European Journal of Operational Research 320 (2025) 115–131

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Production, Manufacturing, Transportation and Logistics

Exact and heuristic approaches for the ship-to-shore problem

M. Wagenvoort [a,*], P.C. Bouman [a], M. van Ee [b], T. Lamballais Tessensohn [c], K. Postek [d]

[a] Erasmus University Rotterdam, Erasmus School of Economics, Econometric Institute, P.O. Box 1738, 3000 DR Rotterdam, The Netherlands
[b] Netherlands Defence Academy, Faculty of Military Sciences, Het Nieuwe Diep 8, 1781 AC Den Helder, The Netherlands
[c] TNO, Military Operations, Oude Waalsdorperweg 63, 2597 AK The Hague, The Netherlands
[d] Independent researcher, Rotterdam, The Netherlands

ARTICLE INFO

Keywords:
OR in defence
Ship-to-shore problem
Branch-and-price
Integer linear programming
Computational complexity

ABSTRACT

After a natural disaster such as a hurricane or flooding, the navy can help by bringing supplies, clearing roads, and evacuating victims. If destinations cannot be reached over land, resources can be transported using smaller ships and helicopters, called connectors. To start aid on land as soon as possible this must be done efficiently. In the ship-to-shore problem, trips with their accompanying resources are determined while minimising the makespan. Limited (un)loading capacities, heterogeneous connector characteristics and constraints posed by priority of the resources and grouping of the resources (resource sets) all require that the connector trips are carefully coordinated. Despite the criticality of this coordination, existing literature does not consider resource sets and has only developed heuristics. We provide a formulation that incorporates resource sets and develop (i) an exact branch-and-price algorithm and (ii) a tailored greedy heuristic that can provide upper bounds. We find that 84% of our 98 practical instances terminate within an hour in on average 80 s. Our greedy heuristic can find optimal solutions in two-thirds of these instances, mostly for instances that are very constrained in terms of the delivery order of resources. When improvements are found by the branch-and-price algorithm, the average gap with the makespan of the greedy solution is 40% and, in most cases, these improvements are obtained within three minutes. For the 20 artificial instances, the greedy heuristic has consistent performance on the different types of instances. For these artificial instances improvements of on average 35% are found in reasonable time.

1. Introduction

After a natural disaster such as a hurricane, the navy can provide aid by bringing supplies, helping to clear roads, and evacuating victims. In case of coastal areas, the navy provides support by transporting supplies from ships to the shore using smaller ships and helicopters, called *connectors*. For example, the US military delivers supplies in relief missions through a floating dock from which smaller and lighter vessels make deliveries to the pier (Debusmann Jr., 2024). This has to be done efficiently for the help on land to start as soon as possible. As the planning of such an operation may depend on various situational parameters, such as weather conditions, the planning of such an operation has to be done fast. In addition to humanitarian purposes, this problem can be encountered in other military 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, which is a type of transportation problem. In the ship-to-shore problem, connectors pick-up resources, such as personnel and vehicles, from ships, and deliver them to the shore. Hence, it can be seen as a pick-up and delivery vehicle routing problem (PDVRP) (Zachariadis et al., 2016). However, large naval operations typically require the coordinated delivery of various types of resources, which we identified based on various interviews with experts on military operations at the Defence and Security unit of the Netherlands Organisation for Applied Scientific Research (TNO), a Dutch national research institute. To ensure this coordinated delivery, additional decisions on how to load the heterogeneous resources on the connectors are needed. Compared to the traditional PDVRP, the ship-to-shore problem focuses on including the various types of coordination required within large scale operations, while reducing the movement of connectors to round-trips between the various loading and unloading spots at the ships and the shore.

One way in which coordination between the deliveries of the resources is imposed is through priority levels. Resources with a lower priority can only be delivered after items with a higher priority have arrived on the shore. This helps ensure that the command-and-control

* Corresponding author.
E-mail address: wagenvoort@ese.eur.nl (M. Wagenvoort).

https://doi.org/10.1016/j.ejor.2024.08.017
Received 11 January 2023; Accepted 15 August 2024
Available online 19 August 2024

structure remains clear and ensures there are clear phases in the execution of the plan. Between these phases, resources should not be mixed, as it is imposed that preparatory measures should be completed before expensive equipment can be safely deployed. Strict priority orderings can also exist in other related problems. For example, consider the installation of a wind farm where resources should be brought from the shore to the sea. Here it is preferable to deliver expensive components only after the foundation is completed. Another way coordination is imposed is by requiring groups of resources to be delivered at the same time or immediately after each other. Namely, units may have trained with particular vehicles and other units. They can be temporarily separated while being transported, but should be able to work together on the shore. These types of coordination do not exist in the PDVRP. Despite their importance in practical operations, these coordinating constraints have not been considered in prior research on the ship-to-shore problem (Christafore Jr., 2017; Danielson, 2018; Strickland, 2018).

The aim of the ship-to-shore problem to minimise the duration of the operation, such that all resources are on land as soon as possible. In the operation, connectors are loaded at a ship, called the sea base (SB), after which the resources are transported to a landing area (LA) to be unloaded such that the connector can return to a (possibly different) SB for its next trip. Note that only one SB and one LA is visited in each trip for safety reasons. The SBs and LAs accommodate different types of connectors. For example, helicopters can only be loaded at a landing platform on the deck of the ship, while surface connectors cannot use this landing platform. Connectors also have different dimensions, fuel capacities, fuel consumption rates, speeds and weight capacities. Furthermore, the speed of a connector depends on its design speed, whether it is loaded or not, and the state of the sea, *i.e.* the wind and the waves.

The ship-to-shore problem can include various operational constraints that require coordination between the schedules of the different connectors. Our problem formulation has three such constraints. Firstly, there is limited (un)loading capacity, as there is a limited number of (un)loading spots, putting a constraint on the number of connectors that can be (un)loaded at the same time. Secondly, we consider a heterogeneous fleet of connectors with varying speeds, fuel capacity and consumption, and dimensions, affecting the set of resources that can be transported simultaneously. Thirdly, we require coordination between the delivery of the resources in the two ways described before. Namely, we consider priority levels and groups of resources, called *resource sets*, that should be delivered together. The time period in which the resources from a resource set are delivered is called a *delivery wave* for that resource set. There is no ordering imposed between the different waves with the same priority and these delivery waves can (partially) overlap. Note that this implies that the minimum number of connectors should be such that each resource set can be delivered using each connector at most once.

An input to this problem are the ways in which connectors can be loaded during a trip. In the ship-to-shore application, there are very specific practical constrains that are very challenging to incorporate in the model. We therefore focus on optimising the transportation schedule and take the ways connectors can be loaded as input.

The main contributions of this paper are as follows. First, we develop a formulation for the ship-to-shore problem that allows for coordination between the resources, and we prove the ship-to-shore problem to be NP-hard. We then develop two solution methods: An exact branch-and-price algorithm in which heuristic pricing is used in combination with an exact pricing method, and a tailored greedy heuristic that can also be used as an upper bound in the formulation and algorithm. A branch-and-price algorithm is used as preliminary results showed that the integrated problem did not work well. We conduct computational experiments with instances from practice to show that the branch-and-price algorithm is able to solve the majority of these instances within an hour. Finally, we investigate under which circumstances which method is preferred. We observe that the greedy algorithm performs particularly well on instances where resource set constraints exist, but no priorities are defined. We furthermore observe that the exact method has the highest potential to improve solutions when resource set constraints are not present. We additionally use 20 artificial instances to compare with our practical instances. These show that for general instances there is no difference in the performance of the greedy heuristic. The artificial instances confirm the ability of the branch-and-price algorithm to find improvements compared to the solution of the greedy heuristic fast.

The paper is organised as follows. In Section 2, we formally introduce the ship-to-shore problem and the time–space network which we use to solve this problem. Section 3 gives an overview of the related literature and how the problem differs from a PDVRP. Our mathematical model and proof of NP-hardness are given in Section 4. Our branch-and-price algorithm is provided in Section 5 and our greedy heuristic is described in Section 6. In Section 7, we describe the experimental set-up and analyse the performance of our exact algorithm and heuristic. We end with a conclusion in Section 8.

2. Problem definition

In this section, we formally define the problem and its notation. An overview of the notation can also be found in the supplementary materials.

In the ship-to-shore problem we have a set of sea bases Σ and a set of landing areas Λ that each have a set of (un)loading locations, $\mathcal{P}_i$ for $i \in \Sigma$ and $\mathcal{D}_i$ for $i \in \Lambda$, respectively. The set of loading locations and unloading locations can then be defined as $\mathcal{P} = \cup_{i\in\Sigma}\mathcal{P}_i$ and $\mathcal{D} = \cup_{i\in\Lambda}\mathcal{D}_i$, respectively. We denote the set of connectors, *i.e.*, smaller ships and helicopters used for transporting the supplies, as $\mathcal{C}$. Let $\mathcal{M}$ be the set of resource types for which the dimensions are known, and n_m the demand for resource type $m \in \mathcal{M}$. Here, we aggregate resources with the same origin, destination, priority level, and resource set, if assigned any. It is not possible to aggregate resources that are not identical in the origin, destination, priority, and/or resource set, as, for example, personnel should stick together with their unit and it is therefore specified what their origin is. Personnel with different priority levels, origins, and/or destinations, are thus considered different resource types. We define the meaning of priority level and resource set later in this section.

For each connector $c \in \mathcal{C}$, $\mathcal{L}^c$ denotes the set of feasible ways connectors can be loaded, called *loadings*. Here $l \in \mathcal{L}^c$ 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 feasibly fit together on connector c, and have the same priority level, origin and destination. This implies that exactly one loading needs to be selected for each trip made by a connector from an SB to an LA. Travel times between the locations are denoted by t_{ijc} for $i, j \in \mathcal{P} \cup \mathcal{D}$ and $c \in \mathcal{C}$. When $i \in \mathcal{P}$ and $j \in \mathcal{D}$, t_{ijc} is determined using the speed of connector $c \in \mathcal{C}$ while it is loaded, and when $i \in \mathcal{D}$ and $j \in \mathcal{P}$, using the speed while it is empty. The (un)loading time for a connector $c \in \mathcal{C}$ at location $i \in \mathcal{P} \cup \mathcal{D}$ 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.

Due to the (un)loading capacities, at most one connector can be located at each $i \in \mathcal{P} \cup \mathcal{D}$ at all times. Furthermore, the fuel level of a connector should be non-negative at all times. All resources have a priority level and our convention shall be that the lower the number, the higher the priority. Hence, all priority $\pi \in \{1, \dots, \Pi - 1\}$ resource types should be delivered at their destination, before the unloading of priority $\pi + 1$ resource types starts. Furthermore, we have a collection of resource sets $s \in \mathcal{S}$, where $\mathcal{S}$ is a partition of (a subset of) $\mathcal{M}$, *i.e.* not all resources are necessarily part of a resource set. The delivery of resource types within a resource set should start within ϵ time of each other. Namely, if unloading of a connector with resources from resource set s starts at time t and takes t' time, then unloading the

next connector containing resources from resource set s should start at time $t + t' + \epsilon$ at the latest. We call these requirements the resource set constraints. 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 resource types, but only to those of that same type.

A solution to this problem defines for each connector $c \in C$, a sequence of trips from a loading location $i \in \mathcal{P}$ to an unloading location $j \in \mathcal{D}$ where for each trip a loading is assigned, such that the makespan is minimised and all constraints are met.

2.1. The time–space network

To solve the problem, we make use of a time–space network. In the time–space network we work with discrete time periods. This time discretisation results in loss of 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 size of the time–space network and solution quality. In practice, 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). In the remainder of this section, we explain how the time–space network is constructed.

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. To model this problem, we therefore opt for the usage of a time–space network inspired by the approach of Chardaire et al. (2005). To construct a time–space network for an instance, an upper bound on the number of required time periods is required. To this extend, we will use the upper bound obtained using the greedy heuristic described in Section 6 denoted by T.

Let $G = (\mathcal{N}, \mathcal{A})$ be the time–space network with $\mathcal{N}$ the set of nodes and $\mathcal{A}$ the set of arcs. The nodes in the time–space network are a combination of a location and a time period and the arcs denote a transition through time and space.

There are four types of nodes: starting, ending, (un)loading, and waiting nodes. The starting and ending nodes denote the starting and ending locations of the connectors which are denoted by τ_c and τ'_c for connector $c \in C$, respectively. Hence, the set of starting and ending nodes is defined as $\tau = \{(\tau_c, 1) \mid c \in C\}$ and $\tau' = \{(\tau'_c, T) \mid c \in C\}$, respectively. (Un)loading nodes correspond to (un)loading locations at the SBs and LAs and are defined as $\mathcal{P}' = \{(i,t) \mid i \in \mathcal{P}, t = 1, \dots, T\}$ and $\mathcal{D}' = \{(i,t) \mid i \in \mathcal{D}, t = 1, \dots, T\}$ for loading and unloading activities, respectively. The interpretation of node (i,t) is that the connector is (un)loaded at location i during time period t. Waiting nodes are added to model that a connector waits for a free (un)loading spot or travels below maximum speed. The set of waiting locations is denoted by $\mathcal{W} = \{(i,c) \mid i \in \Sigma \cup \Lambda, c \in C\}$ and the corresponding nodes by $\mathcal{W}' = \{(i,t) \mid i \in \mathcal{W}, t = 1, \dots, T\}$. The total set of nodes can then be defined as $\mathcal{N} = \mathcal{P}' \cup \mathcal{D}' \cup \mathcal{W}' \cup \tau \cup \tau'$.

The set of arcs $\mathcal{A}$ consists of four types of arcs: starting, ending, travelling, and waiting arcs. The starting and ending arcs denote the departure and return of a connector from its starting or ending location and are defined as $\{(i,j) \mid i \in \tau, j \in \mathcal{P}'\}$ and $\{(i,j) \mid i \in \mathcal{D}', j \in \tau'\}$, respectively. Furthermore, we include the option of not using a connector by including an arc from node $(\tau_c, 1)$ to (τ'_c, T). Travelling arcs correspond to a feasible transition in both time and space between a loading location i and unloading location j. A feasible transition in space implies that i and j can both be accessed by one of the available connectors. A transition is feasible in time when the time difference of node i and j is at least equal to the travel time t_{ijc} for some connector $c \in C$. It is possible that a connector does not directly proceed to the next location due to the priority, resource set, and capacity constraints, but either waits for a spot to be free or travels below maximum speed. For a travel time of t_{ijc} between loading location i and unloading location j, or vice versa, by connector $c \in C$, let Δ be the equivalent number of time periods. Then, arcs would have to be added from time period t to $t+\Delta+1, t+\Delta+2, \dots, T$. To avoid adding all possible transitions between the different locations, we use the previously defined waiting nodes. With the use of these waiting nodes, we only have to add the shortest arc between locations. Namely, from each loading node an arc is added to an unloading node or waiting node at an LA when feasible in time and space, and from each unloading node an arc is added to a loading node or waiting node at an SB when feasible in time and space. Finally, waiting arcs are added between waiting nodes to allow for a connector to stay at a waiting location, namely the arcs $\{(j,t),(j,t+1) \mid j \in \mathcal{W}, t = 1, \dots, T-1\}$. Hence the number of arcs in the network is linear in the number of SBs, LAs, connectors, and time periods.

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 Fig. 1. 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 after unloading in time period 10.

Finally, we have to define the deliveries of resources from resource sets in terms of the time–space network. For a resource set, it is imposed that in a delivery wave, deliveries of its resources take place within ϵ time of each other. In the time–space network, we therefore impose that resources from a resource set should be delivered in consecutive time periods.

The above described time–space network can be used to model the movement of connectors through both time and space. A solution for the ship-to-shore problem consists of, for each connector, a path through the time–space network and for each trip from an SB to an LA, the loading that should be transported. Which loadings can be used on a specific arc while satisfying the priority and resource set constraints, will depend on the loadings that were assigned in the trips preceding that arc. Therefore, simply duplicating all arcs from an SB to an LA node for each loading would result in some paths through the time–space network being infeasible with respect to these constraints. We describe how loadings are considered in Section 5.2.

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. The ship-to-shore problem

The ship-to-shore problem has been addressed in some research. Villena (2019) solves the ship-to-shore problem using an Integer Linear Programming model to minimise the makespan. The author considers

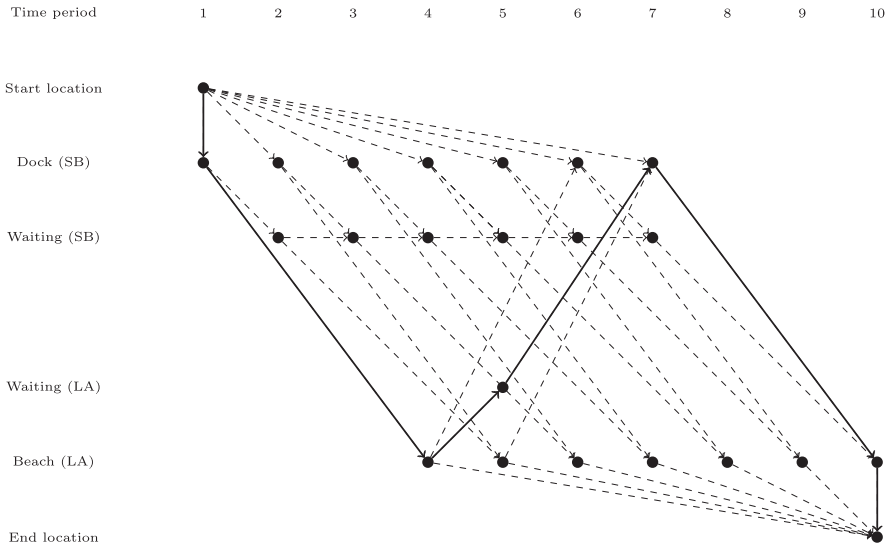


Fig. 1. Example of a time–space network for one connector with one SB containing one dock and one LA containing one beach. Here all possible arcs are given (the bold and dashed lines combined) for a connector that takes two time periods to travel from the SB to the LA and one time period to travel from the LA to the SB. The bold lines are an example of a path through the network.

the priority levels of the connectors, however, the author does not consider (un)loading capacities and the fact that some resources are complementary and should belong to the same wave. Fuel capacities are imposed by defining a maximum distance that the connectors can travel and thus, refuelling is not allowed.

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 than the satisfied 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, resource set constraints are hard to impose as the set of trips selected in the first phase could result in an infeasible solution.

3.2. Related problems

There exist problems that have a similar structure to 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 to 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. Kulshrestha et al. (2014) aim to minimise the evacuation time by assigning buses to 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 in which each location should be visited, which can be seen as a strict type of priority constraints. They do not consider resource sets, and fuel 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. Priorities arise since certain components are required at the start, while others are only required later on. Resource set constraints can be interpreted as the delivery of components that have to be used together in 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. 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.

Another related problem is the pick-up from and delivery to offshore platforms. Here, vessels have to be scheduled to visit offshore oil and gas platforms to execute deliveries and pick-ups, *e.g.* the delivery of equipment and collection of waste. The vessels have capacity constraints, but also capacity constraints for (un)loading at the offshore platforms can exists. This problem is studied by Gribkovskaia et al. (2008) and Cuesta et al. (2017) and solved using a tabu search and adaptive large neighbourhood search, respectively.

More generally, the ship-to-shore problem can be linked to a pick-up and delivery vehicle routing problem (PDVRP), where items should be collected at a pick-up location and then delivered to its delivery location. However, the location that is visited determines the resource that is transported, while in the ship-to-shore problem it has to be determined which resources are picked-up each time a connector visits the location. 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. For each of these aspects, we briefly explain the difference between related aspects of the PDVRP and the ship-to-shore problem.

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 set constraints could be interpreted as requiring all items with the same pick-up and delivery pair to be delivered at the same time, while allowing for split deliveries. Synchronised visits in a vehicle routing problem (VRP) have been considered to allow for driver transitioning or executing tasks to be performed by more vehicles, but differ from the ship-to-shore problem where visits should be synchronised in a delivery wave that can contain multiple consecutive time periods (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). Refuelling options have been studied in relation to the VRP since the introduction of alternative fuel-powered vehicles that have a more limited driving range and more limited and costly refuelling options compared to traditional vehicles. We can model the fuel level in the ship-to-shore problem in a similar way as is done in the electrical VRP. Desaulniers et al. (2002), Hiermann et al. (2016), Schneider et al. (2014) assume vehicles remain at a charging location until the batteries are full. However, in the ship-to-shore problem, refuelling can occur simultaneously with loading a connector, and can be terminated before the connector is fully refuelled, which requires an additional constraint.

The (un)loading capacities are essential in the ship-to-shore problem, as disregarding them can lead to infeasible schedules where the capacities are exceeded. 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 loading bays at which vehicles can be loaded, resulting in the vehicle routing problem with docking constraints (Rieck & Zimmermann, 2010).

4. Mathematical model and complexity

In this section, we discuss the computational complexity of the ship-to-shore problem defined in Section 2. In Section 4.1, the computational complexity of the ship-to-shore problem and two special cases are determined to be NP-hard. Thereafter, we provide a mathematical model in Section 4.2 to solve the problem using the time–space network approach defined in Section 2.1.

4.1. Computational complexity

To proof that the ship-to-shore problem defined in Section 2 is NP-hard, we consider the decision version of the problem. In this problem, the question is whether all resources can be transported in at most N 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.

Theorem 1. *The decision version of the ship-to-shore problem with a single sea base, landing area, connector and priority level; with zero (un)loading times, fuel consumption rates and resource sets; and with unit travel times and multiple resource types, is strongly NP-complete.*

Proof. This special case is in NP as it can be checked in polynomial time whether demand is satisfied and the makespan is at most N.

We will use a reduction from Set Cover, which is known to be strongly NP-complete (Karp, 1972), to prove that this special case is 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 $N = 2K - 1$.

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 at the same time or directly after one another. This problem can thus be interpreted as the delivery of fuel to petrol stations from a central depot. This special case differs from the special case in Theorem 1 as it contains multiple destinations and connectors, but resources that only differ in their destination.

Theorem 2. *The decision version of the ship-to-shore problem with one sea base, resource types that only differ in their destination, multiple connectors, (un)loading times equal to zero, unit travel times between the sea base and the landing areas, one priority level, no resource sets, and a fuel consumption rate equal to zero, is strongly NP-complete.*

Proof. This special case is in NP as it can be checked in polynomial time whether demand is satisfied and the makespan is at most N.

We will use a reduction from 3-Partition, which is known to be strongly NP-complete (Garey & Johnson, 1979), to prove that this special case is NP-complete. In 3-Partition, we are given $3m$ numbers, $a_1, \dots, a_{3m}$. 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 $S_1, \dots, S_m \subseteq \{1, \dots, 3m\}$ 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 m landing areas, each with a demand of B. Hence, there are m resource types with demand B that only differ in the destination. Then, create $3m$ connectors, where connector i can ship a_i units of a resource per trip. Finally, set $N = 1$.

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. ■

Table 1
List of sets, variables and parameters for the Integer Linear Programming formulation.

Set	Explanation
$\mathcal{C}$	Set of connectors
$\mathcal{P}'$	Set of loading nodes
$\mathcal{D}'$	Set of unloading 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 resource with priority $\pi \in \{1,\dots,\Pi\}$ is unloaded
T_{π}^{end}	Time period at which the last resource with priority $\pi \in \{1,\dots,\Pi\}$ is unloaded
v_{st}	A binary variable equal to 1 if resources from set $s \in S$ 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 $s \in S$ 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 $s \in S$ 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

A similar reduction from Partition (Karp, 1972) shows that the problem is (weakly) 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 these theorems. Thus, the ship-to-shore problem is NP-hard.

Corollary 1. *The ship-to-shore problem is strongly NP-hard.*

4.2. Integer linear programming formulation

In this section we introduce a linear programming formulation to solve the ship-to-shore problem. To solve the problem, we need to find a route for each connector, which consists of a path through the time–space network, such that a loading to be transported is assigned for each trip from an SB to an LA. The ship-to-shore problem is then equivalent to assigning exactly one route to each connector such that the makespan is minimised, all resources are transported, and all constraints regarding (un)loading capacities, resource set constraints, and priority levels are met. First, some notation is introduced in Table 1.

Furthermore, we introduce the following notation. Let $\mathcal{R}(c)$ for $c \in \mathcal{C}$ be the set of routes for connector c and $\mathcal{R}(c,m)$ for $c \in \mathcal{C}$ and $m \in \mathcal{M}$ be the set with routes for connector c that delivers at least one resource of type m. For $c \in \mathcal{C}$ and $i \in \mathcal{N}$, we define $\mathcal{R}(c,i)$ as the set of routes for connector c that visit node i. Finally, $\mathcal{R}(c,\{\pi,t\})$ for $c \in \mathcal{C}$, $\pi \in \{1,\dots,\Pi\}$ and $t \in \{1,\dots,T\}$ denotes the set of routes for connector c that deliver priority π resources in time period t and $\mathcal{R}(c,\{s,t\})$ for $c \in \mathcal{C}$, $s \in S$ and $t \in \{1,\dots,T\}$ is the set of routes for connector c that deliver resources from resource set s in time period t.

Now, an Integer Linear Programming (ILP) formulation can be defined as follows:

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

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

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

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

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

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

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

$$T_{\pi}^{end} \geq t y_{\pi t} \qquad t \in \{1,\dots,T\}, \pi \in \{1,\dots,\Pi\} \tag{8}$$

$$T_{\pi+1}^{start} \geq T_{\pi}^{end} \qquad \pi \in \{1,\dots,\Pi-1\} \tag{9}$$

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

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

$$v_{st} - v_{s,t-1} = w_{st}^{start} - w_{s,t-1}^{end} \qquad s \in S, t \in \{2,\dots,T\} \tag{12}$$

$$w_{st}^{start} + w_{s,t-1}^{end} \leq 1 \qquad s \in S, t \in \{2,\dots,T\} \tag{13}$$

$$\sum_{t=1}^{T} w_{st}^{start} \leq 1 \qquad s \in S \tag{14}$$

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

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

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

$$T_{\pi}^{start}, T_{\pi}^{end} \in \mathbb{N}^{+} \qquad \pi \in \{1,\dots,\Pi\} \tag{18}$$

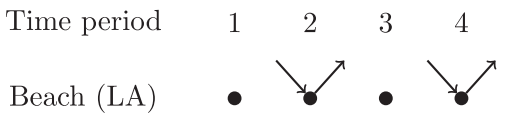


Fig. 2. Wave example 1 (infeasible).

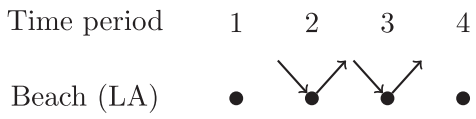


Fig. 3. Wave example 2 (feasible).

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

$$w_{st}^{start}, w_{st}^{end} \in \mathbb{B} \qquad s \in \mathcal{S}, t \in \{1, \dots, T\} \tag{20}$$

$$t_{span} \in \mathbb{N}^{+}. \tag{21}$$

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. Note that a route can also be a route directly from the start location of a connector to the end location of that connector, *i.e.* not all connectors have to transport resources. 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 $T(1 - y_{\pi t})$in constraint (7) ensures that no bound is imposed on T_{π}^{start} when $y_{\pi t}$ equals 0, *i.e.* when no priority π resource types are delivered in time period t. We impose a strict priority ordering, where priority levels cannot be mixed even if they correspond to different locations. However, a more lenient interpretation where priority levels are destination specific is possible in our model. Namely, we can define variables T_{π}^{start} and T_{π}^{end} for each destination separately and add constraints (9) for each separate location.

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_{st}^{start} and w_{st}^{end} denote whether a delivery wave of resource set s starts or ends in time period $t = 1, \dots, T$, respectively. Constraints (10) and (11) jointly set v_{st} equal to 1 if there is at least one route selected that delivers at least one resource from resource set s in time period t. Here $|\mathcal{D}'|$ is used as there can be at most $|\mathcal{D}'|$ connectors delivering resources in a single time period. Constraints (12) and (13) jointly set w_{st}^{start} and w_{st}^{end} equal to 1 if period t is part of the delivery wave for resource set s and if $t-1$ is not or $t+1$ is not, respectively. An example of how these constraints work is given in Example 2. Constraint (14) ensures that there is only one wave for each resource set.

Example 2. To illustrate, Figs. 2 and 3 present the nodes of a beach (LA) 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 Fig. 2, v_{s2}, v_{s4}, w_{s2}^{start}, w_{s2}^{end}, and w_{s4}^{start} are forced to one and all other variables v_{st} and w_{st}^{end} are forced to zero. This solution violates constraint (14) and hence is infeasible. In Fig. 3, v_{s2}, v_{s3}, w_{s2}^{start} and w_{s3}^{end} 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.

5. Branch-and-price algorithm

In the ILP formulation defined in Section 4.2, we consider routes which are paths through the time–space network where each trip from an SB to an LA has an assigned loading. To avoid enumerating an exponential number of routes, we resort to column generation within a branch-and-price framework (Barnhart et al., 1998). We solve a restricted master problem (RMP) containing a subset of all feasible routes and use dual information of the LP-relaxation of this RMP to find improving routes via a Pricing Problem (PP), until no improving routes exist. We solve the PP using a fast heuristic labelling algorithm which is backed up by an exact MIP formulation to ensure we find an improving column if one exist. As the optimal solution of the LP-relaxation of the RMP may be fractional, branch-and-price is used to ensure we find an exact integer solution.

The RMP and PP used in the column generation are explained in Sections 5.1 and 5.2. Thereafter, the branching strategies will be explained in Section 5.3. We conclude with some ideas to strengthen the formulation in Section 5.4.

5.1. 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. However, contrary to the mathematical model in Section 4.2, we only consider a subset of the routes, namely $\mathcal{R}' \subseteq \mathcal{R}$. The set of routes is extended using the PP within the branch-and-price framework.

To ensure that a feasible solution can be found for any subset of routes, variables $p_m \in \mathbb{R}^+$ for $m \in \mathcal{M}$ and $p_c \in \mathbb{R}^+$ for $c \in C$ are introduced and added to the left hand side of constraints (2) and (3), respectively. 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.

5.2. 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 negative reduced costs. These routes can then be added to the set of available routes in the RMP, such that it can be solved again with a larger set of routes. From the RMP defined in Section 5.1, the reduced costs for a route r by connector $c \in C$ can be found as:

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

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.

The PP can be solved using a mixed-integer programming model using the formulation provided in Appendix A. Alternatively, it can be solved as a shortest path problem with resource constraints (SPPRC) on the time–space network using an appropriate labelling algorithm (Irnich & Desaulniers, 2005). Since solving the PP in an exact manner as an SPPRC is very time-consuming when all resources/constraints are considered, we opt for a hybrid approach. First, we use a heuristic

pricing method, where we limit the number of labels we store in each node. Second, once this heuristic procedure fails at finding routes with reduced costs below some threshold, we switch to exact pricing using the mixed-integer programming model until the node is solved to optimality.

In the labelling algorithm, we define a label as $L(c, f, \pi, S)$, where c denotes the reduced costs, f denotes the current fuel level, π denotes the current priority level, and $S \subseteq \mathcal{S}$ denotes the set of resource sets of priority level π of which at least one resource has been delivered. Labels are created by extending paths through the time–space network and appropriately updating the reduced costs based on the dual variables, the fuel level, the current priority level and the current set of resource sets covered. Whenever a path is extended from a node at an SB to a node at an LA, a label is constructed for each possible loading. Namely, if the current priority level is π, it is only possible to assign loadings with priority level $\pi' \geq \pi$. Furthermore, due to the delivery waves, if resources of a resource set in $S \subseteq \mathcal{S}$ are already delivered in a route, it is not possible to assign a loading that contains resources of a resource set in S as this route cannot be used. Label $L_1(c_1, f_1, \pi_1, S_1)$ then strictly dominates label $L_2(c_2, f_2, \pi_2, S_2)$ if (i) $c_1 \leq c_2$, (ii) $f_1 \geq f_2$, (iii) $\pi_1 = \pi_2$, (iv) $S_1 \subseteq S_2$, and (v) at least one of these conditions is strict.

We use this labelling algorithm heuristically by limiting the number of labels at each node (Desaulniers et al., 2002). Here we use a different parameter for the sink and the bound on the number of labels at the other nodes in the network. Furthermore, to limit the number of routes that are added to the RMP in each iteration, a similarity score is used to select routes that are disjoint in the nodes that it visits and/or resources that it delivers (Breugem, 2020). Namely, we sort the candidate routes in ascending order of reduced costs and select the first route. Then, for each following candidate route x, we compute its similarity score with all routes y that are already selected. The route x is selected if none of the similarity scores exceeds 0.5, *i.e.* each route is at least 50% dissimilar to the other routes. We define the similarity score of routes x and y as an overlap coefficient known as the Szymkiewicz–Simpson coefficient which is based on both the nodes visited and resources delivered (Simpson, 1947; Szymkiewicz, 1934):

$$\frac{1}{2}\frac{|\mathcal{N}(x) \cap \mathcal{N}(y)|}{\min\left\{|\mathcal{N}(x)|, |\mathcal{N}(y)|\right\}} + \frac{1}{2}\frac{\sum_{m\in\mathcal{M}} \min\{n_{xm}, n_{ym}\}}{\min\left\{\sum_{m\in\mathcal{M}} n_{xm}, \sum_{m\in\mathcal{M}} n_{ym}\right\}}, \tag{23}$$

for $\mathcal{N}(x)$ the set of nodes in route x, and n_{xm} the number of resources of type m delivered in route x. We perform this step for each connector separately and do not compare routes of different connectors as the demand of a resource can be fulfilled by different connectors.

5.3. 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.

Let a_{ij} be a binary variable representing the usage of an arc from i to j and z_{li} a binary variable representing the delivery of loading l to location i. We then apply branching 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 does not guaranteed 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 PPs. 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 not used in the corresponding PPs.

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 delivery of l at i is forbidden in the PPs. 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 delivery of l' to i is forbidden in the PPs.

For imposing an arc or a delivery, the arc/loading removal operations above 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 require constraints that would not be needed, *e.g.* in the classical vehicle routing problem where each node should be visited. These constraints ensure that among the available routes, at least one is selected that respects the given arc/delivery imposition:

$$\sum_{c\in\mathcal{C}} \sum_{r\in\mathcal{R}'(c,\{i,j\})} x_{rc} \geq 1 \qquad (i, j) \in \mathcal{A}^o \tag{24}$$

$$\sum_{c\in\mathcal{C}} \sum_{r\in\mathcal{R}'(c,\{l,i\})} x_{rc} \geq 1, \qquad (l, i) \in \mathcal{L}^o \tag{25}$$

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

Adding constraints (24) and (25) to the restricted master problem changes the objective function of the pricing problem. Hence, the following terms are added to the objective in (27):

$$-\sum_{(i,j)\in\mathcal{A}^o} \lambda_{ij}^{(24)} a_{ij} - \sum_{(l,i)\in\mathcal{L}^o} \lambda_{li}^{(25)} z_{li}, \tag{26}$$

where $\lambda_{ij}^{(24)}$ and $\lambda_{li}^{(25)}$ are the dual variables of constraints (24) and (25), respectively.

There are multiple ways to select which arc or delivery to branch on. Here, the least, *i.e.* closest to 0 or 1, and most, *i.e.* closest to 0.5, fractional arc or delivery strategy will be used. 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, after preliminary experimentation, the depth first approach is used. Again, ties are broken arbitrarily.

5.4. Strengthening the formulation

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. 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 on the optimal makespan by relaxing the constraints on the (un)loading capacities and resource sets. This relaxation can be solved as a job-shop scheduling problem in which resource-transporting trips to the shore are assigned to the connectors and can easily be solved in a pre-processing phase. Because

this lower bound can be higher than the optimal value 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.

6. Greedy heuristic

The greedy heuristic 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 loading location to pick up a specific set of resources, followed by a visit to an unloading location to deliver these resources. A forbidden list is used to denote trips that are not allowed to be added as they have previously resulted in infeasible solutions.

In the algorithm, we need to use some measure to determine which trip should be added. Since we are interested in minimising the makespan, this measure should include the change in the makespan when the trip is executed. When solely the change in the makespan is used, faster connectors are preferred over slower, but larger, connectors. We therefore want to compensate a larger change in the makespan with the quantity of resources that are transported. As there are both people and large vehicles that can be transported, using the number of resources that are transported is not an appropriate measure. This would namely imply that large resources, which can only be transported by a certain connectors, are left until the smaller resources are transported. Therefore, we have chosen to use the surface area measured in square metres, a in Step 2, that is transported as a measure of the quantity of resources that are transported.

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 forbidden. Here both the capacity constraints at the different locations and the priority constraints are taken into account. Furthermore, we keep track of the fuel level of the connector, hence if necessary, the connector remains one or more additional time periods at the SB for refuelling purposes.

STEP 2: For each loading with priority level π, determine the total area in m^2 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: While there is an incomplete resource set, *i.e.* a trip was added in which part of the resources from a resource set are delivered, the completion of the delivery of this resource set is prioritised before considering the delivery of resources that are not part of this resource set. This is done by repeating versions of Steps 1–3 until all resources in the resource sets are completed. In Step 1, besides the capacity and priority constraints, the resource sets now also have to be considered, *i.e.* only consecutive time periods to the current delivery of the incomplete resource set are considered. In Step 2, only the area of the resources that belong to the incomplete resource set are considered. If no feasible pair exists in Step 3, *i.e.* the resource set constraint cannot be met, the last trip that was added to the schedule is removed and marked as forbidden.

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.

7. Computational experiments

In this section we present the results of computational experiments on both artificial instances and instances constructed using data from the Royal Netherlands Navy. In these experiments, we first analyse the effect of using the bounds as explained in Section 5.4 and the heuristic pricing method as explained in Section 5.2. Thereafter, we analyse the performance of the branch-and-price algorithm and the greedy heuristic for different types of instances.

In Section 7.1 we describe the data and instance construction, and in Section 7.2 the corresponding results are presented.

7.1. Experimental design

We consider both artificial instances and instances constructed using data from the Royal Netherlands Navy. Each instance consists of a demand set containing the resources that should be delivered and the corresponding SBs and LAs, and a supply set in terms of the available connectors and (un)loading capacities at the SBs and LAs. Furthermore, an instance is characterised by the operational constraints that are considered. We define the following naming scheme for instances: *d-s-o*, where *d* denotes the demand set, *s* the supply set, and *o* 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 for the instances constructed with data from the Royal Netherlands Navy. Thereafter, we describe the artificial instances.

7.1.1. Royal netherlands navy instances

In this section we describe the instances constructed with data from the Royal Netherlands Navy. The demand at the shore consists of a set of resource types that have to be transported. For each resource type, the origin and destination is defined. Furthermore, it is denoted what the priority number of the resource type is and to what resource set it belongs to, if any.

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 2. 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 1 A and 2 A contain one priority level, *i.e.* all resources have the same priority level.

The supply consists of the number of available connectors and the (un)loading capacity at both the SBs and LAs. There are three types of loadings spots: docks, davits, and landing platforms, and there is one type of unloading spot: landing zones. The different types of (un)loading spots can be used by different connector types. Namely, landing platforms 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 3.

The operational constraints consist of, amongst others, constraints regarding the order of the delivery of the resources. As we are interested in the performance of our branch-and-price algorithm and our greedy heuristic under different circumstances, we vary these operational constraints. These constraints are the priority constraints and resource set constraints. For instances without a priority ordering, this implies that all resources have the same priority level. When either or both of these coordinating constraints are absent from an instance, there are fewer constraints. This affects the number of choices in each step of the greedy heuristic and the number of variables and constraints in the branch-and-price algorithm.

We define four options for *o*. 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

Table 2
Overview of the demand sets containing the identifier, number of SBs, number of LAs, the distance between the SBs and LAs in nautical miles, the number of priority levels, the number of resource types, and the number of resource sets for each demand set.

Demand set	#SBs	#LAs	Distance SB - LA (nm)	# Priority levels	# Resource types	# Resource sets
1A	2	2	15	1	10	1
1B	2	2	15	2	15	2
1C	2	2	15	2	18	2
2A	2	3	15	1	10	1
2B	2	3	15	2	15	2
2C	2	3	15	2	20	2
3A	2	1	15	2	10	2
3B	2	1	15	2	18	3
3C	2	1	15	2	25	3

Table 3
Overview of the supply sets containing the identifier, the number of connectors, the number of different connector types, and the number of each type of (un)loading spot available at each SB/LA for each supply set.

Supply set	# Connectors	# Connector types	# Docks per SB	# Davits per SB	# Landing platforms 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

Table 4
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 96 instances.

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

denote the case in which the resource set constraints are added to case N. The full and default model we consider contains both constraints, 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 results in the instances shown in Table 4. Here 'All' implies that all four subsets of the operational constraints are considered. The instances used for only options P and N imply that the other 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 1 A and 2 A 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 96 instances.

7.1.2. Artificial instances

In this section we describe the artificially constructed instances. The instances and solutions can be obtained from Wagenvoort (2023). The artificial instances are constructed from a set of connectors and a set of available resources. From these sets, five combinations of a demand and supply set are generated. We consider all four configurations of the operational constraints, resulting in 20 instances.

We consider four different types of connectors and seven different types of resources. For the connector types, we consider one type of helicopter and three types of surface connectors: large, medium, and small. For each connector type, given in Table 5, we specify the (un)loading time, capacity, fuel related parameters, speed when loaded and empty, and at which locations it can be (un)loaded. For each of the resources, given in Table 6, we specify the size of the resource and for each connector type whether it can be placed on this connector type. Note that for simplicity we have defined the capacity of a connector and the size of a resource in terms of one dimension only. We consider a loading feasible for a connector type when it does not contain any resources that are not compatible with the connector type and the sum of the sizes of the resources in the loading does not exceed the capacity of the connector.

To construct the artificial instances, we randomly select a set of connectors, where we always include at least one connector of type 'Large' to ensure feasibility. We then randomly select a set of resources, for which we randomly generate a priority level, resource set, and quantity. Here we select the smaller resources with a higher probability. Furthermore, we generate for each resource an origin and destination. We let there be either one or two SBs and one or two LAs reachable by surface connectors and at most one LA reachable by helicopters. The distance between the SBs and LAs is set to 15 nautical miles for each instance.

Table 5
Overview of connector types used for the artificial instances. For each connector type we indicate the (un)loading times in minutes, the capacity for resources, the fuel capacity, the fuel consumption rate (per minute), the refuel rate (per minute), the speed while being (un)loaded in nautical miles per hour, and the type of (un)loading locations it can access.

Type	Loading time (min.)	Unloading time (min.)	Capacity	Fuel capacity	Fuel consumption (per min.)	Refuel rate (per min.)	Speed loaded (knts)	Speed empty (knts)	Location compatibility
Large	15	15	150	10	0.005	0.15	10	12	dock, beach
Medium	10	10	75	10	0.005	0.15	20	25	dock, beach
Small	5	5	25	10	0.01	0.15	30	30	dock, davit, beach
Helicopter	5	5	8	10	0.01	0.15	125	125	landing platform (SB), landing zone (LA)

Table 6
Overview of resource types used for the artificial instances, their size, and what connector can carry them.

Name	Size	Connector compatibility			
		Large	Small	Medium	Helicopter
Pax	1	X	X	X	X
VehA	60	X			
VehB	40	X			
VehC	35	X			
VehD	30	X	X		
VehE	25	X	X		
VehF	20	X	X		

We construct five demand sets Di and supply sets Si for $i \in \{1, \dots, 5\}$. The contents of these five demand and supply sets can be found in the supplementary materials. These can be used to obtain all 20 instances, namely we consider instances $Di - Si - o$ for $i \in \{1, \dots, 5\}$ and $o \in \{\text{F, P, W, N}\}$. We then consider all maximal loadings for each connector based on the compatibility of the resources and the size of the connector. The loadings can be found in Wagenvoort (2023).

7.2. 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 96 GB RAM. We choose a cut-off point of one hour for the branch-and-price algorithm. Whenever we refer to the gap, we mean the optimality gap between the lower bound and the best integer solution.

For the labelling algorithm, parameters must be defined. Namely, we have to determine the maximum number of labels to store at each node, the maximum number of labels to store at the sink, and the threshold for switching from the heuristic labelling algorithm to the exact MIP. Based on early experiments, we choose the following parameters. We allow, in each pricing problem, at most ten labels at the sink and at most five at all other nodes, *i.e.* at most ten columns per connector are added in each iteration. When no route with negative reduced costs smaller than -0.05 is found by the labelling algorithm, we switch to the MIP and use the MIP until the node is solved to optimality.

The remainder of the section is structured as follows. First, the effect of using the bounds and the labelling algorithm is analysed. Second, the performance of the branch-and-price algorithm and greedy heuristic are discussed. Finally, the performance of the artificial instances are analysed to compare with the findings of the instances from the Royal Netherlands Navy.

7.2.1. The effect of bounds and labelling

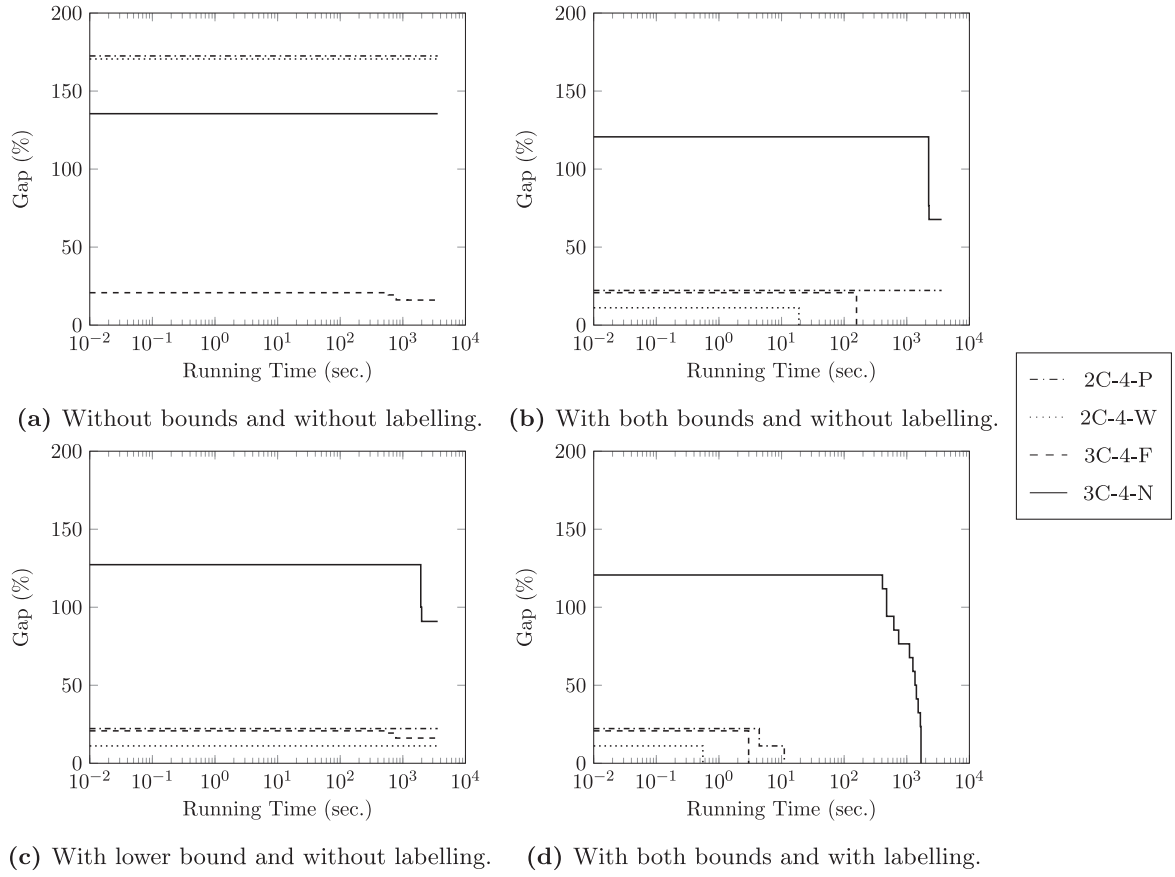
Intuitively, using the bounds to strengthen the formulation as explained in Section 5.4 and using heuristic pricing in combination with the exact MIP as explained in Section 5.2, speeds-up the branch-and-price algorithm. To validate our intuition, some instances with different characteristics and demand are run for different configurations. We run each instance for both the most and least fractional branching rule with a cut-off point of one hour and use the result of the branching strategy with the lowest running time or lowest gap after one hour. The change in the optimality gap over time for this branching strategy is presented in Fig. 4.

Fig. 4(a) shows that none of the instances terminate within an hour. In fact, only for one instance, instance 3C-4-F, the gap decreases within an hour. For three of the four instances, the lower bound obtained by solving the machine scheduling problem as explained in Section 5.4, is better than the initial root node without the bound. Therefore, in Fig. 4(c), we see a decrease in the gap for these instances. When both bounds are added, see Fig. 4(b), we see that instances 3C-4-F and 2C-4-W now terminate within an hour. Note that the gap instantly drops to zero from the initial gap. Since we are using a depth first approach, it can occur that the lower bound within the branching tree could not have been updated before termination of the algorithm. This results in an instant drop in the gap. We also see that the gap obtained for instance 3C-4-N when both bounds are used is lower compared to the gap when only the lower bound is used. Fig. 4(d) shows the change in the gap over time when the heuristic labelling algorithm is used in combination with the exact MIP. We see that now also instance 3C-4-N terminates and that all other instances terminate faster.

7.2.2. Results of the branch-and-price algorithm and greedy heuristic

In this section, the results of the branch-and-price algorithm and the greedy heuristic are presented. Here both bounds and the labelling heuristic are used in the branch-and-price algorithm. We analyse the performance of the branch-and-price algorithm and the greedy heuristic based on the instance characteristics, namely, the instance type (A, B, or C), and which operational constraints are present in the instance. Summarising results for the different instance types are given in Table 7, full results can be found in the supplementary materials.

In Table 7 we see that all small instances (type A) terminate within an hour and that, with the exception of the instances with only priority constraints (P), all terminate within a minute. All type B instances terminate within approximately three minutes, with the exception of the instances with only priority constraints (P). For type C instances, all the instances with only the resource set constraints (W) terminate within an hour. We thus see, both from the percentage of instances that terminate within an hour and the running times, that the instances with only the priority constraints are hardest to solve. On average, 84% of all instances could be solved within an hour of which 75% are solved in approximately three minutes.



(a) Without bounds and without labelling. (b) With both bounds and without labelling.

(c) With lower bound and without labelling. (d) With both bounds and with labelling.

Fig. 4. The change in the optimality gap, the gap between the lower bound and best integer solution, over time in the branch-and-price algorithm with a cut-off point of one hour for different configurations. For the configurations we consider with/without actively using the lower and upper bound as described in Section 5.4, and with/without heuristic labelling as described in Section 5.2.

Table 7
Performance measures of the branch-and-price algorithm with a cut-off point of one hour. This includes the percentage of instances that terminate within an hour, the average number of columns generated, the average number of times the labelling heuristic is used per node, the average number of times the exact MIP is used per node, and percentiles of the total running times. Here o denotes the set of operational constraints regarding the order of the resources that are present, namely either the full set (F), only the priority constraints (P), only the resource set constraints (W), or neither of these constraints (N).

o	Type	Num. Instances	Term. <1 h (%)	Avg. # Columns	Avg. # Heur	Avg. # MIP	Running time (sec.)				
							min	25th	50th	75th	max
F	A	3	100	140.00	5.00	1.00	0.14	0.19	0.24	0.29	0.34
	B	7	100	6422.29	5.50	1.00	0.36	0.43	0.65	2.21	2.84
	C	8	87.50	2045.88	7.26	1.12	0.65	0.99	1.23	2.28	3600
	Total	18	94.44	3623.71	6.27	1.06	0.14	0.49	1.10	2.45	3600
P	A	3	100	27 058.50	1.42	1.00	14.81	588.84	1162.87	1736.90	2310.93
	B	12	58.33	15 306.13	3.06	1.00	0.42	136.76	3600	3600	3600
	C	9	22.22	22 156.00	1.31	1.02	10.64	3600	3600	3600	3600
	Total	24	50.00	19 656.33	2.10	1.01	0.42	714.30	3600	3600	3600
W	A	9	100	165.60	5.20	1.80	0.13	0.13	0.18	0.70	1.02
	B	7	100	448.33	5.44	1.00	0.42	0.46	0.63	0.74	1.06
	C	8	100	960.63	5.92	1.00	0.27	0.55	1.04	2.91	8.17
	Total	24	100	589.63	5.58	1.21	0.13	0.42	0.70	1.10	8.17
N	A	9	100	7556.00	2.80	1.00	0.05	0.10	4.31	21.08	58.81
	B	12	100	3060.00	1.78	1.01	0.27	1.38	6.80	93.55	182.08
	C	9	77.78	182 202.30	3.87	1.00	0.31	1.31	88.25	2643.81	3600
	Total	30	93.33	66 659.00	2.72	1.01	0.05	0.57	7.65	120.33	3600
Total		96	84.38	23 781.08	4.12	1.07	0.05	0.55	1.64	182.08	3600

Regarding the usage of the MIP after heuristic labelling, we observe that the average number of times the MIP is used is only slightly above one, indicating that the heuristic labelling algorithm is able to find good negative reduced costs columns. We see that the number of generated columns varies a lot. If the number of columns is high, this can be caused by two things, the number of iterations at a node is high resulting in many routes, and/or the number of nodes in the branching tree is high resulting in many nodes that are solved. For the 'P' and 'N' instances, we see that, although the labelling heuristic is only used between two and three times on average, the number of columns is significantly larger compared to the 'F' and 'W' instances. These results thus show that, as we would expect, these less constrained instances have more feasible routes making the problem harder to solve.

Comparing the results of the branch-and-price algorithm with the solutions of the greedy heuristic, we find that for two thirds of the instances the greedy solution is proven to be optimal. Table 8 shows the percentage of instances for which the greedy solution is proven to be optimal, the percentage of instances for which the branch-and-price algorithm finds improvement, and the average gap with the greedy solution and average time to obtain these improvements for these instances. We see from these results that the greedy heuristic performs best when there are resource sets (F and W). In the cases with resource set constraints and no priority levels (W), all greedy solutions are optimal, and proven to be optimal within an hour. When priority levels do exist (F), the greedy heuristic finds the best and optimal solution in almost 90% of the cases. When an improvement is found, the solution of the greedy heuristic has, on average, a makespan that is 12% higher compared to the optimum. These improvements are found in a few seconds. Although instances with both constraints (F) are more constrained compared to instances with only resource set constraints (W), we observe that improvements are found in F instances while for all W instances the solution of the greedy heuristic is optimal. As the greedy heuristic is an iterative method, it does not anticipate on future trips. Hence, it might select connectors at the end of a priority level that are required at the start of the next priority level. The branch-and-price algorithm is then able to find a solution where these essential connectors are not used at the end of a priority level resulting in a better solution.

For instances with no resource set constraints (P and N), we see that the greedy heuristic performs worse. Furthermore, not all large instances (type B and C instances) without resource set constraints terminate within an hour. This is the case for all instances with resource set constraints. Hence, we see that while there are instances where no improvement compared to the solution of the greedy heuristic is found, the solution of the greedy heuristic is also not proven to be optimal. In 37.50% and 50.00% of the instances with (P) and without (N) priority constraints, respectively, the greedy solution is known to be optimal. In 33.33% and 46.67% of the instances, respectively, we find an improvement. On average, the gaps between the solution of the greedy heuristic and the solution of the branch-and-price algorithm are 40%. If we compare the gaps for the different instances types (types A, B and C), we see that the smaller the instances, the larger the gaps. This can be explained by the smaller makespan, *i.e.* a difference in the makespan of one time period gives a larger gap for smaller instances where the makespans are smaller. When improvements can be found, these are found within five minutes in most of the cases. Only for the largest instances (type C) when there are no operational constraints regarding the order of delivery (N), there are instances where it takes more than five minutes.

Overall, we thus see that the branch-and-price algorithm is able to solve the majority of the instances in limited time. We see that instances with only priority constraints are hardest to solve, while instances with both constraints or only resource set constraints are easiest to solve. Although priority constraints restrict the solution space compared to having neither of these constraints, they do not restrict it as much as resource set constraints while still having to consider some coordination between the schedules of the different connectors. Furthermore we find that when resource set constraints exist, the greedy solution is often optimal. When these constraints are not present, improvements were found compared to the solution of the greedy heuristic in about 40% of the instances. These improvements are on average found within a few minutes and have an average gap of approximately 40% with the solution of the greedy heuristic.

7.2.3. Artificial instances

The results in Section 7.2.2 show that the greedy heuristic performs well when instances are constrained, especially when resource set constraints exist. In this section, we compare the performance of the greedy heuristic and branch-and-price algorithm for the artificial instances and compare this finding with the results from the instances from the Royal Netherlands Navy.

The 20 artificial instances are run under the same configurations as the instances from the Royal Netherlands Navy. The full results of the artificial instances can be found in the supplementary materials. Summarising results comparing the performance of the greedy heuristic and branch-and-price algorithm can be found in Table 9. The results in this table show that for some instances the greedy heuristic finds the optimal solution, but that quite often improvement is found. When improvement is found, the average gap between the greedy heuristic and the best solution found by the branch-and-price algorithm is quite large, on average 35%. Furthermore, generally this solution is found relatively soon, on average within a couple of minutes.

When we compare the results of the artificial instances in Table 9 with the results of the instances from the Royal Netherlands Navy in Table 8, we notice the difference in the results for the different types of instances. Namely, while the greedy heuristic is optimal for all instances with resource set constraints from the Royal Netherlands Navy, this is not the case for the artificial instances. A potential reason is that instances from the Royal Netherlands Navy are constructed by experts that have knowledge of the capacities and the potential loadings on the connectors that can be encoded in defining the resource sets.

8. Conclusion

In this paper we provide a formulation for the ship-to-shore problem that allows for coordination between the connectors. We prove that the ship-to-shore problem is NP-hard, even in restricted special cases. We develop (i) a branch-and-price algorithm, and (ii) a tailored greedy heuristic. Our branch-and-price algorithm makes use of an upper and lower bound and we incorporate a pricing heuristic that improves the running time of the algorithm. We investigate, using data from the Royal Netherlands Navy, under which circumstances, which method is preferred. We find that the branch-and-price algorithm is able to solve the majority of the instances within an hour and that it performs best in very constrained cases in terms of the coordination of delivering sets of resources. We also find that the greedy heuristic is often able to find the optimal solution in such restricted cases. However, in less restricted cases the branch-and-price algorithm finds an improvement compared to the greedy algorithm in approximately 40% of the instances. For those instances, the average gap with the greedy heuristic is around 40% and those improvements are found within a few minutes.

We then use artificial instances to compare with the instances from the Royal Netherlands Navy. We find no difference in the performance of the greedy heuristic under different circumstances for these instances. This shows that potentially the greedy heuristic performs well when coordination between the resources is required due to a bias in the instances constructed by experts in the field. The artificial instances do confirm the ability of the branch-and-price algorithm to find improvements fast. Namely, for the instances where improvement compared to the solution from the greedy heuristic is found, the average gap with the solutions of the branch-and-price algorithm is 35% and found within a few minutes.

Table 8
The effect of priority level constraints and resource set constraints on the performance of the greedy heuristic with a cut-off point of one hour for the branch-and-price algorithm on the instances from the Royal Netherlands Navy. Here o 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). For each set of operational constraints, we denote for each instance type separately as well as for all instances types combined, the number of such instances, the % of these instances for which we know the greedy heuristic found the optimal solution, and the % of instances for which the BP found improvement. For the instances for which improvement was found, the average gap with the solution from the greedy heuristic is given as well as the average time it took to find the best integer solution.

					Instances with improvement	
o	Instance type	Num. Instances	Greedy optimal (%)	BP finds improvement (%)	Avg. Gap greedy (%)	Avg. T. till best Sol. (sec.)
F	A	3	100	0	–	–
	B	7	85.71	14.29	14.29	2.69
	C	8	87.50	12.50	10.00	5.85
	Total	18	88.89	11.11	12.14	4.27
P	A	3	66.67	33.33	127.27	14.81
	B	12	50.00	16.67	38.96	91.64
	C	9	11.11	55.56	24.89	31.93
	Total	24	37.50	33.33	41.21	44.72
W	A	9	100	0	–	–
	B	7	100	0	–	–
	C	8	100	0	–	–
	Total	24	100	0	–	–
N	A	9	88.89	11.11	63.64	8.51
	B	12	33.33	66.67	44.35	50.05
	C	9	33.33	55.56	24.18	484.24
	Total	30	50.00	46.67	38.52	202.15
Total		96	66.67	25.00	37.22	133.19

Table 9
The effect of priority level constraints and resource set constraints on the performance of the greedy heuristic with a cut-off point of one hour for the branch-and-price algorithm on the artificial instances. Here o 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). For each set of operational constraints, we denote the % of these instances for which we know the greedy heuristic found the optimal solution, and the % of instances for which the BP found improvement. For the instances for which improvement was found, the average gap with the solution from the greedy heuristic is given as well as the average time it took to find the best integer solution.

			Instances with improvement	
o	Greedy optimal (%)	BP finds improvement (%)	Avg. Gap greedy (%)	Avg. T. till Best Sol. (sec.)
F	20	40	25.48	1032.01
P	40	20	47.83	104.43
W	20	80	42.05	32.09
N	20	80	30.10	15.00
Average	25	55	35.22	214.25

Therefore, in practice when these coordinating constraints exist, current practices mimicked by the greedy heuristic, might perform well. However, the branch-and-price algorithm is able to find large improvements to the solutions of the greedy heuristic fast, hence it is worthwhile to try and find a better schedule as large gains can be made. There are a number of options for further research related to our model. 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. In that case, this would require 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, but there may be opportunity for better heuristics in different situations. For less constrained problems, one can apply additional local search steps 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 tailored heuristic works well for other similar applications, 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.

CRediT authorship contribution statement

M. Wagenvoort: Conceptualization, Formal analysis, Methodology, Project administration, Writing – original draft. **P.C. Bouman:**

Table A.10
List of sets, variables and parameters for the Pricing Problem.

Set	Explanation
$\mathcal{A}$	Set of arcs
$\mathcal{D}'$	Set of landing area nodes
$\mathcal{L}$	Set of loadings
$\mathcal{P}'$	Set of sea base nodes
$\mathcal{T}$	Set of time periods
$\mathcal{W}'$	Set of waiting nodes
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
n_{ml}	The number of resources from type $m \in \mathcal{M}$ in loading $l \in \mathcal{L}$
$p_{l\pi}$	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 $c \in \mathcal{C}$

Methodology, Supervision, Writing – review & editing. **M. van Ee:** Methodology, Supervision, Writing – review & editing. **T. Lamballais Tessensohn:** Conceptualization, Data curation, Supervision. **K. Postek:** Conceptualization, Data curation, Methodology, Supervision, Writing – review & editing.

Acknowledgements

We would like to thank Dr. Kerry M. Malone from TNO for the feedback she provided on previous versions of the paper. This research was made possible by TNO in collaboration with Erasmus University Rotterdam and the Netherlands Defence Academy.

Appendix A. Mathematical formulation of the pricing problem

In the Pricing Problem (PP), we aim to find a route such that the reduced costs (22) are minimised. Additionally to the notation introduced in Table 1, the notation in Table Table A.10 is used in the PP.

We can then define the PP for a specific connector $c \in C$ as follows:

$$\min \; -\sum_{m\in\mathcal{M}} \lambda_m^{(2)} b_m + \lambda^{(3)} + \sum_{i\in\mathcal{N}(c)} \sum_{j\in\mathcal{N}(c):(i,j)\in\mathcal{A}(c)} \lambda_i^{(4)} a_{ij} + \sum_{t=1}^{T} \sum_{\pi=1}^{\Pi} \sum_{l\in\mathcal{L}(c,\pi)} \sum_{i\in\mathcal{D}'(c,t)} \lambda_{ct\pi}^{(5)} z_{li} + \sum_{s\in S} \sum_{l\in\mathcal{L}(c,s)} \sum_{t=1}^{T} \sum_{i\in\mathcal{D}'(c,t)} \left(\lambda_{st}^{(10)} + \lambda_{st}^{(11)}\right) z_{li} + \lambda_c^{(15)} d \tag{27}$$

$$\text{s.t.} \sum_{j\in\mathcal{N}(c):(\tau,j)\in\mathcal{A}(c)} a_{\tau j} = 1 \tag{28}$$

$$\sum_{i\in\mathcal{N}(c):(i,\tau')\in\mathcal{A}(c)} a_{i\tau'} = 1 \tag{29}$$

$$\sum_{i\in\mathcal{N}(c):(i,j)\in\mathcal{A}(c)} a_{ij} - \sum_{i\in\mathcal{N}(c):(j,i)\in\mathcal{A}(c)} a_{ji} = 0 \quad j \in \mathcal{N}(c) \setminus \{\tau, \tau'\} \tag{30}$$

$$f_i \le \sum_{j\in\mathcal{N}(c):(i,j)\in\mathcal{A}(c)} a_{ij} \quad i \in \mathcal{P}'(c) \tag{31}$$

$$u_\tau = Q_c \tag{32}$$

$$u_j \le Q_c - h_c(t_j - t_i)a_{ij} \quad (i,j) \in \mathcal{A}(c) \tag{33}$$

$$u_j \le u_i + g_c f_i - h_c(t_j - t_i)a_{ij} + Q_c(1 - a_{ij}) \quad (i,j) \in \mathcal{A}(c) \tag{34}$$

$$\sum_{l\in\mathcal{L}(c)} z_{lj} = \sum_{i\in\mathcal{P}'(c)\cup\mathcal{W}'(c):(i,j)\in\mathcal{A}(c)} a_{ij} \quad j \in \mathcal{D}'(c) \tag{35}$$

$$2z_{lj} \le k_{lj} + \sum_{i\in\mathcal{P}'(c)\cup\mathcal{W}'(c):(i,j)\in\mathcal{A}(c)} k_{li} a_{ij} \quad l \in \mathcal{L}(c), j \in \mathcal{D}'(c) \tag{36}$$

$$b_m \le n_m \quad m \in \mathcal{M} \tag{37}$$

$$b_m \le \sum_{l\in\mathcal{L}(c,m)} \sum_{i\in\mathcal{D}'(c)} n_{ml} z_{li} \quad m \in \mathcal{M} \tag{38}$$

$$\sum_{l\in\mathcal{L}(c,\pi)} \sum_{j\in\mathcal{D}'(c,t)} z_{lj} \le y_{\pi t} \quad t \in \{1, \dots, T\}, \pi \in \{1, \dots, \Pi\} \tag{39}$$

$$\sum_{\pi=1}^{\Pi} y_{\pi t} \le 1 \quad t \in \{1, \dots, T\} \tag{40}$$

$$T_\pi^{start} \le t y_{\pi t} + T(1 - y_{\pi t}) \quad t \in \{1, \dots, T\}, \pi \in \{1, \dots, \Pi\} \tag{41}$$

$$T_\pi^{end} \ge t y_{\pi t} \quad t \in \{1, \dots, T\}, \pi \in \{1, \dots, \Pi\} \tag{42}$$

$$T_{\pi+1}^{start} \ge T_\pi^{end}$$

$$\pi \in \{1, \dots, \Pi\} \tag{43}$$
$$\sum_{i \in \mathcal{D}'(c)} \sum_{l \in \mathcal{L}(c,s)} z_{li} \leq 1$$
$$s \in \mathcal{S} \tag{44}$$
$$\sum_{i \in \mathcal{D}'(c,t)} \sum_{l \in \mathcal{L}(c)} t z_{li} \leq d$$
$$t \in \{1, \dots, T\} \tag{45}$$
$$a_{ij} \in \mathbb{B}$$
$$(i,j) \in \mathcal{A}(c) \tag{46}$$
$$b_m \in \mathbb{N}^+$$
$$m \in \mathcal{M} \tag{47}$$
$$d \in \mathbb{N}^+ \tag{48}$$
$$T_\pi^{start}, T_\pi^{end} \in \mathbb{N}^+$$
$$\pi \in \{1, \dots, \Pi\} \tag{49}$$
$$f_i \in \mathbb{B}$$
$$i \in \mathcal{P}'(c) \tag{50}$$
$$u_i \in \mathbb{R}^+$$
$$i \in \mathcal{N}(c) \tag{51}$$
$$w_{st}^{start}, w_{st}^{end} \in \mathbb{B}$$
$$s \in \mathcal{S}, t \in \{1, \dots, T\} \tag{52}$$
$$y_{\pi t} \in \mathbb{B}$$
$$t \in \{1, \dots, T\}, \pi \in \{1, \dots, \Pi\} \tag{53}$$
$$z_{li} \in \mathbb{B}.$$
$$l \in \mathcal{L}(c), i \in \mathcal{D}'(c) \tag{54}$$

The aim is to minimise the reduced costs (22) as represented by the objective (27).

Constraints (28), (29) and (30) 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 (31)). It is assumed that all connectors start with a full tank (constraint (32)). Constraints (33) and (34) 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 (35) and (36) 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 (38) 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 (37) to avoid deducting the reduced costs of the resource in the objective too much.

To satisfy the priority order, constraints (39)–(43) are imposed and to ensure that there is at most one delivery for each resource set, constraints (44) 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 resource set 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 (45).

Appendix B. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.ejor.2024.08.017.

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).

Breugem, T. (2020). *Crew Planning at Netherlands Railways: Improving Fairness, Attractiveness, and Efficiency* Ph.D. thesis, ERIM PhD Series in Research in Management.

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* Master thesis, Naval Postgraduate School Monterey United States.

Cuesta, E. F., Andersson, H., Fagerholt, K., & Laporte, G. (2017). Vessel routing with pickups and deliveries: an application to the supply of offshore oil platforms. *Computers & Operations Research*, *79*, 140–147.

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

Debusmann Jr., B. (2024). How the US military plans to construct a pier and get food into gaza. *British Broadcasting Corporation (BBC)*, URL https://www.bbc.com/news/world-us-canada-68534370.

Desaulniers, G., Desrosiers, J., & Solomon, M. M. (2002). Accelerating strategies in column generation methods for vehicle routing and crew scheduling problems. In *Essays and surveys in metaheuristics* (pp. 309–324). Springer.

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

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

Gribkovskaia, I., Laporte, G., & Shlopak, A. (2008). A tabu search heuristic for a routing problem arising in servicing of offshore oil and gas platforms. *Journal of the Operational Research Society*, *59*(11), 1449–1459.

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.

Irnich, S., & Desaulniers, G. (2005). Shortest path problems with resource constraints. In *Column generation* (pp. 33–65). Springer.

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*.

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.

Simpson, G. G. (1947). Holarctic mammalian faunas and continental relationships during the cenozoic. *Geological Society of America Bulletin*, *58*(7), 613–688.

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

Szymkiewicz, D. (1934). Une contribution statistique à la géographie floristique. *Acta Societatis Botanicorum Poloniae*, *11*(3), 249–265.

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.

Wagenvoort, M. (2023). *Ship-to-Shore Artificial Instances*. Zenodo, http://dx.doi.org/10.5281/zenodo.10216198.

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.

Zachariadis, E. E., Tarantilis, C. D., & Kiranoudis, C. T. (2016). The vehicle routing problem with simultaneous pick-ups and deliveries and two-dimensional loading constraints. *European Journal of Operational Research*, *251*(2), 369–386.

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.