al. (2005) all include splitting of orders, but do not consider time windows for the orders. Many different solution methods are used to solve the problems, most of them heuristic, but some exact. As seen in Subsection 3.2.2, many of the solution methods for the operational HRP try to exploit certain structures or aspects of the problem in order to solve it more efficiently. With all the extensions included in our problem, this becomes an essential part of our solution approach as well. Our solution method is a matheuristic utilising a labeling algorithm and a MIP model, and is different from the solution methods found in the literature review.

Two extensions that are present in our problem, but to our knowledge not included in any other paper on the HRP, are the option to visit multiple heliports during a route and choosing the heliports to pickup/deliver each order at. These are only relevant in problems that include multiple heliports and where the heliports and installations are located such that it is possible, and potentially beneficial, to fly to an installation from several heliports. Among the papers where this is the case, none include these two options. Adding these options could result in better utilisation of helicopters, and therefore potentially reduce distance flown and the number of helicopters needed to complete all orders. Both chartering and using helicopters are expensive, and reducing the need for helicopters or flying distance would result in significant cost savings. We found in our project report, Oppedal and Thorstein (2019), that these two extensions showed promise with only a few orders included. It is therefore of interest to study whether the same is the case with more orders included.

The main contributions of this thesis are an arc-flow model that includes the two new extensions described above and all extensions discussed in Section 3.1, as well as a new matheuristic used to solve the problem. The matheuristic is used to solve the problem with a high number of orders included, and study the benefit of visiting several heliports during a route and choosing which heliports to pickup/deliver each order at. The inclusion of splitting of orders in the problem is also analysed, as this is one of the extensions with the most impact on problem complexity.

4 | Arc-Flow Model

In this chapter we present the complete mathematical formulation for the problem in this thesis, which is the arc-flow model presented in Section 4.1. This model is equal to the one presented in our project report, Oppedal and Thorstein (2019), which was inspired by the arc-flow model presented in Haugseth et al. (2019). Preprocessing that can be done to improve the computational time of the model is discussed in Section 4.2.

4.1 Arc-flow model

This section presents the mathematical arc-flow model made for the problem in this thesis. All the sets, indices, parameters and variables used in the model are listed and briefly explained, but some additional explanation is needed for some of the sets.

Let O be the set of all transportation orders, indexed by o. Each order has one or more corresponding pickup and delivery nodes, which are included in the sets N_o^P and N_o^D, respectively. The number of nodes included in these sets depends on whether the personnel are transported to or from the installation. If the order requires transportation from a heliport to an installation, a pickup node for each heliport is added to N_o^P and a single delivery node at the installation is included in N_o^D. Likewise, if the order requires transportation from an installation to a heliport, N_o^P contains a single pickup node at the installation and N_o^D contains a delivery node for each heliport. The two possibilities are illustrated in Figure 4.1, which shows an example with two heliports and one order at an installation.

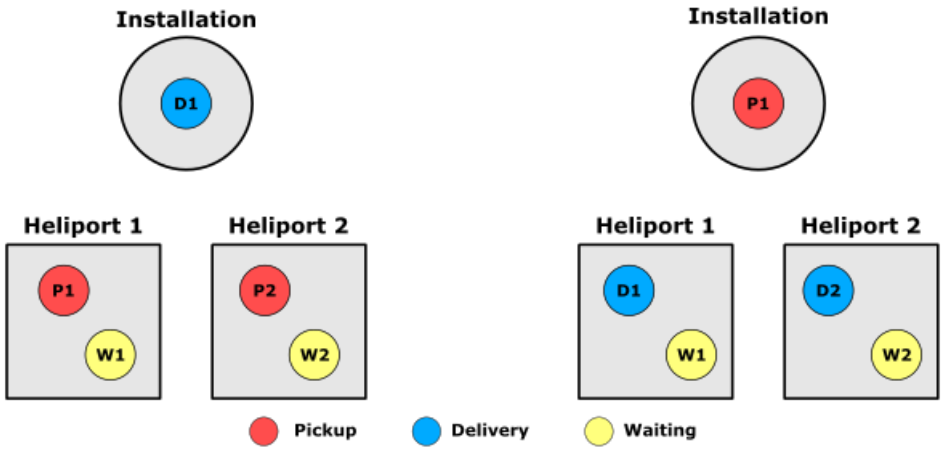


Figure 4.1: An illustration of pickup, delivery, and waiting nodes with one installation and two heliports for the arc-flow model.

The left part of the figure illustrates the nodes included for an order requiring transportation from a heliport to an installation. There is a delivery node at the installation, and a pickup node at both heliports. On the right, there is a single pickup node at the installation, and a delivery node at both heliports. This because the order requires transportation from an installation to a heliport. All nodes for an order o is added to the set N_o, which means that $N_o = N_o^P \cup N_o^D$.

All nodes located at heliports are included in N^{HP}, and this set has several subsets. $N^{P,HP}$ is the set of pickup nodes at heliports, and $N^{D,HP}$ is the set of delivery nodes at heliports. In addition, we have a set of waiting nodes, N^W. The waiting nodes are introduced in the model to handle the special case when a helicopter delivers an order at an installation, and the next order is a pickup at an installation for which the time window has not opened yet. In this case the helicopter needs a node to wait at, which is the purpose of the waiting nodes. In Figure 4.1, there is one waiting node included at each heliport. Together, these sets make up N^{HP}, that is $N^{HP} = N^{P,HP} \cup N^{D,HP} \cup N^W$. Similarly, there is a set of all nodes located at installations, N^I, $N^I = N^{P,I} \cup N^{D,I}$.

The set N^{PDW} contains all pickup, delivery and waiting nodes, $N^{PDW} = N^{HP} \cup N^I$. All nodes in N^{PDW} can be visited by every helicopter. Every helicopter has its own start and end node, and we therefore have the set $N_h = s(h) \cup N^{PDW} \cup e(h)$ that includes all nodes a helicopter h can visit. The nodes are indexed in increasing order, and each node is given an index based on the following scheme:

$$\text{start nodes} < \text{pickup nodes} < \text{delivery nodes} < \text{end nodes} < \text{waiting nodes}.$$

Between trips a helicopter is required to be empty, and it needs to have the opportunity to wait for the next time window to open. To achieve this, let A^E be the set of arcs between the last node of one trip and the first node of the next trip. On these arcs the helicopters are allowed to wait and must be empty. Table 4.1 lists all arcs included in A^e:

Table 4.1: All arcs included in A^E, for which a helicopter must be empty and has the opportunity to wait.

i	**j**	**Further explanation**
$N^{D,I}$	$N^{P,HP} \cup N^W$	Between deliveries at installations and pickups at heliports or waiting nodes.
$N^{D,HP}$	$N^{P,HP} \cup N^{P,I}$	Between deliveries at heliports and pickups at heliports or pickups at installations.
N^W	$N^{P,I}$	Between waiting nodes and pickups at installations.

Sets

O: Set of all orders

H: Set of helicopters

N_h: Set of all nodes that can be visited by helicopter h.

N^P: Set of all pickup nodes

N^D: Set of all delivery nodes

N^W: Set of waiting nodes

N^{PDW}: Set of all pickup, delivery and waiting nodes

N_i: Set of all nodes in the same geographical location as node i, including i itself, but excluding start and end nodes

N^I: Set of all nodes located at installations

$N^{P,I}$: Set of all pickup nodes located at installations

$N^{D,I}$: Set of all delivery nodes located at installations

N^{HP}: Set of all nodes located at a heliport

$N^{P,HP}$: Set of all pickup nodes located at heliports

$N^{D,HP}$: Set of all delivery nodes located at heliports

N_o: Set of all nodes for order o

N_o^P: Set of all pickup nodes for order o

N_o^D: Set of all delivery nodes for order o

A^E: Set of arcs between the end node of one trip and the starting node of the next trip.

L: Set of all legs, with highest value $\overline{L}$

Indices

i, j: Node

o: Order

h, h': Helicopter

l: Leg number

Parameters

C_{hij}: Cost of travelling with helicopter h from node i to j

C_h^F: Fixed cost of using helicopter h

F_{hij}: Fuel consumption when travelling with helicopter h from node i to j

F_{hi}^{max}: Maximum fuel level for helicopter h when leaving node i

F_{hi}^{min}: Minimum fuel level for helicopter h when leaving node i

L^{max}: Maximum number of legs allowed between pickup and delivery of a passenger

P_o: Total number of passengers to be transported for order o

Q_h: Seat capacity for helicopter h

T_{hij}: Time used when travelling with helicopter h from node i to j

T_i^{max}: Upper time limit at node i

T_i^{min}: Lower time limit at node i

T_{ij}^F: Fixed time used at node i when travelling to node j

T^I: Fixed time used at installation

Variables

x_{hijl}: Equal to 1 if helicopter h travels from node i to node j on leg l, 0 otherwise

y_h: Equal to 1 if helicopter h is used, 0 otherwise

f_{hi}: Fuel level for helicopter h when leaving node i

q_{hi}: Number of occupied seats for helicopter h when leaving node i

p_{hi}: Number of passengers picked up or delivered by helicopter h at node i

t_{hi}: Time when arriving with helicopter h at node i

$\lambda_{hh'ij}$: Equal to 1 if helicopter h visits node i and helicopter h' visits node j, 0 otherwise

$\lambda'_{hh'ij}$: Equal to 1 if helicopter h' visits node j before helicopter h visits node i, where node i and j are in the same geographical position, 0 otherwise

$$\min \sum_{h\in H}\sum_{i\in N_h}\sum_{j\in N_h}\sum_{l\in L} C_{hij}x_{hijl} + \sum_{h\in H} C_h^F y_h \tag{4.1}$$

subject to

$$\sum_{h\in H}\sum_{i\in N_o^P}\sum_{j\in N_h}\sum_{l\in L} x_{hijl} \geq 1, \qquad o\in O, \tag{4.2}$$

$$\sum_{i\in N_o^P}\sum_{j\in N_h}\sum_{l\in L} x_{hijl} \leq 1, \qquad h\in H, o\in O, \tag{4.3}$$

$$\sum_{i\in N_o^D}\sum_{j\in N_h}\sum_{l\in L} x_{hijl} \leq 1, \qquad h\in H, o\in O, \tag{4.4}$$

$$\sum_{j\in N_h}\sum_{l\in L} x_{hijl} \leq 1, \qquad h\in H, i\in N^W, \tag{4.5}$$

$$\sum_{i\in N_o^P}\sum_{j\in N_h}\sum_{l\in L} x_{hijl} - \sum_{i\in N_o^D}\sum_{j\in N_h}\sum_{l\in L} x_{hijl} = 0, \qquad h\in H, o\in O, \tag{4.6}$$

$$\sum_{j\in N_h}\sum_{l\in L} x_{hijl} - \sum_{j\in N_h}\sum_{l\in L} x_{hjil} = 0, \qquad h\in H, i\in N_h\setminus\{s(h),e(h)\}, \tag{4.7}$$

$$\sum_{j\in N^P} x_{hs(h)j0} - y_h = 0, \qquad h\in H, \tag{4.8}$$

$$\sum_{i\in N^D}\sum_{l\in L} x_{hie(h)l} - y_h = 0, \qquad h\in H, \tag{4.9}$$

$$\sum_{j\in N_h}\sum_{l\in L} x_{hs(h)jl} \leq 1, \qquad h\in H, \tag{4.10}$$

$$\sum_{i\in N_h}\sum_{l\in L} x_{hie(h)l} \leq 1, \qquad h\in H, \tag{4.11}$$

$$\sum_{j\in N_h} x_{hjil} = \sum_{j\in N_i} x_{hijl} + \sum_{j\in N_h\setminus N_i} x_{hij(l+1)}, \qquad h\in H, i\in N_h\setminus\{s(h),e(h)\}, l\in L\setminus\{\overline{L}\}, \tag{4.12}$$

$$\sum_{i\in N_h}\sum_{j\in N_o^D}\sum_{l\in L} l x_{hijl} - \sum_{i\in N_h}\sum_{j\in N_o^P}\sum_{l\in L} l x_{hijl} \leq L^{max}, \qquad h\in H, o\in O, \tag{4.13}$$

$$\sum_{i\in N_h}\sum_{j\in N_h} x_{hijl} \le \sum_{i\in N_h}\sum_{j\in N_h} M_h^L x_{hij(l-1)}, \qquad h\in H, l\in L\setminus\{0\}, \tag{4.14}$$

$$x_{hijl} = 0, \qquad h\in H, i\in N^{PDW}, j\in (N_i \mid j\ge i+1), l\in L, \tag{4.15}$$

$$f_{hi} - F_{hij} - f_{hj} + M_{hij}^F(1-\sum_{l\in L} x_{hijl}) \ge 0, \qquad h\in H, i\in N_h, j\in N^I, \tag{4.16}$$

$$F_{hi}^{min} \le f_{hi} \le F_{hi}^{max}, \qquad h\in H, i\in N_h, \tag{4.17}$$

$$q_{hi} + p_{hj} - q_{hj} - Q_h(1-\sum_{l\in L} x_{hijl}) \le 0, \qquad h\in H, i\in N_h, j\in N^P, \tag{4.18}$$

$$q_{hi} - p_{hj} - q_{hj} - Q_h(1-\sum_{l\in L} x_{hijl}) \le 0, \qquad h\in H, i\in N_h, j\in N^D, \tag{4.19}$$

$$q_{hi} - Q_h(1-\sum_{l\in L} x_{hijl}) \le 0, \qquad h\in H, (i,j)\in A^E, \tag{4.20}$$

$$p_{hi} \le q_{hi} \le Q_h \sum_{j\in N_h}\sum_{l\in L} x_{hijl}, \qquad h\in H, i\in N^P, \tag{4.21}$$

$$0 \le q_{hi} \le Q_h \sum_{j\in N_h}\sum_{l\in L} x_{hijl} - p_{hi}, \qquad h\in H, i\in N^D, \tag{4.22}$$

$$\sum_{j\in N_h}\sum_{l\in L} x_{hijl} \le p_{hi} \le P_o \sum_{j\in N_h}\sum_{l\in L} x_{hijl}, \qquad h\in H, o\in O, i\in N_o, \tag{4.23}$$

$$\sum_{h\in H}\sum_{i\in N_o^P} p_{hi} = P_o, \qquad o\in O, \tag{4.24}$$

$$\sum_{h\in H}\sum_{i\in N_o^D} p_{hi} = P_o, \qquad o\in O, \tag{4.25}$$

$$\sum_{i\in N_o^P} p_{hi} - \sum_{i\in N_o^D} p_{hi} = 0, \qquad h\in H, o\in O, \tag{4.26}$$

$$t_{hi} + T_{ij}^F + T_{hij} - t_{hj} + M_{hij}^{T1}(1-\sum_{l\in L} x_{hijl}) \ge 0, \qquad h\in H, i\in N_h, j\in N_h \mid (i,j)\notin A^E, \tag{4.27}$$

$$t_{hi} + T^F_{ij} + T_{hij} - t_{hj} - M^{T2}_{hij}(1 - \sum_{l \in L} x_{hijl}) \leq 0, \qquad h \in H, i \in N_h, j \in N_h, \tag{4.28}$$

$$\sum_{i \in N^P_o} t_{hi} + \sum_{i \in N^P_o} \sum_{j \in N^D_o} \sum_{l \in L} (T_{hij} + T^F_{ij}) x_{hijl} \leq \sum_{j \in N^D_o} t_{hj}, \qquad h \in H, o \in O, \tag{4.29}$$

$$\sum_{j \in N_h} \sum_{l \in L} T^{min}_i x_{hijl} \leq t_{hi} \leq \sum_{j \in N_h} \sum_{l \in L} T^{max}_i x_{hijl}, \qquad h \in H, i \in N_h \setminus \{e(h)\}, \tag{4.30}$$

$$\sum_{i \in N_h} \sum_{l \in L} T^{min}_{e(h)} x_{hie(h)l} \leq t_{he(h)} \leq \sum_{i \in N_h} \sum_{l \in L} T^{max}_{e(h)} x_{hie(h)l}, \qquad h \in H, \tag{4.31}$$

$$t_{hi} - t_{h'j} - T^I + (T^I + T^{max}_j)(1 - \lambda'_{hh'ij}) \geq 0, \qquad h \in H, h' \in H \setminus \{h\}, i \in N^I, j \in N_i, \tag{4.32}$$

$$\sum_{i' \in N_h} \sum_{l \in L} x_{hii'l} + \sum_{j' \in N_{h'}} \sum_{l \in L} x_{h'jj'l} \leq \lambda_{hh'ij} + 1, \qquad h \in H, h' \in H \setminus \{h\}, (i,j) \in (N^I \times N^I \mid j \in N_i), \tag{4.33}$$

$$\lambda'_{hh'ij} + \lambda'_{h'hji} \geq \lambda_{hh'ij}, \qquad h \in H, h' \in H \setminus \{h\}, (i,j) \in (N^I \times N^I \mid j \in N_i), \tag{4.34}$$

$$x_{hijl} \in \{0,1\}, \qquad h \in H, i \in N_h, j \in N_h, l \in L, \tag{4.35}$$

$$y_h \in \{0,1\}, \qquad h \in H, \tag{4.36}$$

$$\lambda_{hh'ij} \in \{0,1\}, \qquad h \in H, h' \in H \setminus \{h\}, i \in N^I, j \in N_i, \tag{4.37}$$

$$\lambda'_{hh'ij} \in \{0,1\}, \qquad h \in H, h' \in H \setminus \{h\}, i \in N^I, j \in N_i. \tag{4.38}$$

The objective function (4.1) minimises the cost related to chartering the helicopters and traversing the arcs. Constraints (4.2) ensure that at least one pickup node is visited for each order, while constraints (4.3) and (4.4) ensure that a helicopter visits at most one pickup and one delivery node for each order. A restriction of one or fewer visits to waiting nodes for every helicopter is given by constraints (4.5). Constraints (4.6) force the same helicopter that picks up an order to visit an associated delivery node for that order. Flow in and out of a node is ensured by constraints (4.7), while constraints (4.8) and (4.9) ensure that a helicopter starts at its start node and ends at its end node, if the helicopter is in use. A maximum of one arc out of a

start node and into the end node is forced by constraints (4.10) and (4.11).

The leg counter is updated in constraints (4.12), and constraints (4.13) restrict the maximum number of legs between pickup and delivery. Constraints (4.14) is symmetry breaking constraints for legs, with $M_h^L = min\{|O|, \frac{2Q_h}{min_{i \in N^P}\{p_{hi}\}}\} = min\{|O|, 2Q_h\}$. The lowest possible value for p_{hi} when visiting a pickup node is one, hence the latter part becomes $2Q_h$. Constraints (4.15) break symmetry between nodes at the same geographical location by forcing the helicopter to visit the node with a higher index before a node with lower index. Fuel flow is taken care of by constraints (4.16), with $M_{hij}^F = F_{hi}^{max} + F_{hij} - F_{hj}^{min}$. The maximum and minimum limit for fuel at every node are set by constraints (4.17).

The passenger flow for pickup and delivery nodes is given by constraints (4.18) and (4.19). Constraints (4.20) ensure that the helicopter is empty at the end of a trip. Upper and lower limits for seat capacity are given by constraints (4.21) and (4.22), and upper and lower limits of number of passengers picked up are given by constraints (4.23). Constraints (4.24) and (4.25) force all orders to be completed, while constraints (4.26) ensure that a helicopter delivers the same number of passengers as it picked up.

Time flow constraints are given by (4.27) and (4.28), with $M_{hij}^{T1} = T_j^{max} - T_{ij}^F - T_{hij}$ and $M_{hij}^{T2} = T_i^{max} + T_{ij}^F + T_{hij}$. Constraints (4.29) force pickup nodes for an order to be visited before delivery nodes. Upper and lower time limits for nodes are given by constraints (4.30) and (4.31). Constraints (4.32) - (4.34) ensure that simultaneous presence of helicopters at an installation is not allowed at any time. Binary restrictions for variables are given by constraints (4.35) - (4.38).

4.2 Preprocessing and symmetry

The arc-flow model contains a lot of symmetry due to several helicopters, multiple legs, and distribution of passengers to transport. This results in many possible arcs to be considered and potentially very long computational time needed. Because of this, it is of interest to reduce symmetry and remove as many arcs as possible. This section presents how several arcs are removed from the model and how time windows at heliports have been reduced.

The mathematical model already includes constraints (4.14) which limit the symmetry regarding leg number for each helicopter by only allowing a leg number to be used if the previous leg number has been used. We further break symmetry with constraints (4.15), which remove possible combinations of nodes at the same geographical location. A helicopter must visit the node with the higher index number first. Because of the indexing of nodes, this also ensures that the helicopters visit delivery nodes before pickup nodes.

The number of arcs can be reduced by removing arcs that the helicopters are not allowed to traverse, as seen in Table 4.2:

Table 4.2: List of arcs that can be removed due to not being legal to traverse.

i	j	Further explanation
N_h	$s(h)$	From all nodes to start nodes.
N^P	$e(h)$	From pickup nodes to end nodes.
$s(h)$	$N_h \| j \notin N^P$ $(l = 0)$	From start nodes to all nodes that are not pickup nodes, only on leg 0.
$s(h)$	N_h $(l \neq 0)$	From start nodes to all other nodes, on all legs that are not 0.
$e(h)$	N_h	From end nodes to all other nodes.
N_o^D	N_o^P	From delivery nodes to pickup nodes in the same order.
$N^{P,I}$	$N^{P,HP}$	From pickup nodes at installations to pickup nodes at heliport.
$N^{D,HP}$	$N^{D,I}$	From delivery nodes at heliports to delivery nodes at installations.
N^{HP}	$N^{HP} \| j \notin N_i$	From nodes at heliports to nodes at a different heliport.
$N_h \| i \notin N^{D,I}$	N^W	From all nodes that are not deliveries at installations to waiting nodes.
N^W	$N_h \| j \notin N^{P,I}$	From waiting nodes to all nodes that are not pickups at installations.

An arc can also be removed if the travel time between the two nodes violates the time windows by either resulting in too late or too early arrival. The following equations are used to remove these arcs:

$$T_{hi}^{min} + T_{ij}^F + T_{hij} > T_{hj}^{max}, \qquad h \in H, i \in N_h, j \in N_h \tag{4.39}$$

$$T_{hi}^{max} + T_{ij}^F + T_{hij} < T_{hj}^{min}, \qquad h \in H, i \in N_h, j \in N_h \tag{4.40}$$

Time windows for the nodes at heliports can also be reduced. In our problem each order has a time window for when it must be either picked up, or delivered, at an installation. Hence, only the nodes at installations are given time windows based on the orders. The corresponding nodes at the heliports have time windows initially set to the whole day, but can be limited with the use of the following equations:

$$T_{hj}^{min} = T_{hi}^{min} + T_{ij}^F + T_{hij}, \qquad h \in H, o \in O, i \in N_o^P \cap N^{P,I}, j \in N_o^D \cap N^{D,HP} \tag{4.41}$$

$$T_{hi}^{max} = T_{hj}^{max} - (T_{ij}^{F} + T_{hij}), \qquad h \in H, o \in O, i \in N_o^P \cap N^{P,HP}, j \in N_o^D \cap N^{D,I} \tag{4.42}$$

Equations 4.41 limit the start of the time window for delivery nodes at heliports, by adding the shortest possible travel time between the two nodes, which corresponds to travelling directly from i to j. In a similar fashion, equations 4.42 limit the end of the time window for pickup nodes at heliports.

5 | Matheuristic

In this chapter, our solution method for the problem studied in this thesis is presented. Our method of choice is a matheuristic that makes use of operators to change the solution, a labeling algorithm, and mixed integer programming. The reasons for choosing this method are discussed in Section 5.1. An overview of the matheuristic is given in Section 5.2, and the construction heuristic used to create an initial solution is presented in Section 5.3. Finally, the main parts of the matheuristic are presented: the operators used to change the solution in Section 5.4, the labeling algorithm in Section 5.5, and the mixed integer programming problem that assigns passengers to routes in Section 5.6.

5.1 Introduction to the matheuristic

The arc-flow model presented in Chapter 4 can be solved by implementing it directly into a commercial solver such as Gurobi or Fico Xpress. For a problem as complex as the one studied in this thesis, this would yield poor results, as the computational time would be too high to be useful with more than a few orders. The number of extensions increases the computational complexity significantly, and especially the possibility to split orders makes the problem notably harder to solve. Parragh et al. (2014), which was discussed in Subsection 3.1.7, state that with their branch-and-price algorithm, they "are able to solve a number of instances to optimality, but instances of realistic size (with more than 40 requests) are still out of reach". As a result of this, they decided to develop a VNS algorithm in addition to the exact method. Both Sierksma and Tijssen (1998), who include splitting of orders, and Moreno et al. (2005) also find that heuristics are needed to provide good solutions to realistic-size problems fast. Similarly to the problem in Parragh et al. (2014), the problem studied in this thesis is difficult to solve to optimality quickly, especially as it includes extensions not present in Parragh et al. (2014), such as a heterogeneous fleet and multiple depots. Moreover, as described in Section 3.4, this problem includes new extensions that are not found in other HRPs. These increase the problem complexity even further. Nevertheless, the arc-flow model has been tested, and results are shown in Section 6.2. The results show that the computational time increases significantly with only a small increase in the number of orders in the instances. Because of this, solution methods other than implementing the mathematical model into a solver are needed.

Many of the heuristics reviewed in Chapter 3 have in common that they are based on removing and inserting orders to find better solutions. This has worked for problems that include splitting of orders as well, like in Parragh et al. (2014) and Haddad et al. (2018). A problem with this approach is that when the number of orders reaches a high level, the number of possible insertion points for the orders increases

significantly, especially with splitting allowed. If the computational time is to be kept short, one cannot search through all the insertion points needed to find a good solution. In the HRP, however, orders are picked up at, or delivered to, the same set of installations. An installation can have several orders related to it, and a helicopter can serve more than one order at at an installation during a visit. Because of this, a helicopter route in the HRP always has an associated flight sequence that is shorter than or equal to the sequence of orders. Furthermore, the restriction that a passenger can travel a maximum of L^{max} legs, limits the trip length of a helicopter to $2L^{max}$ installations. If $L^{max} = 2$, a person that is picked up at the heliport must be delivered at the second installation or earlier, and a person that is picked up at an installation can visit at most one installation before the heliport. With a limit on the length of the flight sequence of trips, the number of possible flight sequences is reduced even more. Using the shorter flight sequence as the basis for changes in a solution should therefore reduce the number of changes needed to find a solution that is optimal, or close to optimal.

When flight sequences are used to make changes to solutions instead of routes, there needs to be an additional step in the heuristic. Flight sequences are not unique in the same way as routes, and a single flight sequence can generate numerous routes. How many routes that are generated depend on the number of orders present at the installations and their time windows. After routes for a flight sequence have been generated through a route generation procedure, the last step is to assign passengers to the routes. This could be done heuristically, but if the number of routes is sufficiently small, the allocation of passengers can be done quickly by solving a MIP model. Information from the solution of the MIP model is then used to make changes to the flight sequences in the solution.

In the arc-flow model from Section 4.1, every order have multiple nodes, both pickup and delivery nodes. One node is located at the installation and one at each heliport, making the total number of nodes for an order equal to the number of heliports plus one. With the heuristic approach described above, this is no longer necessary, and the total number of nodes can be reduced. The orders are now represented by a single node, located at the installations, while the heliports are represented by their own heliport nodes. This is illustrated in Figure 5.1, which shows an example with two heliports, one installation and one order. In the left part of the figure, an order requiring delivery at the installation is included, and results in a single order node. The same is the case in the right part, but with a pickup node. Each of the two heliports has its own node.

In addition, we introduce a new type of node called “dummy nodes”, that are discussed in detail in Section 5.5. One dummy node is created for each installation, and they are used to visit an installation without handling any orders, or performing any pickups or deliveries.

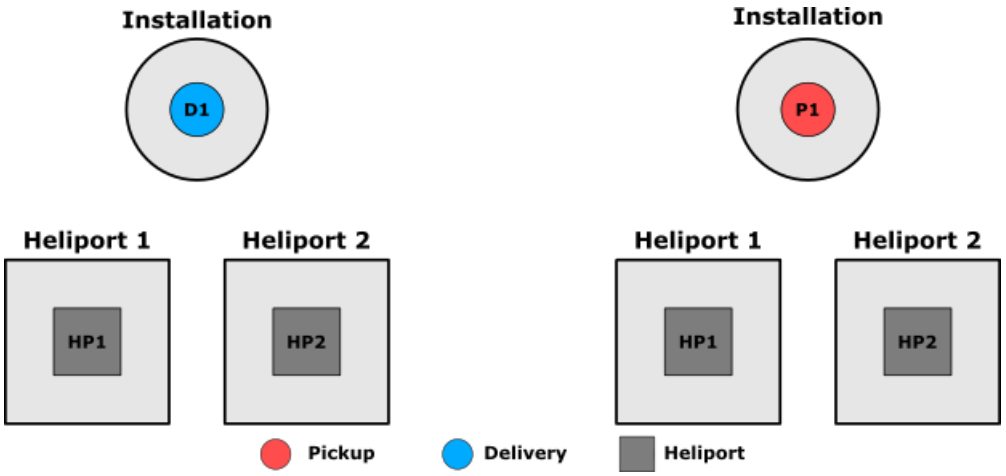


Figure 5.1: An illustration of pickup, delivery, and heliport nodes with one installation and two heliports for the matheuristic.

5.2 Overview of the matheuristic

Our solution method for the problem studied in this thesis is illustrated in Algorithm 5.1. The algorithm is initiated by constructing an initial solution, s^{init}, with a construction heuristic in the "ConstructionHeuristic" function, explained in detail in Section 5.3. Solutions from the construction heuristic are guaranteed to be feasible with regards to all restrictions in the problem, and the best solution from multiple runs is returned as s^{init}. Both the best known solution, s^{best}, and the best feasible solution, $s^{F,best}$, are set equal to the initial solution. $\mathcal{S}^{best}$ is a set of solutions used to store the best solutions found throughout the search, and is initialised empty. The set of potential good routes, $\mathcal{R}^{pot}$, is also initialised with the routes included in s^{init}, $\mathcal{R}^{s^{init}}$. $\mathcal{R}^{pot}$ is used to store all routes that have been used in a good solution, namely s^{best} or $s^{F,best}$. The argument for keeping the old routes in the pool of routes to choose from, is that the routes have previously been useful, and could be so again after changes have been made to the flight sequences. We initialise and update a set of operator parameters, $\mathcal{P}$, and which operator to use is based on these parameters. This set contains a counter, α, an upper limit for the counter, $alpha^{max}$, upper limits for the use of each operator, $\alpha_1 - \alpha_5$, and a weight for two of the operators, τ. These parameters are updated throughout the search, based on whether a new s^{best} is found or not.

The main part of the algorithm is then started, and a single iteration is described by line 9-33. Each iteration includes using an operator to change flight sequences, generating new routes with a labeling algorithm, solving a MIP model, and potentially updating solutions if a better solution is found. First, based on the last solution found, s, and the values of the parameters in $\mathcal{P}$, a change is made to one or more flight sequences in the solution using the "ChangeFlightSequences" function. This function is presented in Section 5.4, and returns a set of changed flight sequences, $\mathcal{F}$. We then use a labeling algorithm, the "LabelingAlgorithm" function, to generate all feasible routes, R^{gen}, from the flight sequences in $\mathcal{F}$. A detailed

description of the labeling algorithm is given in Section 5.5. The best combination of routes and the distribution of passengers are then found by using the "SolveMIP^{WSP}" function. A MIP model which includes all routes in both $\mathcal{R}^P$ and $\mathcal{R}^{gen}$ is solved. This is one of two versions of the MIP model used in the matheuristic. The main difference between them is that this one does not include constraints for simultaneous presence of helicopters at installations, from now on referred to as simultaneous presence. The two versions of the MIP model are explained in Section 5.6.

Algorithm 5.1 Overview of the matheuristic.

1: s^{init} = ConstructionHeuristic()
2: $s^{best} = s^{init}$
3: $s^{F,best} = s^{init}$
4: $\mathcal{S}^{best} = \emptyset$
5: $\mathcal{R}^{pot} = \mathcal{R}^{s^{init}}$
6: $\mathcal{P}(\alpha, \alpha^{max}, \alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \tau)$
7: $\mathcal{P}$ = UpdateOperatorParameters($\mathcal{P}$)
8: **while** Time limit not reached **do**
9: $\mathcal{F}$ = ChangeFlightSequences(s,$\mathcal{P}$)
10: $\mathcal{R}^{gen}$ = LabelingAlgorithm($\mathcal{F}$)
11: $\mathcal{R}^{MIP} = \mathcal{R}^{pot} \cup \mathcal{R}^{gen}$
12: s = SolveMIP^{WSP}($\mathcal{R}^{MIP}$)
13: **if** $c(s) < c(s^{best})$ **then**
14: $s^{best} = s$
15: $\mathcal{R}^{pot} = \mathcal{R}^{pot} \cup \mathcal{R}^{s}$
16: **end if**
17: **if** PotentiallyFeasible(s) **then**
18: s = SolveMIP($\mathcal{R}^s$)
19: **if** s is feasible $\wedge$ $c(s) < c(s^{F,best})$ **then**
20: $s^{F,best} = s$
21: **end if**
22: **end if**
23: **if** $\alpha = \alpha^{max}$ **then**
24: $\mathcal{S}^{best} = \mathcal{S}^{best} \cup s^{best}$
25: s = ConstructionHeuristic*()
26: $s^{best} = s$
27: $\mathcal{R}^{pot} = \mathcal{R}^{s}$
28: **end if**
29: $\mathcal{P}$ = UpdateOperatorParameters($\mathcal{P}$)
30: **if** Penalty criterion met **then**
31: $\mathcal{S}^{best} = \mathcal{S}^{best} \cup s^{best}$
32: UpdatePenalty(Penalty criterion)
33: **end if**
34: **end while**
35: **return** $s^{F,best}$

After obtaining a solution s from the MIP model, we check whether the objective value of s, $c(s)$, is better than the objective value of s^{best}, $c(s^{best})$. If this is the case, we set s^{best} to be s and update $\mathcal{R}^{pot}$ by adding the new routes used in the solution, R^s. The "PotentiallyFeasible(s)" function, described in Section 5.6, is used to check whether s could be a new $s^{F,best}$, and if so, the routes from s, $\mathcal{R}^s$, are used when solving the second version of the MIP model in the "SolveMIP" function. If the resulting solution, s, is feasible and has

a lower objective value than the current best feasible solution, it is set as the new $s^{F,best}$.

After α^{max} iterations have been performed without obtaining a new s^{best}, we create a new solution with the construction heuristic, update $\mathcal{R}^P$, store s^{best} in $\mathcal{S}^{best}$. The use of $*$ indicates that fewer iterations are run in the construction heuristic than when the initial solution is constructed. The algorithm then continues by searching for better solutions with this new solution as the starting point. This is done to prevent the matheuristic from getting stuck with a solution that it is not able to improve any further. α^{max} should therefore be given a value that allows a sufficient number of improvement operators to be used to efficiently search for better solutions, while not getting stuck at a single solution. The parameters in $\mathcal{P}$ are updated using the "UpdateOperatorParameters" function, described in Section 5.4.

In both versions of the MIP model there is a penalty associated with not completing the orders. After certain criteria in the search are met, the penalty is updated using the "UpdatePenalty" function, based on the criterion met. The use of penalty is explained in Subsection 5.6.3.

Using the classifications given by Archetti and Speranza (2014), the matheuristic fits into the class of decomposition approaches. The matheuristic can be decomposed into three parts: a heuristic part that changes the flight sequences, an exact labeling algorithm that generates routes from the flight sequences, and an exact MIP model that assigns passengers to the routes. Fuel feasibility is taken care of in the heuristic search, leg feasibility both in the search and the labeling algorithm, time feasibility during the labeling algorithm, and simultaneous presence and capacity of helicopters in the MIP model.

5.3 Construction heuristic

In order for the matheuristic to search for better solutions, it needs to have an initial solution to start from. The first step is therefore to run a construction heuristic to find a starting solution. Although the starting solution does not have to be feasible, we force the construction heuristic to find only feasible solutions. This is done to reduce the risk of really bad starting points for the algorithm, and to ensure that a feasible solution is found, even for the hardest instances to solve.

Two main approaches were considered for the construction heuristic: one where flight sequences are constructed and routes generated from the flight sequences, and one where routes are generated directly. The first of the two is comparable to how the main algorithm works, but can be difficult to use when making an initial solution. Without any knowledge on what time an installation should be visited, or if orders at the same installation can be served at the same time or not, it can be hard to construct flight sequences that are time and capacity feasible and provide a good starting solution. For this reason, a randomised greedy heuristic, based on inserting orders in the first feasible position, is used. To reduce the complexity of the

construction heuristic and provide an initial solution fast, we do not allow splitting of orders. When inserted, and order must be handled completely. An overview of the construction heuristic is given by Algorithm 5.2.

Algorithm 5.2 Construction heuristic.

```
1: R^C = ∅
2: for all HP in Heliports do
3:     O^HP = All orders handled by HP
4:     Randomise(O^HP)
5:     Start new route R
6:     while O^HP ≠ ∅ do
7:         for all o ∈ O^HP do
8:             if FeasibleInsertion(o, R) then
9:                 R = R ∪ o
10:            end if
11:        end for
12:        if No order inserted then
13:            if At installation then
14:                R = R∪ HP
15:            end if
16:            if At heliport then
17:                R^C = R^C ∪ R
18:                Start new route R
19:            end if
20:        end if
21:    end while
22: end for
23: return R^C
```

First of all, the empty set of complete routes, $\mathcal{R}^{\mathcal{C}}$, is defined. Then, we make use of the knowledge that in a good solution the heliports mostly serve the installations close to the heliport. We therefore arrange the orders into sets for each heliport, $\mathcal{O}^{HP}$, where all orders in a set is served from that heliport. The rest of the construction heuristic is executed for each heliport, HP, separately. We start by assigning all orders handled by HP to $\mathcal{O}^{HP}$ and randomising it. Next, a route $\mathcal{R}$ starting at HP is created. Then, while there are orders left in $\mathcal{O}^{HP}$, we attempt to insert order o at the back of $\mathcal{R}$. If it is not feasible to insert any orders in the route, it is checked whether the last insertion is a heliport or an order. If it is an order, a heliport is inserted and we continue to try inserting orders. When the last insertion is a heliport and it is not possible to extend $\mathcal{R}$ any more, it is added to $\mathcal{R}^{\mathcal{C}}$. A new route $\mathcal{R}$ is then created, and the algorithm continues until all orders for all heliports are included in a route.

5.4 Changing flight sequences

To be able to search the solution space and find better solutions to the matheuristic, changes to the solutions have to be made. We do this by making changes to the flight sequences of the routes that are part of the solution. Whereas heuristics that remove and insert orders have a direct relationship between insertions and the cost of a solution, there is no such relationship for changes in a flight sequence and the cost of a solution. Although inserting an installation can allow a previously incomplete order to be handled completely, time

windows for other orders may no longer fit with the route. This makes it harder to evaluate moves in the solution space. In order to search the solution space as efficiently as possible, we therefore make changes to flight sequences both based on information from the current solution and randomly. Changes are made by a total of seven different operators, which are explained in this section. Generally, the operators are not allowed to make changes that results in two of the same installation next to each other or more than $2L^{max}$ installations in the flight sequence of a trip. The first operator is used in every iteration of the algorithm, after a different operator has changed a flight sequence. The next four operators try to improve the solution based on observed information, and the last two operators make random changes. This way, we both make changes that are assumed to improve the solution, as well as explore new flight sequences with the random changes.

5.4.1 Operators

Change heliports

In our problem we allow for multiple heliports to be used in a flight sequence, and we therefore need a way to change the heliports in the flight sequences. The operator checks all heliports in the flight sequence, and changes a heliport if it results in reduced distance flown. The start and end heliports can also be changed, but both must be changed to the same heliport in order to satisfy the requirement of starting and ending a flight sequence at the same heliport. Changing a heliport in a flight sequence is always beneficial if it results in a shorter flight distance. The helicopter is still able to reach all time windows for the different orders in the route, as a change of heliports only reduces time consumption from one installation to the heliport, and from the heliport to the next installation. We allow for waiting at the heliports, which means that this operator only reduces the distance flown, and thereby lower transportation costs, while still being able to handle the same orders. The reduced time consumption could, however, result in the helicopter being able to handle even more orders later in the route.

Remove unused installation

The remove unused installation operator removes installations where no orders are handled from a flight sequence, and returns the new flight sequence without these installations. This is only possible if there is a dummy node at the installation, a node where no passengers are picked up or delivered. Dummy nodes are discussed more in Section 5.5. Installations that do not have any orders related to them are only used for waiting, which is not allowed in a feasible solution, and should therefore be removed from the flight sequence. Whenever removing an installation results in two heliports or two of the same installation in a row, one of them is also removed. If the installation is visited at the first or the last leg of a trip, removing it always results in a shorter distance travelled while still being able to handle the same orders, and thus a better solution. When the installation is located in the middle of a trip, removing it does not always lead to a better solution. This is because the time windows at the next installations may no longer fit without the

waiting time.

Insert installation for incomplete order

This operator inserts the installation u related to an order o that is not handled completely in the current solution. An overview of the operator is given in Algorithm 5.3:

Algorithm 5.3 Insert installation for incomplete order.

input: s
$\mathcal{F} = \emptyset$
for all f in $\mathcal{F}^s$ **do**
    Insert u in f in a feasible position so that $\Delta d(f)$ is minimised
    if $|\mathcal{F}| < K^{IO} \quad \vee \quad \Delta d(f) < \max_{f' \in \mathcal{F}} \Delta d(f')$ **then**
        $\mathcal{F} = \mathcal{F} \cup f$
        if $|\mathcal{F}| > K^{IO}$ **then**
            Remove f' for which $\Delta d(f') = \max_{f' \in \mathcal{F}} \Delta d(f')$ from $\mathcal{F}$
        end if
    end if
end for
Return $\mathcal{F}$

Installation u is inserted in a flight sequence f so that the additional distance flown, $\Delta d(f)$, is minimised. For an insertion to be feasible, u must be in a position where o can be handled, depending on whether it is a pickup or delivery, and visited within the time window of o. This is done for every f in $\mathcal{F}^s$, the set of all flight sequences in the current solution s. Inserting an installation for o does not guarantee that o is handled completely and it could lead to other orders not being handled completely because of time windows and capacity constraints. The installation is therefore inserted in multiple flight sequences. To limit the number of routes generated from the flight sequences, only the K^{IO} flight sequences with the lowest $\Delta d(f)$ are returned. A flight sequence f, with the new installation included, is added to the set of new flight sequences, $\mathcal{F}$, if there are fewer than K^{IO} flight sequences in $\mathcal{F}$, or the distance flown in f is shorter than the maximum distance flown in $\mathcal{F}$. The flight sequence with the highest Δd in $\mathcal{F}$ is then removed if $|\mathcal{F}| > K^{IO}$.

Remove short flight sequence

Using the information that the cost of chartering a helicopter is high, and therefore makes up a large part of the total transportation cost, we expect a good solution to utilise each flight sequence as much as possible. Therefore, this operator removes a flight sequence by moving all its installations to other flight sequences. For a flight sequence, f', to be removed it must satisfy the condition of $|f'| \leq K^{RSF}$. Choosing a low value for K^{RSF} reduces the number of possible flight sequences to be removed, but also limits the number of flight sequences used in the labeling algorithm, and thus how many routes that are generated. Choosing the size of K^{RSF} is therefore a trade-off. Using the operator to remove a flight sequence f' is shown in Algorithm 5.4. u is an installation to be removed from f', and f is a flight sequence in the set of flight sequences from the current solution $\mathcal{F}^s$. To be able to insert u without changing the original flight sequences, we make the

copy f^{new}. The added distance by inserting u in f^{new} is $\Delta d(f^{new})$. The flight sequence with the lowest added distance is f^{best}, and this is added to the set of flight sequences, $\mathcal{F}$, that is returned at the end of the algorithm. To ensure the possibility of inserting multiple installations into the same flight sequence, f^{best} is also added to $\mathcal{F}^s$.

Algorithm 5.4 Remove short flight sequence

```
input: s
F = ∅
for all u ∈ f' do
    Δd(f^best) = 0
    for all f ∈ F^s do
        f^new = f
        Insert u in f^new so that u can be reached within the same time window and Δd(f^new) is minimised
        if Δd(f^best) = 0  ∨  Δd(f^new) < Δd(f^best) then
            f^best = f^new
        end if
    end for
    F = F ∪ f^best
    F^s = F^s ∪ f^best
end for
Return F
```

Generate routes for current flight sequences

In the MIP model, the only routes that are considered are the routes in $\mathcal{R}^{pot}$ and the routes generated from the last run of the labeling algorithm. When the routes in $\mathcal{R}^{pot}$ changes, the routes in the solution may no longer be the optimal route for its flight sequence. This operator is used to generate routes for the flight sequences used in the current solution to see whether small changes in its routes are beneficial, for example if an installation is unnecessary. To limit the number of routes generated, we do this separately for each flight sequence in the current solution.

Random swap

The random swap operator is used to swap two locations in different positions, either in different flight sequences or within the same flight sequence, and returns the changed flight sequences. Between two flight sequences we only allow swapping of installations, but within the same flight sequence we allow swapping an installation for a different installation or a heliport. For a swap to be feasible, the installations to be swapped must be different. If, after the swap, the same installation or two heliports occurs two times in a row, one of them is removed.

Insert random installation

The insert random installation operator chooses a random flight sequence and inserts a random installation at a random position. In this operator we also make use of the multi-trip insertion operators from François et al.

(2016). The multi-trip insertion operators consider the insertion of both the installation and heliports, which is done by creating multiple flight sequences. When inserting an installation between two other installations the following flight sequences are created when multi-trip insertion operators are used:

1. Only inserting the installation between the two installations.
2. Inserting the installation between the two installations, and a heliport before the inserted installation.
3. Inserting the installation between the two installations, and a heliport after the inserted installation.
4. Inserting the installation between the two installations, and a heliport both before and after the inserted installation.

The use of the multi-trip operators are shown in Figure 5.2, which shows all the flight sequences created from the original flight sequence.

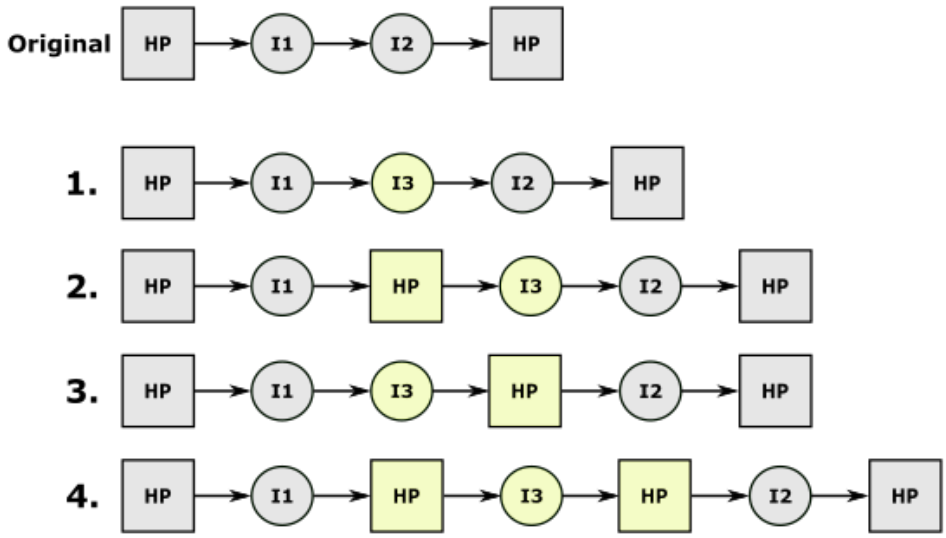


Figure 5.2: An illustration of the flight sequences created when using multi-trip insertion operators.

If the installation is inserted either directly after or before a heliport, the operator only creates two flight sequences. One flight sequence with only the installation inserted, and one flight sequence with the installation and a heliport inserted, either before or after the installation depending on where a heliport already exists.

5.4.2 Combining the operators

The operators are used in a specific order, except for the change heliports operator which is used in every iteration. An overview of how the operators are used can be found in Algorithm 5.5, and is further explained in this subsection. The functions correspond to using the operator with the same name.

Algorithm 5.5 Combining the operators.

```
1: input: s, P(α, α^max, α_1, α_2, α_3, α_4, α_5, τ)
2: if α < α_1 then
3:     F = RemoveUnusedInstallation(s)
4: else if α < α_2 then
5:     F = InsertInstallationForIncompleteOrder(s)
6: else if α < α_3 then
7:     F = RemoveShortFlightSequence(s)
8: else if α < α_4 then
9:     F = GenerateRoutesForCurrentFlightSequences(s)
10: else if α < α_5 then
11:     n^rand = RandomNumber[0, 2τ^init)
12:     if n^rand < τ then
13:         F = SwapRandom(s)
14:     else
15:         F = InsertRandomInstallation(s)
16:     end if
17: end if
18: F = ChangeHeliports(F)
19: return F
```

To keep track of which operator to use, we have the counter α. The first operator used is the remove unused installation operator, and it is used until $\alpha = \alpha_1$. Inserting an installation for an incomplete order is then performed until $\alpha = \alpha_2$, the remove short flight sequence operator until $\alpha = \alpha_3$, and generating routes for current flight sequences until $\alpha = \alpha_4$. For the random swap and insert random installation operators there is a shared upper limit α_5. Which of the two that is used is based on a random number drawn, n^{rand}, and a weight τ. After obtaining a set of flight sequences, $\mathcal{F}$, from one of the other operators, the change heliport operator is used. $\mathcal{F}$ is then returned.

The operator parameters in $\mathcal{P}$ are updated in the "UpdateOperatorParameters" function, as shown in Algorithm 5.6. Whenever s^{best} is updated, the parameters are also updated. Otherwise, α is increased by one. An updated s^{best} can be a caused by either an operator resulting in a better solution, or a new solution constructed by the construction heuristic. In this case, *alpha* is reset to zero, and the operator limits are updated based on information from the new solution, s. The first limit applies to the remove unused installation operator, and is set equal to the number of flight sequences in s containing unused installations, denoted by $|\mathcal{F}^{unused}|$. α_2 is set equal to the sum of the previous limit, α_1, and the number of incomplete orders, $|\mathcal{O}^{inc}|$. The rest of the limits are updated as shown in Algorithm 5.6, where $|\mathcal{F}^{short}|$ is the number of flight sequences shorter than K^{RSF} locations, and $|\mathcal{F}^s|$ is the number of flight sequences in s. $alpha^{rand}$

is a fixed parameter for the number of consecutive iterations where one of the two random operators is used before a new solution should be constructed.

The weight τ is reset to its initial value, τ^{init} if the construction heuristic has been used to create a new solution. τ is increased by a fixed number, $\Delta\tau$, whenever the swap random operator results in a new best solution. Likewise, it is decreased by $\Delta\tau$ whenever the insert random operator results in a better solution. To prevent using only one of these operators we have an upper and lower limit for τ, τ^{max} and τ^{min}.

Algorithm 5.6 Updating the operator parameters in $\mathcal{P}$.

input: s, $\mathcal{P}(\alpha, \alpha^{max}, \alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \tau)$
if s^{best} updated **then**
    $\alpha = 0$
    $\alpha_1 = |\mathcal{F}^{unused}|$
    $\alpha_2 = \alpha_1 + |\mathcal{O}^{inc}|$
    $\alpha_3 = \alpha_2 + |\mathcal{F}^{short}|$
    $\alpha_4 = \alpha_3 + |\mathcal{F}^{s}|$
    $\alpha_5 = \alpha_4 + \alpha^{rand}$
    $\alpha^{max} = \alpha_5$
    if Constructed new solution with construction heuristic **then**
        $\tau = \tau^{init}$
    else
        if Swap random operator used **then**
            if $\tau + \Delta\tau < \tau^{max}$ **then**
                $\tau = \tau + \Delta\tau$
            end if
        else if Insert random installation operator used **then**
            if $\tau - \Delta\tau > \tau^{min}$ **then**
                $\tau = \tau - \Delta\tau$
            end if
        end if
    end if
else
    $\alpha = \alpha + 1$
end if
$\mathcal{P} \leftarrow \alpha, \alpha^{max}, \alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \tau$
return $\mathcal{P}$

5.5 Labeling algorithm

In order to solve the MIP model, routes have to be generated from the changed flight sequences. A labeling algorithm is used to find all possible routes from a flight sequence. The labeling algorithm can be formulated as a shortest path problem with resource constraints (SPPRC), as explained by Irnich and Desaulniers (2005). In a SPPRC, the objective is to find the shortest path through a directed graph $G(\mathcal{N}, \mathcal{A})$ where $\mathcal{N}$ is the set of nodes, and $\mathcal{A}$ is the set of arcs, so that no resource constraints are violated. At every step through the algorithm we have a label, $\mathcal{L}$, that includes information on the current node, how much is used of each resource, and what nodes have previously been visited. A general labeling algorithm starts

by generating a label for every resource feasible move from the start node, and adding them to a set of unprocessed labels, $\mathcal{U}$. Throughout the algorithm the first label in $\mathcal{U}$ is removed from the set. Then, a new label is generated for every possible move by extending the label using resource extension functions. If the new label is resource feasible it is added to $\mathcal{U}$, otherwise it is discarded. When a label reaches the end node, it is added to the set of finished labels. This process continues until the set of unfinished labels is empty.

The labeling algorithm described above creates all possible routes from start to finish in the graph of nodes, and results in a high number of finished labels if the graph is big and resource constraints not tight. In order to reduce the number of labels, dominance is used. A label is said to be dominated by another label if the two labels are at the same node and the resource consumption is equal to or worse than the other label for all resources. If a label is dominated by another label it is discarded, as it never is better to use a dominated label over the one that dominates it.

5.5.1 Generating routes with a labeling algorithm

The labeling algorithm used to generate routes is shown in Algorithm 5.7, and follows the same structure as the general labeling algorithm presented by Irnich and Desaulniers (2005). For the HRP in this thesis, $\mathcal{N}$ in $G(\mathcal{N}, \mathcal{A})$ consists of order and heliport nodes. The labeling algorithm is used to generate routes from the graph $G(\mathcal{N}, \mathcal{A})$ for a given flight sequence. Figure 5.3 illustrates the graph of a flight sequence with two installations included.

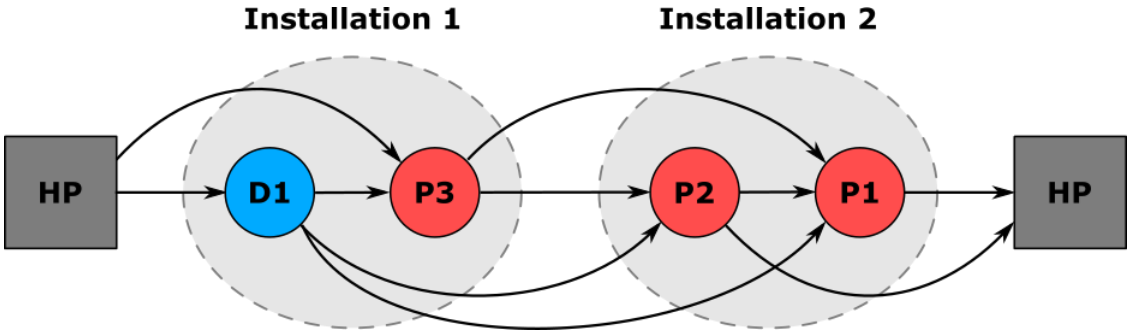


Figure 5.3: The graph for a flight sequence with two installations and two orders at each installation.

At each installation multiple orders can be included as nodes in the graph, but which orders that can be included is limited because of L^{max}. This is further explained in Subsection 5.5.2. The arcs between the nodes, $\mathcal{A}$, correspond to feasible extensions. In a flight sequence the locations must be visited in a fixed order, and therefore all arcs in $G(\mathcal{N}, \mathcal{A})$ are directed and either between nodes at the same installation or installations next to each other. Fuel feasibility is checked before the labeling algorithm is used, and the flight sequence is discarded if it is not fuel feasible for any helicopter. Both the capacity constraints and the simultaneous presence constraints are taken into consideration after routes have been generated. We consider a fixed flight sequence, which means that the travelling costs are equal for all routes generated from the same flight sequence. Travelling costs can therefore be calculated before the labeling algorithm. Time is

the only resource taken into account in our labeling algorithm, as a route is only feasible if each order node can be reached within its time window. The labels used in our labeling algorithm is therefore $\mathcal{L}(i, T_i, X_i)$, where i is the current node and X_i is the sequence of nodes already visited. The time resource window, T_i, consists of both the earliest arrival time, t_i^{min}, and the latest arrival time, t_i^{max}, at node i. Feasible extensions and the resource extension functions are explained in more detail in Subsection 5.5.3.

Algorithm 5.7 Labeling algorithm used to generate all routes for a given flight sequence.

```
1: U = {L(b, T_0, X_0}
2: R^C = ∅
3: while U ≠ ∅ do
4:     L(i, T_i, X_i) = RemoveFirstElement(U)
5:     for all arcs (i, j) ∈ A do
6:         L' = L
7:         if L'(j, T_j, X_j) is a resource feasible extension then
8:             if j = e then
9:                 R^C = R^C ∪ L'
10:            else
11:                U = U ∪ L'
12:            end if
13:        end if
14:    end for
15: end while
16: return R^C
```

The labeling algorithm in Algorithm 5.7 starts with an initial label, $\mathcal{L}(b, T_0, X_0)$, for the starting heliport at the beginning of the flight sequence, which is added to the set of unprocessed labels, $\mathcal{U}$. The set of completed routes, $\mathcal{R}^{\mathcal{C}}$, is initialised empty. Like the general labeling algorithm, the algorithm runs until there are no more labels in $\mathcal{U}$. The first element of $\mathcal{U}$ is removed, and all extensions are checked for feasibility using the resource extension functions and the requirements provided in Subsection 5.5.3. If feasible, the label is added back to $\mathcal{U}$, otherwise it is discarded. Whenever a label reaches the end heliport, e, the route is added to $\mathcal{R}^{\mathcal{C}}$, which is returned at the end of the algorithm.

A difference between the labeling algorithm in Algorithm 5.7 and the general labeling algorithm is the use of dominance between labels. If time windows were not a part of the problem, routes with more orders would be preferable over routes with fewer orders. With time windows included, a route with fewer orders could have more left of the time resource window, and consequently have more feasible extensions than the route with more orders. The only way dominance can occur in our problem is in the case where two routes, r and r' are identical, except for r' having one additional order included. If this additional order does not reduce the time resource window, the time resource window of r and r' would be the same. In this case r' would dominate r. In our problem, if two orders of the same type are going to or from the same installation at approximately the same time, they are represented as one order. This means that the most likely scenario for dominance to happen would be if two orders of different types were given similar time windows at the same installation, meaning that it is preferable that pickups and deliveries happen at the

same time. Since the start of the time windows are generated at random points spanning most of the day, the probability of orders of different types having time windows that do not reduce time resource windows is seen as low. Dominance between labels is therefore not included in the labeling algorithm. If, however, orders were generated in a way where this would be expected to occur more frequently, dominance between labels should be included.

5.5.2 Feasible nodes at each location

The orders that can be considered as nodes at each installation in the flight sequence, and included in $G(\mathcal{N}, \mathcal{A})$, depend on the number of legs between the installation and the heliports. Based on the description given on the maximum number of legs, L^{max}, in Section 5.1, the following applies for any given installation in a flight sequence:

- Orders requiring pickup are included if there are at most L^{max} legs to the next heliport.
- Orders requiring delivery are included if there are at most L^{max} legs from the previous heliport.

Because of this, several orders can be excluded based on the position of the installations and the number of legs to the heliports. Figure 5.4 illustrates an example with three installations and $L^{max} = 2$.

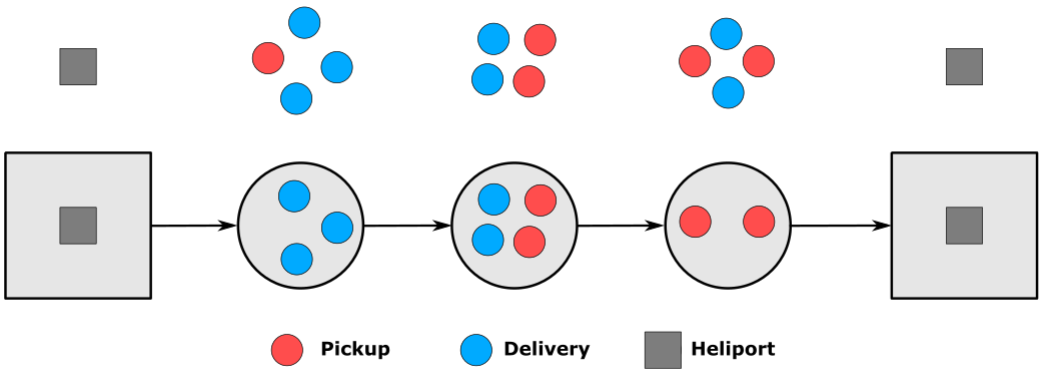


Figure 5.4: An illustration of the nodes included at each location in a flight sequence, with $L^{max} = 2$.

For the first installation, only orders that require delivery to the given installation are considered, while only orders requiring pickup are considered at the third installation. For the second installation, both pickups and deliveries are included as there are no more than two legs between the installation and both of the heliports. For every occurrence of a heliport we also add a corresponding heliport node to $G(\mathcal{N}, \mathcal{A})$.

As described in Chapter 2, it is not allowed to visit an installation without performing any pickups or deliveries. In other words, to stop at an installation just to wait is forbidden. The labeling algorithm discards any routes that do not visit an order node at every installation. This could be a problem if the flight sequence includes an installation that prohibits the labeling algorithm from generating a potentially good

route. To address this issue we introduce "dummy nodes". Dummy nodes are nodes that have no pickups or deliveries and have time windows that span the whole day. They can only be visited if the last visited node was at the previous installation, and the node after the dummy has to be at the next installation. It follows that if a solution uses a route with a dummy node, the installation is visited just to wait, and it is considered an unused installation. The remove unused installation operator, presented in Section 5.4, is used to remove these unused installations from flight sequences.

We want dummy nodes to be used to be able to remove unnecessary installations from the flight sequences, but originally there is no benefit to using routes with dummy nodes. If a route with a dummy node can be used in a solution, a route with an order node at the same installation could also be used if the capacity constraint are not violated, and the installation would then not be defined as an unused installation. To incentivise the use of dummy nodes whenever possible, we reduce the cost of flight sequences with an unused installation by a minimal amount. This way, routes with dummy nodes are used over an otherwise equal route when profitable, but it does not interfere with the solution in any other way.

5.5.3 Resource extension functions and feasible extensions

In this subsection we present the resource extension functions used when extending labels in the labeling algorithm, as well as the requirements for an extended label to be feasible. We only consider time window constraints in our labeling algorithm, and therefore only need resource extension functions for the earliest arrival time, t_i^{min}, and the latest arrival time, t_i^{max}, at node i. Using the same notation as in Chapter 4, the resource extension functions when extending a label from node i to j are therefore limited to the following functions:

$$t_j^{min} = max\{T_j^{min}, t_i^{min} + T_{ij}^F + T_{ij}\} \tag{5.1}$$

$$t_j^{max} = min\{T_j^{max}, t_i^{max} + T_{ij}^F + T_{ij}\} \tag{5.2}$$

T_j^{min} and T_j^{max} are the earliest possible and latest possible arrival time at node j, respectively. T_{ij}^F is the fixed time used at node i when travelling to node j. T_{ij} represents the same time consumption as T_{hij} in the arc-flow model. The helicopters in the real life problem have similar cruising speeds, and we have therefore simplified our model by setting the cruising speed constant and equal for all helicopters. As a result, the resource extension function for time is equal for every helicopter, and the labeling algorithm has to be used only once in each iteration. If the cruising speeds were different, routes would have to be generated for every helicopter separately.

Additionally, the sequence of nodes currently visited is updated with the inclusion of the new order node:

$$X_j = X_i \cup j \tag{5.3}$$

For an extension from i to j to be feasible following requirements must be satisfied:

$$t_j^{min} \leq T_j^{max} \tag{5.4}$$

$$t_j^{max} \geq T_j^{min} \tag{5.5}$$

Figure 5.5 illustrates how the resource extension functions work for a flight sequence with two installations, and a total of three orders included.

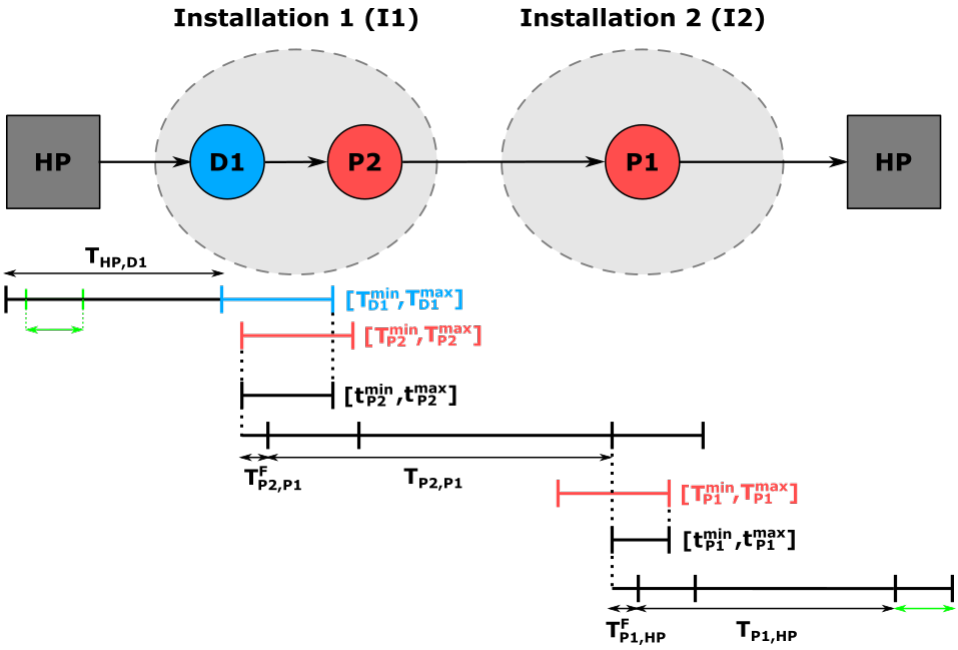


Figure 5.5: An illustration of the resource extension of the time windows for a flight sequence with two installations.

The first order in the route, $D1$, can always be included, with $t_{D1}^{min} = T_{D1}^{min}$ and $t_{D1}^{max} = T_{D1}^{max}$. Extending to the next order node, $P2$, is also feasible, but the time resource window at $P2$ is smaller than for the previous node. As the node is located at the same installation both $T_{D1,P2}^{F}$ and $T_{D1,P2}$ are zero, and the new time resource window has $t_{P2}^{min} = T_{P2}^{min}$ and $t_{P2}^{max} = T_{D1}^{max}$. When extending to the next node, $P1$, at the next installation, both the fixed time at the previous installation and the travel time is included. At node $P1$ we have $t_{P1}^{min} = T_{P2}^{min} + T_{P2,P1}^{F} + T_{P2,P1}$ and $t_{P1}^{max} = T_{P1}^{max}$, where $T_{P2,P1}^{F}$ is equal to the fixed time at an installation, T^I. The extension is feasible since neither equation 5.4 nor equation 5.5 are violated, as seen in the figure. At the last node, the heliport node, the time resource window is the same, but with the fixed time and travel time added, which is illustrated by the green time range in Figure 5.5. This results in the green time range at the first heliport, which illustrates the time range where a helicopter must start its trip

in order to handle all orders within their time windows.

We also require that if j is an order node at the same installation as i, then $i > j$ must be true. This in order to avoid symmetries at the same installation, and reduce the number of labels created. The extension from $D1$ to $P2$ is feasible since a delivery node has a higher index value than a pickup node, as in the arc-flow model. The introduction of dummy nodes can potentially result in a massive increase in labels as the time windows of dummy nodes do not consume much of the time resource. Because of this we limit the number of dummy nodes in a route to K^D. For an extension to be feasible it is therefore a requirement that there exists at most K^D dummy nodes in the label.

5.6 Assigning passengers to routes

In this section we describe the last part of the matheuristic, the assignment of passengers to the routes created by the labeling algorithm. First, we present the model without the constraints for simultaneous presence at installations. Second, we complete the full model by adding constraints for simultaneous presence. Finally, we discuss how penalty is used to guide the search towards the desired solutions.

5.6.1 The relaxed version of the mixed integer programming model

Below, the relaxed version of the MIP model is presented, which is solved in the "SolveMIP^{WSP}" function in Algorithm 5.1. The model follows the same style of notation as the arc-flow model in Section 4.1, and all notation needed for the relaxed model is defined below.

R is the set of all routes sent to the model, both from the labeling algorithm and previous solutions, corresponding to $\mathcal{R}^{MIP}$ in the main algorithm. Routes that contain order o are included in R_o, and O contains all orders. All helicopter types that have the necessary range to travel route r are included in H_r. S_r is the set of trips on route r, and I_{rs} is the installations included in trip s of route r, except for installations with a dummy node. Since an installation can be visited multiple times in a trip, we also need V_{rsu} to know have many visits a helicopter makes to the same installation u on trip s on route r. O_{rsuv} is the set of orders included at visit v at installation u on trip s on route r.

O_{rl} is the set of orders handled on leg l of route r. This includes all orders where the passengers have to be onboard the helicopter on leg l to be delivered to their destination if they are assigned to route r. O_{rl} is linked with another set, L_r. L_r is the set of constraining legs for route r, the legs where the passenger capacity of the helicopter is the most constrained. For a route r and every trip s, the leg l where O_{rl} is the largest is a constraining leg. These are not the only possible constraining legs for route r. To illustrate the concept of constraining legs we use an example with $L^{max} = 2$. In this case, every trip handling a pickup or a delivery has between one and three constraining legs. Trips with only deliveries or pickups have one

constraining leg, which is the one where O_{rl} is the largest. This constraining leg is always the first or the last on the trip, depending on whether deliveries or pickups are present. If trips have both pickups and deliveries, the first and last leg are always constraining, except for when there are two installations and pickups are completed at the first installation and deliveries at the second. In this case, there are four possibilities:

1. Only pickups at the first installation and only deliveries at the second → Leg two is the only constraining leg.
2. Both pickups and deliveries at the first installation, and only deliveries at the second → Leg one and leg two are constraining.
3. Only pickups at the first installation, and both pickups and deliveries at the second → Leg two and leg three are constraining.
4. Both pickups and deliveries at both installations → All legs, one, two, and three, are constraining.

To illustrate some of the cases explained, we show the constraining legs for two flight sequences, one with three installations and one with two installations, in Figure 5.6.

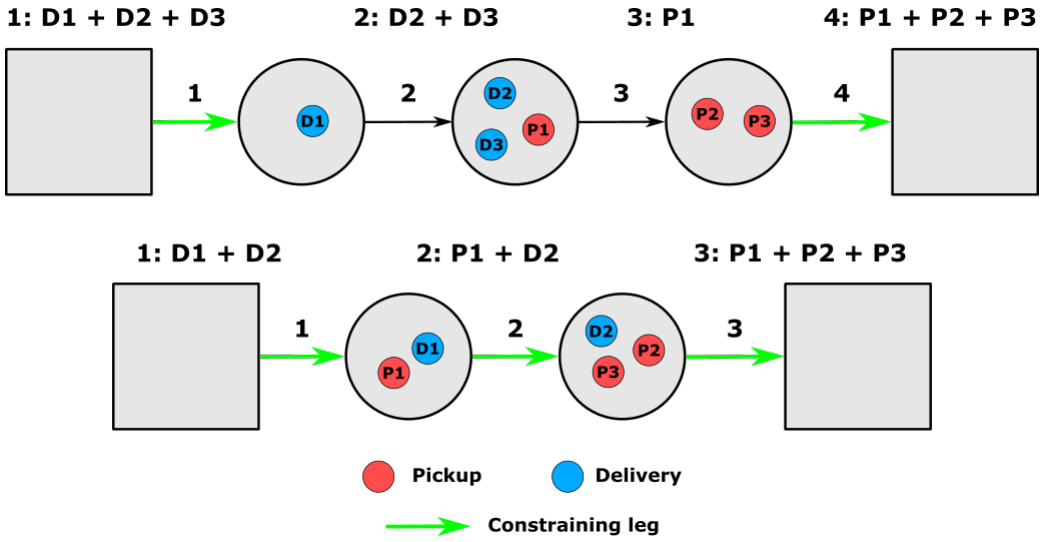


Figure 5.6: An illustration of constraining legs for flight sequences with three and two installations with $L^{max} = 2$.

Above each leg we have included the orders handled by the helicopter on that leg. As seen for the first flight sequence, the first leg is constraining since it contains both the orders handled on the second leg, and an additional order. The same logic applies for the last leg. The number of passengers in the helicopter is always higher on the constraining legs than on other legs, and thus we only need capacity constraints for these legs. In the second flight sequence, all three legs are constraining. This is the fourth possibility listed above.

The MIP model, including notation for all sets, indices, parameters and variables used, is presented below:

Sets

R: Set of all routes

R_o: Set of all routes that include order o

H_r: Set of all helicopter types able to handle route r

L_r: Set of constraining legs for route r

O: Set of all orders

O_{rl}: Set of orders handled on leg l of route r

S_r: Set of trips on route r

I_{rs}: Set of all installations in the flight sequence of route r on trip s without dummy nodes

V_{rsu}: Set of visits to installation u on trip s on route r

O_{rsuv}: Set of orders included at visit v at installation u on trip s of route r

Indices

o: Order

r: Route

h: Helicopter

l: Leg

s : Trip

u : Installation

v: Visit

Parameters

C^F_{rh}: Fixed cost of using route r with helicopter h

C^D_{rh}: Transportation cost of travelling the total distance of route r with helicopter h

C^P: Penalty cost per passenger not transported

P_o: Total number of passengers to be transported in order o

Q_h: Seat capacity for helicopter h

T^T_{rs}: Travel time for trip s on route r

T^{max}_{rs}: Upper time limit for starting trip s on route r

T^{min}_{rs}: Lower time limit for starting trip s on route r

T^{HP}: Fixed time used at heliport

Variables

δ_{rh}: Equal to 1 if route r and helicopter h is used, 0 otherwise

p_{ro}: Number of passengers in order o transported on route r

t_{rs}: Starting time for trip s on route r

ω_o : Number of passengers in order o not transported

$$\min \sum_{r\in R}\sum_{h\in H_r}(C^F_{rh}+C^D_{rh})\delta_{rh}+\sum_{o\in O}C^P\omega_o \tag{5.6}$$

subject to

$$\sum_{o\in O_{rl}} p_{ro} \le \sum_{h\in H_r} Q_h\delta_{rh}, \qquad r\in R, l\in L_r, \tag{5.7}$$

$$\sum_{r\in R_o} p_{ro} \ge P_o-\omega_o, \qquad o\in O, \tag{5.8}$$

$$\sum_{o\in O_{rsuv}} p_{ro} - \sum_{h\in H_r}\delta_{rh} \ge 0, \qquad r\in R, s\in S_r, u\in I_{rs}, v\in V_{rsu}, \tag{5.9}$$

$$T_{rs}^{min} \sum_{h \in H_r} \delta_{rh} \leq t_{rs} \leq T_{rs}^{max} \sum_{h \in H_r} \delta_{rh}, \qquad r \in R, s \in S_r, \tag{5.10}$$

$$t_{rs} + (T_{rs}^{T} + T^{HP}) \sum_{h \in H_r} \delta_{rh} \leq t_{r(s+1)}, \qquad r \in R, s \in S_r \setminus \{|S|\}, \tag{5.11}$$

$$\sum_{h \in H_r} \delta_{rh} \leq 1, \qquad r \in R, \tag{5.12}$$

$$\delta_{rh} \in \{0, 1\}, \qquad r \in R, h \in H_r, \tag{5.13}$$

$$\omega_o \geq 0, \qquad o \in O. \tag{5.14}$$

The objective function (5.6) minimises the total cost, both related to flying the routes and chartering the helicopters, and the penalty for not completing the orders. Constraints (5.7) ensure that the helicopter capacity is not exceeded on the constraining legs. In constraints (5.8) it is stated that each order must be completed or a penalty must be taken for each passenger not transported. To make sure that a helicopter cannot visit an installation just to wait, constraints (5.9) prohibit visiting installations without picking up or delivering at least one passenger. Constraints (5.10) set the upper and lower time windows for the start of trips, and constraints (5.11) make sure that a new trip is not started before the previous trip has finished. Finally, constraints (5.12) force a maximum of one helicopter to be used on a route, and constraints (5.13) and (5.14) set the binary restrictions and lower bound for the δ and ω variables.

5.6.2 Including constraints for simultaneous presence

The relaxed version of the MIP model is solved in every iteration of the matheuristic, but it cannot guarantee feasibility with regards to simultaneous presence. If the number of orders is not very high and the orders are spread out throughout the day, the probability of a solution violating the rule of simultaneous presence is low. Adding constraints for simultaneous presence makes the model more time consuming to solve, and the full model is therefore only used to check whether a solution found with the relaxed model can be feasible when also considering simultaneous presence. If a solution, s, from the relaxed model satisfies a set of criteria in the "PotentiallyFeasible" function in Algorithm 5.1, then the full model is solved for the routes from s, $\mathcal{R}^s$. The following criteria must be satisfied:

- The solution is different from previously found solutions.
- There is no dummy node in any route.

- The maximum number of passengers not handled in any order is not higher than the difference in capacity between the largest and smallest helicopter:

$$\omega_o \leq \max_{h \in H^{Type}}(Q_h) - \min_{h \in H^{Type}}(Q_h), \qquad o \in O \tag{5.15}$$

The second criteria may not be satisfied in the situation where a dummy node is located at an installation between two other installations, and thereby only used to wait for time windows to open. This is not acceptable for a feasible solution to our problem. In the third criteria, H^{Type} is the set of all helicopter types. If equation 5.15 is satisfied there is a possibility that all orders can be completed by just changing to a helicopter with greater capacity.

To complete our MIP model, we introduce additional notation and three new constraints. The set $I^{SP}_{rr'ss'}$ is comprised of installations that have overlapping time windows for a trip s on route r and a trip s' on route r'. These are the installations where simultaneous presence of helicopters could be a problem.

Sets

$I^{SP}_{rr'ss'}$: Set of installations that have overlapping time windows on trip s on route r and trip s' on route r'

Indices

r': Route

h': Helicopter

s' : Trip

v' : Visit

Parameters

T_{rsuv} : Travel time for visit v to installation u in trip s on route r

T^I: Fixed time used at installation

Variables

$\sigma_{rr'ss'uvv'}$: Equal to 1 if visit v at installation u on trip s on route r happens at least T^I time units after visit v' at installation u on trip s' on route r', 0 otherwise

$$(t_{rs} + T_{rsuv}) - (t_{r's'} + T_{r's'uv'} + T^I) + M^{SP}_{rr'ss'uvv'}(1 - \sigma_{rr'ss'uvv'}) \geq 0,$$
$$r \in R, r' \in R \setminus \{r\}, s \in S_r, s' \in S_{r'}, u \in I^{SP}_{rr'ss'}, v \in V_{rsu}, v' \in V_{r's'u}, \tag{5.16}$$

$$\sigma_{rr'ss'uvv'} + \sigma_{r'rs'suv'v} \geq \sum_{h \in H_r} \delta_{rh} + \sum_{h' \in H_{r'}} \delta_{r'h'} - 1,$$
$$r \in R, r' \in R \setminus \{r\}, s \in S_r, s' \in S_{r'}, u \in I^{SP}_{rr'ss'}, v \in V_{rsu}, v' \in V_{r's'u}, \tag{5.17}$$

$$\sigma_{rr'ss'uvv'} \in \{0,1\}, \qquad r \in R, r' \in R \setminus \{r\}, s \in S_r, s' \in S_{r'}, u \in I^{SP}_{rr'ss'}, v \in V_{rsu}, v' \in V_{r's'u}. \tag{5.18}$$

Constraints (5.16) ensure that if two helicopters visit the same installation on some visit on some trip on

their routes, there must be a gap of T^I time units between their landings at the installation. The lower bound for the sum of the two sigmas is set in constraints (5.17), and the binary restriction for the sigma is given by constraints (5.18). For constraints (5.16) we have $M^{SP}_{rr'ss'uvv'} = T^{max}_{r's'} + T_{r's'uv'} + T^I - T_{rsuv}$.

5.6.3 Penalty cost for incomplete orders

To be able to search through the solution space more efficiently, we do not force the model to complete all orders fully. Instead, there is a penalty cost, C^P, for every passenger that is not transported. The penalty cost allows the matheuristic to work with solutions that are close to feasible and would be good if the remaining passengers were assigned to routes. For this to work efficiently, C^P has to be set to a suitable value. If it is set too high, solutions with passengers not assigned to routes are unlikely to be chosen, and potentially good solutions could be neglected. On the other hand, if the penalty cost is set too low, solutions where fewer orders are completed could be favoured too much. If the penalty cost is low enough, the model actually chooses to not transport any passengers at all, as that is cheaper than using helicopters. This is potentially a problem when few orders are included.

Although allowing both feasible and infeasible solutions could be beneficial when searching for new solutions, we ultimately want feasible solutions. Because of this, the penalty in the full MIP model is set substantially higher than in the relaxed model. This is done to force the use of helicopters with greater capacity instead of taking the penalty, and, assuming that constraints 5.16 and 5.17 are satisfied, find a feasible solution to the full problem. Another way of pushing the algorithm towards feasible solutions is to increase the penalty when certain criteria are met. In our matheuristic there are two such criteria. The first is that after a set amount of the computational time, the "UpdatePenalty" function is used to double the penalty. The second is that when the total computational time is almost up, we check whether the best solution in the set of best solutions found, $\mathcal{S}^{best}$, contains incomplete orders. If so, the "UpdatePenalty" function is used to update the current solution to the best solution in $\mathcal{S}^{best}$, and adds a new route for every incomplete order, only handling that one order. This creates a feasible solution, as long as constraints 5.16 and 5.17 are not violated. The penalty is then increased to a level where it is never profitable to have any incomplete orders, and the search continues for the remaining time. When there is a large gap in solution value between the best solution and the best feasible solution found, this can help get a new best feasible solution.

6 | Computational Study

In this chapter we present the results from tests performed on the arc-flow model and the matheuristic. The tests were run using a Lenovo NextScale nx360 M5 with two 2.3 GHz Intel E5-2670v3 12 core processors and 64 GB memory, running Linux. The arc-flow model was implemented using Mosel with Xpress version 8.8, and the matheuristic was implemented in C++, using Gurobi 9.0.2 as the solver for the MIP model. We start by explaining our choice of heliports and installations, and the generation of test instances in Section 6.1. In Section 6.2 we briefly present the results from testing the arc-flow model. The computational performance of our matheuristic is discussed in Section 6.3, by studying the effects of increasing the number of orders in the instances. Finally, we study the benefits of including several extensions in Section 6.4.

6.1 Instance generation

Test instances were generated based on information and data provided by Tieto and gathered from "Heliport.no" (Tieto Norway, 2020). The data from Tieto contains information about the orders, including origin, destination, and number of passengers in the order. "Heliport.no" was used to confirm that the chosen installations are frequently in need of transportation, and which heliport each installation is served from.

The location of the selected installations and heliports are shown in Figure 6.1, together with the location IDs of the installations. Two heliports are included, Bergen Airport Flesland (BGO) and Stavanger Airport Sola (SVG). The reason is that they are located close enough to each other to serve a set of the same installations. This is important in this thesis as our main purpose is to find potential benefits of allowing visits to multiple heliports and choosing which heliports to pickup/deliver each order at, and the benefit of splitting orders. In addition, BGO and SVG are two of the most used heliports for offshore helicopter transportation, based on observations from "heliport.no".

We include a total of 11 installations. These can all be reached from both the chosen heliports, which creates a network of nodes where every node in the network can be reached from any other node. Five of the installations are located with approximately the same distance from both heliports, namely Alvheim (ALV), Heimdal (HDA), Jotun A (JOA), Balder (BAF) and Grane (GRA). To reduce the probability of having only a single heliport serve all installations in an instance, we include, for each heliport, three more installations that are more favourable for this heliport. Gullfaks A (GFA), Troll A (TRO) and Oseberg (OSE) are all located closer to BGO, and are therefore more favourable for this heliport. Likewise, Johan Sverdrup (JSF), Gudrun (GUD) and Sleipner A (SLE) are more favourable for SVG. In the construction heuristic, and when

testing our new extensions in later sections, we use the heliport that is used for the installation in real life as the installation's home heliport. This is based on data from "Heliport.no" (Tieto Norway, 2020). We did, however, change ALV to be served from BGO instead of SVG, as we wanted a more even distribution among the heliports, and BGO is actually the closest heliport for this installation. Based on this, installations 1-4 in Figure 6.1 are served by BGO, while installations 5-11 are served by SVG.

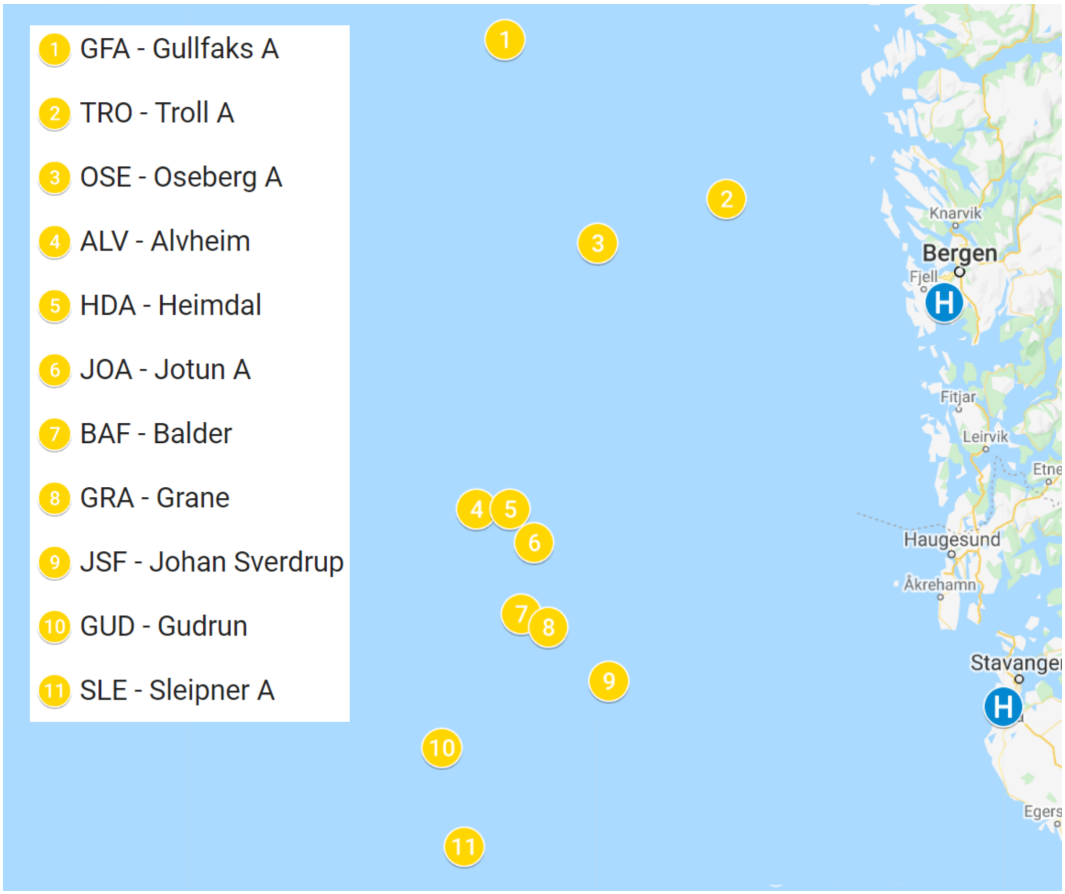


Figure 6.1: Locations of the two heliports and the chosen installations.

In our tests the heliports have the same fleet of helicopters available, which consists of two different helicopter types. The two types are "Sikorsky S92" (S92) and "Super Puma H215" (SP). Information about parameter values for the two helicopter types are gathered from the data provided by Tieto and from technical specifications listed from the manufacturers, which is "Lockheed Martin" (Lockheed Martin, 2020) for the S92 and "Airbus" (Airbus, 2020) for the SP. These parameter values are given in Table 6.1.

Table 6.1: Parameter values for the two helicopter types used in the tests.

Helicopter type	Seat capacity	Cruising speed [km/h]	Fuel capacity [litres]	Average fuel consumption [litre/km]	Chartering cost [-]	Cost per distance [-]
Sikorsky S92	19	252	2900	2.9	10 000	5.75
Super Puma H215	16	252	2300	2.8	10 000	5.00

Seat capacity, cruising speed, fuel capacity and fuel consumption are based on real values for the helicopter, but some are slightly modified. It is possible for the seat capacity for the SP to be the same as the S92, but there are also variants of the SP with a lower seat capacity. To ensure a heterogeneous fleet of helicopters we set the seat capacity lower for the SP. Both helicopters have approximately the same cruising speed, which are set equal to simplify the problem. The S92 has a higher fuel capacity, giving it more range, even with the higher fuel consumption. The values for the cost parameters are not based on real values. Because the S92 has a higher seat capacity and longer range than the SP, it is given a higher cost per distance flown. The chartering cost is the same for both helicopters, and is set much higher than the cost per distance to motivate the use of fewer helicopters.

Instances were created by randomly drawing the number of passengers for each order from a uniform distribution between 6 and 18. By having an upper limit of 18 we ensure that each order can be served by a single helicopter, and do not force splitting of orders. Whether an order should be picked up or delivered at an installation, and which installation it belongs to, were also randomly chosen. The earliest and latest possible time of pickup and delivery were set to 07:00 and 20:59, respectively. The time windows were set to one hour for all orders, and the start of each time window was drawn randomly between the earliest and latest possible time. To avoid having a huge majority of orders at a few installations, we include a rule for the maximum number of orders per installation. If $|O|$ is the total number of orders in an instance, and $|U|$ is the number of installations, then the maximum number of orders at any installation is given by $O^{max,inst} = \frac{2|O|}{|U|}$, rounded to the nearest integer. Two sets of instances are created with increasing number of orders included. The first contains instances with a number of orders ranging from one to nine. The second set contains instances starting at 20 orders, increasing by 5 orders up to 60 orders. For every number of orders a total of five instances were generated. The first set is used to test both the arc-flow model and the matheuristic, and the second set is used to study the matheuristic further.

6.2 Arc-flow results

The arc-flow model in Section 4.1 has been tested with the first set of instances, ranging from one to nine orders. The results are briefly presented in this section. For all instances a total of four helicopters were included, one of each type at both heliports. For the arc-flow model to work correctly, the maximum number of legs a helicopter can travel during a route, $\overline{L}$, must be set sufficiently high. Setting it too high, however, increases the symmetry unnecessarily and leads to higher computational time. $\overline{L}$ was set using the formula $\overline{L} = 2|O|+1$, which guarantees that no feasible solutions are removed from the solution space. The maximum number of legs personnel can be transported, L^{max}, was set to 2. The tests were ran with an upper time limit of three hours, and the results of the tests are shown in Table 6.2.

Table 6.2: Results for the arc-flow model for instances with one to nine orders.

Number of orders	Instances solved to optimality	Average optimality gap	Average computational time
1	5/5	0.0 %	0.01 s
2	5/5	0.0 %	0.04 s
3	5/5	0.0 %	0.35 s
4	5/5	0.0 %	0.79 s
5	5/5	0.0 %	186.79 s
6	5/5	0.0 %	530.45 s
7	5/5	0.0 %	3969.57 s
8	2/5	33.2 %	>7715.22 s
9	0/5	44.0 %	>10800 s

The table shows the number of instances solved to optimality, the average optimality gap and the average computational time for up to nine orders included. If no optimal solution is found within the time limit of three hours, we set the computational time for that instance to be 10800 seconds. The ">" in the table indicates that the arc-flow model requires more than three hours to solve one or more of the instances to optimality. Similarly, if no feasible solution is found within the time limit, we set the optimality gap to 100 % for that instance. As seen from the table, the instances were solved to optimality within one second at four orders and below. However, the computational time increases drastically with just a small increase in the number of orders. At only six orders the average computational time is already at around 530 seconds, and when including only a single additional order it is further increased to over 3900 seconds. With eight orders included the arc-flow model is not solved to optimality for all instances, and we have an average optimality gap of approximately 33.2 %. Furthermore, the computational time has increased to over 7700 seconds. When adding one more order, to nine orders in the instances, none of the instances are solved to optimality within the time limit.

These results, together with the arguments made in Section 5.1, motivate the need for a heuristic approach in order to study the benefits of adding the new extensions to the HRP, especially when the goal is to study instances with a greater and more realistic number of orders included.

6.3 Computational performance of the matheuristic

In this section, the computational performance of the matheuristic is discussed. There exists no method, exact or heuristic, that includes all the extensions in this problem and is able to solve it, other than the arc-flow model. Because of this there are no results or methods to use as a benchmark for our matheuristic with more than eight orders included, and the only optimality gaps available are for our first set of instances. To study the performance of the matheuristic for instances with more orders, measures such as iterations of the algorithm per second, number of routes from labeling, and variance of the objective value are used.

A maximum computational time is used as the termination criteria for the matheuristic, and was set to 20 minutes. Instances with 20, 30, 40, 50, and 60 orders were run for an hour, and the improvement in objective value was found to be relatively small after about 20 minutes for all instances. The construction heuristic performs a maximum of 10 000 iterations, and chooses the best solution found as the initial solution for the search. To avoid spending too much time on finding an initial solution, we also have an upper time limit of 60 seconds for the construction heuristic. When the construction heuristic is run again later in the matheuristic, to create a new starting solution for the search, there is a time limit of 15 seconds. For the instances with the highest number of orders, some iterations generate a high number of routes in the labeling algorithm. To prevent the MIP model from spending too much time on a single iteration, an upper limit of 15 seconds was set for its solution time. The parameters discussed in chapter 5 are listed in Table 6.3, together with the values used during the testing. These values were chosen based on testing of the matheuristic with different values for the parameters.

Table 6.3: Parameter values for the matheuristic, chosen based on testing with different values for the parameters.

Parameter	Value
L^{max}	2
α^{rand}	300
τ^{init}	0.50
τ^{max}	0.75
τ^{min}	0.25
$\Delta\tau$	0.05
K^{IO}	3
K^{RSF}	6
K^{D}	1
C^{P}	500

When making changes with the operators we allow a total of 300 iterations, α^{rand}, using either the swap random or insert random installation operators, before the search is restarted with a new solution created by the construction algorithm. This allows the matheuristic to search for improvements to the current solution, but also prevents it from being stuck at a solution where no improvements can be found.

6.3.1 Comparison to the exact solutions from the arc-flow model

The matheuristic has been tested with the instances that was solved to optimality with the arc-flow formulation, and the results are compared in this subsection. Because the arc-flow model is unable to solve instances with more than eight orders to optimality, it is expected that the matheuristic finds good solutions quickly. It can be seen from Table 6.4 that this is the case, as the optimal solution was found for almost all instances. In the few cases where the optimal solution was not found, the optimality gap was minimal, with the average gap at six orders being less than 0.2 %. The instances were initially run with a time of only a few seconds, and then time was increased if necessary. Although it is difficult to estimate from these instances

how the matheuristic performs with more orders included, the good results indicate that the matheuristic may be effective with more than eight orders as well.

Table 6.4: Results from using the matheuristic to solve the same instances as the arc-flow model.

Number of orders	Number of instances	Average optimality gap
1	5	0.0 %
2	5	0.0 %
3	5	0.0 %
4	5	0.0 %
5	5	0.0 %
6	5	<0.2 %
7	5	0.0 %
8	2	0.0 %

6.3.2 Performance for the second set of instances

The matheuristic has been tested for the second set of instances, described in Section 6.1, where the number of orders included is between 20 and 60. The algorithm was run a total of 10 times for each instance, and the results for each instance represents an average of these 10 runs. Most of the results in this subsection are presented based on the number of orders included, and represent an average of the five instances with this number of orders.

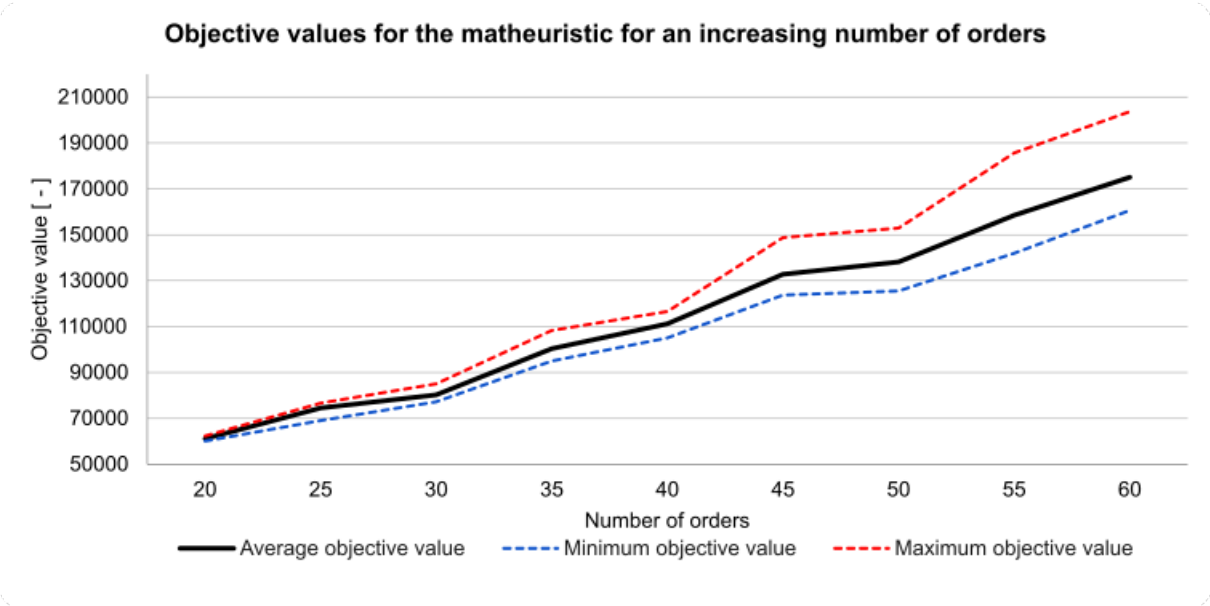


Figure 6.2: Average objective values for an increasing number of orders, with calculated minimum and maximum objective values.

The average objective value for the second set of instances is shown by the black line in Figure 6.2. To study the variation in the objective value for different runs of the instances, we calculate, for each instance,

the percentage difference between the average value of all runs and both the maximum and minimum value obtained. Then we calculate the average percentage differences for all instances with the same number of orders. These average percentages are then used to calculate the maximum objective value and the minimum objective value for different number of orders included, represented by the red and blue lines in Figure 6.2, respectively. With 20 to 40 orders there is almost no variation in the results, which means that the algorithm finds close to the same solution in every run for all five instances. This changes when the number of orders increases, and from around 45 orders and upwards there is a larger difference between the maximum and minimum objective value for the runs. For these instances there are too many orders included for the algorithm to consistently find the same solutions.

This increasing variation is further illustrated in Figure 6.3, which shows the average relative standard deviation (RSD) for an increasing number of orders. The standard deviation (SD) is calculated based on the objective values from all runs of an instance. To get the relative standard deviation for each instance we divide the SD by the average objective value. For every number of orders we calculate the average RSD based on the RSD of the five instances with that number of orders included. The average RSD increases with the number of orders. For 20 orders the average RSD is low, at approximately 1 %, and it remains stable at approximately 3-4 % from 25 to 40 orders. After 40 orders there is a bigger increase. This correlates with Figure 6.2, as both show that it gets more difficult for the algorithm to find the same solutions when the number of orders reaches 45 and above.

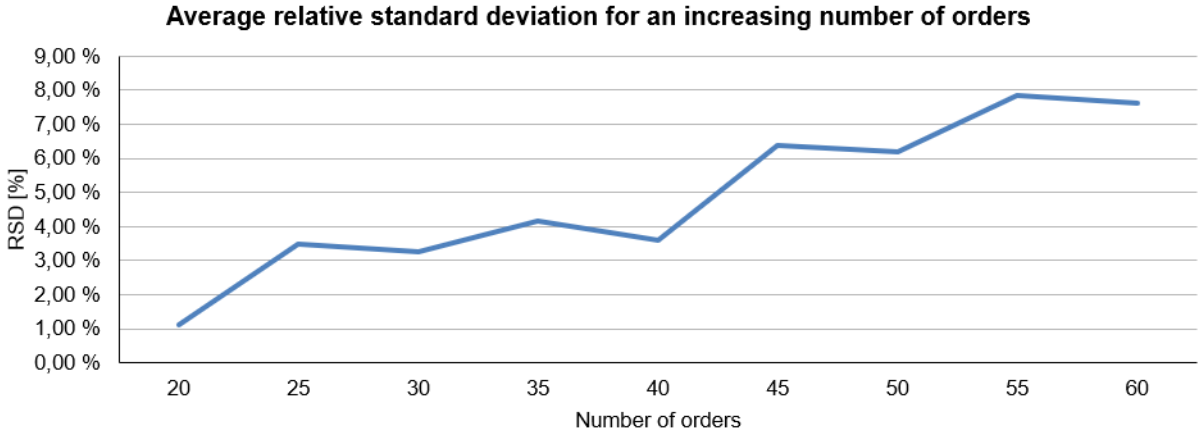
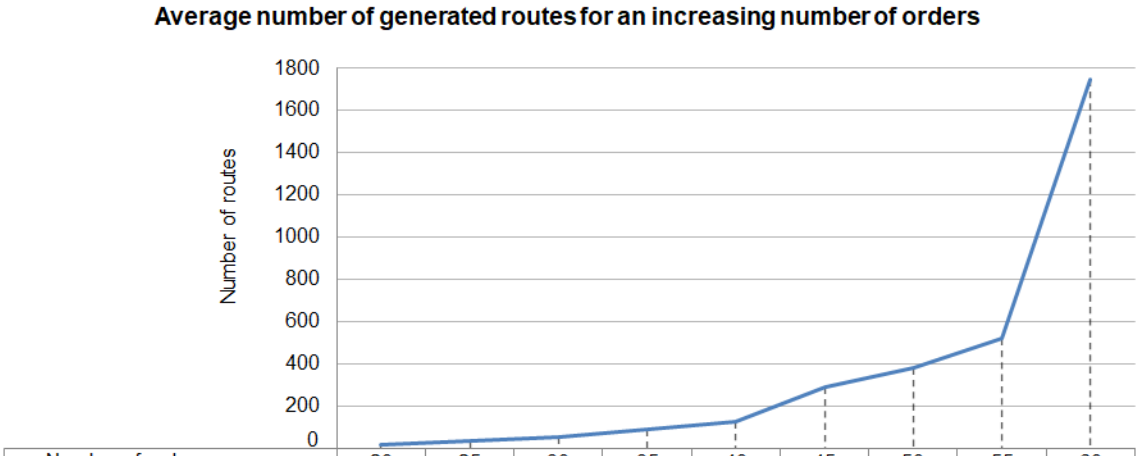


Figure 6.3: Average relative standard deviation for an increasing number of orders included in the instances.

The high number of orders leads to many routes generated in the labeling algorithm. As seen in Figure 6.4 the average number of routes, rounded to the nearest integer, is quite consistent from 20 to 40 orders. The table in Figure 6.4 also shows the average maximum number of routes generated for the different number of orders, averaged for all runs and instances. It can be seen that the average maximum number of routes generated increases as well. After 40 orders both the average and maximum number of routes increase greatly. The average number increases a lot from 55 to 60 orders, which is partly caused by the high maximum number

of routes generated. At 60 orders the maximum number of routes generated is very high, which is caused by some combination of flight sequences resulting in a large number of routes in the labeling algorithm.



Number of orders	20	25	30	35	40	45	50	55	60
Average number of routes	16	36	50	91	123	287	381	518	1 749
Average max. number of routes	197	603	987	1 689	1 788	4 354	6 680	5 721	12 301

Figure 6.4: Average number of routes from the labeling algorithm for an increasing number of orders.

With 45 or more orders included, when on average hundreds or thousands of routes are generated by the labeling algorithm, the matheuristic is not able to perform as many iterations as with fewer orders. More time is spent at both route generation and solving the MIP model, but especially the latter is time consuming with a lot of routes. Figure 6.5 shows the development of the average number of iterations per second for an increasing number of orders. As expected, the average number of iterations per second decreases as the number of orders increases, and it seems to be close to inversely proportional to the average number of routes. After 45 orders the algorithm is able to perform fewer than 1.6 iterations per second, and at 60 orders the number is down to 0.6 iterations per second. This can explain the increased average RSD of the objective values when more than 45 orders are included, as the number of iterations is too low to effectively search the solution space.

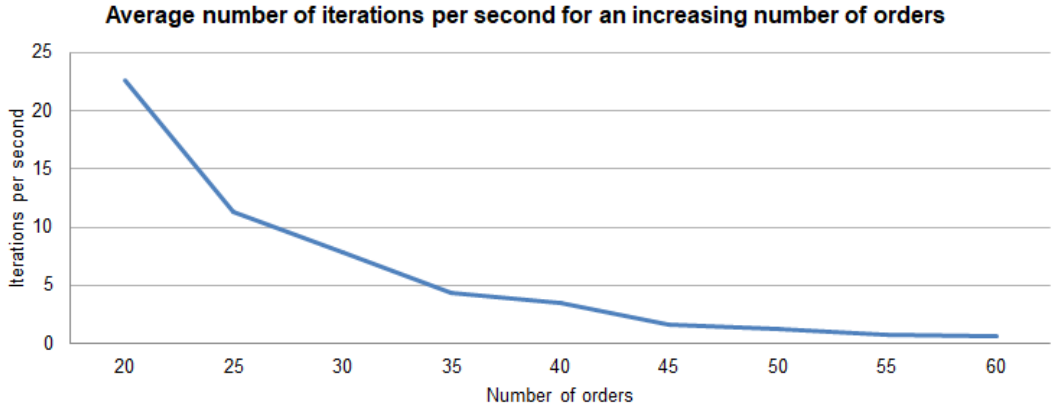


Figure 6.5: Average iterations per second for an increasing number of orders.

Even though the average number of iterations decreases and the variation in objective value increases when more orders are included, our algorithm is still able to make significant improvements to the solution throughout the search. Table 6.5 shows the average percentage improvement for the different number of orders, when comparing the best feasible solution to the initial solution, created by the randomised greedy insertion in the construction heuristic presented in Section 5.3. The average improvement stays above 25 % up til 40 orders, above 20 % between 45 and 55 orders, and only drops below 20 % at 60 orders, where it is at 17.29 %.

Table 6.5: Average improvement from the initial solution.

Number of orders	Average improvement from initial solution
20	26.50 %
25	27.89 %
30	32.20 %
35	26.00 %
40	25.15 %
45	20.97 %
50	21.78 %
55	20.42 %
60	17.29 %

6.3.3 Time windows

A parameter that can have a substantial impact on the performance of the matheuristic is the width of the time windows. When the time window for pickup, or delivery, of an order at an installation is widened, the number of possible arcs, $\mathcal{A}$, in the graph $G(\mathcal{N}, \mathcal{A})$ increases. With an increased number of arcs, the number of possible routes increases as well. To test the effect of expanding the time windows, all instances with 20, 30, and 40 orders have been run with time windows of two hours instead of one hour. The average number of routes from the labeling algorithm and number of iterations per second are compared to the the results with one hour time windows in Table 6.6:

Table 6.6: Average number of routes and iterations per seconds for one hour and two hour time windows.

	One hour time windows		Two hour time windows	
Number of orders	Average number of routes	Iterations per second	Average number of routes	Iterations per second
20	16	22.6	43	11.4
30	50	7.8	191	2.5
40	123	3.5	643	0.7

At 20 orders we see that the number of routes is almost three times as big with two hour time windows, and the number of iterations per second is halved. At 30 orders, the number of routes is almost four times as big with two hours, and iterations per second less than a third of with one hour time windows. The difference is even greater at 40 orders, where the number of routes is more than five times as large with two hour time

windows, and iterations per second only a fifth of the number with one hour time windows. This clearly shows the impact of the width of time windows, even with a moderate number of orders in the instances.

Table 6.7: Decrease in objective value and number of helicopters saved when increasing time windows to two hours.

Number of orders	Average decrease in objective value	Average number of helicopters saved
20	18.67 %	0.82
30	4.62 %	0.08
40	3.90 %	0.26

Increasing time windows clearly has an impact on the computational performance of the matheuristic, but it also contributes positively to the objective value. Table 6.7 shows the average decrease in objective value and the average number of helicopters saved when changing the time windows from one to two hours. For 20 orders, the objective value decreases by as much as approximately 19 %, and the number of helicopters needed is reduced by more than 0.8 on average. The difference is smaller at 30 and 40 orders, but there are still savings related to having wider time windows. From a cost perspective it is therefore favourable with wider time windows, but this also creates inconvenience for personnel. With wider time windows the personnel must be available for transportation for a longer time, and could potentially be waiting for a several hours before being transported. Width of time windows is therefore a trade-off between better solutions, and the inconvenience of passengers as well as difficulty of solving the problem.

6.4 Benefits of the extensions

In this section, the benefit of adding three of our extensions to the HRP is discussed. Two of them are problem specific and are referred to as the new extensions. These two extensions, allowing helicopters to use multiple heliports during a route and choosing which heliports to use for pickup/delivery of each order, are not previously found in the literature. They were, however, tested for instances with up to seven orders included in our project report, Oppedal and Thorstein (2019), and were found to show promise with regards to reducing costs. The last one, splitting of orders, has been studied previously, but not in a problem as rich as the one studied in this thesis. All of these extensions increase the complexity of the problem, and if their inclusion does not improve the solution to the problem, the only thing they do is making the problem harder to solve. The objective of this section is therefore to find out whether including the three extensions could be worth it with a more realistic number of orders than in Oppedal and Thorstein (2019).

We start off by studying the benefits of including the two new extensions. To do this we run tests on the same instances as previously, but without the two new extensions included, and we refer to this as WE (without extensions). In this case, a helicopter can only use its home heliport, and all orders at an installation must be served from the same heliport. Splitting orders between helicopters is still allowed. Routes may now have to be changed because the helicopters are not be able to reach the same time windows if they

cannot travel to the closest heliport during a trip. Furthermore, when orders have to be handled from a specific heliport, orders cannot be included in routes starting from the other heliport. This can result in both longer distance flown and more helicopters being used. The results for WE are compared to the objective values with all extensions included, referred to as the standard case, and the average objective values are shown in Figure 6.6.

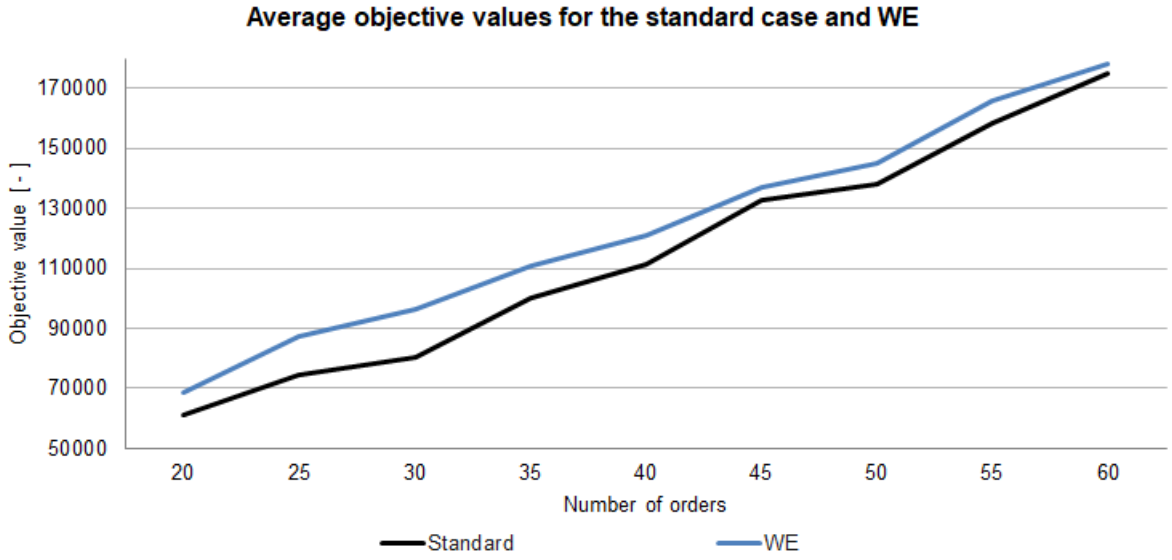


Figure 6.6: Average objective values for the standard case and WE.

In this figure the black line corresponds to the objective values for the standard case, and WE is the blue line. As expected, the average objective value is lower for the standard case. The standard case is a relaxation of WE, and the the objective value of the exact solution to the standard case is consequently equal to or better than the objective value for WE. For this reason, a well-performing heuristic should also find at least as good solutions in the standard case. To further study the benefits of the two new extensions we use Figure 6.7, which shows the percentage increase in objective value for WE compared to standard case, and Table 6.8, which shows the average number of helicopters saved.

From Figure 6.7 we can see that the difference in costs for WE reaches a maximum at 30 order with an increase of approximately 20 %. The objective value is on average 9.3 % higher, and the smallest increase is 1.9 % at 60 orders. It can be seen that the difference in cost is decreasing after 30 orders, and there could be several reasons for this. Firstly, as explained in the previous section, the performance of the matheuristic in the standard case seems to fall for instances with more than 45 orders. If the matheuristic is struggling to find a solution close to the optimal one, it could be less likely that solutions that benefit from the new extensions are found.

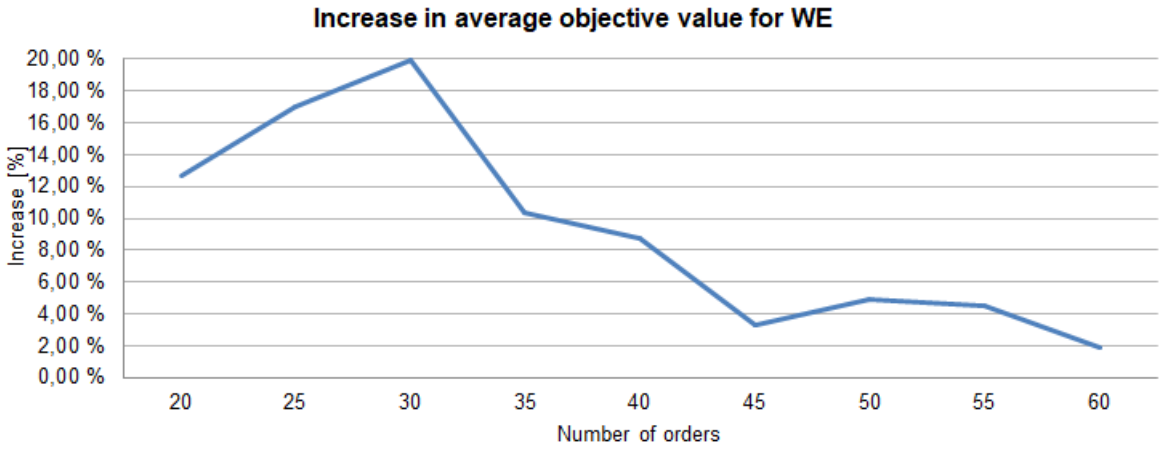


Figure 6.7: Average increase in objective value for WE compared to the standard case.

Secondly, having more orders could reduce the benefit by itself. Allowing helicopters to use multiple heliports and choosing which heliports to use for pickup/delivery of each order, let the helicopters operate more efficiently by letting them fly to the closest heliport and pick up, or deliver, the passengers where it fits the rest of the route best. This reduces the flying distance and the number of helicopters needed. When the number of orders in an instance reaches a high level there are enough time windows open at all time so that a helicopter can serve installations efficiently using only its home heliport. In this situation it is also more likely that an order belonging to an installation can be served from the installations home heliport, as the order probably fits in one of the existing routes using that heliport. With a high number of orders, the standard case therefore becomes more similar to WE.

Lower objective values for the standard case, compared to WE, can be caused by either fewer helicopters used, shorter distance travelled, or a combination of both. The difference in objective values can be used as an implication that a solution is better in a real life scenario, but the costs of chartering and using helicopters used in this thesis are not based on real data. Because of this, the difference in objective values cannot be used as a definitive measure of how much better a solution is. Instead, the number of helicopters used can be measured. If the number of helicopters used in a solution is lower than in another solution, the first solution is expected to be better in real life as well. Table 6.8 shows the average number of helicopters saved when comparing the standard case to WE. At both 25 and 30 orders we are on average able to save more than one helicopter, and as many as 1.30 at 30 orders. In a real life scenario this could provide huge savings for the offshore oil companies. The potential to save helicopters is generally quite high, and is close to 0.5 or above for almost all number of orders. The average is approximately 0.69 helicopters saved.

Table 6.8: Average number of helicopters saved when comparing the standard case to WE.

Number of orders	Average number of helicopters saved
20	0.60
25	1.16
30	1.30
35	0.88
40	0.70
45	0.32
50	0.54
55	0.46
60	0.24

Seeing that excluding the two extensions increase the objective value and number of helicopters needed, it is of interest to study which one that contributes the most to the increase. We therefore performed tests on the same instances, but without allowing a helicopter to use multiple heliports, which we refer to as OH (one heliport). In this case, splitting of orders and choosing which heliports to use for pickup/delivery of each order are included. The results are shown as the red line in Figure 6.8.

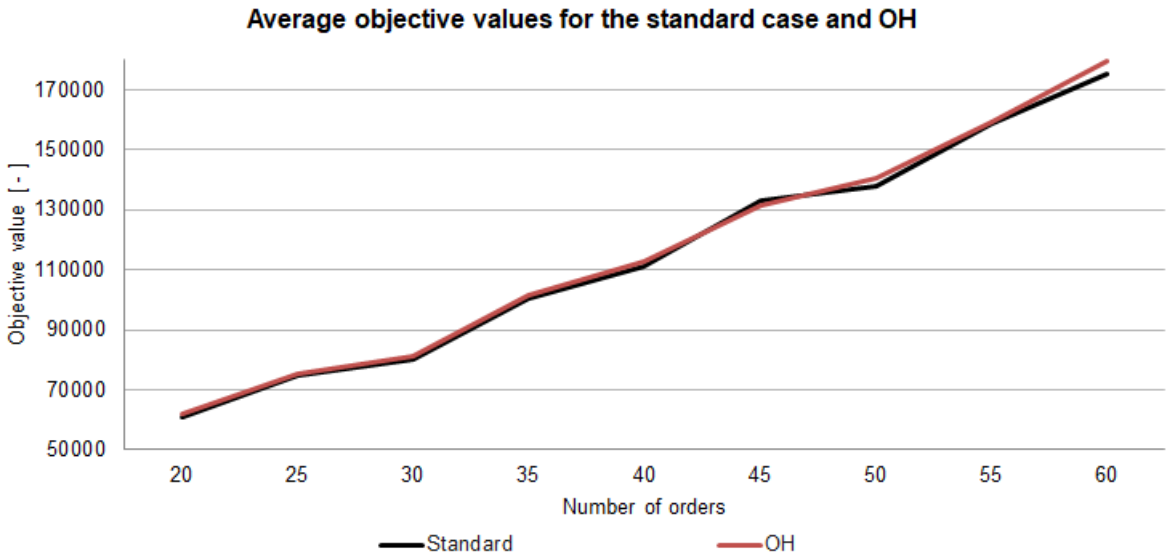
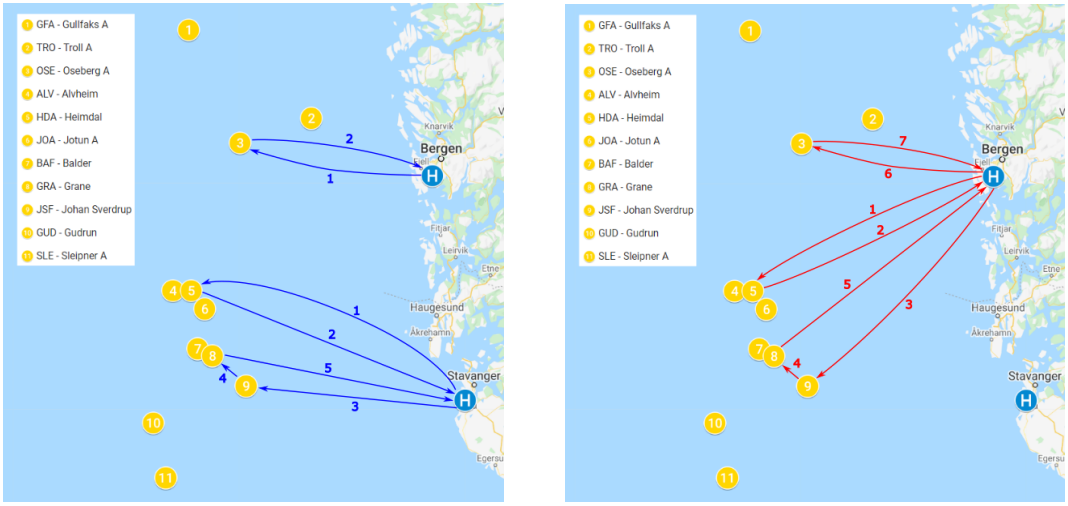


Figure 6.8: Average objective values for the standard case and OH.

It can be seen that these results are almost identical to the standard case, the black line, just slightly deteriorated. The differences are so small that the average number of helicopters saved is approximately 0.05. This indicates that allowing a helicopter to use multiple heliports does not show much promise by itself, and that allowing the model to choose where to pick up, or deliver, an order must be included to achieve significant benefits from the new extensions.

To illustrate how the extensions give better solutions we use Figure 6.9 and Figure 6.10. These are the flight sequences from the results of testing an instance with four orders included, which we are able to verify the optimality of with the arc-flow model.

Figure 6.9: Flight sequence of the one helicopter needed in the standard case with four orders included.



(a) Flight sequences of the two helicopters needed for WE.

(b) Flight sequence of the one helicopter needed for OH, which is longer than for the standard case.

Figure 6.10: An illustration of the flight sequences of the results without the new extensions included. An instance with four orders is used.

Figure 6.9 shows that in the standard case we only need a single helicopter to handle all orders in the instance, and the helicopter utilises both heliports to do this as cost efficient as possible. Without the extensions included the flight sequences change. As seen in Figure 6.10a we need two helicopters to serve the same four orders, and including the extensions could therefore reduce overall costs greatly. In the case of OH, Figure 6.10b, when only removing the ability to visit multiple heliport during a route, all orders are handled by one helicopter, and the helicopter uses only its main heliport to do so. This gives a slight increase

in total distance flown, but the increase in objective value is not much compared to having both extensions included.

The last of the three extensions, splitting of orders, was tested with the two new extensions included. The average increase in objective value, compared to the standard case, is shown in Figure 6.11.

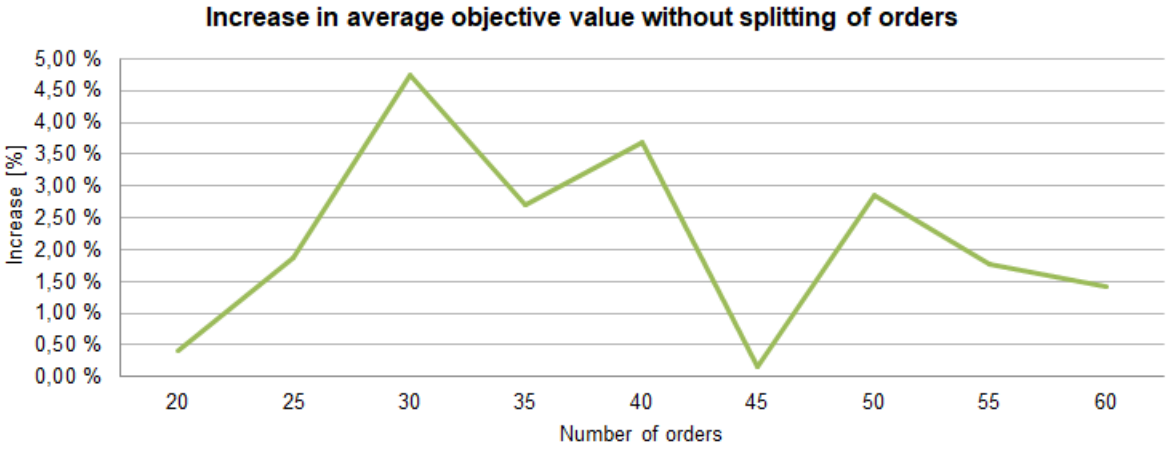


Figure 6.11: Average increase in objective value when not including splitting of orders, compared to the standard case.

Generally, there is an increase in the average objective value, but the differences are small. The average increase is approximately 2.2 %, with a maximum of 4.8 % at 30 orders and a minimum of 0.2 % at 45 orders, and the average number of helicopters saved is close to zero for almost all instances. The difference is smaller than one might expect, and there are several possible reasons. Since each order can be delivered by a single helicopter, it is never profitable to split an order if it can be delivered by one helicopter without changing anything else in the route. We therefore need several orders with similar time windows for splitting to be efficient. This could be one of the reasons why the value of splitting increases from 20 to 30 orders. The benefit of splitting remains approximately the same from 30 to 50 orders, except for at 45 orders. There is a decrease from 50 to 60 orders, which could be caused by the model being much easier to solve without splitting included. From Section 6.3 we see that the matheuristic struggles to consistently find good solutions with more than 45 orders included, but the model without splitting does not have that problem to the same extent. The model without splitting is able to carry out, on average, 96 % more iterations per second compared to the full model.

7 | Concluding Remarks

The problem studied in this thesis is the planning of transportation of personnel to and from offshore installations, with the use of helicopters from heliports situated along the coast of Norway. Personnel going to or from the same installation at the same time are organised into orders. The objective is to minimise the total cost, consisting of the helicopter chartering cost and cost per distance flown, while completing all orders and adhering to all regulations. A flight sequence is defined as the sequence of heliports and installations visited by a helicopter during a day. When replacing the installations in a flight sequence with the orders handled at each installation, we get what is defined as a route. We present a model for the offshore helicopter routing problem (HRP) with two new extensions. The first is to allow a helicopter to visit multiple heliports during a route, and by this being able to utilise other heliports if it results in reduced distance flown. The second extension is that we have the option to choose which heliports to use for pickup/delivery of each order, making it possible for a heliport to serve all installations within the range of the helicopters. As seen in the literature review there are no studies that include these extensions. To solve the problem, a decomposition matheuristic has been developed. In the matheuristic, changes are made to flight sequences instead of routes, exploiting characteristics of the problem. Routes are then easily generated from the flight sequences using a labeling algorithm, and a mixed integer programming model is used to assign passengers to the routes.

Based on the tests performed, the matheuristic seems to perform well. Compared to the results from solving the arc-flow model with a commercial solver, the matheuristic is able to find the optimal solution for almost every instance, with only a very small optimality gap for a few instances. As the arc-flow model is not able to solve instances with more than eight orders to optimality, we are only able to use it as a benchmark for this number of orders. Further testing of the matheuristic has been done with more orders, ranging from 20 to 60 orders included. Significant improvements are made to the initial solution created by the randomised greedy construction heuristic, with the objective value generally reduced by approximately 25 % at the end of the search. The results show that there is little variation in the objective values for different runs of the same instances with up to 40 orders, but with 45 orders or more the variation increases. The reason is likely to be the increased average number of routes generated in the labeling algorithm with more orders, which in turn leads to fewer iterations performed per second.

We find that including the new extensions results in significant savings. When they are not included, the objective value is on average increased by approximately 10 %. More importantly, the average number of helicopters saved is also relatively high with the extensions included. The average for all instances is approximately 0.7 helicopters saved, and we are able to save at least one helicopter for several instances. Based

on tests, allowing helicopters to use multiple heliports during a route does not seem to be that beneficial by itself, and the option to choose which heliport to use for each order must be included to achieve significant benefits. Splitting of orders does not result in a notable difference in the number of helicopters used.

The matheuristic seems to be a promising solution method based on the tests performed in this thesis, but we are not able to identify exactly how close to the optimal objective value it gets. Nevertheless, we find that including both the new extensions results in a significant decrease in solution value. More notably, the extensions result in helicopters saved, which could provide huge savings for the offshore oil companies, and therefor be advantageous when planning the transportation of personnel to offshore installations.

8 | Future Research

Even though the problem studied in this thesis is a rich variant of the HRP and includes multiple extensions, there are several possible problem features that can be studied further. This chapter discusses some of those features, both in terms of how the problem can be extended and how it can be solved differently.

There are still aspects of the real life HRP that have not been included in this problem, and some of them could be explored further. Uncertainty regarding weather, for example, is important for the planning of helicopter routes. That uncertainty could be included in a stochastic model using weather data as a parameter, and routes could be chosen based on the probabilities of bad weather. Another way to take the weather uncertainty into consideration could be to incentivise flexibility and robustness in a solution, and thereby prevent slight delays in the schedule from making the rest of the route infeasible. This could be done by rewarding additional waiting times at heliports, or requiring there to be some time left of the time windows of the orders after completing them.

The planning horizon is also something that could be changed, expanding it from one to several days. A helicopter could be allowed to end a day at a different heliport than it started at, and only return to its home heliport at the end of the planning horizon. Other possible changes to the problem are adding orders between installations or adding orders that are time connected, meaning that one has to be served a set amount of time before the other.

Although the helicopters in this problem are heterogeneous, we use their similarities to simplify the problem. More heterogeneous helicopters could be used, for example by having different cruising speeds. Instead of having a fixed cruising speed for each helicopter type, one could also have a varying speed depending on the weight of the helicopter and weather conditions. Additionally, the fuel consumption could be set as a variable of speed and helicopter weight to create a more realistic scenario.

As mentioned in Chapter 6, it is difficult to know exactly how well the matheuristic performs due to the lack of exact methods to compare with. It would therefore be of interest to develop an exact method that could be used as a benchmark for our and future heuristics. Other heuristics, such as a population based heuristic, could be developed as well, as they have shown promising results for both VRPs and HRPs.

Reference List

Abbasi-Pooya, A. and Husseinzadeh Kashan, A. (2017). "New mathematical models and a hybrid Grouping Evolution Strategy algorithm for optimal helicopter routing and crew pickup and delivery". In: *Computers & Industrial Engineering* 112, pp. 35–56.

Aghezzaf, E.-H., Raa, B., and Van Landeghem, H. (2006). "Modeling inventory routing problems in supply chains of high consumption products". In: *European Journal of Operational Research* 169.3, pp. 1048–1063.

Airbus (2020). *H215*. Airbus. URL: `https://www.airbus.com/helicopters/civil-helicopters/heavy/h215.html` (visited on 06/02/2020).

Alonso, F., Alvarez, M. J., and Beasley, J. E. (2008). "A tabu search algorithm for the periodic vehicle routing problem with multiple vehicle trips and accessibility restrictions". In: *Journal of the Operational Research Society* 59.7, pp. 963–976.

Archetti, C., Guastaroba, G., and Speranza, M. G. (2014). "An ILP-refined tabu search for the Directed Profitable Rural Postman Problem". In: *Discrete Applied Mathematics* 163.1, pp. 3–16.

Archetti, C. and Speranza, M. G. (2012). "Vehicle routing problems with split deliveries". In: *International Transactions in Operational Research* 19.1, pp. 3–22.

Archetti, C. and Speranza, M. G. (2014). "A survey on matheuristics for routing problems". In: *EURO Journal on Computational Optimization* 2.4, pp. 223–246.

Azi, N., Gendreau, M., and Potvin, J.-Y. (2010). "An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles". In: *European Journal of Operational Research* 202.3, pp. 756–763.

Azi, N., Gendreau, M., and Potvin, J.-Y. (2014). "An adaptive large neighborhood search for a vehicle routing problem with multiple routes". In: *Computers & Operations Research* 41, pp. 167–173.

Baldacci, R., Bartolini, E., Mingozzi, A., and Valletta, A. (2011). "An Exact Algorithm for the Period Routing Problem". In: *Operations Research* 59.1, pp. 228–241.

Baldacci, R., Battarra, M., and Vigo, D. (2008). "Routing a Heterogeneous Fleet of Vehicles". In: *Operations Research/ Computer Science Interfaces Series*. Ed. by B. Golden, S. Raghavan, and E. Wasil. Vol. 43, pp. 3–27.

Baldacci, R., Battarra, M., and Vigo, D. (2009). "Valid inequalities for the fleet size and mix vehicle routing problem with fixed costs". In: *Networks* 54.4, pp. 178–189.

Baldacci, R. and Mingozzi, A. (2008). "A unified exact method for solving different classes of vehicle routing problems". In: *Mathematical Programming* 120, pp. 347–380.

Barbarosoğlu, G., Özdamar, L., and Çevik, A. (2002). "An interactive approach for hierarchical analysis of helicopter logistics in disaster relief operations". In: *European Journal of Operational Research* 140.1, pp. 118–133.

Berbeglia, G., Cordeau, J.-F., Gribkovskaia, I., and Laporte, G. (2007). "Static pickup and delivery problems: a classification scheme and survey". In: *TOP* 15.1, pp. 1–31.

Caballero-Morales, S.-O. and Martinez-Flores, J.-L. (2019). "Helicopter routing model with non-deterministic failure rate for evacuation of multiple oil platforms". In: *Computers & Industrial Engineering*, p. 105669.

Cattaruzza, D., Absi, N., and Feillet, D. (2016). "Vehicle routing problems with multiple trips". In: *4OR* 14.3, pp. 223–259.

Contardo, C. and Martinelli, R. (2014). "A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints". In: *Discrete Optimization* 12, pp. 129–146.

Cordeau, J.-F. and Laporte, G. (2007). "The dial-a-ride problem: models and algorithms". In: *Annals of Operations Research* 153.1, pp. 29–46.

Danna, E. and Le Pape, C. (2005). "Branch-and-Price Heuristics: A Case Study on the Vehicle Routing Problem with Time Windows". In: *Column Generation*. Ed. by G. Desaulniers, J. Desrosiers, and M. M. Solomon, pp. 99–129.

Dayarian, I., Crainic, T. G., Gendreau, M., and Rei, W. (2015). "A column generation approach for a multi-attribute vehicle routing problem". In: *European Journal of Operational Research* 241.3, pp. 888–906.

Desaulniers, G. (2009). "Branch-and-Price-and-Cut for the Split-Delivery Vehicle Routing Problem with Time Windows". In: *Operations Research* 58.1, pp. 179–192.

Dror, M., Laporte, G., and Trudeau, P. (1994). "Vehicle routing with split deliveries". In: *Discrete Applied Mathematics* 50.3, pp. 239–254.

Fernández-Cuesta, E., Norddal, I. K., Andersson, H., and Fagerholt, K. (2017). "Base location and helicopter fleet composition in the oil industry". In: *INFOR* 55.2, pp. 71–92.

Fisher, M. L. and Jaikumar, R. (1981). "A generalized assignment heuristic for vehicle routing". In: *Networks* 11.2, pp. 109–124.

Flisberg, P., Lidén, B., and Rönnqvist, M. (2009). "A hybrid method based on linear programming and tabu search for routing of logging trucks". In: *Computers & Operations Research* 36.4, pp. 1122–1144.

François, V., Crama, Y., and Laporte, G. (2016). "Large neighborhood search for multi-trip vehicle routing". In: *European Journal of Operational Research* 255.2, pp. 422–441.

Golden, B., Assad, A., Levy, L., and Gheysens, F. (1984). "The fleet size and mix vehicle routing problem". In: *Computers & Operations Research* 11.1, pp. 49–66.

Gribkovskaia, I., Gullberg, B. O., Hovden, K. J., and Wallace, S. W. (2006). "Optimization model for a livestock collection problem". In: *International Journal of Physical Distribution & Logistics Management* 36.2. Ed. by G. Britta and v. H. Remko, pp. 136–152.

Gribkovskaia, I., Halskau, Ø., and Kovalyov, M. Y. (2015). "Minimizing takeoff and landing risk in helicopter pickup and delivery operations". In: *Omega* 55, pp. 73–80.

Haddad, M. N., Martinelli, R., Vidal, T., Martins, S., Ochi, L. S., Souza, M. J. F., and Hartl, R. (2018). "Large neighborhood-based metaheuristic and branch-and-price for the pickup and delivery problem with split loads". In: *European Journal of Operational Research* 270.3, pp. 1014–1027.

Halskau, Ø. (2014). "Offshore Helicopter Routing in a Hub and Spoke Fashion: Minimizing Expected Number of Fatalities". In: *Procedia Computer Science*, pp. 1124–1132.

Halvorsen-Weare, E. E. and Fagerholt, K. (2013). "Routing and scheduling in a liquefied natural gas shipping problem with inventory and berth constraints". In: *Annals of Operations Research* 203.1, pp. 167–186.

Haugseth, A., Høyland, V., and Nafstad, G. M. (2019). "Exact and Heuristic Solution Methods for an Offshore Helicopter Transportation Problem". In: *Master's Thesis:*

Hermeto, N. d. S. S., Ferreira Filho, V. J. M., and Bahiense, L. (2014). "Logistics network planning for offshore air transport of oil rig crews". In: *Computers & Industrial Engineering* 75, pp. 41–54.

Hernandez, F., Feillet, D., Giroudeau, R., and Naud, O. (2014). "A new exact algorithm to solve the multi-trip vehicle routing problem with time windows and limited duration". In: *4OR* 12.3, pp. 235–259.

Husseinzadeh Kashan, A., Abbasi-Pooya, A., and Karimiyan, S. (2019). "A Rig-Based Formulation and a League Championship Algorithm for Helicopter Routing in Offshore Transportation". In: *Proceedings of the 2nd International Conference on Data Engineering and Communication Technology*. Ed. by A. J. Kulkarni, S. C. Satapathy, T. Kang, and A. H. Kashan. Advances in Intelligent Systems and Computing. Springer Singapore, pp. 23–38.

Irnich, S. and Desaulniers, G. (2005). "Shortest Path Problems with Resource Constraints". In: *Column Generation*. Ed. by G. Desaulniers, J. Desrosiers, and M. M. Solomon, pp. 33–65.

Kek, A. G. H., Cheu, R. L., and Meng, Q. (2008). "Distance-constrained capacitated vehicle routing problems with flexible assignment of start and end depots". In: *Mathematical and Computer Modelling* 47.1, pp. 140–152.

Koç, Ç., Bektaş, T., Jabali, O., and Laporte, G. (2016). "Thirty years of heterogeneous vehicle routing". In: *European Journal of Operational Research* 249.1, pp. 1–21.

Korsvik, J. E., Fagerholt, K., and Laporte, G. (2011). "A large neighbourhood search heuristic for ship routing and scheduling with split loads". In: *Computers & Operations Research* 38.2, pp. 474–483.

Laporte, G. (1992). "The vehicle routing problem: An overview of exact and approximate algorithms". In: *European Journal of Operational Research* 59.3, pp. 345–358.

Li, F., Golden, B., and Wasil, E. (2007). "A record-to-record travel algorithm for solving the heterogeneous fleet vehicle routing problem". In: *Computers & Operations Research* 34.9, pp. 2734–2742.

Li, J., Li, Y., and Pardalos, P. M. (2016). "Multi-depot vehicle routing problem with time windows under shared depot resources". In: *Journal of Combinatorial Optimization* 31.2, pp. 515–532.

Lockheed Martin (2020). *SIKORSKY S-92 MULTI-MISSION HELICOPTER*. URL: `https://lockheedmartin.com/content/dam/lockheed-martin/rms/documents/s-92/Sikorsky-S92-multi-mission-helicopter-brochure.pdf` (visited on 06/02/2020).

Menezes, F., Porto, O., Reis, M. L., Moreno, L., Aragao, M. P. de, Uchoa, E., Abeledo, H., and Nascimento, N. C. do (2010). "Optimizing Helicopter Transport of Oil Rig Crews at Petrobras". In: *Interfaces* 40.5, pp. 408–416.

Mitra, S. (2008). "A parallel clustering technique for the vehicle routing problem with split deliveries and pickups". In: *Journal of the Operational Research Society* 59.11, pp. 1532–1546.

Montoya-Torres, J. R., López Franco, J., Nieto Isaza, S., Felizzola Jiménez, H., and Herazo-Padilla, N. (2015). "A literature review on the vehicle routing problem with multiple depots". In: *Computers & Industrial Engineering* 79, pp. 115–129.

Moreno, L., Aragao, M. P. de, Porto, O., and Reis, M. (2005). "Planning offshore helicopter flights on the campos basin". In: *XXXVII Simpósio Brasileiro de Pesquisa Operational (SBPO)*, pp. 96–108.

Moreno, L., Aragao, M. P. de, and Uchoa, E. (2006). "Column Generation Based Heuristic for a Helicopter Routing Problem". In: *Experimental Algorithms*. Ed. by C. Àlvarez and M. Serna. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 219–230.

Norsk Petroleum (2020a). *Employment*. Norwegianpetroleum.no. URL: `https://www.norskpetroleum.no/en/economy/employment/` (visited on 06/22/2020).

Norsk Petroleum (2020b). *Fields on the Norwegian continental shelf*. Norwegianpetroleum.no. URL: `https://www.norskpetroleum.no/en/facts/field/` (visited on 06/29/2020).

Norsk Petroleum (2020c). *Production forecasts*. Norwegianpetroleum.no. URL: `https://www.norskpetroleum.no/en/production-and-exports/production-forecasts/` (visited on 06/22/2020).

Norsk Petroleum (2020d). *The government's revenues*. Norwegianpetroleum.no. URL: `https://www.norskpetroleum.no/en/economy/governments-revenues/` (visited on 06/22/2020).

Nowak, M., Ergun, O., and White, C. C. (2008). "Pickup and Delivery with Split Loads". In: *Transportation Science* 42.1, pp. 32–43.

Nowak, M., Ergun, O., and White, C. C. (2009). "An empirical study on the benefit of split loads with the pickup and delivery problem". In: *European Journal of Operational Research* 198.3, pp. 734–740.

Oppedal, E. and Thorstein, K. L. (2019). "Exact Optimisation of an Extended Model for the Offshore Helicopter Routing Problem". In: *TIØ4500 - Managerial Economics and Operations Research, Specialization Project*.

Parragh, S. N., Pinho de Sousa, J., and Almada-Lobo, B. (2014). "The Dial-a-Ride Problem with Split Requests ans Profits". In: *Transportation Science* 49, pp. 311–334.

Qian, F., Gribkovskaia, I., and Halskau sr., Ø. (2011). "Helicopter routing in the Norwegian oil industry: Including safety concerns for passenger transport". In: *International Journal of Physical Distribution & Logistics Management* 41.4, pp. 401–415.

Qian, F., Gribkovskaia, I., Laporte, G., and Halskau sr., Ø. (2012). "Passenger and pilot risk minimization in offshore helicopter transportation". In: *Omega* 40, pp. 584–593.

Renaud, J., Laporte, G., and Boctor, F. F. (1996). "A tabu search heuristic for the multi-depot vehicle routing problem". In: *Computers & Operations Research* 23.3, pp. 229–235.

Romero, M., Sheremetov, L., and Soriano, A. (2007). "A Genetic Algorithm for the Pickup and Delivery Problem: An Application to the Helicopter Offshore Transportation". In: *Theoretical Advances and Applications of Fuzzy Logic and Soft Computing*. Ed. by O. Castillo, P. Melin, O. M. Ross, R. Sepúlveda Cruz, W. Pedrycz, and J. Kacprzyk. Advances in Soft Computing. Berlin, Heidelberg: Springer, pp. 435–444.

Rosa, R. d. A., Machado, A. M., Ribeiro, G. M., and Mauri, G. R. (2016). "A mathematical model and a Clustering Search metaheuristic for planning the helicopter transportation of employees to the production platforms of oil and gas". In: *Computers & Industrial Engineering* 101, pp. 303–312.

Salhi, S., Imran, A., and Wassan, N. A. (2014). "The multi-depot vehicle routing problem with heterogeneous vehicle fleet: Formulation and a variable neighborhood search implementation". In: *Computers & Operations Research*. Recent advances in Variable neighborhood search 52, pp. 315–325.

Savelsbergh, M. W. P. and Sol, M. (1995). "The General Pickup and Delivery Problem". In: *Transportation Science* 29.1, pp. 17–29.

Sierksma, G. and Tijssen, G. A. (1998). "Routing helicopters for crew exchanges on off-shore locations". In: *Annals of Operations Research* 76.0, pp. 261–286.

Stålhane, M., Andersson, H., Christiansen, M., Cordeau, J.-F., and Desaulniers, G. (2012). "A branch-price-and-cut method for a ship routing and scheduling problem with split loads". In: *Computers & Operations Research* 39.12, pp. 3361–3375.

Statistics Norway (2020). *Employment*. URL: `https://www.ssb.no/arbeid-og-lonn/statistikker/regsys/aar/2018-03-08` (visited on 06/22/2020).

Tieto Norway (2020). *Heliport.no*. URL: `https://heliport.no/` (visited on 06/01/2020).

Velasco, N., Castagliola, P., Dejax, P., Guéret, C., and Prins, C. (2009). "A Memetic Algorithm for a Pick-Up and Delivery Problem by Helicopter". In: *Bio-inspired Algorithms for the Vehicle Routing Problem*. Ed. by F. B. Pereira and J. Tavares. Studies in Computational Intelligence. Berlin, Heidelberg: Springer, pp. 173–190.

Xu, Y., Wang, L., and Yang, Y. (2012). "A New Variable Neighborhood Search Algorithm for the Multi Depot Heterogeneous Vehicle Routing Problem with Time Windows". In: *Electronic Notes in Discrete Mathematics*. EURO Mini Conference 39, pp. 289–296.