Computers and Operations Research 128 (2021) 105158

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/caor

An exact solution method for a rich helicopter flight scheduling problem arising in offshore oil and gas logistics

Gaute Messel Nafstad [a], Amund Haugseth [a], Vebjørn Høyland [a], Magnus Stålhane [a,*]

[a] Norwegian University of Science and Technology, Department of Industrial Economics and Technology Management, Trondheim, Norway

ARTICLE INFO

Article history:
Received 2 January 2020
Revised 31 July 2020
Accepted 19 November 2020
Available online 29 November 2020

Keywords:
Offshore logistics
Helicopter routing
Multi-trip VRP
Multi-depot VRP
Temporal synchronization

ABSTRACT

This paper studies the problem of creating an optimal flight schedule for a heterogeneous fleet of helicopters tasked with transporting personnel to, from, and between offshore installations. The problem can be modelled as a rich vehicle routing problem and combines the following properties from the vehicle routing literature: pickup and delivery structure, heterogeneous fleet operating out of multiple depots, multi-trip, and temporal synchronization of transportation tasks. We present compact and extended mathematical models of the problem, where the extended model is based on generating all trips apriori. When solving the extended model we apply delayed constraint generation (DCG) to parts of the model to speed up the solution process. Computational results are presented that show that the extended formulation and solution method can solve realistic instances of the problem within one hour. The results further show that the DCG method works significantly better than using lazy constraints from a commercial solver, especially when the number of transportation tasks requiring temporal synchronization becomes large.

1. Introduction

Since the discovery of the Ekofisk field in 1969, the Norwegian oil and gas sector has developed into the country's largest industry (Norwegian Petroleum, 2020a). Today, Norway is the seventh largest producer of natural gas, and the 15th largest producer of oil worldwide (British Petroleum Company, 2020). The yearly accumulated export value for crude oil, natural gas, and condensate is almost 435 billion NOK (≈ 44 billion USD) (Statistics Norway, 2019), and approximately 140,000 people are, directly or indirectly, employed in the petroleum industry (Statistics Norway, 2019). At the end of 2019, there were 87 fields in production on the Norwegian continental shelf (NCS), while another 95 oil and gas discoveries are candidates for development (Norwegian Petroleum, 2020b). The total petroleum resources on the NCS are estimated to be 15,6 billion standard cubic meter, of which only 7.5 billion has been produced, sold and delivered thus far (Norwegian Petroleum, 2020b).

These petroleum resources are extracted by offshore installations that are either fixed oil platforms or floating production vessels. These installations are operated by personnel who needs to be transported to and from shore. This transportation was earlier conducted by sea vessels, but for the last decades it has almost exclusively been handled by helicopters. In 2018, over 350,000 unique deployments of personnel were conducted on the NCS according to the number of bookings made in the Dawinci Industry Hub (TietoEVRY, 2020). This equals approximately 1000 persons transported every day, which corresponds to over 50 fully loaded Sikorsky S-92 helicopters, with a capacity of 19 passengers (Aerospace Technology, 2018). Helicopter transportation is expensive, and one operator on the NCS estimates annual total costs of chartering and operating helicopters to be around two billion NOK. Today, chartering of helicopters, and planning their flights, is done manually by each company operating installations on the NCS. However, because of the complexity of the planning problem, it is likely that significant savings can be achieved by applying operational research methods to solve the problem.

In this paper we study the problem of operating a heterogeneous fleet of helicopters to transport personnel between offshore installations and/or heliports onshore. Personnel is available at his/her pickup location at a given time, and has a latest time for when he/she is to arrive at the destination. The transportation of personnel is mainly done from a heliport onshore to an installation offshore or vice versa, but transportation between offshore installations also occur. The latter is typically transportation of personnel between floating hotel facilities (*flotels*) and production platforms, or when moving expert personnel between installations.

* Corresponding author.
E-mail address: magnus.staalhane@ntnu.no (M. Stålhane).

https://doi.org/10.1016/j.cor.2020.105158
0305-0548/

Further, there exist temporal dependencies that affect the allowed pickup and/or delivery time of certain personnel. For key personnel it is required that the replacement must arrive at the installation at least a minimum amount of time before he/she can depart to allow time for a debriefing. It also happens that certain personnel travel back and forth between a heliport and an installation on a single day to perform some key task, in which case they must be given sufficient time at the installation to perform their intended task.

To transport their personnel, each operator has a fleet of helicopters at its disposal. The helicopters are stationed at different heliports along the coast, and each helicopter operates to and from their assigned heliport. The fleet of helicopters is heterogeneous with respect to the number of seats available for passengers, the fixed cost of operating each day, the fuel consumption per hour, and the capacity of their fuel tanks. Flight time and fuel consumption between locations are assumed to be based the distance between them and the cruising speed of each helicopter, and not affected by wind conditions or the number of personnel on board. It is further assumed that the time spent at each offshore installation is constant, regardless of the number of personnel (dis)embarking, and that helicopters leave the offshore installation immediately afterwards.

A trip is defined as a flight sequence starting at a heliport, visiting installations in a given order and returning to the initial heliport. In addition to the limitations of the helicopter stated above, each trip is limited by safety regulations stipulating a maximum number of landings any personnel transported can partake in. Helicopters may perform several trips each day, and the sequence of trips conducted by a given helicopter on a given day is defined as a route. Between trips the helicopter must wait a given amount of time to allow for refueling, which can only be done at heliports, and required rest for the pilots. Due to the limited number of helipads (usually just one) at an offshore installation, it is also necessary to ensure that no two helicopter routes are scheduled to visit the same installation at the same time.

The objective in this problem is to design a set of routes for the helicopters to fly, minimizing costs associated with operating a helicopter and the distance flown, while transporting all personnel from(to) their intended origin(destination), and adhering to all the limitations described above.

The purpose of this paper is to present an exact solution method for the problem of planning helicopter flights that transport personnel to, from, and between offshore installations. The problem statement is based on the operations on the NCS, however, the same (or similar) problems are faced by operators of offshore installations all over the world. A new mathematical formulation is presented, that take into account several aspects of the problem that has previously not been addressed in the literature. Further, we develop a second mathematical model where all possible trips are generated apriori, and the concatenation of trips into routes for each helicopter is handled by the model. We further develop a customized labeling algorithm to generate trips and apply delayed constraint generation when solving this model using branch-and-bound. To make the solution method even more effective, we preprocess both arcs and time windows, and add customized branching strategies. A computational study shows the effectiveness of the solution methodology.

The remainder of the paper is organized as follows: In Section 2 we give an overview of the literature on helicopter planning, and a brief overview of the literature on relevant variants of the vehicle routing problem. Then, in Section 3 we present both a compact and an extended mathematical model of the problem, before discussing in detail the different parts of our solution methodology in Section 4. Finally, we test the computational performance of the proposed solution method in Section 5, before giving some concluding remarks in Section 6.

2. Related literature

Optimal transportation of personnel to and from offshore oil installations have been studied since the late 1980s when Galvão and Guimarães (1987, 1990) presented a heuristic solution method, and a decision support system utilizing this heuristic, to optimally transport crew to and from oil platforms off the coast of Brazil. The first paper presenting a mathematical model of the operational planning of personnel transport by helicopters between heliports onshore and offshore platforms is that of Sierksma et al. (1998), who present both exact and heuristic approaches for solving the problem. The objective is to route each helicopter such that a given number of personnel are replaced at each location, and the total transportation costs are minimized. The problem is modelled as a split delivery vehicle routing problem, where more than one helicopter may meet the demand at a given offshore installation.

Since then, similar problems have been studied by Moreno et al. (2005), Moreno et al. (2006) and Menezes et al. (2010), who all consider transportation of personnel from heliports to offshore oil platforms off the Brazilian coast. Moreno et al. (2005) study the problem of planning trips for each helicopter for a single day, where the trips are limited by the number of intermediate landings for both orders and trips, and the objective is to minimize costs related to takeoff and landing, distance and hours travelled, and personnel not transported. To solve this problem they propose a heuristic that constructs a set of trips and aggregates the trips into routes for each helicopter. Moreno et al. (2006) extend the work of Moreno et al. (2005) by adding column generation procedures to the heuristic, utilizing information from the LP relaxation to construct helicopter routes. The problem also involves the element of an imposed waiting time between trips. The work is further extended by Menezes et al. (2010) who use a column generation-based solution approach for the same problem.

Another aspect of operational offshore helicopter planning is safety and risk. This has been studied by Qian et al. (2011) and Qian et al. (2014) who both analyze and propose approaches to minimize takeoff and landing risk, and suggest new operational procedures intended to create safer flight schedules for pickups and deliveries of personnel.

Hermeto et al. (2014) and Fernández-Cuesta et al. (2017) solve strategic problems where the goal is to minimize the long-term costs related to opening and operating airfields, and transporting workers to and from offshore installations. Hermeto et al. (2014) apply a simple routing system which assume that each helicopter only visits one platform on each trip, whereas Fernández-Cuesta et al. (2017) establish a more complex routing component where a route may visit several offshore installations. In addition, Fernández-Cuesta et al. (2017) incorporate the possibility of leasing hubs for refueling off the coast.

What separates the problem studied in this paper from earlier works, is that we consider transportation of personnel between offshore installations, in addition to between the installations and the heliport. Further, we consider time dependencies between transportation of key personnel and enforce that no two helicopters can be present at an installation at the same time. We thus have a richer problem, incorporating more of the real-life constraints and dependencies.

The problem studied in this paper can be seen as a rich variant of the vehicle routing problem (VRP) with time windows. It combines several variants and extensions of the VRP where each aspect has a rich body of research literature thoroughly reviewed in dedicated surveys. The problem is a pickup and delivery problem (PDP) (Berbeglia et al., 2007), with multiple depots (Montoya-Torres et al., 2015), heterogeneous fleet (Koç et al., 2016), multiple trips (Cattaruzza et al., 2016) and temporal synchronization of

operations (Drexl, 2012). In the following we will classify our problem within the classification schemes presented by the surveys cited above, and present the most recent papers on exact solution methods for each variant and extension.

According to the classification scheme of PDPs presented by Berbeglia et al. (2007), a PDP is categorized as a one-to-one-PDP (1-1-PDP) if it has exactly one pickup node and one delivery node for all orders and categorized as a one-to-many-to-one-PDP (1-M-1-PDP) if orders consist of either pickups at the depot and delivery to customers, or vice versa. If each customer has both positive pickup and delivery demands it is categorized as a problem with *combined demands*. According to this classification, the problem studied in this paper is a combination of the 1-M-1-PDP and the 1-1-PDP, however, as seen in Section 3, we model it as a strictly 1-1-PDP. State of the art exact solution methods for the 1-1-PDP are presented by Ropke and Cordeau (2009) and Baldacci et al. (2011). The former present a solution method based on branch-price-and-cut, while the latter is based on dual bounding techniques combined with enumeration of all routes with reduced cost in the interval between the best known primal and dual solutions. Both solution methods use a set partitioning model where each variable represents a feasible route for each vehicle.

In the survey on VRP with heterogeneous fleet given by Koç et al. (2016), they divide the literature into categories based on whether fleet composition is fixed (HF) or a decision variable (FSM), whether the objective includes fixed costs, variable costs, or both, and whether the problem includes time windows. According to their classification the problem studied here is a HFTW(D), where the problem has a fixed fleet, includes time windows and the variable part of the costs are related to the distance travelled. Exact solution methods for this problem are presented by Baldacci et al. (2010) who apply a dual bounding algorithm and route enumeration, and Pessoa et al. (2018) who combine a route enumeration scheme for small vehicles with a tailored branch-price-and-cut algorithm. Both methods rely on set partitioning models.

A literature review on the multi depot vehicle routing problem is given by Montoya-Torres et al. (2015), which categorizes the multi depot VRP into two categories with single or multiple objectives. The problem studied in this paper belongs to the former category, for which recent exact algorithms are presented by Baldacci and Mingozzi (2009) and Contardo and Martinelli (2014). As with the problem variants described above, dual bounding and branch-and-price solution methods are used in this paper, both based on set partitioning models. It is also worth mentioning that an extension of this problem, known as the location routing problem, where the location of the depots must also be decided, has gotten significant attention in the literature (Drexl and Schneider, 2015).

In the survey on multi-trip VRPs given by Cattaruzza et al. (2016), they identify four main academic extensions of the problem: time windows, service dependent loading times, limited trip duration, and profits. The problem in this paper include time windows and limited trip durations, but not the other two. An exact solution method to the multi-trip VRP is proposed by Mingozzi et al. (2013), while an exact method for an extension of the problem which include both time windows and limited trip durations is given by Hernandez et al. (2016). Both papers base their solution algorithm on first generating trips for the vehicles and then concatenating these trips into feasible routes for each vehicle.

A survey on VRPs with synchronization is presented by Drexl (2012). According to his classification scheme, the problem studied in this paper includes temporal operation synchronization with precedence, both with respect to when personnel is picked up/delivered and when the offshore installations are visited. Dohn et al. (2011) present four different formulations for handling temporal synchronization, including handling the synchronization through branching on the time windows. All the models are based on a branch-and-price approach. For the case of synchronized VRP with pickup and deliveries, a comparison of branch-and-price based solution methods, is conducted by Gschwind (2015).

Common for all the extensions and variants of the VRP surveyed, is that the solution methodology is based on a (generalized) set partitioning formulation, where each variable represents a path through the problem defining network. For the PDP, MDVRP, and HFTW, the problem can be formulated as a standard set partitioning model where each variable represents a feasible path through the problem defining network, and there is one constraint per customer, and one per vehicle. For the synchronization aspect, additional constraints have to be added to the model to ensure that the different vehicles interact in accordance with the synchronization requirements. For the multi-trip VRP, the complexity of the model depends on whether the variables in the model represent single trips, or a complete route for each vehicle. In the former case, one needs to coordinate the sequence and start times of the trips for each vehicle, leading to additional constraints in the model.

3. Mathematical models

When modelling this problem it is natural to aggregate all personnel going from the same heliport to the same installation (or vice versa) and that has the same time restrictions and dependencies, into one order. In this section we present two mathematical models for the transportation of a set of orders using helicopters. In Section 3.1 we present a compact mathematical formulation of the problem based on commodity flows along the arcs of the problem defining graph. Then in Section 3.2 we present a model based on pregeneration of all possible trips for each helicopter from its heliport to a set of installations and back to its heliport.

3.1. Compact mathematical formulation

The problem of serving n orders may be modelled on a graph where the pickup and delivery location of orders are represented by nodes. Let $N^P = \{1, 2, \dots n\}$ be the set of pickup nodes, and $N^D = \{n+1, n+2, \dots, 2n\}$ be the set of delivery nodes, where $(i+n)$ is the delivery node of the order picked up at node i. Denote the number of personnel picked up or delivered at node i as P_i, and let $\underline{T}_i$ and $\overline{T}_i$ denote the earliest and latest time a node may be serviced, respectively. Let $N^C \subset N^D \times N^P$, indexed by (i,j), be the set of connected nodes, i.e a pair of nodes where node i must be visited at least T^C_{ij} time units before j. Let N^I_i represent the set of nodes located at the same installation/heliport as node i. Further, we have a set of refueling nodes N^F located at heliports where the helicopters may refuel between trips during the day, and a set $N^S \subset N^P \bigcup N^D$ containing all nodes that are not located at a heliport. Fig. 1 shows an example of how nodes are distributed across two heliports and three offshore installations.

Let H be the set of helicopters, indexed by h, where each helicopter has a given seat capacity K_h and a fixed cost C^H_h that occurs if the helicopter is being used. Let $s(h)$ and $e(h)$ be the start node and end node, respectively, representing the heliport where helicopter h starts and ends its day. With each helicopter we associate a graph $G_h = (N_h, A_h)$ where $N_h = N^P \bigcup N^D \bigcup N^F_h \bigcup \{e(h), s(h)\}$, and $N^F_h \subset N^F$ is the set of refueling nodes located at the same heliport as nodes $e(h)$ and $s(h)$. Let $\overline{F}_{hi}$ and $\underline{F}_{hi}$ denote the maximum and minimum level of fuel on board helicopter h when arriving at node i, respectively.

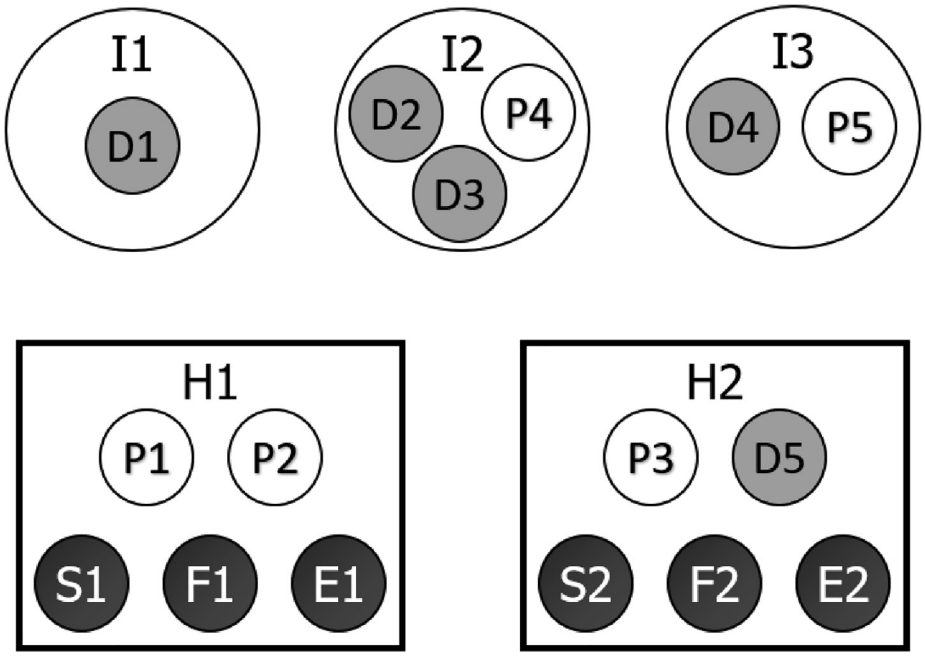

Fig. 1. Example of nodes on installations and heliports. The white nodes represent pickup nodes, the grey represent delivery nodes and the black nodes, S, F, and E, represent start, refueling and end nodes for helicopters stationed at heliport number 1 and 2, respectively.

The set $A_h \subset N_h \times N_h$ define the arcs that are feasible for the helicopter to traverse. First, to avoid sub-tours at an installation, and ensuring that we deliver before picking up, we remove all arcs (i,j) from A_h if $j \in N_i^I$ and $i < j$. Second, to enforce refueling every time the helicopter returns to the heliport, we remove all arcs between a pickup and a delivery node at a heliport, and ensure that no arc from a node at another installation can go directly to a pickup node at a heliport. Thus all helicopters arriving at heliports must visit (a subset of) the delivery nodes, then the refueling node, and finally (a subset of) the pickup nodes.

With each arc $(i,j) \in A_h$ we associate a cost C_{hij}, fuel consumption F_{hij}, and a time T_{hij} for traversing it. Let T^L be the landing time at any installation, and let T_{ij}^L be the landing time at node i when travelling to node j specifically. Note that both T_{hij} and T_{ij}^L are 0 if $j \in N_i^I$, i.e. node i and j are at the same offshore installation. Due to safety regulations there is a maximum number of legs travelled between the pickup and delivery of an order, given by $\overline{L}$. Let $K = \{1, \ldots, \overline{K}\}$ be the set of possible leg numbers, indexed by k, where $\overline{K}$ is an upper bound on the number of legs a helicopter can fly on a given day.

Let the binary variable x_{hijk} be equal to one if helicopter h travels from node i to j on leg k, and zero otherwise. Variable p_{hi} is the number of passengers when leaving node i, and f_{hi} is the fuel level and t_{hi} is the time when helicopter h is arriving at node i. The binary variable z_h equals one if helicopter h is used, and zero otherwise. Further, we introduce the binary variable γ_{hij} which is equal to one if helicopter h service both nodes $i \in N$ and $j \in N_i^I$ and zero otherwise, while δ_{ij} is equal to zero if two different helicopters service nodes i and j and i is serviced at least T^L time units before j.

For ease of exposition, the constraints in the model below are defined using sets of nodes and may therefore contain combinations of the indices h, i, and j for which the corresponding arc $(i,j) \in A_h$ does not exist (e.g. when $i = j$). In these cases the corresponding variable x_{hijk} can be assumed to take the value zero. Using this notation, the problem can be formulated as follows:

$$\min \sum_{h\in H}\sum_{i\in N_h}\sum_{j\in N_h}\sum_{k\in K} C_{hij}x_{hijk} + \sum_{h\in H} C_h^H z_h, \tag{1}$$

subject to:

$$\sum_{h\in H}\sum_{j\in N_h}\sum_{k\in K} x_{hijk} = 1, \quad i \in N^P, \tag{2}$$

$$\sum_{j\in N_h}\sum_{k\in K} x_{hijk} - \sum_{j\in N_h}\sum_{k\in K} x_{h(n+i)jk} = 0, \quad h \in H,\ i \in N^P, \tag{3}$$

$$\sum_{j\in N_h}\sum_{k\in K} x_{hijk} - \sum_{j\in N_h}\sum_{k\in K} x_{hjik} = 0, \quad h \in H,\ i \in N_h \setminus \{s(h), e(h)\}, \tag{4}$$

$$\sum_{j\in N^P} x_{hs(h)j1} = z_h, \quad h \in H, \tag{5}$$

$$\sum_{i\in N^D}\sum_{k\in K} x_{hie(h)k} = z_h, \quad h \in H, \tag{6}$$

$$\sum_{j\in N_h} x_{hjik} = \sum_{j\in N_i^I} x_{hijk} + \sum_{j\in N_h\setminus N_i^I} x_{hij(k+1)},$$
$$h \in H,\ i \in N_h \setminus \{s(h), e(h)\}, k \in K \setminus \{|K|\}, \tag{7}$$

$$\sum_{j\in N_h}\sum_{k\in K} k x_{hj(i+n)k} - \sum_{j\in N_h}\sum_{k\in K} k x_{hjik} \leqslant \overline{L}, \quad h \in H,\ i \in N^P, \tag{8}$$

$$f_{hi} - F_{hij} - f_{hj} + M_{hij}^F\left(1 - \sum_{k\in K} x_{hijk}\right) \geqslant 0,$$
$$h \in H,\ i \in N_h,\ j \in N_h \setminus N_h^F, \tag{9}$$

$$\underline{F}_{hi} \leqslant f_{hi}, \leqslant \overline{F}_{hi}, \quad h \in H,\ i \in N_h, \tag{10}$$

$$p_{hi} + P_j - p_{hj} - K_h\left(1 - \sum_{k\in K} x_{hijk}\right) \leqslant 0, \quad h \in H,\ i \in N_h,\ j \in N^P, \tag{11}$$

$$p_{hi} - P_j - p_{hj} - K_h\left(1 - \sum_{k\in K} x_{hijk}\right) \leqslant 0, \quad h \in H,\ i \in N_h,\ j \in N^D, \tag{12}$$

$$p_{hi} - p_{hj} - K_h\left(1 - \sum_{k\in K} x_{hijk}\right) \leqslant 0, \quad h \in H,\ i \in N_h,\ j \in N_h^F, \tag{13}$$

$$p_{hi} = 0, \quad h \in H,\ i \in N_h^F, \tag{14}$$

$$P_i \sum_{k\in K}\sum_{j\in N_h} x_{hijk} \leqslant p_{hi} \leqslant K_h z_h, \quad h \in H,\ i \in N^P, \tag{15}$$

$$0 \leqslant p_{h(i+n)} \leqslant (K_h - P_{i+n}) \sum_{k \in K} \sum_{j \in N_h} x_{h(i+n)jk}, \quad h \in H, i \in N^P, \tag{16}$$

$$t_{hi} + T^L_{ij} + T_{hij} - t_{hj} + M'_{hij}\left(1 - \sum_{k \in K} x_{hijk}\right) \geqslant 0, \quad h \in H, i \in N_h, j \in N_h \setminus N^F_h, \tag{17}$$

$$t_{hi} + T^L_{ij} + T_{hij} - t_{hj} + M''_{hij}\left(1 - \sum_{k \in K} x_{hijk}\right) \leqslant 0, \quad h \in H, i \in N_h, j \in N_h, \tag{18}$$

$$t_{hi} + \sum_{j \in N_h} \sum_{k \in K} x_{hijk}\left(T_{hi(i+n)} + T^L_{i(i+n)}\right) \leqslant t_{h(i+n)}, \quad h \in H, i \in N^P, \tag{19}$$

$$\sum_{h \in H} t_{hi} + T^C_{ij} \leqslant \sum_{h \in H} t_{hj}, \quad (i,j) \in N^C, \tag{20}$$

$$\sum_{h \in H} t_{hi} - \sum_{h \in H} t_{hj} - T^L + \left(T^L + \bar{T}_j\right)\delta_{ij} \geqslant 0, \quad i \in N^S, j \in N^I_i, \tag{21}$$

$$\sum_{i \in N} \sum_{k \in K} x_{hiik} + \sum_{j \in N} \sum_{k \in K} x_{hjjk} \leqslant 2\gamma_{hij}, \quad h, \in H, i \in N^S, j \in N^I_i | i < j, \tag{22}$$

$$\delta_{ij} + \delta_{ji} \leqslant 1 + \sum_{h \in H} \gamma_{hij}, \quad i \in N^S, j \in N^I_i | i < j, \tag{23}$$

$$\sum_{j \in N_h} \sum_{k \in K} \underline{T}_i x_{hijk} \leqslant t_{hi} \leqslant \sum_{j \in N_h} \sum_{k \in K} \bar{T}_i x_{hijk}, \quad h \in H, i \in N_h, \tag{24}$$

$$x_{hijk} \in \{0,1\}, \quad h \in H, (i,j) \in A_h, k \in K, \tag{25}$$

$$z_h \in \{0,1\}, \quad h \in H, \tag{26}$$

$$\gamma_{hij} \in \{0,1\}, \quad h \in H, i \in N_h, j \in N^I_i, \tag{27}$$

$$\delta_{ij} \in \{0,1\}, \quad i \in N, j \in N^I_i. \tag{28}$$

The objective function (1) minimizes the cost related to traversing arcs and utilizing helicopters. The constraints (2) and (3) make sure that all orders are handled and that the same helicopter visits both the pickup node and the corresponding delivery node. Constraints (4)-(6) ensure continuous flight sequences, while beginning and ending in the correct nodes. The arcs are given the correct leg index with constraints (7), while constraints (8) limit the number of legs between a pickup node and the corresponding delivery node. Constraints (9) and (10) handle fuel consumption and limit the fuel levels. The big M is given by $M^F_{hij} = \bar{F}_{hi} + F_{hij} - F_{hj}$. Embarking and disembarking of personnel at pickup and delivery nodes are enforced with constraints (11) and (12), respectively, while constraints (13) and (14) ensure that no personnel is on board the helicopter when refueling. Constraints (15) and (16) limit the number of passengers on board a helicopter when leaving a pickup and delivery node, respectively. Updating the departure times from each node visited is handled by constraints (17) and (18). Note that the refueling nodes are not part of constraints (17) because, unlike at installation nodes, you are allowed to wait at the heliports before refueling. The big Ms are given as $M'_{hij} = \bar{T}_j + T^L_{ij} + T_{hij}$ and $M''_{hij} = \bar{T}_i + T^L_{ij} + T_{hij}$, respectively. Constraints (19) ensure that the pickup node is visited earlier than the corresponding delivery node. The minimum time difference between handling the connected orders is enforced through constraints (20). Constraints (21)-(23) ensure that no two helicopters are present at an installation simultaneously. The upper and lower limit of the departure time are stated with constraints (24). Finally, constraints (25)-(28) give the domain of each variable.

3.2. Trip-based mathematical formulation

An alternative to the model presented above, is a formulation where each variable represents a trip beginning and ending at a heliport. Let R_h be the set of all possible trips for helicopter h, indexed by r, and let $S = \{1, \ldots, |S|\}$ be the set of trip numbers that a helicopter may perform during a day, indexed by s, indicating which trip number it is currently performing. The parameter C^T_{hr} is the cost related to travelling trip r with helicopter h, and the binary parameter A_{ihr} indicates whether a node i is visited on trip r with helicopter h or not. The time helicopter h uses on trip r is denoted T_{hr}, while the time from the start of a trip to a specific node i is denoted T_{ihr}. Let $\underline{T}_{hr}$ be the earliest and $\bar{T}_{hr}$ be the latest possible time that trip r can be initiated with helicopter h. The parameter T^H is the minimum time consumption at a heliport between trips. The binary variable λ_{hrs} equals one if helicopter h travels trip r on trip number s, and zero otherwise. Variable t_{hs} represents the point in time when helicopter h starts trip number s from the heliport, while variable t_i is the point in time when node i is visited. With this notation, the the trip-based model is formulated as follows:

$$\min \sum_{h \in H} \sum_{r \in R_h} \sum_{s \in S} C^T_{hr} \lambda_{hrs} + \sum_{h \in H} \sum_{r \in R_h} C^H_h \lambda_{hr1}, \tag{29}$$

subject to:

$$\sum_{h \in H} \sum_{r \in R_h} \sum_{s \in S} A_{ihr} \lambda_{hrs} = 1, \quad i \in N^P, \tag{30}$$

$$\sum_{r \in R_h} \lambda_{hrs} \leqslant 1, \quad h \in H, s \in S, \tag{31}$$

$$\sum_{r \in R_h} \lambda_{hrs} \geqslant \sum_{r \in R_h} \lambda_{hr(s+1)}, \quad h \in H, s \in S \setminus \{|S|\}, \tag{32}$$

$$\sum_{r \in R_h} \underline{T}_{hr} \lambda_{hrs} \leqslant t_{hs} \leqslant \sum_{r \in R_h} \bar{T}_{hr} \lambda_{hrs}, \quad h \in H, s \in S, \tag{33}$$

$$t_{hs} + \sum_{r \in R_h} \left(T^H + T_{hr}\right)\lambda_{hrs} \leqslant t_{h(s+1)} + M_h\left(1 - \sum_{r \in R_h} \lambda_{hr(s+1)}\right), \quad h \in H, s \in S \setminus \{|S|\}, \tag{34}$$

$$t_j - t_i \geqslant T^C_{ij}, \quad (i,j) \in N^C, \tag{35}$$

$$t_i \geqslant t_{hs} + \sum_{r \in R_h} T_{ihr} \lambda_{hrs} - M_{ih}\left(1 - \sum_{r \in R_h} A_{ihr} \lambda_{hrs}\right), \quad i \in N, h \in H, s \in S, \tag{36}$$

$$t_i \leqslant t_{hs} + \sum_{r \in R_h} T_{ihr} \lambda_{hrs} + \bar{T}^P_i\left(1 - \sum_{r \in R_h} A_{ihr} \lambda_{hrs}\right), \quad i \in N, h \in H, s \in S, \tag{37}$$

$$t_i - t_j \geqslant T^L - \left(\bar{T}_j + T^L - \underline{T}_i\right)\delta_{ij}, \quad i \in N^S, j \in N^I_i, \tag{38}$$

$$\sum_{r \in R_h} \sum_{s \in S} A_{ihr} \lambda_{hrs} + \sum_{r \in R_h} \sum_{s \in S} A_{jhr} \lambda_{hrs} \geqslant 2\gamma_{hij}, \quad h \in H, i \in N^S, j \in N^I_i | i < j, \tag{39}$$

$$\underline{T}_i \leqslant t_i \leqslant \bar{T}_i, \quad i \in N, \tag{40}$$

$$\delta_{ij} + \delta_{ji} \leqslant 1 + \sum_{h \in H} \gamma_{hij}, \quad i \in N^S, j \in N^I_i | i < j, \tag{41}$$

$$\gamma_{hij} \in \{0,1\}, \quad h \in H, i \in N^S, j \in N^I_i, \tag{42}$$

$$\delta_{ij} \in \{0,1\}, \quad i \in N, j \in N^I_i, \tag{43}$$

$$\lambda_{hrs} \in \{0,1\}, \quad h \in H, r \in R_h, s \in S. \tag{44}$$

The objective function (29) minimizes the cost of the selected trips and the fixed cost of using helicopters. Constraints (30) ensure that all orders are handled. Constraints (31) and (32) make sure that a helicopter can maximum do one trip for each trip number, and that a trip can be assigned to a trip number, only if the previous trip number has already been used. In constraints (33) the time windows for when a helicopter can initiate a trip are bounded by which trip the helicopter is using. If a helicopter does not use trip number s, the starting time of that trip number is set to 0. Constraints (34) state the earliest point in time a helicopter can initiate a trip, with the big M given as: $M_h = \max_{r \in R_h}\{\bar{T}_{hr} + T_{hr}\} + T^H$. Constraints (35) ensure that there is sufficient time between the service of two connected orders. Constraints (36) and (37) ensure that the service time of each order is calculated correctly. The big

M is set to $M_{ih} = \max_{r \in R_h}\{\overline{T}_{hr}\} - \underline{T}_i$. Constraints (38)–(41) ensure that no two helicopters are present at any installation simultaneously. Finally, constraints (42)-(44) state the variable domains.

4. Solution method

Mixed integer programming models such as those presented in Section 3 may be solved by commercial solvers using the branch-and-bound (B&B) algorithm. However, a prerequisite for solving the trip-based model formulated in Section 3.2 is that all possible trips for all helicopters can be generated apriori.In the following we describe such a trip generation algorithm in Section 4.1, before proposing an alternative formulation of parts of the trip-based model requiring delayed constraint generation in Section 4.2. In Section 4.3 we describe how the number of arcs in the problem defining graph may be reduced, and how the width of the time windows may be decreased, before finally introducing some aggregated branching rules used when solving the trip-based mathematical model in Section 4.4.

4.1. Apriori trip generation

The trips in the trip based model are generated apriori using a labeling algorithm. Labeling algorithms are commonly used to solve shortest path problem with resource constraints (SPPRC) (Irnich and Desaulniers, 2005). For the trip generation algorithm presented here, a labeling algorithm is run for each helicopter $h \in H$ and its corresponding graph G_h presented in Section 3. We do, however, make one minor modification to G_h: since we are generating trips, and refueling is done between trips, the set of refueling nodes N_h^F is removed from the graph. A pseudo code for the trip generation is provided in Algorithm 1.

A label is used to store a partial path through the network, and the resources accumulated along that path. A label $L(i, \mathcal{R}, L^*)$, contains the last node on the path, i, a set of resources, $\mathcal{R}$, and a pointer to the parent label, L^*. The parent label pointer is necessary in order to retrace the path represented by the label. The labeling algorithm starts by initiating a set of unprocessed labels, U, only consisting of the start label $L(s(h), \mathcal{R}_0, NULL)$. The initial set of resources are referred to as $\mathcal{R}_0$, and the parent label pointer is a zero pointer, $NULL$. In the algorithm, a selected label $L = (i, \mathcal{R}, *)$, representing a partial path from node s to i with accumulated resources $\mathcal{R}$, is removed from U. This partial path is extended along all arcs $(i,j) \in A_h$ to create new partial paths, each one represented by a label $L\prime$. The resource consumption of label $L\prime$ is calculated according to so-called *resource extension functions*, given by $f_{ij}(\mathcal{R})$, where $\mathcal{R}$ is the resource consumption of label L. The resource consumption of each extended label is then checked to see if it is within the *resource window* $[a_j, b_j]$. If it is, then the new label is added to U unless $j = e(h)$, in which case the label has been extended to the sink node. The algorithm terminates when there are no unprocessed labels left in U. In the following we describe which resources are stored in $\mathcal{R}$, what the resource extension functions look like, and what the resource window is for each resource.

Table 1
Label attributes and initial values.

Resource	Description	Initial value
$\mathcal{O}$	Open orders (started, but not yet completed)	$\varnothing$
$\mathcal{U}$	Unreachable orders	$\varnothing$
$\mathcal{L}_o$	Legs travelled for order o	0
f	Remaining fuel	$\overline{F}_{hs(h)}$
$\underline{t}$	Earliest possible arrival at the current node	$\underline{T}_{hs(h)}$
$\overline{t}$	Latest possible arrival at the current node	$\overline{T}_{hs(h)}$
t_i^C	Time counter for node i	0

Algorithm 1. Label algorithm for trip generation for helicopter h.

```
R_h = ∅
U = {L(s, R_0, NULL)}
while U≠∅
L = (i, R, *) =RemoveFirst(U)
for (i,j) ∈ A
L' = (j, f_ij(R), L)
if a_j ⩽ f_ij(R) ⩽ b_j
if j = e(h)
R_h = R_h ∪ {L'}
else
U = U ∪ {L'}
end if
end if
end for
end while
```

4.1.1. Resources

The resources are used to model constraints (3)-(12), (15)–(19) and (24)–(25) in the mathematical model. The set of resources kept in a label is described in Table 1, as well as their value in the initial label. The resource set, $\mathcal{R}$, is defined as $\mathcal{R} = \left\{\mathcal{O}, \mathcal{U}, \mathcal{L}_o, f, t, \bar{t}, t_i^C\right\}$, and $\mathcal{R}_0$ is the set of initial resource values.

The resources f, $\underline{t}$, and $\bar{t}$ keep track of the fuel left on board the helicopter and the earliest and latest feasible arrival time at node i. Further, the resource $\mathcal{O}$ keeps track of the orders on board the helicopter when reaching node i, while the resource $\mathcal{L}_o$ keep track of the number of legs travelled for each order $o \in \mathcal{O}$. The set $\mathcal{U}$ keeps track of orders which can no longer be serviced on the partial path. This may be because the order has already been served on the (partial) path, or because the time windows of the pickup and/or delivery node makes it impossible to service the order on any feasible extension of the path. Finally, the resource t_i^C keeps track of the amount of time that has elapsed since the visit at node i. The cost of travelling a trip can be calculated after a label has reached the end node, and is therefore not tracked.

4.1.2. Resource extension functions

This subsection presents the resource extension functions used when extending labels in the labeling algorithm.

$$f_{ij}^{\mathcal{O}}(\mathcal{O}(L)) = \begin{cases} \mathcal{O}(L) \cup \{j\} & j \in N^P \\ \mathcal{O}(L) \setminus \{j-n\} & j \in N^D \\ \mathcal{O}(L) & j = e(h) \end{cases} \tag{45}$$

$$f_{ij}^{\mathcal{U}}(\mathcal{U}(L)) = \begin{cases} \mathcal{U}(L) \cup \{j\} & j \in N^P \\ \mathcal{U}(L) & \text{otherwise} \end{cases} \tag{46}$$

$$f_{ij}^{\mathcal{L}_o}(\mathcal{L}_o(L)) = \begin{cases} \mathcal{L}_o(L) + 1 & j \notin N_i^I \wedge o \in \mathcal{O}(L) \\ L_o(L) & \text{otherwise} \end{cases} \tag{47}$$

$$f_{ij}^{f}(f(L)) = f(L) - F_{hij} \tag{48}$$

$$f_{ij}^{\underline{t}}(\underline{t}(L)) = \max\left\{\underline{T}_{hj},\ \underline{t}(L) + T_{ij}^L + T_{hij}\right\} \tag{49}$$

$$f_{ij}^{\bar{t}}(\bar{t}(L)) = \min\left\{\overline{T}_{hj},\ \bar{t}(L) + T_{ij}^L + T_{hij}\right\} \tag{50}$$

$$f_{ij}^{t_k^C}(t_k^C(L)) = \begin{cases} \max\left\{T_{jk}^C,\ t_k^C(L) - T_{hij} - T_{ij}^L\right\} & (j,k) \in N^C \\ \max\left\{0,\ t_k^C(L) - T_{hij} - T_{ij}^L\right\} & \text{otherwise} \end{cases} \tag{51}$$

The sets of open and unreachable orders are extended using Eqs. (45) and (46), respectively. In addition, the set of unreachable orders can be updated by considering the possible extensions. If an extension to a pickup node is found to be infeasible due to lack of fuel, or because the earliest possible arrival time is outside the time window, the pickup node is added to the set of unreachable orders. These resource windows cannot become feasible. Thus, it is impossible for any extension to the end node to handle the order. For a pair of time connected orders, the order related to the service that must be conducted first is added to the set $\mathcal{U}$, when the service related to the other order has been conducted. Eqs. (47) update the number of intermediate landings for each open order, and Eq. (48) update the fuel level. Eqs. (49) and (50) extend the lower and upper limits for the arrival at the current node, while Eqs. (51) update the time counter for all nodes.

4.1.3. Resource windows

An extension to node j is feasible for a label L if criteria (52)–(58) are met. Criterion (52) states that the extension cannot be a part of the set of unreachable orders if the evaluated extension is a pickup node. This criterion ensures that the related order has to be open in order to extend to a delivery node, and that a label cannot be extended to the end node if it has any open orders. Criterion (53) states that the maximum leg counters cannot exceeded their limits. Sufficient fuel level for the helicopter to return to the heliport is stated in criterion (54). Criterion (55) enforces that the number of personnel on board the helicopter does not violate its capacity. Criteria (56) and (57) ensure that the time window at the last node of the path is not violated. Finally, criterion (58) enforces that the time counter resource must be zero or negative, if node j is part of a connected pair of orders.

$$\left\{\begin{array}{ll} j \notin \mathcal{U} & j \in N_h^P \\ (j-n) \in \mathcal{O} & j \in N_h^D \\ \mathcal{O} = \varnothing & j = e(h) \end{array}\right\} \tag{52}$$

$$\mathcal{L}_k \leqslant \overline{L}, k \in \mathcal{O} \tag{53}$$

$$f(L) \geqslant \underline{F}_{hj} \tag{54}$$

$$\sum_{i \in \mathcal{O}(L)} P_i \leqslant K_h \tag{55}$$

$$\overline{t}(L) \geqslant \underline{T}_{hj} \tag{56}$$

$$\underline{t}(L) \leqslant \overline{T}_{hj} \tag{57}$$

$$t_j^C \leqslant 0 \tag{58}$$

4.2. Delayed constraint generation

A drawback of the model presented in Section 3.2 is that there is a large number of constraints ((36)–(43)), and a large number of binary variables (δ and γ) that are needed only to ensure that no two helicopters visit the same installation at the same time. One way to handle this, would be to categorize these constraints as *lazy constraints*, a concept available in most commercial solvers that removes the constraints from the model, and re-inserts them as they are needed throughout the B&B tree. However, these constraints are potentially quite weak because of the big-M formulations used in constraints (36)–(38), and constraints (36) and (37) are quite dense as they sum over the set of trips R_h.

Instead, we can reformulate the part of the mathematical model covering constraints (37)–(43). First, we re-define constraints (36) and (37) to only be active for nodes i that are part of a pairing N^C, by re-writing them as:

$$t_i \geqslant t_{hs} + \sum_{r \in R_h} T_{ihr}\lambda_{hrs} - M_{ih}\left(1 - \sum_{r \in R_h} A_{ihr}\lambda_{hrs}\right),\ (i,*),(*,i) \in N^C, h \in H, s \in S, \tag{59}$$

$$t_i \leqslant t_{hs} + \sum_{r \in R_h} T_{ihr}\lambda_{hrs} + \overline{T}_i^P\left(1 - \sum_{r \in R_h} A_{ihr}\lambda_{hrs}\right),\ (i,*),(*,i) \in N^C, h \in H, s \in S. \tag{60}$$

This is done in order to handle constraints (35). Note that the number of connected orders is usually quite small in practice, leading to few constraints of this type.

Second, to enforce that at most one helicopter is present at each offshore installation at any given time we introduce an alternative formulation, using an exponential number of constraints. Let

$$\Theta_i = \{(h,r,s) | h \in H,\ r \in R_h\, s \in S,\ A_{ihr} = 1\}$$

and for ease of exposition we define $\theta_k = (h_k, r_k, s_k)$. Using this notation we can re-formulate the model as:

$$t_{h_1 s_1} + T_{h_1 r_1 i}\lambda_{h_1 r_1 s_1} - \left(t_{h_2 s_2} + T_{h_2 r_2 j}\lambda_{h_2 r_2 s_2}\right) - T^L \geqslant -M_{h_2}^2\left(1 - \delta_{\theta_1\theta_2}\right),$$
$$i \in N^S, j \in N_i^I, \theta_1 \in \Theta_i, \theta_2 \in \Theta_j | h_1 \neq h_2 \tag{61}$$
$$\delta_{\theta_1\theta_2} + \delta_{\theta_2\theta_1} \geqslant \lambda_{h_1 r_1 s_1} + \lambda_{h_2 r_2 s_2} - 1$$
$$i \in N^S, j \in N_i^I, \theta_1 \in \Theta_i, \theta_2 \in \Theta_j | i < j, h_1 \neq h_2, \tag{62}$$
$$\delta_{\theta_1\theta_2} \in \{0,1\}, \quad i \in N^S, j \in N_i^I, \theta_1 \in \Theta_i, \theta_2 \in \Theta_j | h_1 \neq h_2. \tag{63}$$

Constraints (61) correspond to constraints (36)–(38), with $M_h^2 = \max_{r \in R_h}\{\overline{T}_{hr}\} - \underline{T}_i - T^L$. Note that each of these constraints contains exactly five variables, making them very sparse. Constraints (62) correspond to constraints (39), and constraints (63) put binary restrictions on the δ-variables.

Since there is an exponential number of these constraints, it would be impractical, and in many cases impossible to add all of them to the model apriori. To avoid this we instead use delayed constraint generation, where constraints (37)-(43) are initially removed from the model, and the B&B algorithm is applied to the relaxed model. Whenever a potential best integer solution is found, we check whether it violates any of the constraints (61)–(63). If it does not, then the solution provides a primal bound, otherwise the violated constraints are added to make the solution infeasible, and the B&B node is re-solved.

4.3. Preprocessing

Both the set of arcs for a given helicopter A_h and time windows of some of the orders can be preprocessed in order to reduce the number of variables in the arc-flow model, the number of label extensions in Algorithm 1, and the size of the solution space. This section proposes a number of conditions that imply that an arc can be removed from the graph or that a time window can be narrowed.

4.3.1. Preprocessing of arcs

Consider a helicopter h and the arc $(i,j) \in A_h$. Under the following circumstances may the arc be removed from the problem:

1. Arc is going from a start node or refueling node to a delivery node. $i \in \mathcal{N}^F \bigcup \{s(h)\} \wedge j \in \mathcal{N}^D$
2. Arc from a pickup node to an end node or refueling node. $i \in \mathcal{N}^P \wedge j \in \mathcal{N}^F \bigcup \{e(h)\}$
3. Node i or j, or their corresponding pickup/delivery node, is located at a another heliport than the one used by helicopter h.
4. The nodes i and j are at the same installation, and $j \in N_i^I \wedge i < j$. This also breaks symmetry.
5. Arrival at node j is too late. $\underline{T}_{hi} + T_{ij}^L + T_{hij} > \overline{T}_{hj}$

6. Arrival at node j is too early. $\overline{T}_{hi} + T^L_{ij} + T_{hij} < \underline{T}_{hj}$
7. The nodes are connected in the opposite order. $(j,i) \in N^C$
8. It is impossible to get the required time difference if the arc is used.
   $T_{hij} + T^L_{ij} < T^C_{ij}$ and $(i,j) \in N^C$
9. Given the maximum amount of fuel it is possible to have when arriving at node i, it must be possible to fulfill the orders related to nodes i and j, and travel back to the helicopter's heliport.

$$F_{hij} + F_{h(i+n)(j+n)} + \min \begin{Bmatrix} F_{hj(i+n)} + F_{h(j+n)e(h)} \\ F_{hi(j+n)} + F_{h(i+n)e(h)} \end{Bmatrix} > \overline{F}_{hi} \quad i,j \in N^P \tag{64}$$

$$F_{hij} + F_{hj(i+n)} + F_{h(i+n)e(h)} > \overline{F}_{hi} \quad i \in N^P, j \in N^D \tag{65}$$

$$F_{hij} + F_{hj(j+n)} + F_{h(j+n)e(h)} > \overline{F}_{hi} \quad i \in N^D, j \in N^P \tag{66}$$

$$F_{hij} + F_{hje(h)} > \overline{F}_{hi} \quad i,j \in N^D \tag{67}$$

10. Node i is a pickup node at an installation with a related delivery node located at another installation, while j is a node located on a heliport.
11. Node i is a delivery node located at a heliport and j is a pickup node.
12. The helicopter does not have enough capacity to handle the orders related to nodes i, and j simultaneously. $(P_i + P_j > K_h) \wedge \neg\left(i \in N^D \wedge j \in N^P\right)$.
13. The helicopter does not have enough capacity to handle the order size of node i or j. $\max(P_i, P_j) > K_h$.

Applying all of these rules, we can greatly reduce the solution space of the compact formulation, and the number of extensions made in the labeling algorithm.

4.3.2. Preprocessing of time windows

The time dependent orders create the possibility of reducing their respective time windows. This method is based on the time window reduction presented by Dohn et al. (2011). The reduction of the time windows for a given pairing $(i,j) \in N^C$ is formulated in Table 2.

4.4. Aggregated branching variables

The choice of what variables to branch on in each node in the B&B tree, affects the effectiveness of the algorithm (Gamrath et al., 2015). Dumas et al. (1991) and Andersson et al. (2011) include variables in their PDPs for the sole purpose of branching on them. Dumas et al. (1991) branch on variables related to each order in a PDP, while Andersson et al. (2011) use aggregated branching variables, also referred to as constraint branching, in the maritime pickup and delivery problem with split loads.

Aggregated branching variables have been utilized in the the trip based model. Here, variables with the highest branching priority are a set of variables μ_{hs}, which equals 1 if helicopter h travels any trip on trip number s, and 0 otherwise. Eq. (68) provides the connection between these variables and the original variables in the model.

$$\mu_{hs} = \sum_{r \in R_h} \lambda_{hrs}, \quad h \in H, s \in S \tag{68}$$

The set of aggregated branching variables with the second highest priority, are similar to the binary variables applied by Andersson et al. (2011). The binary variable κ_{ih}, equals 1 if helicopter h conducts a pickup at node i, and 0 otherwise. The variables are connected with the original variables through Eq. (69).

$$\kappa_{ih} = \sum_{r \in R_h} \sum_{s \in S} A_{ihr} \lambda_{hrs}, \quad i \in N^P, h \in H \tag{69}$$

Table 2
Preprocessing of time windows for connected orders.

Node	Pre-reduction	Post-reduction
i	$[\underline{T}_i, \overline{T}_i]$	$\left[\underline{T}_i, \min\left\{\overline{T}_i, \overline{T}_j - T^C\right\}\right]$
j	$[\underline{T}_j, \overline{T}_j]$	$\left[\max\left\{\underline{T}_j, \underline{T}_i + T^C\right\}, \overline{T}_j\right]$

5. Computational study

In this section we perform a study of the computational performance of the two mathematical models presented in Section 3, when solved using Gurobi 9.0.0. All tests were conducted on a Lenovo NextScale nx360 M5 computer, with an Intel E5-2643v3 six core 2×3.40GHz processor and 512.0 GB installed memory, running on a Linux operating system. The test instance generation and labeling algorithm were implemented in C++.

In Section 5.1 we explain how the test instances used in this computational study were generated, before we compare the computational performance of the compact and trip-based model formulations in Section 5.2. Further, we test the effect the number of connected orders in the set N^C has on the computational time in Section 5.3, before finally comparing the computational effect of solving instances with a single, rather than multiple, heliport (s) in Section 5.4.

5.1. Test instance generation

The input for the test instance generation was extracted from flight records for helicopters on the Norwegian continental shelf in 2018. Selected heliports were Bergen Airport Flesland and Stavanger Airport Sola, which are the two most used heliports on the NCS. In addition we selected ten installations, which have frequent flights to or from these heliports, and are reachable from both heliports. This created a network where all locations can be reached from an arbitrary location in the network. A map showing the location of the heliports and the offshore installations can be seen in Fig. 2.

Each instance is characterized by the following features:

- Number of orders, i.e. the instance size.
- Number of personnel in each order.
- Pickup and delivery location of each order.
- Time windows for each order.
- What orders that are time connected orders, and required time differences.

Three instances were generated per instance size. The number of personnel per order, was drawn randomly in the interval between 6 and 10, and the time windows were given widths of one hour. The number of orders between installations were set to be 25 % of the total orders in an instance, although flight records suggest that the actual percentage is closer to 5 %. This relatively high percentage imposes a stronger need for coordination and synchronization between helicopters, as helicopters must conduct more visits at installations. We thus get test instances where this aspect is prominent.

The helicopters used in the test instances were the Super Puma and Sikorsky S-92, which are the most used helicopter types on the

Fig. 2. Overview of the two airports (squares) and the location of the 10 offshore installations (crosses) used to generate the test instances.

NCS. The number of helicopters included in each instance depends on the number of orders. For a given number of orders, n, the number of helicopters included, $|H|$, was given by $|H| = 2 \cdot \lceil \frac{n}{5} \rceil + 2$. This number just needs to be large enough to make the instance feasible, since a part of the objective is to minimize the number of helicopters used. The helicopters were evenly distributed among the two heliports.

The testing of the solution methods is divided into three parts. For the first part, presented in Section 5.2, one pair of time connected orders was included in each instance. In the second part of the testing, presented in Section 5.3 we increased the number of pairs of connected orders in each instance from 0 and up to 8. In both these parts, the time connected orders were determined by randomly choosing the pickup and the delivery of two different orders. Finally, in the third part of the testing, presented in Section 5.4, we removed the heliport Flesland from each instance and moved all pickup and delivery nodes, and helicopters, to Sola Airport.

5.2. Comparison of models

A comparison of the computational effort needed to solve the compact and trip based models, respectively, using a commercial solver for instances between 8 and 16 orders is presented in Table 3. For the compact model we give the optimality gap at the termination of the B&B search (Opt. gap) and the computing time in seconds (Comp. time). For the trip-based model we give the optimality gap, the time spent generating trips (Trip gen. time), the time spent solving the mathematical model using a commercial solver (B&B time), and the total computing time (Comp. time) which is the sum of the two. Maximum computing time for both solution methods was set to 3600 seconds.

As can be seen from the table, the compact model struggles to solve instances with 10 orders, and once the number of orders is increased to 12 or more, none of the instances may be solved within one hour. Most of the instances also have an optimality gap of 100%, indicating that not even a feasible solution was found to the problem within the time limit. However, the trip based model can solve all instances from 8 to 16 orders within two seconds.

It is clear from the results presented in Table 3 that the trip-based model outperforms the compact model in terms of solution quality within a limited computing time when applying standard commercial solvers to both model formulations. To see how the trip-based model handles larger test instances, we compare three solution methods based on this model formulation. The first is to use standard branch-and bound (B&B) to solve the model presented in Section 3, while the second one is to define constraints (37)–(43) as *lazy constraints* during the execution of the B&B algorithm (Lazy). The third is to use delayed constraint generation (DCG), as described in Section 4.2.

Common for all the solution methods, is that all trips have to be generated apriori. In Table 4 we report the time used for generating trips, and the number of trips generated, for instances with 17 to 52 orders. Each row is an aggregation of 9 instances, with 3 instances of each order size. For each of these sets of 9 instances, we give the minimum, average, and maximum time, and number of trips, respectively. Unsurprisingly, the general trend is that both the time and number of trips increase with an increasing number of orders, however, there are some exceptions. E.g. the maximum time spent on the instances with an order size in the 44–46 interval is larger than the maximum time spent on instances with an order size in the 50–52 interval. In general we see that the total time spent is less than one minute for all the instances up to 40 orders, and even for some of the instances up to 52 orders. In the worst case over all instances tested, it takes less than 12 min (720 s) to generate all trips. We further observe that the number of trips generated varies a lot within each order interval, with e.g. the instances in the 44–46 interval ranging from 100,000 to 10,000,000 trips. This indicates that the distribution of orders across heliports and offshore installations, and the possibilities to remove arcs and reduce time windows, have a large impact on the number of feasible trips for each helicopter.

The results of running the three solution methods for larger instances are presented in Table 5. For each of the solution methods, and each order size interval, we give the number of instances solved to optimality (# Opt.), the average optimality gap (Opt. gap), and the total computing time in seconds (Comp. time). The last line of the table (Tot./Avg.) gives the total number of instances solved, and the average optimality gap and computing time, respectively, over all tested instances. For each instance, the maximal time of

Table 3
Table comparing the computational performance of the compact model presented in Section 3.1 and the trip-based model presented in Section 3.2. For the compact model we present the optimality gap (Opt. gap) and the computing time (Comp. time), while for the trip-based model we present the optimality gap (Opt. gap) the time spent generating trips (Trip gen. time), the time spent in the B&B search (B&B time) and the total computing time (Comp. time).

	compact model		trip-based model			
# Orders	Opt. gap	Comp. time [*s*]	Opt. gap	Trip gen. time [*s*]	TtB&B time [*s*]	Comp. time [*s*]
8	0.00 %	37	0.00 %	0.01	0.03	0.04
8	0.00 %	11	0.00 %	0.00	0.03	0.03
8	0.00 %	298	0.00 %	0.01	0.12	0.13
10	100.00 %	3600	0.00 %	0.01	0.32	0.33
10	0.00 %	619	0.00 %	0.01	0.09	0.10
10	25.20 %	3600	0.00 %	0.01	0.31	0.32
12	100.00 %	3600	0.00 %	0.01	0.33	0.35
12	100.00 %	3600	0.00 %	0.01	0.04	0.05
12	53.65 %	3600	0.00 %	0.01	0.44	0.45
14	100.00 %	3600	0.00 %	0.03	1.12	1.15
14	100.00 %	3600	0.00 %	0.01	0.28	0.29
14	100.00 %	3600	0.00 %	0.02	0.52	0.54
16	100.00 %	3600	0.00 %	0.04	1.48	1.52
16	100.00 %	3600	0.00 %	0.02	0.86	0.88
16	100.00 %	3600	0.00 %	0.05	1.73	1.77

Table 4
This table gives an overview of the minimum, average, and maximum time spent generating trips using Algorithm 1, and the minimum, average, and maximum number of trips produced for subsets of 9 test instances with an increasing number of orders.

	Time generating trips			Number of trips		
Orders	Min	Average	Max	Min	Average	Max
17–19	0.01	0.05	0.20	845	4 231	18 115
20–22	0.01	0.11	0.39	1 390	8 427	25 902
23–25	0.03	0.11	0.32	2 868	8 616	22 230
26–28	0.08	0.40	1.97	6 720	21 087	75 117
29–31	0.09	0.74	2.23	6 790	34 368	87 416
32–34	0.19	5.51	18.50	13 472	172 859	535 536
35–37	1.30	7.18	16.14	50 616	198 262	441 960
38–40	1.56	11.56	42.35	53 442	278 478	871 101
41–43	3.66	35.85	82.94	103 800	780 779	1 697 120
44–46	4.36	123.47	696.44	132 187	2 137 780	10 902 628
47–49	16.13	92.33	274.94	762 245	2 021 542	4 945 039
50–52	15.51	218.86	621.90	942 007	3 793 540	9 411 108

Table 5
Table comparing solving the trip-based model with standard B&B, with B&B using lazy constraints (Lazy), and delayed constraint generation (DCG), respectively. For each method we give the number of instances solved to optimality (# Opt.), the average optimality gap (Opt. gap), and the average computing time (comp. time).

	B&B			Lazy			DCG		
Orders	# Opt.	Opt. gap	Comp. time [*s*]	# Opt.	Opt. gap	Comp. time [*s*]	# Opt.	Opt. gap	Comp. time [*s*]
17–19	9	0.00 %	16.36	9	0.00 %	5.49	9	0.00 %	4.73
20–22	9	0.00 %	190.40	9	0.00 %	54.31	9	0.00 %	10.39
23–25	9	0.00 %	275.82	9	0.00 %	33.98	9	0.00 %	44.76
26–28	8	1.58 %	928.97	9	0.00 %	395.29	9	0.00 %	63.68
29–31	7	2.57 %	1541.25	9	0.00 %	969.34	8	0.62 %	772.64
32–34	1	20.61 %	3233.70	3	35.80 %	2540.17	3	16.18 %	2579.32
35–37	1	28.76 %	3413.04	3	28.07 %	2935.91	3	14.84 %	2585.45
38–40	0	23.51 %	3600.00	2	67.07 %	3365.38	2	41.16 %	3151.38
41–43	0	53.18 %	3600.00	0	68.79 %	3600.00	0	39.12 %	3600.00
44–46	0	64.36 %	3600.00	0	48.61 %	3600.00	0	71.34 %	3600.00
47–49	0	90.69 %	3600.00	0	100.00 %	3600.00	0	100.00 %	3600.00
50–52	0	91.34 %	3600.00	0	100.00 %	3600.00	0	100.00 %	3600.00
Tot/Avg	44	31.38 %	2300.0	53	37.36 %	2058.3	52	31.94 %	1967.7

the B&B search is adjusted to account for the time spent generating the trips so that the maximal total computing time is one hour (3600 s).

As can be seen from the table, all methods solve all instances with up to 25 orders, though the standard B&B method is significantly slower than the other two. For the instances with between 26 and 31 orders, the three methods can solve 15, 18, and 17 out of 18 instances, respectively. Once the number of orders becomes greater than 31, less than half of the instances within each order size interval can be solved, and once the number of orders passes 40, none of the methods can solve any instance to optimality within one hour, though feasible solutions can be found for many of the instances. Overall, we see that both delayed constraint generation and applying lazy constraints outperforms standard B&B, both when it comes to the number of instances solved, and the average computing time. It is, however, notable that B&B finds feasible solutions to some of the largest instances, which neither of the other solution methods do. Comparing DCG with Lazy shows

that neither outperforms the other. Using lazy constraints we are able to solve one more instance to optimality than when using DCG, while the latter has a lower average optimality gap and computing time. Thus, for instances with one pair of connected orders, both of these methods seem to be equally good at solving the problem.

5.3. Testing the effect of connected orders

In the tests presented above, the instances contained only one pair of connected orders. In order to test the effect of the number of connected orders on the computational performance of the proposed methods, we have generated 5 additional instances with 30 orders each. Each of these instances has been solved 9 times, with an increasing number of connected orders, from 0 to 8 pairs. The computational performance of these tests can be seen in Table 6. Here we have aggregated the results according to the number of connected orders, and give the number of instances solved to optimality within one hour (# Opt.), the average optimality gap (Opt. gap), and the computing time (Comp. time). For the DCG method we also list the average number of times constraints (61)–(63) were added during the B&B search (# Cons.).

The table shows that the trend from the previous tests, where the standard B&B performed inferior to the other two methods, is present in these tests as well. It solves the least number of instances, and has a higher average computing time than the other two. What is interesting in Table 6 is that the delayed constraint generation seems to significantly outperform the B&B with lazy constraints, which spends, on average, about 50 % longer, and solves 4 instances less within one hour, compared with the DCG method. The general trend for all three methods is also that increasing the number of connected orders, increases the computing time, however, there are some outliers. E.g. when using lazy constraints, the instances without any connected orders take, on average, longer to solve than 1–6 pairs of connected orders, and it is unable to solve one of the instances.

In Table 7 we give the average number of trips generated (# Trips), and the average number of B&B nodes generated (# B&B nodes), when using the three different methods proposed in the paper. First, we notice that the number of trips is reduced significantly, when the number of connected orders increase. This is due to the preprocessing techniques described in Section 4.3, where we both remove arcs and narrow time windows for the nodes representing these orders. There thus exists fewer feasible paths through the problem defining network, reducing the number of trips generated by the labeling algorithm described in Section 4.1. When the number of connected orders reaches 8, we see an almost 30 % reduction in the the average number of trips, compared with when there are no connected orders.

When comparing the number of nodes in the B&B tree, there is also a trend that the delayed constraint generation creates a larger B&B tree than the other two. In some cases, this may be explained by the fact that the other two methods did not terminate their search. However, e.g. for 2 connected orders we can see that using lazy constraints gives roughly 53,000 B&B nodes on average, while delayed constraint generation gives 145,000 nodes when all instances are solved to optimality. Despite generating much larger B&B trees, we still get significantly shorter computing times, when using DCG, indicating that less time is used in each node. This is likely due to the fact that fewer constraints are added to the formulation at a time, and the added constraints are less dense. Both these factors are known to influence the time it takes to solve a linear program.

5.4. Testing the effect of one vs two heliports

To show that the presented methodology is also viable in the single heliport case, and to compare the computational performance, we have altered the instances presented in Section 5.1 by moving all pickup and delivery nodes located at the heliport in Bergen, to Stavanger. The results of using the DCG solution method on the test instances from 17 to 37 orders are presented in Table 8. Each row is an aggregation of nine instances, while the first three columns presents the number of instances solved to optimality (# Opt), the average optimality gap (Opt. gap), and the average total computing time (Comp time). The last two columns show the percentage deviation in objective value (Δ obj.) and the deviation in the number of trips generated (Δ trips), respectively, when comparing the results with the original test instances with two heliports. When calculating these deviations, we have only considered the subset of instances where both versions have been solved to optimality.

As seen in Table 8 we can solve all instances up to 31 order to optimality, and further 8 of the instances from 32 to 37 orders. Comparing these results to those presented in Table 3 we can see that the DCG solution method can solve 3 more instances, and has, on average, both smaller optimality gaps and shorter computing times. We also see that the number of trips generated is not effected much by the reduction from two to one heliports. Thus, we may conclude that the methodology works equally well for instances with a single heliport.

Studying the results presented in Table 8, we further see that the objective values, on average, increase slightly when going from two to one heliports. This is not surprising, as the helicopters have to fly a longer distances to service those installations closest to Bergen. Comparing the optimal solutions of the test instances with one and two heliports, the number of helicopters used are mostly the same in both cases, however, there are both instances where it increases and decreases by one.

Table 6
Table comparing solving the trip-based model with B&B, Lazy, and DCG, respectively, when the number of connected orders is increased. For each method we give the number of instances solved to optimality (# Opt.), the average optimality gap (Opt. gap), and the average computing time (Comp. time). For the DCG method, the number of times constraints were added is also listed (# Cons).

	B&B			Lazy			DCG			
$\lvert\mathcal{N}^C\rvert$	# Opt.	Opt. gap	Comp. time	# Opt.	Opt. gap	Comp. time	# Opt.	Opt. gap	Comp. time	# Cons.
0	5	0.00 %	2595.24	4	0.33 %	1668.36	5	0.00 %	471.59	73.6
1	3	0.95 %	2804.86	5	0.00 %	1095.65	5	0.00 %	359.72	76.0
2	3	0.95 %	2501.21	5	0.00 %	1089.97	5	0.00 %	737.95	94.6
3	3	1.44 %	2368.23	5	0.00 %	1001.42	5	0.00 %	816.27	151.8
4	3	3.15 %	3067.16	5	0.00 %	1034.59	5	0.00 %	1177.73	99.6
5	3	4.03 %	2469.74	4	0.72 %	1545.19	5	0.00 %	909.73	137.0
6	4	1.58 %	2223.34	4	0.33 %	1583.26	4	1.34 %	1194.56	144.0
7	3	3.27 %	2222.25	4	0.20 %	1722.28	5	0.00 %	737.87	145.2
8	3	4.19 %	2215.46	3	3.47 %	2058.27	4	0.94 %	1829.65	161.8
Tot/Avg	30	2.17 %	2496.39	39	0.56 %	1422.11	43	0.25 %	915.01	120.4

Table 7
Table listing the average number of trips generated (# Trips) and the average number of B&B nodes (# B&B nodes) generated when solving the trip-based model with B&B, Lazy, and DCG, respectively, when the number of connected orders is increased.

$\lvert\mathcal{N}^C\rvert$	# Trips	# B&B nodes		
		B&B	Lazy	DCG
0	47695.2	57440.8	77568.0	69721.2
1	45564.4	33261.0	53397.0	55336.6
2	45151.4	29664.0	53671.8	144530.4
3	42890.4	21094.4	59297.4	132799.2
4	41573.0	47016.0	88488.6	345396.4
5	38420.2	30898.8	93327.4	271439.8
6	36524.6	38656.6	207333.6	327316.6
7	33749.8	34993.0	194810.0	147105.4
8	30298.8	35063.6	171104.2	635135.6

Table 8
Table comparing solving the test instances with one and two heliports using the DCG solution method. The table lists the number of instances solved to optimality (# Opt), the average optimality gap (Opt. gap) and total computing time (Comp time), as well as the average difference in objective value objective value (Δ obj.) and number of trips (Δ trips) when using one instead of two heliports.

Orders	# Opt.	Opt. gap	Comp time [s]	Δ obj.	Δ trips
17–19	9	0.00 %	3.90	1.21 %	-5.56
20–22	9	0.00 %	8.51	0.41 %	-196.22
23–25	9	0.00 %	59.13	1.38 %	-98.00
26–28	9	0.00 %	95.60	1.99 %	-597.33
29–31	9	0.00 %	376.69	0.34 %	175.33
32–34	4	14.14 %	2273.47	0.15 %	2199.11
35–37	4	6.20 %	2378.85	1.13 %	-4214.44

6. Concluding remarks

In this paper we have studied a rich helicopter flight scheduling problem from the offshore oil and gas industry. The problem consists of designing routes for helicopters to transport personnel either between heliports onshore and offshore installations or between offshore installations. The problem can be modelled as a rich vehicle routing problem, which includes a pickup and delivery structure, heterogeneous fleet of vehicles, multiple trips, multiple depots, and temporal synchronization of transportation tasks.

To solve this problem, we have presented a trip-based model, where all trips are generated apriori, and a mathematical model that puts these trips together to form feasible routes. To further improve the solution time when solving these models using branch-and-bound (B&B), we have re-formulated the model using an exponential number of constraints which is added to the model using delayed constraint generation (DCG) during the B&B search. The computational experiments indicate that when the number of temporal dependencies in the model become large, the delayed constraint generation method outperforms both standard B&B, and using the *lazy constraints* option built into (most) commercial solvers. Further investigation of the computational results indicate that the using DCG generates larger B&B trees than the other two methods, and thus the explanation for its improved performance is likely to stem from the fact that it adds fewer and sparser constraints to the model, thus affecting the time needed to solve each node in the B&B tree less than when adding lazy constraints. Finally, testing of the DCG indicates that there is little difference in the computational effort needed to solve instances with one and two heliports.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Gaute Messel Nafstad: Conceptualization, Methodolody, Software, Investigation, Writing - review & editing, Visualization. **Amund Haugseth:** Conceptualization, Methodolody, Software. **Vebjørn Høyland:** Conceptualization, Methodolody, Software. **Magnus Stålhane:** Conceptualization, Methodolody, Supervision, Writing - original draft, Writing - review & editing.

Acknowledgements

We would like to thank the two referees for their valuable comments and suggestions. We would further like to thank TietoEVRY for their cooperation and for providing us with industry data.

References

Andersson, H., Christiansen, M., Fagerholt, K., 2011. The maritime pickup and delivery problem with time windows and split loads. INFOR 49 (2), 79–91.
Baldacci, R., Bartolini, E., Mingozzi, A., 2011. An exact algorithm for the pickup and delivery problem with time windows. Oper. Res. 59 (2), 414–426.
Baldacci, R., Bartolini, E., Mingozzi, A., Roberti, R., 2010. An exact solution framework for a broad class of vehicle routing problems. Comput. Manage. Sci. 7 (3), 229–268.
Baldacci, R., Mingozzi, A., 2009. A unified exact method for solving different classes of vehicle routing problems. Math. Program. 120 (2), 347.
Berbeglia, G., Cordeau, J.-F., Gribkovskaia, I., Laporte, G., 2007. Static pickup and delivery problems: a classification scheme and survey. TOP 15 (1), 1–31.
Aerospace Technology, 2018. Norway's largest industries. URL:https://www.aerospace-technology.com/projects/s92/..
British Petroleum Company, 2020. bp Statistical Review of World Energy 2020. URL: https://www.bp.com/content/dam/bp/business-sites/en/global/corporate/pdfs/energy-economics/statistical-review/bp-stats-review-2020-full-report.pdf..
Cattaruzza, D., Absi, N., Feillet, D., 2016. Vehicle routing problems with multiple trips. 4OR 14 (3), 223–259.
Contardo, C., Martinelli, R., 2014. A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints. Discr. Optim. 12, 129–146.
Dohn, A., Rasmussen, M.S., Larsen, J., 2011. The vehicle routing problem with time windows and temporal dependencies. Networks 58 (4), 273–289..
Drexl, M., 2012. Synchronization in vehicle routing–a survey of VRPs with multiple synchronization constraints. Transp. Sci. 46 (3), 297–316.

Drexl, M., Schneider, M., 2015. A survey of variants and extensions of the location-routing problem. Eur. J. Oper. Res. 241 (2), 283–308.
Dumas, Y., Desrosiers, J., Soumis, F., 1991. The pickup and delivery problem with time windows. Eur. J. Oper. Res. 54, 7–22.
Fernández-Cuesta, E., Norddal, I.K., Andersson, H., Fagerholt, K., 2017. Base location and helicopter fleet composition in the oil industry. In: INFOR: Information Systems and Operational Research, pp. 71–92. URL:https://doi.org/10.1080/03155986.2016.1262583.
Galvão, R., Guimarães, J., 1987. An algorithm for helicopter routing in the support of offshore oil exploration. In: Proceedings of the XX Brazilian OR Symposium, pp. 96–108.
Galvão, R.D., Guimarães, J., 1990. The control of helicopter operations in the Brazilian oil industry: Issues in the design and implementation of a computerized system. Eur. J. Oper. Res. 49 (2), 266–270.
Gamrath, G., Melchiori, A., Berthold, T., Gleixner, A.M., Salvagnin, D., 2015. Branching on multi-aggregated variables. In: Michel, L. (Ed.), Integration of AI and OR Techniques in Constraint Programming. Springer International Publishing, Cham, pp. 141–156.
Gschwind, T., 2015. A comparison of column-generation approaches to the synchronized pickup and delivery problem. Eur. J. Oper. Res. 247 (1), 60–71.
Hermeto, N. d. S.S., Filho, V.J.M.F., Bahiense, L., 2014. Logistics network planning for offshore air transport of oil rig crews. Computers & Industrial Engineering 75, 41 – 54..
Hernandez, F., Feillet, D., Giroudeau, R., Naud, O., 2016. Branch-and-price algorithms for the solution of the multi-trip vehicle routing problem with time windows. Eur. J. Oper. Res. 249 (2), 551–559.
Irnich, S., Desaulniers, G., 2005. Shortest path problems with resource constraints. In: Desaulniers, G., Desrosiers, J., Solomon, M.M. (Eds.), Column Generation. GERAD 25th Anniversary Series. Springer, pp. 33–65.
Koç, Ç., Bektaş, T., Jabali, O., Laporte, G., 2016. Thirty years of heterogeneous vehicle routing. Eur. J. Oper. Res. 249 (1), 1–21.
Menezes, F., Porto, O., Reis, M.L., Moreno, L., Aragão, M.P. d., Uchoa, E., Abeledo, H., Nascimento, N.C. d., 2010. Optimizing helicopter transport of oil rig crews at Petrobras. Interfaces 40 (5), 408–416..
Mingozzi, A., Roberti, R., Toth, P., 2013. An exact algorithm for the multitrip vehicle routing problem. INFORMS J. Comput. 25 (2), 193–207.
Montoya-Torres, J.R., Franco, J.L., Isaza, S.N., Jiménez, H.F., Herazo-Padilla, N., 2015. A literature review on the vehicle routing problem with multiple depots. Comput. Ind. Eng. 79, 115–129.
Moreno, L., de Aragão, M.P., Uchoa, E., 2006. Column generation based heuristic for a helicopter routing problem. In: Àlvarez, C., Serna, M. (Eds.), Experimental Algorithms. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 219–230.
Moreno, L., Poggi de Aragao, M., Porto, O., Reis, M., 2005. Planning offshore helicopter flights on the Campos Basin. XXXVII Simpósio Brasileiro de Pesquisa Operacional (SBPO), Gramado, Brazil, pp. 967–976..
Norwegian Petroleum, 2020a. ECONOMY. URL: https://www.norskpetroleum.no/en/economy/..
Norwegian Petroleum, 2020b. Everything you need to know about norwegian petroleum activities. URL: https://www.norskpetroleum.no/en/.
Pessoa, A., Sadykov, R., Uchoa, E., 2018. Enhanced branch-cut-and-price algorithm for heterogeneous fleet vehicle routing problems. Eur. J. Oper. Res. 270 (2), 530–543.
Qian, F., Gribkovskaia, I., Sr, Øyvind Halskau, 2011. Helicopter routing in the norwegian oil industry: Including safety concerns for passenger transport. Int. J. Phys. Distrib. Logist. Manage. 41 (4), 401–415. URL:https://doi.org/10.1108/09600031111131959.
Qian, F., Strusevich, V., Gribkovskaia, I., Halskau, Ø., 2014. Minimization of passenger takeoff and landing risk in offshore helicopter transportation: models, approaches and analysis. Omega 51, 93–106.
Ropke, S., Cordeau, J.-F., 2009. Branch and cut and price for the pickup and delivery problem with time windows. Transp. Sci. 43 (3), 267–286.
Sierksma, G., Tijssen, G.A., Jan 1998. Routing helicopters for crew exchanges on off-shore locations. Ann. Oper. Res., 261–286
Statistics Norway, 2019. External trade in goods. URL:https://www.ssb.no/en/utenriksokonomi/statistikker/muh/aar..
Statistics Norway, 2019. Spillover-effects from the offshore petroleum to the mainland economy. URL:https://www.ssb.no/nasjonalregnskap-og-konjunkturer/artikler-og-publikasjoner/_attachment/405655?_ts=16ecb1da138..
TietoEVRY, 2020. Dawinci industry hub - changing perspectives on oil & gas logistics. URL:https://www.tietoevry.com/en/industries/oil-and-gas/Integrated-logistics-management/..