An Exact and Heuristic Approach for the Ship-to-Shore Problem

M. Wagenvoort [*][†][‡][§] P.C. Bouman [†] M. van Ee [‡] T. Lamballais Tessensohn [§]
K. Postek [¶]

Econometric Institute Research Papers, EI 2022-05
April 2022

Abstract
After a hurricane, for example hurricane Irma in Sint-Maarten, the navy can provide aid by bringing supplies, helping to clear roads and evacuating victims. If the destinations cannot be reached over land via a port, resources can be transported using smaller ships and helicopters, called connectors. This has to be done efficiently so that the supply provision can start as soon as possible. Planning such an operation is known as the ship-to-shore problem, which is a combination of a heterogeneous vehicle routing and a bin-packing problem, which we prove to be an $\mathcal{NP}$-hard problem. The aim is to schedule the connector trips to the shore and determine what resources should be loaded onto the connectors for each of their trips while minimising the duration of the operation. Connectors have different sizes, weight capacities, speeds and (un)loading times. Scheduling such an operation is currently done manually, which we mimic using a greedy heuristic. We aim to determine the quality of the greedy heuristic and determine in what cases it performs well and in what cases improvements can be found. To determine the quality, we solve the problem using a branch-and-price algorithm in which we take a set of ways to feasibly load the connectors as input. We use data provided by the Royal Netherlands Navy to construct 98 instances and find that the greedy heuristic can find an optimal solution in the majority of the cases. However, when the problem is less constrained in terms of the requirements regarding the order in which the resources should be delivered, the greedy heuristic is more likely to choose a suboptimal trip and can result in a solution that is far from optimal.

Keywords: Ship-to-shore problem, branch-and-price, integer linear programming, computational complexity

[*]Corresponding author: `wagenvoort@ese.eur.nl`

[†]Erasmus University Rotterdam, Erasmus School of Economics, Econometric Institute, P.O. Box 1738, 3000 DR Rotterdam, The Netherlands

[‡]Netherlands Defence Academy, Faculty of Military Sciences, Het Nieuwe Diep 8, 1781 AC Den Helder, The Netherlands

[§]TNO, Military Operations, Oude Waalsdorperweg 63, 2597 AK The Hague, The Netherlands

[¶]Delft University of Technology, Faculty of Electrical Engineering, Mathematics and Computer Science, Delft Institute of Applied Mathematics, Mekelweg 4, 2618 CD Delft, The Netherlands.

1 Introduction

After a natural disaster, such as a hurricane, the navy can provide aid by, for example, bringing supplies, helping to clear roads, and evacuating victims. In case of coastal areas, the navy provides support by transporting supplies using smaller ships and helicopters, called *connectors*. This transport has to be done efficiently for the help on land to start as soon as possible. Also the planning of such an operation has to be done fast because knowledge of the situation changes quickly, changing what needs to be delivered where, requiring the plan to be altered on the fly. In addition to humanitarian purposes, this problem can be encountered in other operations such as assault, withdrawal, raid, or support of other operations (Maritime Warfare Centre, 2019).

Planning such an operation is known as the ship-to-shore problem. The aim is to minimise the duration of the operation by scheduling connector round-trips to the shore and determining what resources are transported in these trips. We assume the set of feasible loadings, sets of resources that fit together on a connector, is given as input. This problem can be seen as a pick-up and delivery vehicle routing problem in which a route containing trips to the shore and back is assigned to each connector. Alternatively, it can be seen as a machine scheduling problem in which each machine represents a connector and jobs correspond to the resource-transporting trips.

Although the problem has been addressed by some researchers (Christafore Jr., 2017; Danielson, 2018; Strickland, 2018; Villena, 2019), the existing formulations do not allow to determine the optimal solution while integrating all constraints considered in this paper, for example, related to the order in which groups of resources are to be delivered. Examples of other problems that are of a similar structure are evacuations and the installation of wind farms at sea. The problem can also be linked to a pick-up and delivery problem.

In practice, these operations are planned manually by iteratively adding trips to the shore for the connectors. However, since the previously-mentioned research does not cover all the constraints in this problem, it is not known what the quality of these manually constructed plannings are. Therefore, we are interested in the quality of these solutions and in what cases significant improvements can be made.

We are the first to model a general and integrated model for the ship-to-shore problem that includes constraints regarding the simultaneous delivery of sets of resources and the number of available (un)loading spots. We provide two results related to the computational complexity of this problem and develop an exact branch-and-price algorithm and a greedy heuristic that will serve as a reflection of the manually-constructed schedules. We use 98 feasible instances from the Royal Netherlands Navy to analyse the performance of the branch-and-price and greedy algorithms. The results show that the greedy heuristic performs well in very constrained cases, but is outperformed by the exact method in other cases.

The paper is organised as follows. The problem and its constraints are described in Section 2. Section 3 gives an overview of the related literature. Section 4 presents a proof of $\mathcal{NP}$-hardness and a description of the greedy heuristic and the branch and price algorithm and Section 5 the results. We end with a conclusion in Section 6.

2 Problem Description

In the ship-to-shore problem, resources have to be transported from landing platform docks, the sea base(s) (SB), to the shore, the landing area(s) (LA). To transport these resources, connectors, smaller ships and helicopters, are used. A landing platform dock contains different locations at which connectors can be loaded. Helicopters can be loaded at a landing spot on the deck of the ship while the surface connectors can be loaded at either a dock or a davit. After a connector is loaded, the resources are transported to the shore, where they are unloaded from the connector and the connector can return for its next trip. An example of a landing platform dock is given in Figure 1. This ship has a deck with landing spots on which helicopters can land, docks for surface connectors that can be accessed through the opening at the rear and davits to load surface connectors on the side.

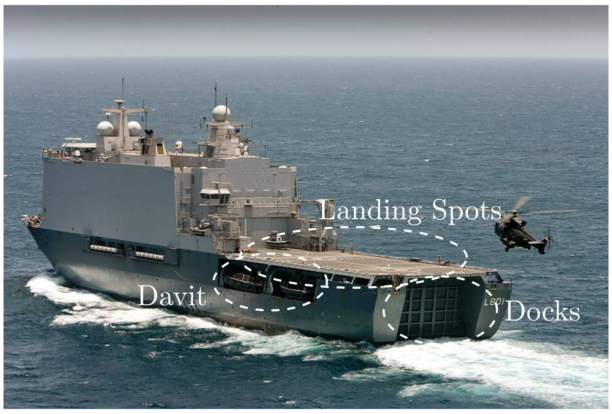


Figure 1: Zr. Ms. Johan de Witt (Maritime Warfare Centre, 2019)

Figure 2: Landing Craft Utility (Maritime Warfare Centre, 2019)

Figure 3: Fast Raiding, Interception & Special forces Craft (Maritime Warfare Centre, 2019)

There are several types of connectors, two examples of which are given in Figures 2 and 3. The different types of connectors have different dimensions, fuel capacities, fuel consumption rates, speeds and weight capacities. Furthermore, the locations that can be visited by each connector type can differ. Namely, surface connectors can only reach the shore, whereas helicopters can also reach destinations further land inwards. Also, large connectors, such as a Landing Craft Utility (Figure 2), can only be loaded at a dock, whereas smaller connectors, such as a Fast Raiding, Interception & Special forces Craft (Figure 3), can also be loaded at a davit. The speed of a connector depends on its design speed, whether it is loaded or not and on the state of the sea, *i.e.*, the wind and the waves.

During the operation, the main goal is to transport the resources while minimising the makespan (the total duration of the operation). However, the problem can include various constraints.

Firstly, there is limited (un)loading capacity, as there is a limited number of landing spots, docks and davits at a sea base and a limited number of spots to unload on land. Thus, only a limited number of connectors can be (un)loaded at the same time.

Secondly, connectors have a fuel capacity. Hence, they may have to be refuelled at a sea base during the operation to ensure they can complete their trips in the operation.

Thirdly, resources have to be delivered with a given priority to the landing area. Resources with a lower priority can only be delivered when all items with a higher priority have already been delivered. Our convention throughout the paper shall be that the lower the number, the higher the priority, *i.e.*, priority 1 has the highest priority. Additionally, it is not possible for resources with different priority numbers to be transported on the same connector simultaneously. In addition to priority levels, some resources have to be delivered to a landing area at the same time or immediately after each other, for example supplies, vehicles to transport the supplies, plus the required personnel to carry out the delivery. When resources belong together, we say that they belong to the same resource set. The set of consecutive periods in which the resources from a resource set are delivered is called a delivery wave for this resource set. There is no ordering imposed between the different waves with the same priority and these delivery waves can (partially) overlap.

Finally, the space on the connectors and their weight capacity is limited. Hence, a connector can only be loaded with a set of resources that fit on the connector in terms of its weight capacity and dimensions. We call such a set of resources that feasible fit together on a connector a loading.

3 Literature Review

We first focus on the literature related to the ship-to-shore problem. Thereafter, we link it to some other practical problems with similar characteristics and we describe how the ship-to-shore problem can be interpreted as a special case of the Pick-up and Delivery Vehicle Routing Problem.

3.1 Literature on the Ship-to-Shore Problem

The ship-to-shore problem has been addressed in some research. Villena (2019) solved the ship-to-shore problem using an Integer Linear Programming model to minimise the makespan. She considers the priority levels of the connectors, however, she does not consider (un)loading capacities and the fact that some resources are complementary and should belong to the same wave. She imposes fuel capacities by imposing a maximum distance that the connectors can travel and thus, does not allow for refuelling.

Christafore Jr. (2017), Danielson (2018) and Strickland (2018) generate schedules for the ship-to-shore problem using a three-phase approach. In the first phase, a quickest flow problem is considered. In this problem, the aim is to maximise the total demand satisfied given a fixed number of time periods. If the actual demand is higher then the demand, the number of time periods is increased. When the number of time periods is large enough for all demand to be satisfied, the corresponding number of connector trips per connector type can be determined. In this phase, capacity constraints are disregarded and hence no direct schedule can be constructed from these trips. In the second phase, trips are assigned to connectors in an assignment problem using a heuristic. Finally, in the last phase, a schedule is constructed from the output of the assignment problem such that the makespan is minimised and (un)loading capacities are satisfied. Christafore Jr. (2017) and Strickland (2018) only consider the transport of fuel, while Danielson (2018) extends this framework to multiple commodities.

In the three-phase approach from Christafore Jr. (2017), Danielson (2018) and Strickland (2018), (un)loading capacities are disregarded when the set of trips to be executed are determined in the first phase. Therefore, this set of trips could be suboptimal compared to the set of trips that result from an integrated approach. Furthermore, wave constraints are hard to impose as the set of trips selected in the first phase could result in an infeasible solution in the second and third phase.

3.2 Literature on Related Problems

There exist problems that have a similar structure as the ship-to-shore problem. First, two specific examples are given. Second, its relation to the Pick-up and Delivery Vehicle Routing Problem is explained.

An example of a problem with similar characteristics as the ship-to-shore problem is an evacuation where people are located at one or multiple locations and have to be brought to one or multiple shelter locations. Some locations might have to be evacuated first because they are more in danger and hence have a higher priority. The question is then what trips the vehicles, *e.g.*, buses, have to make to evacuate the area as quickly as possible. As with the ship-to-shore problem, the construction of an evacuation plan has to be done fast. Kulshrestha et al. (2014) aim to minimise the evacuation time by determining which bus should be assigned to which pick-up location, while incorporating uncertainty in the demand. They do not consider priorities, resource sets, fuel capacities, and (un)loading capacity at the locations and assume that buses only travel to a fixed pick-up point. Zhao et al. (2020) use a heuristic to determine the allocation of buses to trips with the aim of minimising both the in-bus travel time and the waiting time of the evacuees. They incorporate time-windows for the locations, *i.e.*, the time intervals within which the location should be visited to pick up the evacuees, which can be seen as a strict type of priority constraints. They also do not consider resource sets, fuel capacities and (un)loading capacities.

Another example of a problem with similar characteristics is the installation of a wind farm. Components of the wind mills and the underlying wind farm infrastructure have to be transported from the shore to the sea using vessels. There are priorities because certain components are required at the start, while others are only required later on. Resource sets can be interpreted as components that have to be used together to complete the next step in the installation process. Vessels are rented and hence to minimise the costs, the installation of the wind farm has to be completed as fast as possible. The time to construct a schedule for this problem is however less critical compared to the ship-to-shore problem and in evacuations, since this is a project that can be planned in advance. Ursavas (2017) uses a Benders decomposition approach to determine the time at which a particular vessel should start a certain building process and what loading is selected in each tour. Weather predictions are included as the weather has a big influence on the time that certain steps in the building process require.

More generally, the ship-to-shore problem can be linked to a Pick-up and Delivery Vehicle Routing Problem (PDVRP). In this problem, for each item that should be transported, it should first be collected at a pick-up location and then delivered to its delivery location. The decisions in the PDVRP are however simpler compared to the ship-to-shore problem. Namely, in the PDVRP, the location that is visited determines the resources that should be transported, while in the ship-to-shore problem, the loading that should be transported has to be determined too. Therefore, unless the general PDVRP is extended to a split delivery problem, each location is visited exactly once as opposed to the ship-to-shore problem in which trips are made back and forth between a limited set of locations (Nowak, 2005). In general, this problem does not contain priorities, resource sets, refuelling constraints, and (un)loading capacities at the locations.

Time windows, which are a common extension of the basic PDVRP (Ticha et al., 2017), can be seen as a type of priority ordering between the resources. However, the difference is that there is no strict ordering, but only a partial ordering when using time windows. Furthermore, the time windows impose the constraint that the pick-up and delivery occur during a certain time window, which is not necessarily the case in the ship-to-shore problem.

Resource sets could be interpreted as requiring all items with the same pick-up and delivery pair to be picked up and delivered at the same time, while allowing for split deliveries. Alternatively, a resource set could be defined as a set of items with the same pick-up location, but different delivery locations, or vice versa. Synchronised customer visits in a vehicle routing problem with time windows can be solved using, *e.g.*, a branch-and-price algorithm and an adaptive large neighbourhood search (Bredstrom & Rönnqvist, 2007; Drexl, 2012; Liu et al., 2019).

Fuel is sometimes considered in transportation problems with the aim of minimising these costs (Xiao et al., 2012) or the emissions (Behnke et al., 2021). However, we are interested in incorporating the need to refuel during the operation. Due to the emergence of vehicles using alternative fuel-powered vehicles, the green vehicle routing problem emerged. In this problem, vehicles have a maximum driving range, which is incorporated in the construction of the routes (Erdoğan & Miller-Hooks, 2012; Moghdani et al., 2021). Alternatively, charging of electric vehicles is possible at recharging stations (Desaulniers et al., 2016; Hiermann et al., 2016; Schneider et al., 2014), which is similar to the ship-to-shore problem, with the slight practical difference that refuelling is done at the sea base while the connector is being loaded for its next trip.

The (un)loading capacities are an essential aspect of the ship-to-shore problem, as disregarding them can lead to infeasible schedules in which more connectors are scheduled to be (un)loaded than there is space for. A limited number of (un)loading spots is realistic for routing problems, as loading capacity might be limited at a depot. However, these are usually disregarded in the basic vehicle routing problem. In practice, depots have a limited number of loadings bays at which vehicles can be loaded, resulting in the vehicle routing problem with docking constraints (Rieck & Zimmermann, 2010).

4 Methodology

In this section, we present a greedy heuristic and a branch-and-price algorithm that can be used to solve the ship-to-shore problem. Before the algorithms are introduced in Section 4.3 and 4.4, the computational complexity of the problem is discussed in Section 4.1 and we describe how the problem is defined using a time-space network in Section 4.2.

4.1 Problem Definition and Computational Complexity

In the ship-to-shore problem, the aim is to transport all resources from the sea base(s) to the landing area(s) using (a) connector(s) while minimising the makespan, *i.e.*, the duration of the operation. Here, we take into account constraints regarding the number of (un)loading spots, fuel levels, connector weight and space capacities, priority levels of the resources, and resource waves. We identified two special cases of the ship-to-shore problem for which it can be shown that they are $\mathcal{NP}$-hard. First, the notation of the ship-to-shore problem is introduced. Second, the computational complexity of the ship-to-shore problem is discussed by describing each of the special cases and providing a proof of $\mathcal{NP}$-hardness.

We thus have a set of sea bases $\mathcal{P}$, landing areas $\mathcal{D}$ and connectors $\mathcal{C}$. Let $\mathcal{M}$ be the set of resource types, where each resource type $m \in \mathcal{M}$ refers to a set of identical resources with the same priority level $\pi_m \in \{1, \dots, \Pi\}$, origin $p_m \in \mathcal{P}$, destination $d_m \in \mathcal{D}$, and resource set, if assigned to any. The demand for each resource type $m \in \mathcal{M}$ is n_m. For each connector $c \in \mathcal{C}$, $\mathcal{L}^c$ denotes the set of feasible loadings. A loading $l \in \mathcal{L}^c$ is a vector of length $|\mathcal{M}|$ representing the number of resources of each type $m \in \mathcal{M}$ that can be transported together, *i.e.,* they can feasible fit together on connector c, belong to the same priority level, and have the same origin and destination. Travel times between the locations are denoted by t_{ijc} for $i, j \in \mathcal{P} \cup \mathcal{D}$ and $c \in \mathcal{C}$. Here for a connector $c \in \mathcal{C}$, t_{ijc} for $i \in \mathcal{P}$ and $j \in \mathcal{D}$ is determined using the speed of connector c while it is loaded, and t_{ijc} for $i \in \mathcal{D}$ and $j \in \mathcal{P}$ is determined using the speed of connector c while it is empty. The loading and unloading time for a connector $c \in \mathcal{C}$ at location $i \in \mathcal{P}$ and $j \in \mathcal{D}$, respectively, is denoted by t'_{ic} For each connector $c \in \mathcal{C}$ we denote the fuel capacity, fuel consumption rate, and refuelling rate by Q_c, h_c, and g_c, respectively, for each connector $c \in \mathcal{C}$.

The aim is to transport all resources while minimising the makespan. Here we need to adhere to all constraints. This implies that at each point in time at most one connector can be located at each $i \in \mathcal{P} \cup \mathcal{D}$. The fuel level of a connector should be non-negative at each point in time. All priority $\pi \in \{1, \dots, \Pi\}$ resource types should be delivered at their destination, before the unloading of priority $\pi + 1$ resource types can start. Furthermore, we have a collection of resource sets $s \in \mathcal{S}$, where $\mathcal{S}$ is a partition of either $\mathcal{M}$ or a subset of $\mathcal{M}$. The delivery of resource types within a resource set should take place at the same time or directly after each other. If a resource type is not assigned to any resource set, there is no requirement to link the delivery of these resources to the delivery of other resources. If a resource type occurs as a singleton resource set, this means that the delivery of all the demand of that resource type must occur subsequently. Finally, in the decision version of the problem, the question is whether all resources can be transported in at most T periods while adhering to all previously mentioned constraints.

First, we consider the special case in Theorem 1. In this case we have one sea base, one landing area and one connector. We set all (un)loading times equal to 0 and all travel times equal to 1. All resource types have the same priority level and do not belong to a resource set. The fuel consumption rate equals 0, *i.e.*, fuel capacity is disregarded. Note that minimising the makespan is in this case equivalent to minimising the number of trips.

Theorem 1. *The decision version of the ship-to-shore problem with* $|\mathcal{P}| = 1$, $|\mathcal{D}| = 1$, $|\mathcal{C}| = 1$, $t_{ijc} = 1$ *for all* $i, j \in \mathcal{P} \cup \mathcal{D}$ *and* $c \in \mathcal{C}$, $t'_{ic} = 0$ *for all* $i, j \in \mathcal{P} \cup \mathcal{D}$ *and* $c \in \mathcal{C}$, $\Pi = 1$, $\mathcal{S} = \emptyset$ *and* $h_c = 0$ *for all* $c \in \mathcal{C}$ *is strongly* $\mathcal{NP}$*-complete.*

Proof. This special case is in $\mathcal{NP}$ as it can be checked in polynomial time whether demand is satisfied and the makespan is at most T.

We will use a reduction from Set Cover, which is known to be strongly $\mathcal{NP}-$complete (Karp, 1972), to prove that this special case is $\mathcal{NP}$-complete. In Set Cover, we are given a set of elements $E = \{e_1, \dots, e_n\}$ and some subsets of those elements $S_1, \dots, S_m$, where each $S_j \subseteq E$. We say that an element is covered if at least one subset containing the element is chosen. The question is whether we can cover all elements by choosing at most K subsets.

Given an instance of Set Cover, create a resource type for each element, and set its demand equal to 1. Furthermore, create a loading for every subset, where the loading contains the resource types corresponding to the elements from the subset. Finally, set $T = 2K - 1$.

If the instance of Set Cover is a yes-instance, we can use the loadings corresponding to the chosen subsets. Therefore, the created instance of the ship-to-shore instance is also a yes-instance. Similarly, if the ship-to-shore instance is a yes-instance, it naturally follows that the instance of Set Cover is also a yes-instance. ■

Second, consider the special case in Theorem 2. In this case we have one sea base. The resource types that have to be transported only differ in the destination to which they have to be transported, *e.g.*, only fuel has to be transported from the sea base to different locations. Connectors can have various capacities of transporting this resource, *e.g.*, one connector can transport 1000 gallons of fuel at a time, while another connector can transport 2000 gallons. There is one priority level, *i.e.*, the delivery of the resource at one location is not prioritised over other locations, and there are no resource sets, *i.e.*, it is not required to deliver the demand at one location during consecutive periods of time. The fuel consumption rate for the connectors equals 0, *i.e.*, fuel capacity is disregarded. This problem can thus be interpreted as the delivery of fuel to petrol stations from a central depot.

Theorem 2. *The decision version of the ship-to-shore problem with* $|\mathcal{P}| = 1$, $|\mathcal{M}| = |\mathcal{D}|$ *where each resource type only differs in the destination,* $t_{ijc} = 1$ *for all* $i, j \in \mathcal{P} \cup \mathcal{D}$ *and* $c \in \mathcal{C}$, $t'_{ic} = 0$ *for all* $i, j \in \mathcal{P} \cup \mathcal{D}$ *and* $c \in \mathcal{C}$, $\Pi = 1$, $\mathcal{S} = \emptyset$ *and* $h_c = 0$ *for all* $c \in \mathcal{C}$ *is strongly* $\mathcal{NP}$*-complete.*

Proof. This special case is in $\mathcal{NP}$ as it can be checked in polynomial time whether demand is satisfied and the makespan is at most T.

We will use a reduction from 3-Partition, which is known to be strongly $\mathcal{NP}-$complete (Garey & Johnson, 1979), to prove that this special case is $\mathcal{NP}$-complete. In 3-Partition, we are given $3m$ numbers, $a_1, \ldots, a_{3m}$. Each number a_i satisfies $B/4 < a_i < B/2$, where $B = \frac{1}{m} \sum_{i=1}^{3m} a_i$. The question is whether we can partition the numbers in subsets of size 3 with equal sum. More formally, do there exist sets $S_1, \ldots, S_m$ with $|S_j| = 3$ for all j, $S_j \cap S_{j'} = \emptyset$ for all j, j', and $\sum_{i \in S_j} a_i = B$ for all j?

Given an instance of 3-Partition, create $3m$ connectors, where connector i can ship a_i units of the resource per trip. Furthermore, create m landing areas, each with a demand of B. Finally, set $T = 1$.

Suppose that the instance of 3-Partition is a yes-instance. If we send the connectors corresponding to the numbers in S_j to landing area j, all demand is satisfied within one time unit. Therefore, the ship-to-shore instance is also a yes-instance. Conversely, suppose that the ship-to-shore instance is a yes-instance. This means that all demand is satisfied within one time unit. Therefore, the connectors visiting landing area j have a total capacity of B. Furthermore, because $B/4 < a_i < B/2$, we know that each landing area is visited by exactly 3 connectors. Hence, the instance of 3-Partition is also a yes-instance. ■

A similar reduction from Partition (Karp, 1972) shows that the problem is (weakly) $\mathcal{NP}$-hard for two landing areas.

Corollary 1 follows from Theorem 1 and Theorem 2, as the ship-to-shore problem is a generalisation of the special cases in Theorem 1 and Theorem 2. Thus, the ship-to-shore problem is $\mathcal{NP}$-hard.

Corollary 1. *The ship-to-shore problem is strongly $\mathcal{NP}$-hard.*

4.2 The Time-Space Network

Due to (un)loading capacities, there is a maximum number of connectors that can be (un)loaded at the same location simultaneously. Furthermore, as we are considering a heterogeneous fleet of connectors that can reach different loading spots, the type of connectors (un)loading at a location has to be taken into account. Moreover, it is possible that the same connector visits the same location multiple times. Thus, we have to model the movement of connectors over time and allow them to visit the same location multiple times. This can be modelled using a time-space network (Chardaire et al., 2005).

In the time-space network, a node is added for each dock, davit, landing spot, and landing area (this can be a beach or a landing zone further land inward) in each time period. Let $\mathcal{P}'$ be the set of nodes representing the loading locations at the SBs and $\mathcal{D}'$ the set of nodes representing the (un)loading locations at the LAs. Arcs can then be added when a feasible transition though time and space is possible, which we denote by the set $\mathcal{A}$. As connectors execute trips between a SB and a LA, it is not necessary to add arcs between different beaches or between different docks, davits, and landing spots at the SBs.

It is possible that a connector does not proceed to the next location as soon as possible due to the priority, wave, and capacity constraints. The connector can either wait for a spot to be free, without occupying a (un)loading location, or move below maximum speed. This would imply that if, for example, Δ time periods are required to travel from a SB to a LA, arcs have to be added from time period t to $t + \Delta + 1, t + \Delta + 2, t + \Delta + 3, \ldots, T$, with T the total number of time periods in the network. To avoid adding all possible transitions between the different locations, we make use of waiting nodes. A waiting node is added for each connector at each SB and LA. These can be used when the connector does not immediately proceed to the next location, but either waits for an (un)loading spot to become free or moves below maximum speed. With the use of these waiting nodes, we only have to add the shortest arc between SB and LA locations. Hence, the number of arcs in the time-space network is linear in the number of locations, connectors, and time periods.

Connectors can have different speeds depending on the weight of the load. Therefore two different speeds are used for each connector type, one for when the connector is empty and one for when the connector is loaded. Hence, since we are transporting resources from the landing platform docks to the shore, the speed of the loaded connector is used for arcs between SB locations and LA locations and the speed of the empty connector is used for arcs between LA locations and SB locations. As the speeds are different for the different connector types, the arcs that can be used by different connectors can be different.

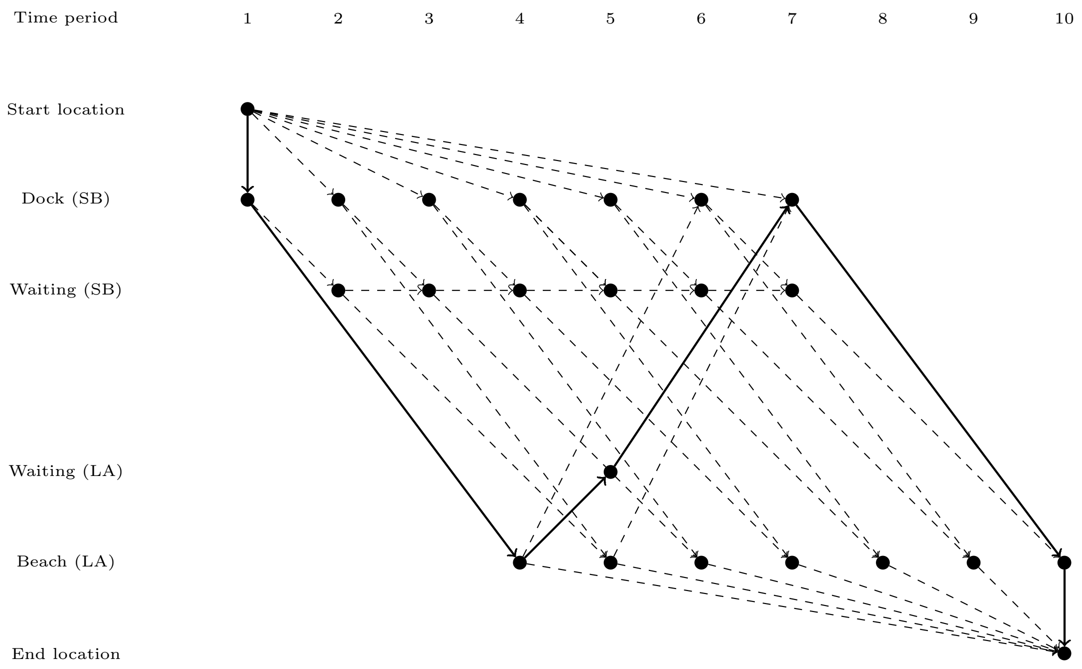


Figure 4: Example of a time-space network for one connector with one SB containing one dock and one LA containing one beach. Here all possible arcs are given (the bold and dashed lines combined) for a connector that takes two time periods to travel from the SB to the LA and one time period to travel from the LA to the SB. The bold lines are an example of a path through the network.

Example 1. *An example of a time-space network for one connector with one SB containing one dock and one LA containing one beach is given in Figure 4. In this example, the connector needs two time periods to travel from the SB to the LA and one time period to travel from the LA to the SB. The dashed and thick lines represent all arcs and the thick lines correspond to one feasible path through the network. We see that the connector is loaded with resources at the SB in time period 1, travels to the beach during time periods 2 and 3 to arrive for unloading in* *time period 4. After visiting the beach, the connector does not immediately proceed to the SB, but waits for one time period before proceeding. After loading at the SB in time period 7, the connector ends its last trip in time period 10.*

As time is discretised in this network, there is some loss in exactness unless the time period length is set to the greatest common divisor of all (un)loading times and travel times (Boland et al., 2019). However, when the time period length decreases, the number of required time periods increases and so does the size of the network. Hence, there is a trade-off between the running time and exactness. The distances between the SBs and LAs are quite large. Thus, the travel times are relatively long compared to the (un)loading time. Therefore, the length of a time period is chosen such that (un)loading can occur in one time period (Amrouss et al., 2017).

To construct the time-space network for an instance, an upper bound on the number of required time periods is required. We will use the output from the greedy heuristic described in Section 4.3 as this upper bound.

4.3 The Greedy Heuristic

The greedy heuristic described below aims to mimic the current scheduling procedure and will also serve as an upper bound of the makespan for the purpose of constructing the time-space network. Routes for the connectors are extended by iteratively adding new trips, *i.e.*, a visit to a dock, davit or landing spot at an SB to pick-up a specific set of resources, followed by a visit to a beach or landing zone at a LA to deliver these resources.

In the greedy heuristic, the following steps are executed:

STEP 0: Let $\pi = 1$ be the current priority level and let $T = 0$ be the last delivery period.

STEP 1: For every connector determine the first possible delivery period t for a loading with priority π resources that is not tabu. Here both the capacity constraints at the different locations and the priority constraints are taken into account.

STEP 2: For each loading with priority level π, determine the total area of resources that have not been delivered yet, let this area be a.

STEP 3: Select the connector-loading pair with minimum $\frac{t-T}{a}$, *i.e.*, the minimum ratio of the change in the makespan divided by the area of resources that are transported. Here, ties are broken arbitrarily. Add this trip to the current schedule and set $T = t$.

STEP 4: When there is a resource set of which only part is delivered yet, the completion of this delivery wave is prioritised. This is done by repeating versions of Steps 1-3 until the waves are completed. In Step 1, besides the capacity and priority constraints, the wave constraints now also have to be considered, *i.e.*, only consecutive time periods to the current (incomplete) wave are considered. In Step 2, only the area of the resources that belong to the incomplete resource set are considered, as completion of the wave is prioritised. If no feasible pair exists in Step 3, the last trip that was added to the schedule is removed and marked as tabu.

STEP 5: If all priority π resources are delivered, $\pi = \pi + 1$.

STEP 6: If all resources are delivered, return T, else go to Step 1.

4.4 The Branch-and-Price Algorithm

Given the time-space network as described in Section 4.2, we can define a route as a path through the time-space network in which for each trip from a SB to an LA, a loading is assigned. The ship-to-shore problem is then equivalent to assigning exactly one route to each connector such that all resources are transported and all constraints regarding (un)loading capacities, wave constraints, and priority levels are met. The complexity of this problem is dependent on the number of possible routes that can be constructed, which can be exponential (even if there are only two possible loadings).

Due to the huge number of possibilities, adding all possible routes to the model results in a high running time. Therefore, to add only a subset of all possible routes that might be beneficial for the solution, column generation can be used. This implies that we do not consider all feasible routes, but generate routes that will improve the solution, *i.e.*, that have negative reduced costs, until no such routes exist anymore. We solve a problem containing part of all the feasible routes, called the restricted master problem (RMP) and use the output of the LP-relaxation of this problem to construct routes in the Pricing Problem (PP) until no more routes with negative reduced costs can be found. To obtain an integer solution, this is embedded in a branch-and-bound framework, called a branch-and-price algorithm (Barnhart et al., 1998). In the algorithm, the solution space is restricted by continuously imposing integrality on fractional variables.

First, the RMP and PP used in the column generation are given in Sections 4.4.1 and 4.4.2. Thereafter, the branching strategies will be explained in Section 4.4.3. We conclude with some ideas to strengthen the formulation in Section 4.4.4.

4.4.1 The Restricted Master Problem

The aim of the RMP is to assign a route to each connector such that the makespan is minimised and all constraints are met. First, some notation is introduced in Table 1.

Table 1: List of sets, variables and parameters for the Restricted Master Problem.

Set	Explanation
$\mathcal{C}$	Set of connectors
$\mathcal{D}'$	Set of landing area nodes
$\mathcal{M}$	Set of resource types
$\mathcal{N}$	Set of nodes
$\mathcal{R}$	Set of routes
$\mathcal{S}$	Set of resource sets
Variables	**Explanation**
x_{rc}	A binary variable equal to 1 if route $r \in \mathcal{R}$ is assigned to connector $c \in \mathcal{C}$, 0 otherwise
$y_{\pi t}$	A binary variable equal to 1 if priority $\pi \in \{1, \dots, \Pi\}$ resources are delivered in time period $t \in \{1, \dots, T\}$
T_{π}^{start}	Time period at which the first priority $\pi \in \{1, \dots, \Pi\}$ resource is unloaded
T_{π}^{end}	Time period at which the last priority $\pi \in \{1, \dots, \Pi\}$ resource is unloaded
v_{st}	A binary variable equal to 1 if resources from set $s \in \mathcal{S}$ are delivered in time period $t \in \{1, \dots, T\}$
w_{st}^{start}	A binary variable equal to 1 if a wave of unloading resources from set $s \in \mathcal{S}$ starts in time period $t \in \{1, \dots, T\}$
w_{st}^{end}	A binary variable equal to 1 if a wave of unloading resources from set $s \in \mathcal{S}$ ends in time period $t \in \{1, \dots, T\}$
t_{span}	The makespan, *i.e.*, the duration of the operation
Parameters	**Explanation**
d_r	The last delivery period of route $r \in \mathcal{R}$
n_m	The number of resources of type $m \in \mathcal{M}$ that have to be transported
n_{rm}	The number of resources of type $m \in \mathcal{M}$ that are transported in route $r \in \mathcal{R}$
T	The number of time periods
Π	The number of priority levels

When parameters are given as input to a set, this implies that it concerns the subset of the routes with those characteristics. For example, $\mathcal{R}(c, i)$ for $c \in \mathcal{C}$ and $i \in \mathcal{N}$ implies the set of routes for connector c that visit node i and $\mathcal{R}(c, \{\pi, t\})$ for $c \in \mathcal{C}$, $\pi \in \{1, \dots, \Pi\}$ and $t \in \{1, \dots, T\}$ implies the set of routes for connector c that deliver priority π resources in time period t.

Now, the integer linear programming version of the RMP can be defined as follows:

$$\min t_{span} \tag{1}$$

$$\text{s.t.} \sum_{c \in \mathcal{C}} \sum_{r \in \mathcal{R}(c,m)} n_{rm} x_{rc} \geq n_m \qquad m \in \mathcal{M} \tag{2}$$

$$\sum_{r \in \mathcal{R}(c)} x_{rc} = 1 \qquad c \in \mathcal{C} \tag{3}$$

$$\sum_{c \in \mathcal{C}} \sum_{r \in \mathcal{R}(c,i)} x_{rc} \leq 1 \qquad i \in \mathcal{N} \tag{4}$$

$$\sum_{r \in \mathcal{R}(c,\{\pi,t\})} x_{rc} \leq y_{\pi t} \qquad c \in \mathcal{C}, t \in \{1, \dots, T\}, \pi \in \{1, \dots, \Pi\} \tag{5}$$

$$\sum_{\pi=1}^{\Pi} y_{\pi t} \leq 1 \qquad t \in \{1, \dots, T\} \tag{6}$$

$$T_{\pi}^{start} \leq t y_{\pi t} + T(1 - y_{\pi t}) \qquad t \in \{1, \dots, T\}, \pi \in \{1, \dots, \Pi\} \tag{7}$$

$$T^{end}_{\pi} \geq t y_{\pi t} \qquad t \in \{1,\dots,T\}, \pi \in \{1,\dots,\Pi\} \quad (8)$$

$$T^{start}_{\pi+1} \geq T^{end}_{\pi} \qquad \pi \in \{1,\dots,\Pi-1\} \quad (9)$$

$$\sum_{c \in \mathcal{C}} \sum_{r \in \mathcal{R}(c,\{s,t\})} x_{rc} \leq |\mathcal{D}'| v_{st} \qquad s \in \mathcal{S}, t \in \{1,\dots,T\} \quad (10)$$

$$v_{st} \leq \sum_{c \in \mathcal{C}} \sum_{r \in \mathcal{R}(c,\{s,t\})} x_{rc} \qquad s \in \mathcal{S}, t \in \{1,\dots,T\} \quad (11)$$

$$v_{st} - v_{s,t-1} = w^{start}_{st} - w^{end}_{s,t-1} \qquad s \in \mathcal{S}, t \in \{2,\dots,T\} \quad (12)$$

$$w^{start}_{st} + w^{end}_{s,t-1} \leq 1 \qquad s \in \mathcal{S}, t \in \{2,\dots,T\} \quad (13)$$

$$\sum_{t=1}^{T} w^{start}_{st} \leq 1 \qquad s \in \mathcal{S} \quad (14)$$

$$t_{span} \geq \sum_{r \in \mathcal{R}(c)} d_r x_{rc} \qquad c \in \mathcal{C} \quad (15)$$

$$x_{rc} \in \mathbb{B} \qquad c \in \mathcal{C}, r \in \mathcal{R}(c) \quad (16)$$

$$y_{\pi t} \in \mathbb{B} \qquad t \in \{1,\dots,T\}, \pi \in \{1,\dots,\Pi\} \quad (17)$$

$$T^{start}_{\pi}, T^{end}_{\pi} \in \mathbb{N}^{+} \qquad \pi \in \{1,\dots,\Pi\} \quad (18)$$

$$v_{st} \in \mathbb{B} \qquad s \in \mathcal{S}, t \in \{1,\dots,T\} \quad (19)$$

$$w^{start}_{st}, w^{end}_{st} \in \mathbb{B} \qquad s \in \mathcal{S}, t \in \{1,\dots,T\} \quad (20)$$

$$t_{span} \in \mathbb{N}^{+}. \quad (21)$$

The objective (1) is to minimise the makespan, *i.e.*, the duration of the operation. This is set by constraints (15). Constraints (2) ensure that all resources are transported from a sea base to a landing area. Constraints (3) assign a route to each connector. To ensure the (un)loading capacities are respected, constraints (4) are imposed. Constraints (5) force the decision variable $y_{\pi t}$ equal to 1 if a route is selected that transports priority $\pi \in \{1,\dots,\Pi\}$ resources in time period $t \in \{1,\dots,T\}$. Constraints (6) ensure that it is not possible to unload resource types from different priorities in the same time period.

Constraints (7) and (8) set the start and end time of unloading resources from a certain priority while constraints (9) ensure the ordering in the priorities. The term $T(1-y_{\pi t})$ ensures that no bound is imposed on T^{start}_{π} when $y_{\pi t}$ equals 0, *i.e.*, when no priority π resource types are delivered in time period t.

All resources in a resource set $s \in \mathcal{S}$ have to be delivered in the same time period or in consecutive time periods, *i.e.*, in the same wave. Constraints (10), (11), (12) and (13) ensure that w^{start}_{st} and w^{end}_{st} denote whether a delivery wave of resource set s starts or ends in time period $t = 1,\dots,T$, respectively. Constraint (14) ensure that there is only one wave for each resource set.

Example 2. *To illustrate, Figures 5 and 6 present the nodes of a beach in a time-space network. The arcs in these figures correspond to the connectors trips to this beach that deliver resources from resource set $s \in \mathcal{S}$. In Figure 5, v_{s2}, v_{s4}, w^{start}_{s2}, w^{end}_{s2}, and w^{start}_{s4} are forced to one and all other variables v_{st} and w^{end}_{st} are forced to zero. This violates constraint* (14) *and hence is infeasible. In Figure 6, v_{s2}, w^{start}_{s2} and w^{end}_{s3} are forced to be equal to 1 and all other v_{st} and*

w_{st}^{end} *are forced to be equal to 0. This satisfies constraint* (14) *and thus forms a feasible wave for resource set s.*

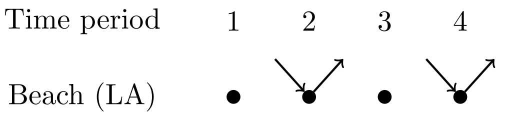


Figure 5: Wave example 1 (infeasible)

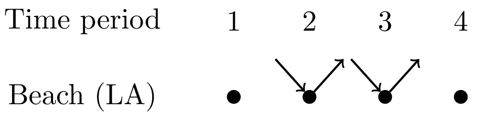


Figure 6: Wave example 2 (feasible)

To ensure that a feasible solution can be found for any subset of routes, variables $p_m \in \mathbb{R}^+$ are introduced for each resource type $m \in \mathcal{M}$ and added to the left hand side of constraints (2) and variables $p_c \in \mathbb{R}^+$ for each connector $c \in \mathcal{C}$ and added to the left hand side of constraints (3). Assigning positive values to these variables is penalised in the objective such that all resources are transported by the routes in an optimal solution, if possible. To ensure that the penalty is high enough to avoid using them if possible, their costs are set to the upper bound on the number of time periods in the time-space network.

4.4.2 The Pricing Problem

After solving the LP-relaxation of the RMP, the dual variables can be used to find new routes that will improve the solution, if any remain. Thus, the objective of the PP is to find routes with minimum reduced costs. From the RMP defined in Section 4.4.1, the reduced costs for a route r by connector $c \in \mathcal{C}$ can be found as:

$$\begin{aligned} RC(x_{rc}) = - \sum_{m \in \mathcal{M}(r)} n_{rm} \lambda_m^{(2)} + \lambda_c^{(3)} + \sum_{i \in \mathcal{N}(r) \setminus \{\tau, \tau'\}} \lambda_i^{(4)} \\ + \sum_{\pi=1}^{\Pi} \sum_{t \in \mathcal{T}(\pi, r)} \lambda_{ct\pi}^{(5)} + \sum_{s \in \mathcal{S}} \sum_{t \in \mathcal{T}(s,r)} \left(\lambda_{s,t}^{(10)} + \lambda_{s,t}^{(11)} \right) + d_r \lambda_c^{(15)} \end{aligned} \tag{22}$$

where $\lambda_m^{(2)}$, $\lambda_c^{(3)}$, $\lambda_i^{(4)}$, $\lambda_{ct\pi}^{(5)}$, $\lambda_{st}^{(10)}$ and $\lambda_c^{(15)}$ are the dual values of the LP-relaxation of the RMP and $\mathcal{T}(i)$ corresponds to the set of time periods in which resource types with characteristic i are delivered. Here it holds that $\lambda_m^{(2)}$, $\lambda_i^{(4)}$, $\lambda_{ct\pi}^{(5)}$, $\lambda_{st}^{(10)}$, $\lambda_c^{(15)} \in \mathbb{R}^+$, while $\lambda_c^{(3)} \in \mathbb{R}$ as these correspond to equality constraints.

Additionally to the notation introduced in Table 1, the following notation is used in the PP.

Table 3: List of sets, variables and parameters for the Pricing Problem.

Set	Explanation
$\mathcal{A}$	Set of arcs
$\mathcal{D}'$	Set of landing area nodes
$\mathcal{D}'_w$	Set of waiting nodes for the landing areas
$\mathcal{L}$	Set of loadings
$\mathcal{P}'$	Set of sea base nodes
$\mathcal{P}'_w$	Set of waiting nodes for the sea bases
$\mathcal{T}$	Set of time periods
Variables	**Explanation**
a_{ij}	A binary variable equal to 1 if arc $(i,j) \in \mathcal{A}$ is used, 0 otherwise
b_m	The number of resources of type $m \in \mathcal{M}$ transported
d	The last delivery period
f_i	A binary variable equal to 1 if the connector refuels at node $i \in \mathcal{P}'$, 0 otherwise
u_i	The fuel level upon reaching node $i \in \mathcal{N}$
z_{li}	A binary variable equal to 1 if loading $l \in \mathcal{L}$ is delivered to $i \in \mathcal{D}'$, 0 otherwise
Parameters	**Explanation**
g_c	The refuelling rate per time period for connector $c \in \mathcal{C}$
h_c	The fuel consumption rate per time period for connector $c \in \mathcal{C}$
k_{li}	A binary parameter equal to 1 if loading $l \in \mathcal{L}$ is available at/heading to location $i \in \mathcal{N}$, 0 otherwise
$p_{l\pi}$	A binary parameter equal to 1 if loading $l \in \mathcal{L}$ has priority level $\pi = \{1, \dots, \Pi\}$, 0 otherwise
Q_c	The fuel capacity of connector $c \in \mathcal{C}$
τ_c	The starting node of connector $c \in \mathcal{C}$
τ'_c	The ending node of connector $c \in \mathcal{C}$

We can then define the PP as follows:

$$\begin{aligned}
\min \; & -\sum_{m\in\mathcal{M}} \lambda_m^{(2)} b_m + \lambda^{(3)} + \sum_{i\in\mathcal{N}(c)} \sum_{j\in\mathcal{N}(c):(i,j)\in\mathcal{A}(c)} \lambda_i^{(4)} a_{ij} \\
& + \sum_{t=1}^{T}\sum_{\pi=1}^{\Pi} \sum_{l\in\mathcal{L}(c,\pi)} \sum_{i\in\mathcal{D}'(c,t)} \lambda_{ct\pi}^{(5)} z_{li} \\
& + \sum_{s\in\mathcal{S}} \sum_{l\in\mathcal{L}(c,s)} \sum_{t=1}^{T} \sum_{i\in\mathcal{D}'(c,t)} \left(\lambda_{st}^{(10)} + \lambda_{st}^{(11)}\right) z_{li} + \lambda_c^{(15)} d && (23)
\end{aligned}$$

$$\begin{aligned}
\text{s.t.} \quad & \sum_{j\in\mathcal{N}(c):(\tau,j)\in\mathcal{A}(c)} a_{\tau j} = 1 && (24) \\
& \sum_{i\in\mathcal{N}(c):(i,\tau')\in\mathcal{A}(c)} a_{i\tau'} = 1 && (25) \\
& \sum_{i\in\mathcal{N}(c):(i,j)\in\mathcal{A}(c)} a_{ij} - \sum_{i\in\mathcal{N}(c):(j,i)\in\mathcal{A}(c)} a_{ji} = 0 & j \in \mathcal{N}(c) \setminus \{\tau, \tau'\} & \quad (26) \\
& f_i \le \sum_{j\in\mathcal{N}(c):(i,j)\in\mathcal{A}(c)} a_{ij} & i \in \mathcal{P}'(c) & \quad (27) \\
& u_\tau = Q_c && (28) \\
& u_j \le Q_c - h_c(t_j - t_i) a_{ij} & (i,j) \in \mathcal{A}(c) & \quad (29)
\end{aligned}$$

$$u_j \leq u_i + g_c f_i - h_c(t_j - t_i)a_{ij} + Q_c(1 - a_{ij}) \qquad (i,j) \in \mathcal{A}(c) \quad (30)$$

$$\sum_{l \in \mathcal{L}(c)} z_{lj} = \sum_{i \in \mathcal{P}'(c) \cup \mathcal{P}'_w(c) : (i,j) \in \mathcal{A}(c)} a_{ij} \qquad j \in \mathcal{D}'(c) \quad (31)$$

$$2z_{lj} \leq k_{lj} + \sum_{i \in \mathcal{P}'(c) \cup \mathcal{P}'_w(c) : (i,j) \in \mathcal{A}(c)} k_{li} a_{ij} \qquad l \in \mathcal{L}(c), j \in \mathcal{D}'(c) \quad (32)$$

$$b_m \leq n_m \qquad m \in \mathcal{M} \quad (33)$$

$$b_m \leq \sum_{l \in \mathcal{L}(c,m)} \sum_{i \in \mathcal{D}'(c)} n_{ml} z_{li} \qquad m \in \mathcal{M} \quad (34)$$

$$\sum_{l \in \mathcal{L}(c,\pi)} \sum_{j \in \mathcal{D}'(c,t)} z_{lj} \leq y_{\pi t} \qquad t \in \{1, \ldots, T\}, \pi \in \{1, \ldots, \Pi\} \quad (35)$$

$$\sum_{\pi=1}^{\Pi} y_{\pi t} \leq 1 \qquad t \in \{1, \ldots, T\} \quad (36)$$

$$T_{\pi}^{start} \leq t y_{\pi t} + T(1 - y_{\pi t}) \qquad t \in \{1, \ldots, T\}, \pi \in \{1, \ldots, \Pi\} \quad (37)$$

$$T_{\pi}^{end} \geq t y_{\pi t} \qquad t \in \{1, \ldots, T\}, \pi \in \{1, \ldots, \Pi\} \quad (38)$$

$$T_{\pi+1}^{start} \geq T_{\pi}^{end} \qquad \pi \in \{1, \ldots, \Pi\} \quad (39)$$

$$\sum_{i \in \mathcal{D}'(c)} \sum_{l \in \mathcal{L}(c,s)} z_{li} \leq 1 \qquad s \in \mathcal{S} \quad (40)$$

$$\sum_{i \in \mathcal{D}'(c,t)} \sum_{l \in \mathcal{L}(c)} t z_{li} \leq d \qquad t \in \{1, \ldots, T\} \quad (41)$$

$$a_{ij} \in \mathbb{B} \qquad (i,j) \in \mathcal{A}(c) \quad (42)$$

$$b_m \in \mathbb{N}^+ \qquad m \in \mathcal{M} \quad (43)$$

$$d \in \mathbb{N}^+ \quad (44)$$

$$T_{\pi}^{start}, T_{\pi}^{end} \in \mathbb{N}^+ \qquad \pi \in \{1, \ldots, \Pi\} \quad (45)$$

$$f_i \in \mathbb{B} \qquad i \in \mathcal{P}'(c) \quad (46)$$

$$u_i \in \mathbb{R}^+ \qquad i \in \mathcal{N}(c) \quad (47)$$

$$w_{st}^{start}, w_{st}^{end} \in \mathbb{B} \qquad s \in \mathcal{S}, t \in \{1, \ldots, T\} \quad (48)$$

$$y_{\pi t} \in \mathbb{B} \qquad t \in \{1, \ldots, T\}, \pi \in \{1, \ldots, \Pi\} \quad (49)$$

$$z_{li} \in \mathbb{B}. \qquad l \in \mathcal{L}(c), i \in \mathcal{D}'(c) \quad (50)$$

The aim is to minimise the reduced costs (22) as represented by the objective (23).

Constraints (24), (25) and (26) ensure that a connector departs from its starting location, terminates at its ending location and ensure flow conservation throughout the network. The connector can only be refuelled at a pick-up location if this location is visited (constraints (27)). It is assumed that all connectors start with a full tank (constraint (28)). Constraints (29) and (30) update the fuel level throughout the network. They ensure that the fuel level is set to $\min\{u_i + g_c, Q_c\}$ after refuelling at $i \in \mathcal{P}'(c)$ and that no constraint is imposed on the fuel level if the arc is not used, but decreased if an arc is used.

Constraints (31) and (32) ensure that the connector can only be assigned one loading for each trip from a SB to a LA and that a loading can only be assigned when the loading is available at the SB and the connector is heading to the LA corresponding to the destination of the loading.

Constraints (34) set b_m to the number of resources of type $m \in \mathcal{M}$ that are transported in the selected loading. This variable is upper bounded by the number of resources of type m that have to be transported in constraints (33) to avoid deducting the reduced costs of the resource in the objective too much.

To satisfy the priority order, constraints (35)-(39) are imposed and to ensure that there is at most one delivery for each resource set, constraints (40) are imposed. If within a route, multiple deliveries with resource types from the same resource set occur, this route cannot be used as this will violate the wave constraints. Imposing this constraint will avoid generating routes that cannot be used in a feasible RMP solution. The last delivery period of the route is determined in constraints (41).

4.4.3 The Branching Strategies

To complete the branch-and-price algorithm, branching strategies have to be defined that will exclude the fractional solution we obtain at a branching node, but do not exclude any feasible integer solutions in that branch.

In the RMP, the integer decision variables are x_{rc} for $r \in \mathcal{R}, c \in \mathcal{C}$. A natural choice would therefore be to branch on x_{rc}, *i.e.*, on a route for a connector c. Branching on a route implies imposing a route in one child node and forbidding the route in the other child node. Imposing a route is easy as this fixes the route for this connector c and hence only the routes of the remaining connectors have to be optimised given this assignment. Forbidding a route, however, implies that for that connector c we need to find a route that differs in at least one aspect (arc or delivery) from the forbidden route r. This implies a disjunction constraint in the PP, making it difficult to solve.

Therefore, branching is applied on the arcs a_{ij} and the deliveries z_{li} used. We need to branch on both variables as solely branching on one of them is not guaranteed to result in an integer solution: branching on the arcs only can mean that a connector is assigned to a set of routes that use the same arcs, but executes different deliveries in different routes; branching on the deliveries only can mean that deliveries are split by the different connectors. We choose to first branch on the arcs until a solution with integer a_{ij}'s is found, whereafter we branch on the deliveries.

When branching on an arc (i, j), one of the children nodes is not allowed to use this arc whence all routes using (i, j) are removed from the corresponding RMP and (i, j) is removed from the PP's. In the other child node, the usage of arc (i, j) is imposed by which, all routes that use arcs (i, j') for $j' \neq j$ or (i', j) for $i' \neq i$ are removed, and these arcs are removed from the corresponding PP's.

Branching on a delivery implies branching on a combination of a loading $l \in \mathcal{L}$ and a delivery node $i \in \mathcal{D}'$. When a combination (l, i) is forbidden, all routes that deliver loading l to node i are removed from the RMP and the corresponding variable z_{li} is removed from the PP's. When a combination (l, i) is obligatory, all routes that deliver loading l' for $l' \neq l$ to node i are removed from the RMP and the corresponding variables $z_{l'i}$ are removed from the PP's.

For imposing an arc or a delivery, the above arc/loading removal operations are not sufficient yet as they affect only variables a_{ij}/z_{li}, and do not enforce the fact that in the RMP, only x_{rc}

decisions have to be allowed that respect the a_{ij}/z_{li} impositions.

For this reason, in the corresponding child nodes we need to add constraints that would not be needed, for example, in the classical vehicle routing problem. The role of these constraints is that among the available routes, at least one is selected that respects the given arc/delivery imposition:

$$\sum_{c\in\mathcal{C}} \sum_{r\in\mathcal{R}(c,\{i,j\})} x_{rc} \geq 1 \qquad (i,j)\in\mathcal{A}^o \tag{51}$$

$$\sum_{c\in\mathcal{C}} \sum_{r\in\mathcal{R}(c,\{l,i\})} x_{rc} \geq 1, \qquad (l,i)\in\mathcal{L}^o \tag{52}$$

where $\mathcal{A}^o$ is the set of arcs that should be used and $\mathcal{L}^o$ and set of obligatory loading and delivery pairs.

Adding constraints (51) and (52) to the restricted master problem changes the objective function of the pricing problem. Hence, the following terms are added to the objective in (23):

$$-\sum_{(i,j)\in\mathcal{A}^o} \lambda_{ij}^{(51)} a_{ij} - \sum_{(l,i)\in\mathcal{L}^o} \lambda_{li}^{(52)} z_{li}, \tag{53}$$

where $\lambda_{ij}^{(51)}$ and $\lambda_{li}^{(52)}$ are the dual variables corresponding to constraints (51) and (52), respectively.

There are multiple ways to select which arc or delivery to branch on. Here, the highest, *i.e.*, closest to 1, and lowest, *i.e.*, closest to 0, fractional arc or delivery rule will be used. To compute the value for each arc and delivery, we will use the formulas $\sum_{c\in\mathcal{C}}\sum_{r\in\mathcal{R}^c(i,j)} x_{rc}$ and $\sum_{c\in\mathcal{C}}\sum_{r\in\mathcal{R}^c(l,i)} x_{rc}$, respectively. Ties are broken arbitrarily.

As with arc or loading selection, different methods exist to determine which node in the branching tree to branch on next. Here, the breadth and depth first approaches are used. Again, ties are broken arbitrarily.

4.4.4 The Makespan Objective

The objective for the ship-to-shore problem is to minimise the makespan. The makespan is determined by constraints (15) in the RMP. However, in the branch-and-price algorithm, decision variables can be fractional and hence constraints (15) bound the makespan by the weighted average of the last delivery period of the assigned routes. Minimising this weighted average gives incentives to combine long routes in which many resources are transported with very short routes to reduce the bound on the makespan. This negatively affects the quality of the solution at a node in the branching tree. Therefore, with the aim of strengthening the formulation, bounds on both the length of routes and on the makespan will be used.

First, we can explicitly bound the lengths of routes. Very short routes cannot be prohibited because it is possible for a connector to have a short route in an optimal solution. However, routes longer than the current upper bound (makespan of the initial greedy solution or the best integer solution so far) can be prohibited as we know these will not be chosen in an optimal integer solution.

Second, we can find a lower bound of the optimal makespan by relaxing the constraints on the (un)loading capacities and resource waves. The problem then becomes a machine scheduling problem which can be easily solved during the pre-processing phase. Because this lower bound can be higher than the optimal solution at the root node of the branching tree, it can improve the estimates of the optimality gaps of the integer solutions from the branch-and-price algorithm. Secondly, we can add this bound explicitly in the RMP as a constraint on the makespan, which will accelerate the solution time per node.

5 Computational Experiments

In this section we present the results of a computational experiment on instances constructed using data from the Royal Netherlands Navy. In this experiment we aim to understand the performance of the greedy heuristic by comparing its results to those of the branch-and-price algorithm and analysing under which circumstances the greedy heuristic performs well. In order to do so we will first analyse the effect of using the bounds explained in Section 4.4.4 in the branch-and-price algorithm using small instances. Thereafter larger instances are used to make a comparison of the solutions obtained by the greedy heuristic and those of the branch-and-price algorithm. To determine under which circumstances the greedy heuristic finds good solutions, operational constraints will be relaxed and the corresponding results compared to the solutions obtained from the branch-and-price algorithm. Finally, as the running time is important, we analyse the performance of the different branching rules described in Section 4.4.3.

In Section 5.1 we describe the data and instance construction, and in Section 5.2 the corresponding results are presented.

5.1 Experimental Design

Instances are constructed using data from the Royal Netherlands Navy. Each instance consists of the demand of resources on the shore and the supply in terms of the available capacity on the amphibious ships and the number of connectors. Furthermore, an instance is characterised by the operational constraints that are considered. We can then define the following naming scheme for instances: d-s-c, where d denotes the demand set, s the supply set, and c the set of operational constraints that are considered. We first discuss each of these three aspects of an instance and then give an overview of the constructed instances.

The demand at the shore consists of a set of resources that have to be transported. For each resource, it is defined what the start location of the resource is, in other words, at which landing platform dock (SB) the resource is located, and what destination (LA) the resource has. Furthermore, it is denoted what the priority number of the resource is and to what subunit the resource belongs. If there are resources with the same subunit, then the resources belong to the same resource set and should be delivered within the same wave.

To test the performance of the greedy heuristic for different sized instances, the demand sets have different sizes, where it holds that iA $\subset$ iB $\subset$ iC, for i $\in \{1, 2, 3\}$. We thus define nine demand sets d, of which the descriptive statistics are shown in Table 5. This includes the number of SBs, LAs, priority levels, resource sets, and the distance between the SBs and LAs.

The table shows that demand sets 1A and 2A contain one priority level, *i.e.*, all resources have the same priority level.

Table 5: Overview of demand sets.

Demand Set	#SBs	#LAs	Distance SB - LA (nm)	# Priority Levels	# Resource Sets
1A	2	2	15	1	1
1B	2	2	15	2	2
1C	2	2	15	2	2
2A	2	3	15	1	1
2B	2	3	15	2	2
2C	2	3	15	2	2
3A	2	1	15	2	2
3B	2	1	15	2	3
3C	2	1	15	2	3

The supply consists of the number of available connectors and the (un)loading capacity at both the SBs and LAs. There are different types of (un)loading spots that can be used by different connector types. Namely, landing spots can only be used by helicopters, while docks and davits can only be used by surface connectors. Furthermore, not all surface connectors can access a davit. This can only be used by the smaller surface connectors. An overview of the supply sets s is given in Table 6.

Table 6: Overview of the supply sets.

Supply Set	# Connectors	# Connector Types	# Docks per SB	# Davits per SB	# Landing Spots per SB	# Landing Zones per LA
1	4	2	1	2	1	2
2	6	2	1	2	1	2
3	12	2	2	4	2	2
4	16	4	2	4	2	2
5	8	3	2	2	2	2

The operational constraints consist of, amongst others, constraints regarding the order of the delivery of the resources. As we are also interested in the circumstances under which the greedy heuristic performs well, we vary for these operational constraints as these have an impact on the feasible set of trips that are considered in each step of the greedy heuristic. The constraints regarding the order of the delivery of resources are the priority constraints and resource set constraints. If priority constraints are relaxed, this implies that all resources have the same priority level. If resource set constraints are relaxed, this implies that we do not impose the delivery of resources within the same subunit to be at the same time. When either or both of these constraints are relaxed, there are thus fewer constraints regarding the ordering which affects from which the greedy heuristic can choose in each step.

We define four options for c. Let N denote the case in which neither the priority nor the resource set constraints are considered. Let P denote the case in which the priority constraints are added to case N, and let W denote the case in which the resource set constraints are added to case N. The full and default model we consider contains both constraints, which cases are denoted by F.

Using the demand sets, supply sets, and the set of operational constraints, we can construct the instances. Large demand sets with low supply in terms of the number of available connectors can result in feasibility issues due to the resource set constraints. Namely, it should be possible to deliver all resources of the same resource set at the same time or shortly after each other. Since travel times between the SBs and LAs are large, it is not possible to assign a connector to multiple trips transporting resources of the same resource set, as this would lead to a violation of the resource set constraint. Therefore, for each resource set in a combination of a demand set and a supply set, we verify whether it is possible to assign loadings to the connectors such that all resources of this set are transported and each connector is used at most once. If this is not possible, the instance is not feasible when resource set constraints are imposed and hence we only consider the instances with options P and N. On the other hand, small demand sets in combination with a large supply set, are not realistic. Hence, not all combinations of demand and supply sets are considered.

Taking both the feasibility as well as the realism of combinations of demand sets and supply sets into account, this results in the instances as denoted in Table 6. Here ‘All’ implies that all four options of the operational constraints are considered. The instances used for only options P and N imply that these combinations of demand set and supply set are infeasible in terms of the resource set constraints, *e.g.*, for demand set 3C with supply set 5. Demand sets 1A and 2A have a single priority level, hence there is no difference between incorporating the priority constraints or not. Therefore, for these demand sets only options W and N are considered. This results in 98 instances in total.

Table 7: Overview of the instances where for each demand set and supply set combination that is used, the corresponding set of operational constraints is denoted. In total, this results in 98 instances.

Supply	Demand Set								
Set	1A	1B	1C	2A	2B	2C	3A	3B	3C
1	W, N	All		W, N	P, N		All	All	
2	W, N	All		W, N	All		All	All	
3			All			All			All
4		P, N	All		All	All		P, N	All
5	W, N	P, N	All	W, N	All	All	All	All	P, N

5.2 Results

In this section, the computational results are presented. The algorithms are implemented in Java using CPLEX 12.10. The experiments are executed on the Dutch national SurfSARA Lisa cluster consisting mostly of nodes with 16 core Intel Xeon 6130 processors and 96GB RAM. While the greedy heuristic runs in almost no time, we choose a cut-off point of one hour for the branch-and-price algorithm. First, the effect of using the upper and lower bound in the branch-and-price algorithm is analysed using the smallest instances (instances iA for i $\in$ {1,2,3}) in

Section 5.2.1. Second, we will compare the performance of the greedy heuristic and branch-and-price algorithm for the full instances in Section 5.2.2. In Section 5.2.3 we will analyse the effect of the operational constraints on the performance of the greedy heuristic. Finally, in Section 5.2.4, the performance of the different branching rules is analysed.

5.2.1 Analysis of the Usage of Bounds in the Branch-and-Price Algorithm

To analyse the performance of the branch-and-price algorithm with and without the bounds incorporated as explained in Section 4.4.4, we use the instances corresponding to the smallest demand sets. Namely, instances 1A-1-W, 1A-2-W, 1A-5-W, 2A-1-W, 2A-2-W, 2A-5-W, 3A-1-F, 3A-2-F and 3A-5-F, where for demand sets 1A and 2A it holds that F is equivalent to W as no priority levels exist. The instances are run with and without upper bound and lower bound and run for all four branching rules. We compare the results of each instance corresponding to the branching rule that yields the best solution. Here best solution implies the solution with the lowest makespan. If one or more branching rules are tied, we choose the solution corresponding to the branching rule that has the shortest running time. In case the branch-and-price algorithm does not terminate within an hour, we first order the branching rules based on the time it took to find the best integer solution, if applicable.

We analyse the effect of using the bounds by comparing the running time. Since we are using a cut-off point of one hour, solutions obtained when the branch-and-price algorithm is cut-off after one hour, are not (proven to be) optimal. Figure 7 shows the running times of the instances with and without the bounds on a logarithmic scale. A table containing the results, included potential gaps, can be found in Appendix A.

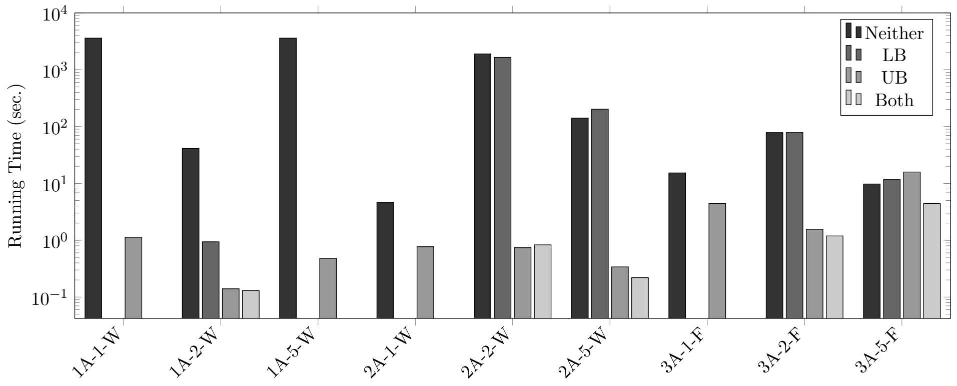


Figure 7: Running time of the branch-and-price algorithm with and without an upper bound (UB) and lower bound (LB). When the LB equals the UB, the branch-and-price is not run and hence the running time is 0.

We observe that adding the upper bound significantly decreases the running time for all instances except 3A-5-F, for which the running time is slightly worse. When the lower bound is used, the effect is dependent on the instance. For some instances, the lower bound equals the makespan of the greedy heuristic, hence the branch-and-price algorithm is not run. For the other instances, the running times are approximately the same, with the exception of instance 1A-2-W, where the running time is significantly lower when the lower bound is added. Using both bounds reduces the running times compared to using neither of the bounds or solely the lower bound.

Compared with solely using the upper bounds, running times are approximately the same, or slightly lower. Hence, in the remainder of the results, both bounds will be incorporated.

5.2.2 The Greedy Heuristic vs the Branch-and-Price Algorithm

To test the performance of the greedy heuristic, we will now compare the solution of the greedy heuristic with the solution of the branch-and-price algorithm. Here we consider the 19 instances with both operational constraints, namely instances with with all operational constraints (c = F), as these contain all constraints that are required in practical operations. The branch-and-price algorithm is run for all four branching rules of which the results corresponding to the branching rule that yields the best solution is reported. This implies the best solution determined by the lexographic order of the lowest makespan, shortest run time and shortest time until the best integer solution. The results for these instances are presented in Table 8, the complete results for all four branching rules are presented in Table 11 in Appendix A. The table denotes for each instance, the bounds, the number of available loadings, the size of the time-space network, and information on the solution and corresponding branching rule, if applicable. The purpose of the number of loadings and the time-space network is to illustrate the size of the instances. Namely, they determine, together with the number of time periods defined by the upper bound from the greedy solution, the number of SBs and LAs, the number of priority levels, and the number of resource sets, the number of variables and constraints in the problem.

Table 8: Results of the greedy heuristic and branch-and-price algorithm for instances with c = F with a cut-off point of one hour. Here the solutions correspond to the solution of the branching rule with the best result in terms of makespan, running time, and number of nodes evaluated.

	Branching				TSN Size				T (sec.) till	
Instance	Rule[1]	UB	LB	#Loadings	#Nodes	#Arcs	Makespan	T (sec.)	Int. Sol.	Gap (%)
3A-1-F[2]		41	41	18	132	4452	41	0		
3A-2-F	BH	27	25	18	82	1743	27	1.19		
3A-5-F	BL	26	11	33	146	4752	26	8.62		
1B-1-F	BH	42	41	38	308	12134	42	18.62		
1B-2-F	-[3]	27	25	38	192	4816	27	3600		7.41
2B-1-F[2]		42	42	16	374	16761	42	0		
2B-2-F	BL	41	25	16	366	16080	41	7.73		
2B-4-F	BH	8	6	62	164	2811	7	8.05	4.49	
2B-5-F	-[3]	32	25	53	428	22127	32	3600		28.00
3B-2-F	DH	59	25	61	210	11151	59	629.61		
3B-5-F	BL	26	11	33	146	4752	26	8.62		
1C-3-F	BH	41	10	10	462	24224	41	34.20		
1C-4-F	BH	14	9	19	264	5608	14	3.67		
1C-5-F	BL	41	19	21	484	26204	41	458.01		
2C-3-F	BL	36	25	26	448	23933	36	25.76		
2C-4-F	DH	11	9	60	182	3649	10	3600	14.03	11.11
2C-5-F	DL	26	19	51	344	14243	26	17.43		
3C-3-F	-[3]	54	26	154	288	18654	54	3600		107.69
3C-4-F	BL	26	11	232	346	14876	26	469.98		

[1]: B = breadth first, D = depth first, L = lowest fractional, H = highest fractional
[2]: UB = LB, hence the branch-and-price algorithm is not run
[3]: all branching rules have the same gap and running time

We observe that there are four instances for which the branch-and-price algorithm does not terminate within an hour. For three of these four instances no improvement is found compared to the solution of the greedy heuristic. In total there are 2 out of 19 instances in which an improvement is found and hence the solution of the greedy heuristic is known to be suboptimal. In 14 out of 19 instances the greedy solution is proven to be optimal.

When considering the performance of the branch-and-price algorithm, we see that the performance of the greedy heuristic is of importance. For example, consider instances 2B-5-F and 2C-5-F which have the same supply, but different demand, namely the demand of 2B is a subset of the demand of 2C. One would thus expect 2B-5-F to be easier to solve compared to 2C-5-F. However, 2B-5-F does not terminate within an hour, while 2C-5-F terminates within 18 seconds. The difference in the running time for the branch-and-price algorithm is caused by the quality of the upper bound obtained from the greedy heuristic which affects the size of the time-space network and hence the number of variables and the size of the feasible region of the problem. Namely, for instances 2C-5-F the greedy heuristic finds a solution with 26 time periods, while it finds a solution of 32 time periods for 2B-5-F. The greedy heuristic iteratively adds trips to a solution, thus it is possible for the heuristic to choose suboptimal trips that are the best option at that moment but have a large, negative, effect on the makespan in the end.

Since the resources in demand set 2B are a subset of the resources in demand set 2C, it should be possible to transport all resources in 2B-5-F in at most 26 time periods. In fact, when the upper bound on 2B-5-F is set to 26, the branch-and-price algorithm is able to show within a minute that 26 is the optimal makepan, implying that the greedy heuristic resulted in a solution that is about 23% above the optimal solution for this instance. However, the branch-and-price algorithm is not even able to improve on the makespan of 32 time periods, which is most likely caused by the ability to combine unnecessarily long routes with very short routes resulting in a very low lower bound, as explained in Section 4.4.4.

We also see that a toolbox with more connectors, although it results in a reduction in the size of the time-space network, does not imply a smaller running time. This is shown by for example instances 1B-1-F and 1B-2-F. This could be caused by the fact that now more routes should be selected for each connector type. Another explanation is that the gap between the bounds is larger in instance 1B-2-F as the lower bound does not consider the (un)loading capacity which has a larger effect on the quality of the lower bound if there are more connectors.

5.2.3 The Effect of Operational Constraints on the Greedy Heuristic

The results in the previous section showed that in 14 out of 19 instances the greedy heuristic found an optimal solution and in 17 out of 19 instances no improvement could be found within an hour. This is an interesting result as it shows that a simple approach that takes from a few milliseconds to a few seconds to solve, gives optimal solutions in many cases. In the greedy heuristic, trips are iteratively added to the planning until a feasible solution is found. If a problem is more constrained in terms of the priorities and resource sets, there are fewer trips that can be added in each iteration of the greedy heuristic and hence the chance of choosing suboptimal trip is lower.

To analyse the effect of the priority and resource set constraints on the performance of the greedy heuristic, we run each instance for the cases in which either or both of these constraints are relaxed. The results of these instances can be found in Appendix A and a summary of the results is given in Table 9. The number of instances for the different categories are different for two reasons. Since demand sets 1A and 2A have a single priority level, for those instances there is no case with priority constraints. Hence, there are six more instances for the cases with only wave constraints and neither constraints compared to the cases with both constraints and only priority constraints. Also, there are five combinations of demand set and supply set that are infeasible due to the resource set constraints.

Table 9 shows the percentage of instances for which the greedy heuristic gave the optimal solution. As not all instances could be solved within an hour, there are also instances for which the solution of the greedy heuristic is the best solution found, but for which it is not known whether this is optimal. For all instances for which improvements compared to the solution of the greedy heuristic were found, the average gap between the solution of the greedy heuristic and the best solution found, the running time to obtain this solution are given, and the percentage of instances for which this best and/or optimal solution was found within five minutes.

We can see from the results in the table that the greedy heuristic also performs well in many cases when there are no priority levels and resource sets. However, in the case where the branch-and-price algorithm can find improvements, the improvements can be very large, especially in case there are no resource set constraints (instances with c = P or c = N). Furthermore, these improvements can be found quite fast, and in quite some cases within five minutes. Hence, the usage of the branch-and-price algorithm can be beneficial to use in problems with few or no priority levels and resource sets.

Table 9: The effect of priority level constraints and resource set constraints on the performance of the greedy heuristic with a cut-off points of one hour for the branch-and-price algorithm. Here c denotes the set of operational constraints regarding the order of the resources that are considered, namely either the full set (F), only the priority constraints (P), only the resource set constraints (W), or neither of these constraints (N). Note that there are no instances for which an improvement was found for the instances with only resource set constraints (c = W).

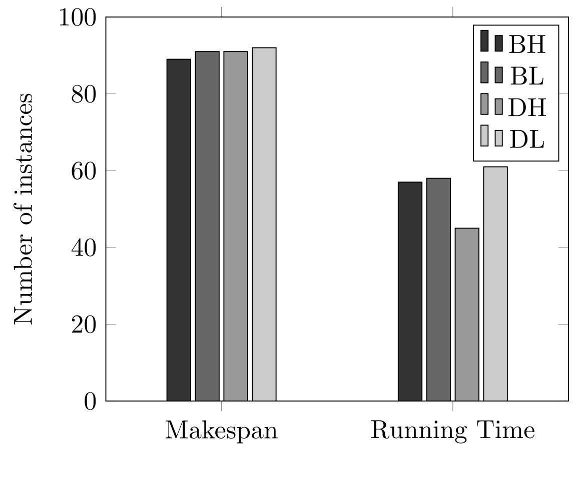
					Instances with Improvement		
c	# Instances	% Instances Greedy Optimal	% Instances Greedy Best	% Instances with Improvement	Avg. Gap (%) Greedy	Avg. T (sec.) till Best Sol.	% within 5 min.
F	19	63.16	89.47	10.53	12.14	9.26	100.00
P	24	12.50	62.50	37.50	52.44	783.63	66.67
W	25	64.00	100.00	0	-	-	-
N	30	13.33	56.67	43.33	36.99	323.03	84.62
Total	98	35.71	75.51	24.49	40.71	469.60	79.17

5.2.4 The Performance of the Branching Rules

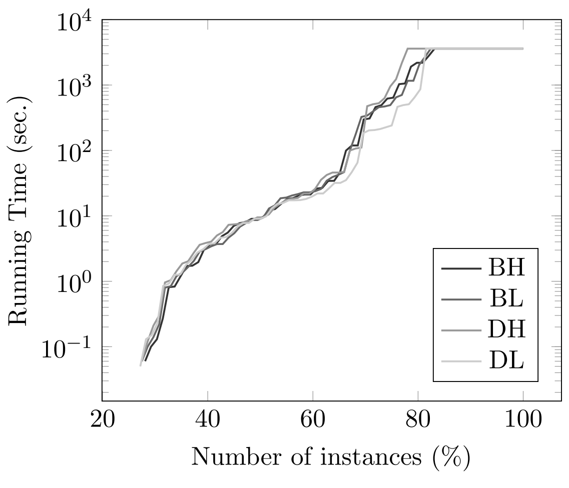
All instances are run for four different branching rules based on the search strategy through the branching tree and the arc or delivery selection procedure. Namely, we use a breadth first (B) or depth first (D) search and select the arc or delivery to branch on according to the highest (H) or lowest (L) fractional value. To compare the performance of the different branching rules, we first evaluate the quality of the solution found for each of the branching rules in terms of the makespan found and then the performance of the branching rule in terms of the running time required.

First, we consider the quality of the branching rules in terms of the makespan. For the branch-and-price algorithm a cut-off point of one hour is used. Hence, if for none of the branching rules the algorithm terminates within an hour, the optimal makespan of the solution is unknown. In those cases we compare the makespan for a branching rule with the best makespan found for that instance. For each branching rule we can then determine the number of instances for which the optimal/best makespan is found using that branching rule. In Figure 8a, the number of instances, out of the 98 instances, for which the different branching rules found the optimal/best makespan is given. We see that the different branching rules find the optimal/best makespan approximately equally often, ranging between 89 and 92.

Second, we consider the quality of the branching rules in terms of the running time. First, we determine how often each branching rule terminates the fastest. Here we only consider the instances for which the branching rule finds the best makespan. Namely, if none of the branching rules terminates within an hour, the running times are equal, however the makespan found using the different branching rules can be different. We then only count the branching rules for which the best makespan was found. Figure 8a shows the number of times a branching rule is the fastest in finding the optimal/best makespan. We see that branching rule DH, while having a high number of instances for which the optimal/best makespan is found, terminates fastest in significantly fewer instances compared to the other three branching rules. These branching rules approximately terminate fastest equally often, with a slightly higher number of instances for branching rule DL.



(a) Number of instances for which the different branching rules found the optimal/best makespan and have the lowest running time.



(b) Performance profile of the running times for the different branching rules, where only instances for which the optimal/best makespan is found are considered.

Figure 8: Performance of the four different branching rules, where a breadth first (B) or a depth first (D) tree search and highest fractional (H) or lowest fractional (L) arc or delivery branching decision are used.

Based on Figure 8a, it seems that branching rule DL performed the best. However, as the number of times the optimal/best makespan was found are approximately equal for all branching rules, this judgement is based on the number of times the branching rule is the fastest. A branching rule that is never fastest but often slightly slower can be be preferred over a branching rule that is most often the fastest, but in other cases significantly slower compared to other branching rules. Therefore, the running times of the different branching rules are ranked and plotted in Figure 8b in a performance profile. This performance profile can be used to analyse the distribution of the running times for the different branching rules.

We observe that until about 60% of the instances solved, the speed of the branching rules is very close and that for about 50% of the instances, the running time is at most ten seconds. After about 60% of the instances solved, DL manages to outperform the other branching rules as its running time distribution lies close to or strictly below the distribution of the other branching rules. Hence, branching rule DL is not only able to solve most instances the fastest, it also outperforms the other branching rules based on the distribution of its running times based on the instances we consider.

6 Conclusion

In this paper we consider the ship-to-shore problem, for which we provide a proof of $\mathcal{NP}$-hardness, even in very restricted special cases. We provide a general and integrated model for the ship-to-shore problem including relevant practical constraints such as the limit on the number of (un)loading spots and the simultaneous delivery of groups of resources. We construct a greedy heuristic that mimics the current scheduling procedure and determine the quality of these solutions by comparing them to the solutions obtained from a branch-and-price algorithm. In the branch-and-price algorithm, we incorporate the usage of an upper and lower bound to strengthen its formulation and reduce its running time significantly. Using data from the Royal Netherlands Navy, we find that the greedy heuristic performs well in highly constrained cases. However, when instances are less constrained, the value of the solution obtained by the greedy heuristic can have a large optimality gap compared to the the solution found by the branch-and-price algorithm.

As distances and therefore travel times are large compared to the (un)loading times in these instances, we chose to set the time period length such that (un)loading for all connectors can take place within one time period. Depending on the application, the loss in exactness can be more significant and a shorter time period length is preferred. This requires small changes to the model, namely, arcs should be added between a node for an (un)loading location in time period t and the node corresponding to the same location in time period $t+1$. Furthermore, when (un)loading at a location takes $n > 1$ time periods, constraints should be added that require a connector to remain at this location for n time periods if it is visited.

The results show that the greedy heuristic performs well in case the problem is very constrained. For less constrained problems, applying, *e.g.*, a local search to the solution of the greedy heuristic, or running the greedy heuristic multiple times with a randomisation parameter, can result in better outcomes. To test whether a heuristic works well in another application, the branch-and-price algorithm can be used as a benchmark.

By using discrete time periods, some slack occurs in the schedule. A potential benefit of this slack is that it can serve as a buffer in case a delay occurs. However, the moments at which this slack occurs are not chosen and therefore accounting for delays while constructing the schedule can result in a lower expected makespan. Hence, for future research, it would be interesting to incorporate the uncertainty about the travel and (un)loading times to construct schedules with a lower expected makespan.

Acknowledgements

We would like to thank Kerry M. Malone from TNO for the feedback she provided on previous versions of the paper.

References

Amrouss, A., El Hachemi, N., Gendreau, M., & Gendron, B. (2017). Real-time management of transportation disruptions in forestry. *Computers & Operations Research*, *83*, 95–105.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W., & Vance, P. H. (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, *46*(3), 316–329.

Behnke, M., Kirschstein, T., & Bierwirth, C. (2021). A column generation approach for an emission-oriented vehicle routing problem on a multigraph. *European Journal of Operational Research*, *288*(3), 794–809.

Boland, N., Hewitt, M., Marshall, L., & Savelsbergh, M. (2019). The price of discretizing time: a study in service network design. *EURO Journal on Transportation and Logistics*, *8*(2), 195–216.

Bredstrom, D., & Rönnqvist, M. (2007). A branch and price algorithm for the combined vehicle routing and scheduling problem with synchronization constraints. *NHH Dept. of Finance & Management Science Discussion Paper*(2007/7).

Chardaire, P., McKeown, G. P., Verity-Harrison, S., & Richardson, S. (2005). Solving a Time-Space Network Formulation for the Convoy Movement Problem. *Operations Research*, *53*(2), 219–230.

Christafore Jr., R. (2017). *Generating ship-to-shore bulk fuel delivery schedules for the Marine Expeditionary Unit* (Tech. Rep.). Naval Postgraduate School Monterey United States.

Danielson, M. E. (2018). *Scheduling amphibious connectors to deliver multiple commodities* (Tech. Rep.). Naval Postgraduate School Monterey United States.

Desaulniers, G., Errico, F., Irnich, S., & Schneider, M. (2016). Exact algorithms for electric vehicle-routing problems with time windows. *Operations Research*, *64*(6), 1388–1405.

Drexl, M. (2012). Synchronization in vehicle routing—a survey of vrps with multiple synchronization constraints. *Transportation Science*, *46*(3), 297–316.

Erdoğan, S., & Miller-Hooks, E. (2012). A green vehicle routing problem. *Transportation research part E: logistics and transportation review*, *48*(1), 100–114.

Garey, M. R., & Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman, San Fransisco.

Hiermann, G., Puchinger, J., Ropke, S., & Hartl, R. F. (2016). The electric fleet size and mix vehicle routing problem with time windows and recharging stations. *European Journal of Operational Research*, *252*(3), 995–1018.

Karp, R. M. (1972). Reducibility among combinatorial problems. In *Complexity of computer computations* (pp. 85–103). Springer.

Kulshrestha, A., Lou, Y., & Yin, Y. (2014). Pick-up locations and bus allocation for transit-based evacuation planning with demand uncertainty. *Journal of Advanced Transportation*, *48*(7), 721–733.

Liu, R., Tao, Y., & Xie, X. (2019). An adaptive large neighborhood search heuristic for the vehicle routing problem with time windows and synchronized visits. *Computers & Operations Research*, *101*, 250–262.

Maritime Warfare Centre. (2019). *Handboek Surface Assault.*

Moghdani, R., Salimifard, K., Demir, E., & Benyettou, A. (2021). The green vehicle routing problem: A systematic literature review. *Journal of Cleaner Production*, *279*, 123691.

Nowak, M. A. (2005). *The pickup and delivery problem with split loads.* Georgia Institute of Technology.

Rieck, J., & Zimmermann, J. (2010). A new mixed integer linear model for a rich vehicle routing problem with docking constraints. *Annals of Operations Research*, *181*(1), 337–358.

Schneider, M., Stenger, A., & Goeke, D. (2014). The electric vehicle-routing problem with time windows and recharging stations. *Transportation science*, *48*(4), 500–520.

Strickland, C. W. (2018). *Generating efficient and robust schedules to deliver bulk fuel via amphibious connectors* (Tech. Rep.). Naval Postgraduate School Monterey United States.

Ticha, H. B., Absi, N., Feillet, D., & Quilliot, A. (2017). Empirical analysis for the vrptw with a multigraph representation for the road network. *Computers & Operations Research*, *88*, 103–116.

Ursavas, E. (2017). A benders decomposition approach for solving the offshore wind farm installation planning at the north sea. *European Journal of Operational Research*, *258*(2), 703–714.

Villena, K. (2019). *Ship to shore transportprobleem tijdens amfibische operaties* (Bachelor thesis). Nederlandse Defensie Academie.

Xiao, Y., Zhao, Q., Kaku, I., & Xu, Y. (2012). Development of a fuel consumption optimization model for the capacitated vehicle routing problem. *Computers & Operations Research*, *39*(7), 1419–1431.

Zhao, X., Ji, K., Xu, P., Qian, W.-w., Ren, G., & Shan, X.-n. (2020). A round-trip bus evacuation model with scheduling and routing planning. *Transportation Research Part A: Policy and Practice*, *137*, 285–300.

Appendices

A Results

Table 10: Results of the branch-and-price algorithm with and without upper bound (UB) and lower bound (LB) for instance 1A-1-W, 1A-2-W, 1A-5-W, 2A-1-W, 2A-2-W, 2A-5-W, 3A-1-F, 3A-2-F and 3A-5-F with a cut-off point of one hour. Here T denotes the makespan of the greedy heuristic and UB and LB denote whether the bound is included and if so, the corresponding value of the bound.

Instance	Branching Rule[1]	T	UB	LB	#Loadings	TSN Size #Nodes	TSN Size #Arcs	Makespan	T (sec.)	T (sec.) till Int. Sol.	Gap (%)
1A-1-W	-[3]	25	-	-	5	164	3574	25	3600		69.95
1A-1-W[2]		25	-	25	5	164	3574	25	0		
1A-1-W	BL	25	25	-	5	164	3574	25	1.13		
1A-1-W[2]		25	25	25	5	164	3574	25	0		
1A-2-W	BL	11	-	-	5	64	632	11	41.25		
1A-2-W	BL	11	-	10	5	64	632	11	0.94		
1A-2-W	BH	11	11	-	5	64	632	11	0.14		
1A-2-W	BH	11	11	10	5	64	632	11	0.13		
1A-5-W	-[3]	9	-	-	9	84	832	9	3600		39.97
1A-5-W[2]		9	-	9	9	84	832	9	0		
1A-5-W	BL	9	9	-	9	84	832	9	0.48		
1A-5-W[2]		9	9	9	9	84	832	9	0		
2A-1-W	BH	25	-	-	5	198	4762	25	4.67		
2A-1-W[2]		25	-	25	5	198	4762	25	0		
2A-1-W	BL	25	25	-	5	198	4762	25	0.77		
2A-1-W[2]		25	25	25	5	198	4762	25	0		
2A-2-W	BH	25	-	-	5	206	5168	25	1898.12		
2A-2-W	BH	25	-	9	5	206	5168	25	1651.41		
2A-2-W	BL	25	25	-	5	206	5168	25	0.74		
2A-2-W	BH	25	25	9	5	206	5168	25	0.83		
2A-5-W	DL	10	-	-	9	120	1667	10	141.38		
2A-5-W	DL	10	-	9	9	120	1667	10	202.57		
2A-5-W	DL	10	10	-	9	120	1667	10	0.34		
2A-5-W	BL	10	10	9	9	120	1667	10	0.22		
3A-1-F	DL	41	-	-	18	132	4452	41	15.30		
3A-1-F[2]		41	-	41	18	132	4452	41	0		
3A-1-F	BH	41	41	-	18	132	4452	41	4.44		
3A-1-F[2]		41	41	41	18	132	4452	41	0		
3A-2-F	DL	27	-	-	18	82	1743	27	46.47		
3A-2-F	DL	27	-	25	18	82	1743	27	78.43		
3A-2-F	BL	27	27	-	18	82	1743	27	1.56		
3A-2-F	BH	27	27	25	18	82	1743	27	1.19		
3A-5-F	DH	26	-	-	33	146	4752	26	9.77		
3A-5-F	BL	26	-	11	33	146	4752	26	11.66		
3A-5-F	DH	26	26	-	33	146	4752	26	15.88		
3A-5-F	DH	26	26	11	33	146	4752	26	4.44		

[1]: B = breadth first, D = depth first, L = lowest fractional, H = highest fractional
[2]: UB = LB, hence the branch-and-price algorithm is not run
[3]: all branching rules have the same gap and running time

Table 11: Performance of the greedy heuristic and branch-and-price algorithm for instances with c = F with a cut-off point of one hour.

Instance	Branching Rule[1]	UB	LB	#Loadings	TSN Size #Nodes	TSN Size #Arcs	Makespan	T (sec.)	T (sec.) till Int. Sol.	Gap (%)
3A-1-F[2]		41	41	18	132	4452	41	0		
3A-2-F	BL	27	25	18	82	1743	27	1.20		
	BH	27	25	18	82	1743	27	1.19		
	DL	27	25	18	82	1743	27	1.27		
	DH	27	25	18	82	1743	27	1.43		
3A-5-F	BL	26	11	33	146	4752	26	8.62		
	BH	26	11	33	146	4752	26	9.00		
	DL	26	11	33	146	4752	26	9.03		
	DH	26	11	33	146	4752	26	9.49		
1B-1-F	BL	42	41	38	308	12134	42	19.05		
	BH	42	41	38	308	12134	42	18.62		
	DL	42	41	38	308	12134	42	19.65		
	DH	42	41	38	308	12134	42	18.82		
1B-2-F	BL	27	25	38	192	4816	27	3600		7.41
	BH	27	25	38	192	4816	27	3600		7.41
	DL	27	25	38	192	4816	27	3600		7.41
	DH	27	25	38	192	4816	27	3600		7.41
2B-1-F[2]		42	42	16	374	16761	42	0		
2B-2-F	BL	41	25	16	366	16080	41	7.73		
	BH	41	25	16	366	16080	41	7.74		
	DL	41	25	16	366	16080	41	7.78		
	DH	41	25	16	366	16080	41	8.24		
2B-4-F	BL	8	6	62	164	2811	7	23.79	17.7	
	BH	8	6	62	164	2811	7	8.05	4.49	
	DL	8	6	62	164	2811	7	18.11	6.76	
	DH	8	6	62	164	2811	7	8.59	4.87	
2B-5-F	BL	32	25	53	428	22127	32	3600		28.00
	BH	32	25	53	428	22127	32	3600		28.00
	DL	32	25	53	428	22127	32	3600		28.00
	DH	32	25	53	428	22127	32	3600		28.00
3B-2-F	BL	59	25	61	210	11151	59	648.55		
	BH	59	25	61	210	11151	59	640.38		
	DL	59	25	61	210	11151	59	649.00		
	DH	59	25	61	210	11151	59	629.61		
3B-5-F	BL	26	11	33	146	4752	26	8.62		
	BH	26	11	33	146	4752	26	9.00		
	DL	26	11	33	146	4752	26	9.03		
	DH	26	11	33	146	4752	26	9.49		
1C-3-F	BL	41	10	10	462	24224	41	34.51		
	BH	41	10	10	462	24224	41	34.20		
	DL	41	10	10	462	24224	41	35.20		
	DH	41	10	10	462	24224	41	35.53		
1C-4-F	BL	14	9	19	264	5608	14	3.70		
	BH	14	9	19	264	5608	14	3.67		
	DL	14	9	19	264	5608	14	3.69		
	DH	14	9	19	264	5608	14	3.84		
1C-5-F	BL	41	19	21	484	26204	41	458.01		
	BH	41	19	21	484	26204	41	460.21		
	DL	41	19	21	484	26204	41	464.53		
	DH	41	19	21	484	26204	41	475.04		
2C-3-F	BL	36	25	26	448	23933	36	25.76		
	BH	36	25	26	448	23933	36	26.13		
	DL	36	25	26	448	23933	36	26.30		

[1]: B = breadth first, D = depth first, L = lowest fractional, H = highest fractional

[2]: UB = LB, hence the branch-and-price algorithm is not run

Table 11: Performance of the greedy heuristic and branch-and-price algorithm for instances with c = F with a cut-off point of one hour.

	Branching				TSN Size				T (sec.) till	
Instance	Rule[1]	UB	LB	#Loadings	#Nodes	#Arcs	Makespan	T (sec.)	Int. Sol.	Gap (%)
	DH	36	25	26	448	23933	36	25.95		
2C-4-F	BL	11	9	60	182	3649	10	3600	17.73	11.11
	BH	11	9	60	182	3649	10	3600	52.75	11.11
	DL	11	9	60	182	3649	11	3600		22.22
	DH	11	9	60	182	3649	10	3600	14.03	11.11
2C-5-F	BL	26	19	51	344	14243	26	18.59		
	BH	26	19	51	344	14243	26	18.10		
	DL	26	19	51	344	14243	26	17.43		
	DH	26	19	51	344	14243	26	18.23		
3C-3-F	BL	54	26	154	288	18654	54	3600		107.69
	BH	54	26	154	288	18654	54	3600		107.69
	DL	54	26	154	288	18654	54	3600		107.69
	DH	54	26	154	288	18654	54	3600		107.69
3C-4-F	BL	26	11	232	346	14876	26	469.98		
	BH	26	11	232	346	14876	26	620.50		
	DL	26	11	232	346	14876	26	508.51		
	DH	26	11	232	346	14876	26	508.79		

[1]: B = breadth first, D = depth first, L = lowest fractional, H = highest fractional

[2]: UB = LB, hence the branch-and-price algorithm is not run

Table 12: Performance of the greedy heuristic and branch-and-price algorithm for instances with c = P with a cut-off point of one hour.

	Branching				TSN Size				T (sec.) till	
Instance	Rule[1]	UB	LB	#Loadings	#Nodes	#Arcs	Makespan	T (sec.)	Int. Sol.	Gap (%)
1B-1-P[2]		41	41	12	292	10934	41	0		
1B-2-P	BL	26	25	12	184	4436	26	3.42		
	BH	26	25	12	184	4436	26	3.71		
	DL	26	25	12	184	4436	26	3.81		
	DH	26	25	12	184	4436	26	4.04		
1B-4-P	BL	17	9	41	338	9280	17	3600		88.89
	BH	17	9	41	338	9280	17	3600		88.89
	DL	17	9	41	338	9280	17	3600		88.89
	DH	17	9	41	338	9280	17	3600		88.89
1B-5-P	BL	33	25	77	388	16860	33	3600		32.00
	BH	33	25	77	388	16860	33	3600		32.00
	DL	33	25	77	388	16860	33	3600		32.00
	DH	33	25	77	388	16860	33	3600		32.00
1C-3-P	BL	39	25	9	438	21808	37	3600	3229.42	48.00
	BH	39	25	9	438	21808	35	3600	3025.42	40.00
	DL	39	25	9	438	21808	25	205.14	205.14	
	DH	39	25	9	438	21808	37	3600	87.42	48.00
1C-4-P	BL	14	9	12	264	5608	11	3600	435.34	22.22
	BH	14	9	12	264	5608	10	3600	2594.38	11.11
	DL	14	9	12	264	5608	10	3600	254.70	11.11
	DH	14	9	12	264	5608	9	107.04	107.03	
1C-5-P	BL	39	25	16	460	23676	34	3600	3350.49	36.00
	BH	39	25	16	460	23676	37	3600	1653.88	48.00
	DL	39	25	16	460	23676	39	3600		56.00
	DH	39	25	16	460	23676	38	3600	90.94	52.00
2B-1-P[2]		42	42	11	368	16186	42	0		
2B-2-P	BL	26	25	11	216	5670	26	5.43		

[1]: B = breadth first, D = depth first, L = lowest fractional, H = highest fractional

[2]: UB = LB, hence the branch-and-price algorithm is not run

Table 12: Performance of the greedy heuristic and branch-and-price algorithm for instances with c = P with a cut-off point of one hour.

Instance	Branching Rule[1]	UB	LB	#Loadings	TSN Size #Nodes	TSN Size #Arcs	Makespan	T (sec.)	T (sec.) till Int. Sol.	Gap (%)
	BH	26	25	11	216	5670	26	5.55		
	DL	26	25	11	216	5670	26	5.47		
	DH	26	25	11	216	5670	26	5.64		
2B-4-P	BL	8	6	37	164	2811	6	2039.66	2039.55	
	BH	8	6	37	164	2811	6	308.22	308.22	
	DL	8	6	37	164	2811	7	3600	8.15	16.67
	DH	8	6	37	164	2811	8	3600		33.33
2B-5-P	BL	26	25	31	344	14243	25	394.44	394.43	
	BH	26	25	31	344	14243	25	1031.91	1031.90	
	DL	26	25	31	344	14243	26	3600		4.00
	DH	26	25	31	344	14243	25	42.20	42.20	
2C-3-P[2]		25	25	14	280	9304	25	0		
2C-4-P	BL	11	9	35	182	3649	9	325.49	325.48	
	BH	11	9	35	182	3649	9	297.76	297.75	
	DL	11	9	35	182	3649	10	3600	183.36	11.11
	DH	11	9	35	182	3649	9	2174.85	2174.85	
2C-5-P	BL	25	19	29	330	13097	19	2678.07	2678.05	
	BH	25	19	29	330	13097	25	3600		31.58
	DL	25	19	29	330	13097	25	3600		31.58
	DH	25	19	29	330	13097	25	3600		31.58
3A-1-P[2]		41	41	13	132	4452	41	0		
3A-2-P	BL	26	25	13	78	1581	26	176.13		
	BH	26	25	13	78	1581	26	2781.73		
	DL	26	25	13	78	1581	26	64.95		
	DH	26	25	13	78	1581	26	3600		4.00
3A-5-P	BL	25	11	22	140	4368	11	1161.93	1161.93	
	BH	25	11	22	140	4368	11	2187.96	2187.96	
	DL	25	11	22	140	4368	11	31.88	31.88	
	DH	25	11	22	140	4368	25	3600		127.27
3B-1-P[2]		74	74	116	264	17586	74	0		
3B-2-P[2]		57	57	58	196	9732	57	0		
3B-4-P	BL	19	11	96	248	7652	19	3600		72.73
	BH	19	11	96	248	7652	19	3600		72.73
	DL	19	11	96	248	7652	19	3600		72.73
	DH	19	11	96	248	7652	19	3600		72.73
3B-5-P	BL	25	11	22	140	4368	11	1161.93	1161.91	
	BH	25	11	22	140	4368	11	2187.96	2187.96	
	DL	25	11	22	140	4368	11	31.88	31.88	
	DH	25	11	22	140	4368	25	3600		127.27
3C-3-P	BL	41	26	146	210	9970	41	3600		57.69
	BH	41	26	146	210	9970	41	3600		57.69
	DL	41	26	146	210	9970	41	3600		57.69
	DH	41	26	146	210	9970	41	3600		57.69
3C-4-P	BL	26	11	212	346	14876	26	3600		136.36
	BH	26	11	212	346	14876	26	3600		136.36
	DL	26	11	212	346	14876	26	3600		136.36
	DH	26	11	212	346	14876	26	3600		136.36
3C-5-P	BL	41	39	205	236	12432	41	3600		5.13
	BH	41	39	205	236	12432	41	3600		5.13
	DL	41	39	205	236	12432	41	3600		5.13
	DH	41	39	205	236	12432	41	3600		5.13

[1]: B = breadth first, D = depth first, L = lowest fractional, H = highest fractional

[2]: UB = LB, hence the branch-and-price algorithm is not run

Table 13: Performance of the greedy heuristic and branch-and-price algorithm for instances with c = W with a cut-off point of one hour.

	Branching				TSN Size				T (sec.) till	
Instance	Rule[1]	UB	LB	#Loadings	#Nodes	#Arcs	Makespan	T (sec.)	Int. Sol.	Gap (%)
1A-1-W[2]		25	25	5	164	3574	25	0		
1A-2-W	BL	11	10	5	64	632	11	0.14		
	BH	11	10	5	64	632	11	0.13		
	DL	11	10	5	64	632	11	0.15		
	DH	11	10	5	64	632	11	0.21		
1A-5-W[2]		9	9	9	84	832	9	0		
1B-1-W[2]		41	41	136	292	10934	41	0		
1B-2-W	BL	26	25	136	184	4436	26	9.40		
	BH	26	25	136	184	4436	26	9.67		
	DL	26	25	136	184	4436	26	9.80		
	DH	26	25	136	184	4436	26	10.57		
1C-3-W	BL	25	9	28	270	8480	25	3.09		
	BH	25	9	28	270	8480	25	3.10		
	DL	25	9	28	270	8480	25	3.09		
	DH	25	9	28	270	8480	25	3.61		
1C-4-W[2]		9	9	30	108	1124	9	0		
1C-5-W	BL	25	18	46	292	9564	25	6.84		
	BH	25	18	46	292	9564	25	7.01		
	DL	25	18	46	292	9564	25	6.83		
	DH	25	18	46	292	9564	25	7.56		
2A-1-W[2]		25	25	5	198	4762	25	0		
2A-2-W	BL	25	9	5	206	5168	25	0.84		
	BH	25	9	5	206	5168	25	0.83		
	DL	25	9	5	206	5168	25	0.85		
	DH	25	9	5	206	5168	25	0.96		
2A-5-W	BL	10	9	9	120	1667	10	0.22		
	BH	10	9	9	120	1667	10	0.27		
	DL	10	9	9	120	1667	10	0.23		
	DH	10	9	9	120	1667	10	0.30		
2B-1-W	BL	41	25	29	364	15881	41	4.64		
	BH	41	25	29	364	15881	41	4.89		
	DL	41	25	29	364	15881	41	4.63		
	DH	41	25	29	364	15881	41	5.02		
2B-2-W	BL	40	25	29	356	15218	40	1.65		
	BH	40	25	29	356	15218	40	1.71		
	DL	40	25	29	356	15218	40	1.78		
	DH	40	25	29	356	15218	40	2.02		
2B-4-W	BL	7	6	68	142	2073	7	3600		16.67
	BH	7	6	68	142	2073	7	3600		16.67
	DL	7	6	68	142	2073	7	3600		16.67
	DH	7	6	68	142	2073	7	3600		16.67
2B-5-W	BL	31	18	62	414	20693	31	99.42		
	BH	31	18	62	414	20693	31	99.38		
	DL	31	18	62	414	20693	31	212.75		
	DH	31	18	62	414	20693	31	99.58		
2C-3-W	BL	11	9	32	98	1187	11	3.71		
	BH	11	9	32	98	1187	11	12.08		
	DL	11	9	32	98	1187	11	2.75		
	DH	11	9	32	98	1187	11	7.42		
2C-4-W	BL	10	9	66	164	2915	10	3600		11.11
	BH	10	9	66	164	2915	10	3600		11.11
	DL	10	9	66	164	2915	10	3600		11.11
	DH	10	9	66	164	2915	10	3600		11.11

[1]: B = breadth first, D = depth first, L = lowest fractional, H = highest fractional

[2]: UB = LB, hence the branch-and-price algorithm is not run

Table 13: Performance of the greedy heuristic and branch-and-price algorithm for instances with c = W with a cut-off point of one hour.

Instance	Branching Rule[1]	UB	LB	#Loadings	TSN Size #Nodes	TSN Size #Arcs	Makespan	T (sec.)	T (sec.) till Int. Sol.	Gap (%)
2C-5-W	BL	25	18	60	330	13097	25	14.44		
	BH	25	18	60	330	13097	25	15.20		
	DL	25	18	60	330	13097	25	15.26		
	BL	25	18	60	330	13097	25	15.43		
3A-1-W[2]		41	41	219	132	4452	41	0		
3A-2-W	BL	27	25	219	82	1743	27	12.96		
	BH	27	25	219	82	1743	27	12.71		
	DL	27	25	219	82	1743	27	13.00		
	DH	27	25	219	82	1743	27	13.47		
3A-5-W	BL	25	11	274	140	4368	25	22.87		
	BH	25	11	274	140	4368	25	21.24		
	DL	25	11	274	140	4368	25	21.99		
	DH	25	11	274	140	4368	25	21.88		
3B-2-W	BL	43	41	266	146	5423	43	46.59		
	BH	43	41	266	146	5423	43	46.43		
	DL	43	41	266	146	5423	43	46.55		
	DH	43	41	266	146	5423	43	46.90		
3B-5-W	BL	25	11	274	140	4368	25	22.87		
	BH	25	11	274	140	4368	25	21.24		
	DL	25	11	274	140	4368	25	21.99		
	DH	25	11	274	140	4368	25	21.88		
3C-3-W	BL	53	25	1021	282	17890	53	3600		112.00
	BH	53	25	1021	282	17890	53	3600		112.00
	DL	53	25	1021	282	17890	53	3600		112.00
	DH	53	25	1021	282	17890	53	3600		112.00
3C-4-W	BL	26	11	1204	346	14876	26	492.98		
	BH	26	11	1204	346	14876	26	492.01		
	DL	26	11	1204	346	14876	26	493.01		
	DH	26	11	1204	346	14876	26	526.67		

[1]: B = breadth first, D = depth first, L = lowest fractional, H = highest fractional

[2]: UB = LB, hence the branch-and-price algorithm is not run

Table 14: Performance of the greedy heuristic and branch-and-price algorithm for instances with c = N with a cut-off point of one hour.

Instance	Branching Rule[1]	UB	LB	#Loadings	TSN Size #Nodes	TSN Size #Arcs	Makespan	T (sec.)	T (sec.) till Int. Sol.	Gap (%)
1A-1-N[2]		25	25	3	164	3574	25	0		
1A-2-N	BL	11	10	3	64	632	11	0.10		
	BH	11	10	3	64	632	11	0.10		
	DL	11	10	3	64	632	11	0.13		
	DL	11	10	3	64	632	11	0.12		
1A-5-N[2]		9	9	5	84	832	9	0		
1B-1-N[2]		41	41	13	292	10934	41	0	0	
1B-2-N	BL	26	25	13	184	4436	26	3600		3.85
	BH	26	25	13	184	4436	26	3600		3.85
	DL	26	25	13	184	4436	26	3600		3.85
	DH	26	25	13	184	4436	26	3600		3.85
1B-4-N	BL	15	9	75	246	5688	15	3600		66.67
	BH	15	9	75	246	5688	14	3600		66.67
	DL	15	9	75	246	5688	9	241.00	241.00	
	DH	15	9	75	246	5688	9	110.79	110.79	

[1]: B = breadth first, D = depth first, L = lowest fractional, H = highest fractional

[2]: UB = LB, hence the branch-and-price algorithm is not run

Table 14: Performance of the greedy heuristic and branch-and-price algorithm for instances with c = N with a cut-off point of one hour.

Instance	Branching Rule[1]	UB	LB	#Loadings	TSN Size #Nodes	TSN Size #Arcs	Makespan	T (sec.)	T (sec.) till Int. Sol.	Gap (%)
1B-5-N	BL	33	25	77	388	16860	32	3600	1596.69	28.00
	BH	33	25	77	388	16860	33	3600		32.00
	DL	33	25	77	388	16860	26	202.51	143.62	
	DH	33	25	77	388	16860	33	3600		32.00
1C-3-N	BL	11	9	24	102	1376	11	2.05		
	BH	11	9	24	102	1376	11	1.96		
	DL	11	9	24	102	1376	11	2.14		
	DH	11	9	24	102	1376	11	2.74		
1C-4-N	BL	10	9	28	156	2192	9	0.80	0.80	
	BH	10	9	28	156	2192	9	0.81	0.81	
	DL	10	9	28	156	2192	9	0.92	0.92	
	DH	10	9	28	156	2192	9	1.04	1.04	
1C-5-N[2]		18	18	38	192	4144	18	0		
2A-1-N[2]		25	25	4	198	4762	25	0		
2A-2-N	BL	10	9	4	56	462	10	0.06		
	BH	10	9	4	56	462	10	0.06		
	DL	10	9	4	56	462	10	0.05		
	DH	10	9	4	56	462	10	0.06		
2A-5-N[2]		9	9	7	96	1124	9	0		
2B-1-N	BL	41	25	12	364	15881	26	39.24	29.92	
	BH	41	25	12	364	15881	26	26.78	21.22	
	DL	41	25	12	364	15881	26	15.76	12.29	
	DH	41	25	12	364	15881	26	17.56	14.28	
2B-2-N	BL	26	25	12	216	5670	25	1.35	1.35	
	BH	26	25	12	216	5670	25	1.73	1.73	
	DL	26	25	12	216	5670	25	1.25	1.25	
	DH	26	25	12	216	5670	25	1.87	1.87	
2B-4-N	BL	8	6	31	164	2811	6	2.85	2.85	
	BH	8	6	31	164	2811	6	34.32	34.32	
	DL	8	6	31	164	2811	6	4.74	4.74	
	DH	8	6	31	164	2811	6	7.29	7.29	
2B-5-N	BL	25	17	28	330	13097	18	704.58	323.89	
	BH	25	17	28	330	13097	18	1908.71	1632.71	
	DL	25	17	28	330	13097	18	226.39	202.92	
	DH	25	17	28	330	13097	20	3600	1512.61	17.65
2C-3-N	BL	11	9	13	98	1187	11	3600		22.22
	BH	11	9	13	98	1187	11	3600		22.22
	DL	11	9	13	98	1187	11	3600		22.22
	DH	11	9	13	98	1187	11	3600		22.22
2C-4-N	BL	11	9	29	182	3649	9	42.42	42.42	
	BH	11	9	29	182	3649	9	20.36	20.36	
	DL	11	9	29	182	3649	9	7.76	7.76	
	DH	11	9	29	182	3649	9	3600		22.22
2C-5-N[2]		18	18	26	222	5912	18	0		
3A-1-N[2]		41	41	75	132	4452	41	0		
3A-2-N	BL	11	9	75	30	202	11	341.26		
	BH	11	9	75	30	202	11	1055.12		
	DL	11	9	75	30	202	11	183.48		
	DH	11	9	75	30	202	11	1227.81		
3A-5-N	BL	18	11	102	98	2128	11	20.50	20.50	
	BH	18	11	102	98	2128	11	119.12	119.11	
	DL	18	11	102	98	2128	11	17.29	17.29	
	DH	18	11	102	98	2128	11	45.93	45.93	

[1]: B = breadth first, D = depth first, L = lowest fractional, H = highest fractional

[2]: UB = LB, hence the branch-and-price algorithm is not run

Table 14: Performance of the greedy heuristic and branch-and-price algorithm for instances with c = N with a cut-off point of one hour.

	Branching				TSN Size				T (sec.) till	
Instance	Rule[1]	UB	LB	#Loadings	#Nodes	#Arcs	Makespan	T (sec.)	Int. Sol.	Gap (%)
3B-1-N[2]		57	57	109	196	9732	57	0		
3B-2-N[2]		41	41	109	132	4452	41	0		
3B-4-N	BL	18	11	153	234	6812	18	3600		63.64
	BH	18	11	153	234	6812	17	3600	2763.90	54.55
	DL	18	11	153	234	6812	12	3600	76.38	9.09
	DH	18	11	153	234	6812	12	3600	78.54	9.09
3B-5-N	BL	18	11	102	98	2128	11	21.50	21.50	
	BH	18	11	102	98	2128	11	119.12	119.12	
	DL	18	11	102	98	2128	11	17.29	17.29	
	DH	18	11	102	98	2128	11	45.93	45.93	
3C-3-N	BL	27	25	437	126	3642	27	3600		8.00
	BH	27	25	437	126	3642	27	3600		8.00
	DL	27	25	437	126	3642	26	861.03	497.75	
	DH	27	25	437	126	3642	26	952.88	669.71	
3C-4-N	BL	25	11	1030	332	13700	25	3600		120.66
	BH	25	11	1030	332	13700	25	3600		120.66
	DL	25	11	1030	332	13700	18	3600	3108.34	58.88
	DH	25	11	1030	332	13700	21	3600	2721.64	85.36
3C-5-N	BL	41	32	525	236	12432	41	3600		28.125
	BH	41	32	525	236	12432	41	3600		28.125
	DL	41	32	525	236	12432	41	3600		28.125
	DH	41	32	525	236	12432	41	3600		28.125

[1]: B = breadth first, D = depth first, L = lowest fractional, H = highest fractional

[2]: UB = LB, hence the branch-and-price algorithm is not run