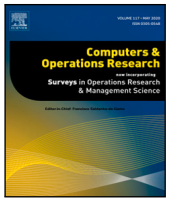
Computers & Operations Research 144 (2022) 105806

Contents lists available at ScienceDirect

Computers and Operations Research

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



The Fish Feed Production Routing Problem

Ivar Brekkå [a], Solveig Randøy [a], Kjetil Fagerholt [a], Kristian Thun [b], Simen Tung Vadseth [a,*]

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

ARTICLE INFO

Keywords:
Maritime logistics
Production routing
Multi-depot
Case study
Heuristic

ABSTRACT

This paper introduces the fish feed production routing problem (FFPRP) faced by Norwegian salmon feed producers. The FFPRP is comprised of a production scheduling problem and a rich vehicle routing problem (VRP) and thus denotes a variant of the integrated production scheduling and vehicle routing problem. We present a discrete time mixed integer programming (MIP) model of the FFPRP. Specifically, the model incorporates a multi-product, multi-trip, and multi-depot setting, where orders are produced at production lines at factories, and delivered within their respective time windows to customers by a heterogeneous fleet of vessels. The main objective is to minimize total costs, including production costs, routing costs, and costs of not delivering orders. We propose a heuristic, combining decomposition and the adaptive large neighborhood search (ALNS) heuristic, to solve the FFPRP. The decomposition-based ALNS heuristic is tested on a number of test instances which are generated based on vessel, factory, and order data from two of Norway's largest fish feed producers. For small problem instances, for which the commercial MIP solver often is able to prove optimality, the average optimality gaps of the proposed heuristic are relatively small, while for the larger problem instances, the heuristic significantly outperforms the commercial MIP solver both in terms of solution quality and run time.

1. Introduction

From 1990 to 2018, the world's total fish consumption increased by 122% (Food and Agriculture Organization of the United Nations, 2020). As global fisheries to a large extent are fully exploited, the supply of wild fish has limited potential to meet the growing demand (Mowi, 2020). The aquaculture industry, on the other hand, has seen vast growth over the last decades. In 2012, the worldwide annual seafood production resulting from the use of fish farms exceeded the production of wild-caught fish (Our World in Data, 2019), and in 2018, the world aquaculture production reached 114.5 million tonnes (Food and Agriculture Organization of the United Nations, 2020). For Atlantic salmon, worldwide production has increased by more than 1000% since 1990 (Mowi, 2020). Norwegian producers contribute with about half of this production (Statistics Norway, 2020).

For Norway's producers and distributors of fish *feed*, the situation is different. These actors are subject to a range of challenges. First, the actors are facing fierce competition, which puts great pressure on margins. Second, the producers' portfolios typically consist of a range of different products, which makes it difficult to find cost-efficient production and inventory plans and corresponding distribution plans. Lastly, the strong bargaining power of the customers results in limited flexibility and additional costs for the producers. Altogether, these challenges serve as motivation for the development of a tool that can support fish feed producers in the planning of their production and distribution.

1.1. Problem description

We consider the fish feed production routing problem (FFPRP), which denotes the problem of finding cost-efficient plans for the production and distribution of fish feed products based on given sets of factories, production lines, products, orders, and vessels. Fish feed *factories* consist of one or several *production lines*, each associated with production capacities for a set of fish feed *products*. Products are produced in batches, where each product is associated with a minimum production quantity for each batch. Starting up the production of a new batch involves additional work and thus extra costs. Each product belongs to a certain *product group*, consisting of products whose production setups are alike. The production machinery at a factory may from time to time be subject to routine maintenance, which involves a full production stop at the factory.

Fish farms – or customers – located at sea along the coast demand fish feed products, and consequently, they place *orders*. Each order is associated with the customer's location and quantities of one or several products, and must be delivered within its corresponding time

* Corresponding author.
E-mail address: simen.t.vadseth@ntnu.no (S.T. Vadseth).

https://doi.org/10.1016/j.cor.2022.105806
Received 30 August 2021; Received in revised form 22 March 2022; Accepted 22 March 2022
Available online 31 March 2022

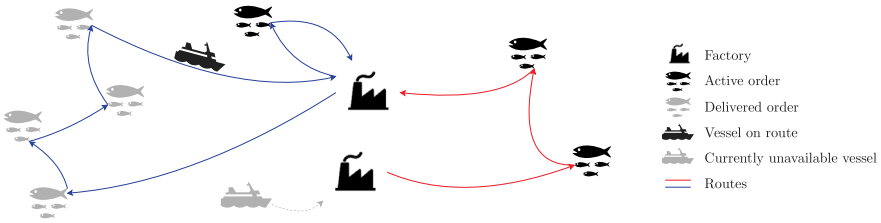


Fig. 1. Illustration of vessel routing.

window. The orders of fish feed products are distributed via sea by a given heterogeneous fleet of *vessels*. Two main types of vessels exist: those transporting orders/products in bags and those transporting in silos. The number of different products on board a vessel of the latter type is limited to the number of silos, as different products cannot be contained within one silo. Regardless of the vessel's type, the load cannot exceed the vessel's weight load capacity. A vessel may be able to handle only some of the orders. As an example, silo ships are not able to deliver orders that are required to be delivered in bags. Also, some fish farms may lack the required equipment or space for a certain vessel to dock or unload, or be located in a shallow fjord where the largest vessels are unable to sail. The orders/products are loaded onto the vessels at the factories. The total number of loading spots at a factory is dynamic, as loading spots sometimes are reserved for vessels delivering raw materials. Raw material delivery times are considered fixed and known prior to the planning. Fish feed products produced, but not yet distributed, are held as inventory. Each factory has a given maximum inventory capacity and is subject to an inventory holding cost.

Each vessel is assigned a *route*. Some vessels are available from the beginning of the planning horizon, while others become available for routing later, as they need to finish their current routes. A vessel always starts and ends its route at a factory (depot). The route's destination factory thus becomes the start position of the corresponding vessel in the subsequent planning period. Since a balanced distribution of vessel start positions is preferable, a maximum number of vessels ending up in the same factory in the end of the planning horizon is imposed. A vessel's route may consist of several *voyages*, meaning that the route may contain intermediate factory visits. Route *activities* include loading of products at factories, unloading of orders at fish farms, sailing between different locations, and waiting to dock. Each vessel is associated with a given sailing speed and must complete its route, that is, arrive at a destination factory, within the planning horizon, which is typically up to two weeks.

Fig. 1 gives an example of the routing of two vessels. The black vessel is currently undertaking the blue route, which consists of two voyages. The gray vessel denotes a currently unavailable vessel. When the vessel becomes available, it may start on its assigned red route, which consists of one voyage. Note that the last visited factory does not have to be the same as the first visited factory, exemplified by the red route.

A visit to a fish farm subject to a disease outbreak may disable certain immediate subsequent visits. Explicitly, a vessel visiting a fish farm with unhealthy fish, a *red* fish farm, cannot visit farms with healthy fish, *green* fish farms, before a *recovery time* has passed. Similarly, a vessel visiting a fish farm with healthy fish located in a *red zone* – that is, in close proximity to unhealthy fish – must let the recovery time pass before it can deliver feeds in areas free of disease. Farms with healthy fish located in red zones are referred to as *yellow* fish farms. All fish farms can again be visited after the required recovery time has passed.

Fig. 2 illustrates the above categorization, using a route including six fish farm visits as an example. The vessel undertaking the route can initially visit all fish farms. The vessel starts by visiting the leftmost fish farms in the order green, yellow, red. Three options exist for the next delivery at a green fish farm, indicated by the dotted arrows. If the sailing time is greater than the recovery time, the vessel can sail directly

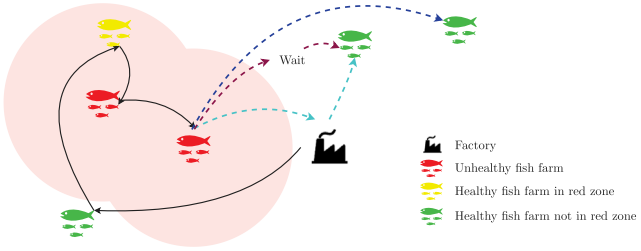


Fig. 2. Illustration of red, yellow, and green fish farms. The dashed arrows indicate the vessel's options when sailing from a red fish farm to a green fish farm.

to the fish farm, indicated by the dark blue arrow. If the sailing time is less than the recovery time, the sailing must be followed by waiting, in order for the vessel to recover before the green fish farm can be visited, indicated by the two purple arrows. As a third option, indicated by the light blue arrows, the vessel can visit the factory, as we assume that this will always give enough time to recover.

In conclusion, a solution to the FFPRP consists of a set of *production schedules*, each assigned to a production line, and a set of *routes*, each assigned to a vessel. The production schedules and the routes must correspond, meaning that the required product quantities must be available at the factory before they can be loaded onto vessels. The FFPRP thus describes the operational problem of short-term production scheduling and vessel routing faced by fish feed producers and distributors. The aim of the FFPRP is to minimize the costs of inventory, transportation, and production setups while serving as many orders as possible.

1.2. Literature review

Although the FFPRP has not previously been studied, its constituents – the production scheduling problem and the vehicle routing problem (VRP) – are well-known within the field of operations research. According to Braekers et al. (2015), VRP models have, however, changed immensely over the last decades, as they increasingly aim to incorporate real-life characteristics and complexities. This has resulted in a range of *extended* VRPs, such as heterogeneous fleet VRP (HFVRP), VRP with time windows (VRPTW) and multi-depot VRP (MDVRP). In the later years, extended VRPs have also to a larger degree been combined to *rich* or *multi-attribute* VRPs (MAVRPs), where several real-life aspects are included simultaneously (Braekers et al., 2015). Although enabling a more accurate representation of reality, rich VRPs may be especially difficult to solve because of the compound, and possibly antagonist, decisions they involve (Vidal et al., 2013).

The routing sub-problem of the FFPRP is concerned with a range of complicating real-life characteristics, such as heterogeneous vessels, delivery time windows, multiple depots, and several depot visits on the same route (multiple trips). In addition, our problem includes *resource synchronization* constraints (Drexl, 2012), as both factory loading spots and the maximum number of vessels having the same factory as final route destination represent resources shared across vessels. For these reasons, we place the problem within the category of rich VRPs.

Most of the VRP attributes contained in the FFPRP are well-described in the operations research literature. Cattaruzza et al. (2018) provide an extensive overview of multi-trip VRPs (MTVRPs), reviewing academic extensions, routing problems combined with production and inventory decisions, and problems within maritime transportation. Several of the reviewed articles are concerned with problems somewhat similar to the FFPRP, including Azi et al. (2014), Adulyasak et al. (2014b), Ullrich (2013), Halvorsen-Weare et al. (2012) and Shyshou et al. (2012). Furthermore, Montoya-Torres et al. (2015) review the literature concerned with multi-depot VRPs. Two of the reviewed articles, Crevier et al. (2007) and Zhen and Zhang (2009), investigate VRPs similar to the routing sub-problem of the FFPRP, incorporating multiple trips, multiple depots, and inter-depot routes, where the latter feature allows vehicles to travel between different depots. However, although the VRP attributes of the FFPRP routing sub-problem seem to be thoroughly researched, they are rarely combined.

Within maritime transportation, several studies consider variants of the VRP that are somewhat similar to the FFPRP. Christiansen et al. (2017) propose a mathematical formulation of a multi-trip VRP used to route fuel supply vessels. The problem considers heterogeneous vessels performing multiple voyages to service customers within their respective time windows. Halvorsen-Weare et al. (2012) also formulate a maritime VRP, referred to as a supply vessel planning problem (SVPP), for the Norwegian oil company Equinor. Specifically, the SVPP considers a set of heterogeneous vessels transporting supplies from an onshore depot to offshore installations. The problem is further studied by Shyshou et al. (2012). Problems formulated specifically for the aquaculture industry are researched by both Agra et al. (2017) and Lianes et al. (2021). The former formulate a maritime inventory routing problem (MIRP) for Norway's largest vertically integrated salmon farmer. The latter study the aquaculture service vessel routing problem (ASVRP), concerned with a set of fish farms requiring services from a heterogeneous fleet of service vessels. Complicating constraints include simultaneous operation of more than one vessel, time windows of service delivery, and precedence requirements for the different services. By this, the ASVRP shares several characteristics with the FFPRP.

Furthermore, we find similar problems designed for the LNG industry. MIRPs for LNG distribution were studied by Grønhaug and Christiansen (2009) and Grønhaug et al. (2010), while the annual delivery program (ADP) problem was first considered by Rakke et al. (2011) and Stålhane et al. (2012). Halvorsen-Weare and Fagerholt (2013) present a simpler version of the ADP problem, which, opposite to the typical MIRP, assumes a predetermined set of orders, and disregards inventory decisions for the customers. Along with the inventory-constrained central depot structure, these characteristics instead represent similarities to the FFPRP. However, the ADP problem denotes a tactical problem, with a typical planning horizon of 12 to 18 months (Mutlu et al., 2016). In contrast, the FFPRP focuses on operational decisions with a planning horizon of up to two weeks.

Both the production scheduling and vehicle routing problem (PS-VRP) and the standard production routing problem (PRP) share characteristics with the FFPRP. The former, described in the review of Moons et al. (2017), considers scheduling problems on an operational level in combination with variants of the VRP. Furthermore, the PS-VRP includes production and inventory decisions similar to those of the FFPRP, where the latter are taken for factories, but not for customers. Thus, the FFPRP may, in fact, be defined as a variant of the PS-VRP incorporating a rich VRP. Both Belo-Filho et al. (2015) and Farahani et al. (2012) look into PS-VRPs.

The standard PRP, as defined by Adulyasak et al. (2014b), integrates two classic optimization problems, the lot-sizing problem and the VRP. However, as opposed to the FFPRP and the PS-VRP, the PRP also takes into account the inventory levels at the customer locations. Several studies on this standard version of the PRP and its benchmark instances have been conducted (Absi et al., 2015; Solyalı and Süral, 2017; Qiu et al., 2018; Li et al., 2019; Manousakis et al., 2022) and the current best results are reported by Vadseth et al. (2022). However, the standard PRP distinguish itself from the FFPRP by using single-trips and having a single depot which all vehicles must return to at the end of each time period. Yet, there exists richer extensions to the standard PRP that share more similarities with the FFPRP. This includes, but is not limited to, the multi-plant perishable food production routing problem with packaging consideration (Li et al., 2020), the multi-trip heterogeneous vehicle routing problem coordinated with production scheduling (Yağmur and Kesen, 2021), and the large multi-product production routing problem with delivery time windows (Neves-Moreira et al., 2018). The latter problem is concerned with a meat processing center with several production lines and a heterogeneous fleet of capacitated vehicles used to deliver the products to stores spread across the country. The problem involves a range of complicating characteristics, many of which are also relevant to the FFPRP such as time windows as well as different production setups for products belonging to different groups. In addition, the rich production routing problem studied by Miranda et al. (2018) also incorporates several relevant characteristics such as time windows, multiple products, a heterogeneous fleet and routes extending over one or more periods.

Both the PS-VRP and the PRP are highly challenging to solve. Furthermore, Moons et al. (2017) argue that, whereas both problems on their own are well-studied separately in the literature, the combination of production scheduling and vehicle routing problems is a rather unexplored research direction. Nevertheless, integration often represents a significant cost advantage, as it enables improvements in the overall resource utilization (Adulyasak et al., 2014b; Chandra and Fisher, 1994).

The authors of the examined literature apply different methods to solve their respective proposed problems. The common factor, however, is the use of heuristics. Braekers et al. (2015) conclude that heuristics and metaheuristics are often more suitable for practical applications of the VRP because real-life problems are considerably large. The same conclusion is drawn by Neves-Moreira et al. (2018), who argue that the dimensions of the PRP make it impossible to be solved exactly by current solution methods. As for the PS-VRP, Moons et al. (2017) conclude that solution methods based on metaheuristics are often applied to find high-quality solutions in reasonable computation time.

To solve their multi-product PRP with time windows, Neves-Moreira et al. (2018) develop a three-phase matheuristic based on a *fix-and-optimize* procedure. The authors argue that their proposed matheuristic is efficient for both the inventory routing problem and the PRP, as it provides new best solutions for several instances within a relatively short run time. Shyshou et al. (2012) apply a large neighborhood search (LNS) heuristic, initially proposed by Shaw (1998), to solve the SVPP. The LNS heuristic is also proposed by Farahani et al. (2012) to solve a variant of the PS-VRP. To solve the ASVRP, Lianes et al. (2021) develop an adaptive large neighborhood search (ALNS) heuristic. The heuristic explores the solution space iteratively by applying different sub-heuristics, chosen based on their historic performance. The authors conclude that the ALNS heuristic yields promising results. An ALNS heuristic is also implemented both by Azi et al. (2014) and Liu et al. (2018). For the latter, experimental results indicate that the ALNS heuristic outperforms existing solution methods, finding new best solutions within less computation time. The ALNS heuristic has also been applied to both the PRP (e.g., Adulyasak et al. (2014a)) and the PS-VRPs (e.g., Belo-Filho et al. (2015)). Lastly, Ullrich (2013) implements a genetic algorithm to solve the integrated machine scheduling and vehicle routing problem.

1.3. Contributions and paper outline

The FFPRP is a real rich integrated routing problem that is new within the field of operations research. Even though the FFPRP shares

characteristics with a range of well-studied problems, such as the extended VRP, the PS-VRP, the PRP, as well as maritime routing problems, the combination of heterogeneous resource-synchronized vessels, time windows, multi-trips, and multi-depots has not previously been investigated. Furthermore, according to Moons et al. (2017), some of the characteristics of the FFPRP are also somewhat unexplored in the context of PS-VRPs. Based on this, our contribution is three-fold. First, we provide a novel mathematical formulation of the FFPRP. Second, we propose an innovative decomposition-based ALNS heuristic to solve the FFPRP. Third, we test the decomposition-based ALNS heuristic on a set of realistic instances based on real data and show that the proposed method provides good solutions and thus has the potential for becoming a valuable decision support tool for the fish feed industry. We also show how it can be used to assess the impact of tactical and strategic decisions.

The remainder of the report is comprised of four sections. Section 2 presents the mathematical model of the FFPRP, before Section 3 describes the decomposition-based ALNS heuristic. The computational study is presented in Section 4, while Section 5 provides the concluding remarks.

2. Mathematical model

This section presents our proposed model formulation of the FFPRP, based on the problem description in Section 1.1. We start in Section 2.1 by discussing our modeling assumptions. Next, Section 2.2 presents the notation, before we define the mathematical model in Section 2.3.

2.1. Modeling assumptions

A vessel route is equivalent to a sequence of *node visits*. An order node visit represents the unloading of an order, which may include quantities of several different products. A factory node visit represents the loading of an order if the visit marks the start of a voyage, or otherwise a route destination. If the same fish farm places multiple orders during the planning horizon, each order is represented by an order node. Consequently, order nodes are only visited once. Factory nodes, on the other hand, may be visited multiple times by multiple vessels. In addition to order nodes and factory nodes, we introduce a *dummy origin node* and a *dummy destination node*. A vessel sails *from* the dummy origin node in the first time period it becomes available for routing. Oppositely, a vessel sails *to* the dummy destination node when it has completed its route.

We develop a discrete time model, where the planning horizon is divided into time periods of equal length. An activity, i.e., loading, unloading, or sailing, is started in one time period and lasts for a given number of time periods. Each vessel becomes available for routing in a given time period, and is then available for the remainder of the planning horizon. We assume that the first and the last activities undertaken by each vessel must be completed within the planning horizon. This implies that two dummy time periods must be added, one before and one after the considered planning horizon, as sailing to or from the dummy node has a duration of one time period.

We consider production cost a sunk cost. The cost of not delivering an order within its delivery time window represents the cost of postponing the order to a later planning horizon, which implies that the total ordered quantity will be produced and delivered eventually, regardless of whether the order is delivered on time or postponed.

We only restrict the total quantity and the total number of different products loaded onto a silo vessel. In reality, however, the quantity loaded onto a vessel is also restricted by the capacity of each silo. This assumption is made in order to avoid further increases in problem complexity. Furthermore, we assume that the time for loading a given vessel is fixed and not dependent on the loading quantity. This is reasonable since the vessels are for most practical cases close to being fully loaded when leaving the factories.

We also assume, based on current practice, that the vessels deliver all of their load during a voyage, and therefore always return empty to a factory.

2.2. Notation

Sets

- $\mathcal{V}$ – Set of vessels
- $\mathcal{N}$ – Set of nodes representing factories or orders
- $\mathcal{N}_v$ – Set of nodes that may be visited by vessel $v \in \mathcal{V}$, $\mathcal{N}_v \subseteq \mathcal{N}$
- $\mathcal{N}^F$ – Set of factory nodes, $\mathcal{N}^F \subset \mathcal{N}$
- $\mathcal{N}_v^F$ – Set of factory nodes that may be visited by vessel $v \in \mathcal{V}$, $\mathcal{N}_v^F \subseteq \mathcal{N}^F$
- $\mathcal{N}^O$ – Set of order nodes, $\mathcal{N}^O \subset \mathcal{N}$
- $\mathcal{N}_v^O$ – Set of order nodes that may be served by vessel $v \in \mathcal{V}$, $\mathcal{N}_v^O \subseteq \mathcal{N}^O$
- $\mathcal{N}^A$ – Set of all nodes, including the dummy origin node and the dummy destination node, $\mathcal{N}^A = \mathcal{N} \cup \{o(v), d(v)\}$
- $\mathcal{A}_v$ – Set of arcs which vessel v may traverse, $\mathcal{A}_v \subset \mathcal{N}^A \times \mathcal{N}^A$
- $\mathcal{A}^W$ – Set of arcs going from red or yellow to green order nodes, as well as arcs going from red to yellow order nodes, $\mathcal{A}^W \subset \mathcal{N}^O \times \mathcal{N}^O$
- $\mathcal{V}^S$ – Set of silo vessels, $\mathcal{V}^S \subseteq \mathcal{V}$
- $\mathcal{V}_i$ – Set of vessels that may visit node $i \in \mathcal{N}, \mathcal{V}_i \subseteq \mathcal{V}$
- $\mathcal{L}$ – Set of production lines at all factories
- $\mathcal{L}_i$ – Set of production lines at factory $i \in \mathcal{N}^F$, $\mathcal{L}_i \subseteq \mathcal{L}$
- $\mathcal{P}$ – Set of products
- $\mathcal{P}_p^G$ – Set of all products within the product group of product $p \in \mathcal{P}$, including p itself, $\mathcal{P}_p^G \subseteq \mathcal{P}$
- $\mathcal{T}$ – Set of time periods
- $\mathcal{T}_i^{TW}$ – Set of time periods within the time window of the order represented by order node $i \in \mathcal{N}^O$, $\mathcal{T}_i^{TW} \subseteq \mathcal{T}$

Parameters

- Q_{lp}^P – Production capacity of product p at production line l, $l \in \mathcal{L}, p \in \mathcal{P}$
- T_{lp}^P – Minimum number of consecutive time periods that production line l must produce product p if it starts production of product p, $l \in \mathcal{L}, p \in \mathcal{P}$
- F_{it} – 1 if factory i can produce in time period t, 0 otherwise, $i \in \mathcal{N}^F, t \in \mathcal{T}$
- T^A – Setup time required when production shifts between products belonging to different product groups
- C_{ip}^S – Setup cost for product p at factory i, $i \in \mathcal{N}^F$, $p \in \mathcal{P}$
- Q_i^I – Inventory capacity at factory i, $i \in \mathcal{N}^F$
- I_{ip}^0 – Initial inventory of product p at factory i, $i \in \mathcal{N}^F, p \in \mathcal{P}$
- C_{ip}^I – Inventory cost for product p at factory i per time period, $i \in \mathcal{N}^F, p \in \mathcal{P}$
- Q_v^V – Vessel v's fish feed weight capacity, $v \in \mathcal{V}$
- M_v^P – Maximum number of different products allowed in silo vessel v, $v \in \mathcal{V}^S$
- D_{ip} – Quantity of product p demanded by order node i, $i \in \mathcal{N}^O, p \in \mathcal{P}$
- T_{vij}^S – Time of sailing from node i to node j for vessel v, $v \in \mathcal{V}, i \in \mathcal{N}_v \cup o(v), j \in \mathcal{N}_v \cup d(v)$
- C_{vij}^T – Cost of sailing from node i to node j for vessel v, $v \in \mathcal{V}, (i,j) \in \mathcal{A}_v$
- M_{it}^V – Maximum number of vessels allowed to load at factory i in time period t, $i \in \mathcal{N}^F, t \in \mathcal{T}$
- M_i^D – Maximum number of vessels which may have factory i as final route destination, $i \in \mathcal{N}^F$
- T_{vij}^W – Required number of waiting time periods before vessel v can visit node j after having sailed from node i, $v \in \mathcal{V}, (i,j) \in \mathcal{A}^W$
- T_{vi}^L – Number of loading or unloading time periods for vessel v at factory or order node i, $v \in \mathcal{V}, i \in \mathcal{N}_v$
- C_i^E – Cost of not delivering the order represented by order node i within the planning horizon, $i \in \mathcal{N}^O$

Decision variables

- g_{lpt} – 1 if production line l produces product p in time period t, 0 otherwise, $i \in \mathcal{N}^F, l \in \mathcal{L}_i, p \in \mathcal{P}, t \in \mathcal{T}$
- a_{lt} – 1 if production line l is not used (is *available*) in time period t, 0 otherwise, $i \in \mathcal{N}^F, l \in \mathcal{L}_i, t \in \mathcal{T}$
- d_{lpt} – 1 if production of product p at production line l starts in time period t, meaning the product was not produced in the previous time period, 0 otherwise, $i \in \mathcal{N}^F, l \in \mathcal{L}_i, p \in \mathcal{P}, t \in \mathcal{T}$
- s_{ipt} – Inventory level of product p at factory i in the end of time period t, $i \in \mathcal{N}^F, p \in \mathcal{P}, t \in \mathcal{T}$
- x_{vijt} – 1 if vessel v starts sailing from node i to node j in time period t, 0 otherwise, $v \in \mathcal{V}, i, j \in \mathcal{N}^A, t \in \mathcal{T}$
- w_{vit} – 1 if vessel v waits to visit node i in time period t, 0 otherwise, $v \in \mathcal{V}, i \in \mathcal{N}_v, t \in \mathcal{T}$
- y_{vit} – 1 if vessel v visits node i in time period t, 0 otherwise. $v \in \mathcal{V}, i \in \mathcal{N}_v, t \in \mathcal{T}$
- z_{vijt} – 1 if vessel v loads the order for order node j at factory node i in time period t, 0 otherwise, $v \in \mathcal{V}, i \in \mathcal{N}_v^F, j \in \mathcal{N}_v^O, t \in \mathcal{T}$
- l_{vpt} – Vessel v's load of product p in the end of time period t, $v \in \mathcal{V}, p \in \mathcal{P}, t \in \mathcal{T}$
- h_{vpt} – 1 if silo vessel v carries product p in time period t, 0 otherwise, $v \in \mathcal{V}^S, p \in \mathcal{P}, t \in \mathcal{T}$
- e_i – 1 if order node i is not delivered within the planning horizon, 0 otherwise, $i \in \mathcal{N}^O$

2.3. Mathematical model

Using the notation presented in Section 2.2, we formulate the mathematical model of the FFPRP as follows.

$$\min z = \sum_{i\in\mathcal{N}^F}\sum_{l\in\mathcal{L}_i}\sum_{p\in\mathcal{P}}\sum_{t\in\mathcal{T}} C^S_{ip} d_{lpt} + \sum_{i\in\mathcal{N}^F}\sum_{p\in\mathcal{P}}\sum_{t\in\mathcal{T}} C^I_{ip} s_{ipt} + \sum_{v\in\mathcal{V}}\sum_{(i,j)\in\mathcal{A}_v}\sum_{t\in\mathcal{T}} C^T_{vij} x_{vijt} + \sum_{i\in\mathcal{N}^O} C^E_i e_i \tag{1}$$

subject to

$$g_{lpt} \le F_{it}, \quad i \in \mathcal{N}^F, l \in \mathcal{L}_i, p \in \mathcal{P}, t \in \mathcal{T} \tag{2}$$

$$g_{lpt} - g_{lp(t-1)} \le d_{lpt}, \quad i \in \mathcal{N}^F, l \in \mathcal{L}_i, p \in \mathcal{P}, t \in \mathcal{T}\backslash\{0\} \tag{3}$$

$$d_{lp0} = g_{lp0}, \quad i \in \mathcal{N}^F, l \in \mathcal{L}_i, p \in \mathcal{P} \tag{4}$$

$$T^P_{lp} d_{lpt} \le \sum_{\tau=t}^{t+T^P_{lp}-1} g_{lpt}, \quad i \in \mathcal{N}^F, l \in \mathcal{L}_i, p \in \mathcal{P}, t \in \mathcal{T} \tag{5}$$

$$a_{lt} + \sum_{p\in\mathcal{P}} g_{lpt} = 1, \quad i \in \mathcal{N}^F, l \in \mathcal{L}_i, t \in \mathcal{T} \tag{6}$$

$$g_{lp(t-1)} \le \frac{1}{T^A}\sum_{\tau=t}^{t+T^A-1} a_{l\tau} + \sum_{q\in\mathcal{P}^G_p} g_{lqt}, \quad i \in \mathcal{N}^F, l \in \mathcal{L}_i, p \in \mathcal{P}, t \in \mathcal{T}\backslash\{0\} \tag{7}$$

$$s_{ip0} = I^0_{ip} - \sum_{v\in\mathcal{V}_i}\sum_{j\in\mathcal{N}^O_v} D_{jp} z_{vij0}, \quad i \in \mathcal{N}^F, p \in \mathcal{P} \tag{8}$$

$$s_{ipt} = s_{ip(t-1)} - \sum_{v\in\mathcal{V}_i}\sum_{j\in\mathcal{N}^O_v} D_{jp} z_{vijt} + \sum_{l\in\mathcal{L}_i} Q^P_{lp} g_{lp(t-1)}, \quad i \in \mathcal{N}^F, p \in \mathcal{P}, t \in \mathcal{T}\backslash\{0\} \tag{9}$$

$$\sum_{p\in\mathcal{P}} s_{ipt} \le Q^I_i, \quad i \in \mathcal{N}^F, t \in \mathcal{T} \tag{10}$$

$$x_{vo(v)f(v)(t^0(v)-1)} = 1, \quad v \in \mathcal{V} \tag{11}$$

$$\sum_{j\in\mathcal{N}}\sum_{t\in\mathcal{T}} x_{vo(v)jt} = 1, \quad v \in \mathcal{V} \tag{12}$$

$$\sum_{(i,j)\in\mathcal{A}_v}\sum_{\tau=t-T^S_{vij}+1}^{t} x_{vij\tau} + \sum_{i\in\mathcal{N}_v}\sum_{\tau=t-T^L_{vi}+1}^{t} y_{vi\tau} + \sum_{i\in\mathcal{N}_v} w_{vit} = 1, \quad v \in \mathcal{V}, t \in \mathcal{T} \tag{13}$$

$$\sum_{v\in\mathcal{V}_i}\sum_{t\in\mathcal{T}^{TW}_i} y_{vit} + e_i = 1, \quad i \in \mathcal{N}^O \tag{14}$$

$$y_{vi(t-T^L_{vi})} = \sum_{j\in\mathcal{N}_v} x_{vijt}, \quad v \in \mathcal{V}, i \in \mathcal{N}_v, t \in \mathcal{T} \tag{15}$$

$$\sum_{j\in\mathcal{N}_v\cup\{o(v)\}} x_{vji(t-T^S_{vji})} + w_{vi(t-1)} = y_{vit} + w_{vit} + x_{vid(v)t}, \quad v \in \mathcal{V}, i \in \mathcal{N}_v, t \in \mathcal{T} \tag{16}$$

$$T^W_{vij} x_{vij(t-T^S_{vij})} \le \sum_{\tau=t}^{t+T^W_{vij}-1} w_{vj\tau}, \quad v \in \mathcal{V}, (i,j) \in \mathcal{A}^W, t \in \mathcal{T} \Big| T^W_{vij} > 0 \tag{17}$$

$$z_{vijt} \le y_{vit}, \quad v \in \mathcal{V}, i \in \mathcal{N}_v^F, j \in \mathcal{N}_v^O, t \in \mathcal{T} \tag{18}$$

$$\sum_{v\in\mathcal{V}_i}\sum_{\tau=t-T^L_{vi}+1}^{t} y_{vi\tau} \le M^V_{it}, \quad i \in \mathcal{N}^F, t \in \mathcal{T} \tag{19}$$

$$l_{vpt^0(v)} = \sum_{i\in\mathcal{N}_v^F}\sum_{j\in\mathcal{N}_v^O} D_{jp} z_{vijt^0(v)}, \quad v \in \mathcal{V}, p \in \mathcal{P} \tag{20}$$

$$l_{vpt} = l_{vp(t-1)} + \sum_{j\in\mathcal{N}_v^O}\left(\sum_{i\in\mathcal{N}_v^F} D_{jp} z_{vijt} - D_{jp} y_{vjt}\right), \quad v \in \mathcal{V}, p \in \mathcal{P}, t \in \mathcal{T} \Big| t > t^0(v) \tag{21}$$

$$\sum_{p\in\mathcal{P}} l_{vp(t-1)} \le Q^V_v\left(1 - \sum_{i\in\mathcal{N}_v^F} y_{vit}\right), \quad v \in \mathcal{V}, t \in \mathcal{T}\backslash\{0\} \Big| t > t^0(v) \tag{22}$$

$$l_{vpt} \le Q^V_v h_{vpt}, \quad v \in \mathcal{V}^S, p \in \mathcal{P}, t \in \mathcal{T} \tag{23}$$

$$\sum_{p\in\mathcal{P}} h_{vpt} \le M^P_v, \quad v \in \mathcal{V}^S, t \in \mathcal{T} \tag{24}$$

$$\sum_{i\in\mathcal{N}_v^F}\sum_{t\in\mathcal{T}} x_{vid(v)t} = 1, \quad v \in \mathcal{V} \tag{25}$$

$$\sum_{v\in\mathcal{V}}\sum_{t\in\mathcal{T}} x_{vid(v)t} \le M^D_i, \quad i \in \mathcal{N}^F \tag{26}$$

$$g_{lpt} \in \{0,1\}, \quad i \in \mathcal{N}^F, l \in \mathcal{L}_i, p \in \mathcal{P}, t \in \mathcal{T} \tag{27}$$

$$a_{lt} \in \{0,1\}, \quad i \in \mathcal{N}^F, l \in \mathcal{L}_i, t \in \mathcal{T} \tag{28}$$

$$d_{lpt} \in \{0,1\}, \quad i \in \mathcal{N}^F, l \in \mathcal{L}_i, p \in \mathcal{P}, t \in \mathcal{T} \tag{29}$$

$$s_{ipt} \ge 0, \quad i \in \mathcal{N}^F, p \in \mathcal{P}, t \in \mathcal{T} \tag{30}$$

$$x_{vijt} \in \{0,1\}, \quad v \in \mathcal{V}, i, j \in \mathcal{N}^A, t \in \mathcal{T} \tag{31}$$

$$w_{vit} \in \{0,1\}, \quad v \in \mathcal{V}, i \in \mathcal{N}_v, t \in \mathcal{T} \tag{32}$$

$$y_{vit} \in \{0,1\}, \quad v \in \mathcal{V}, i \in \mathcal{N}_v, t \in \mathcal{T} \tag{33}$$

$$z_{vijt} \in \{0,1\}, \quad v \in \mathcal{V}, i \in \mathcal{N}_v^F, j \in \mathcal{N}_v^O, t \in \mathcal{T} \tag{34}$$

$$l_{vpt} \geq 0, \quad v \in \mathcal{V}, p \in \mathcal{P}, t \in \mathcal{T} \tag{35}$$

$$h_{vpt} \in \{0,1\}, v \in \mathcal{V}^S, p \in \mathcal{P}, t \in \mathcal{T} \tag{36}$$

$$e_i \in \{0,1\}, \quad i \in \mathcal{N}^O \tag{37}$$

The objective function (1) minimizes the total cost, which is comprised of four parts: production setup costs, inventory costs, transportation costs, and costs of orders not delivered within the planning horizon.

Constraints (2)–(10) define the production sub-problem of the FFPRP. Constraints (2) ensure that production only takes place when the factory does not have production stops. Constraints (3) activate the binary production setup variable used in the objective function. Constraints (4) denote a special variant of this constraint, imposed only on the first time period. Constraints (5) ensure sufficient batch sizes by requiring production to maintain for a given number of consecutive time periods. A production line must either produce exactly one product in the same time period or not be used, as defined by constraints (6). Constraints (7) impose production stop requirements, when production is shifted between products belonging to different product groups. Constraints (8), (9), and (10) constitute the inventory constraints. The first and second group of constraints define the inventory level after the first time period and the remaining time periods, respectively. The third ensure that the inventory capacity is not exceeded.

Constraints (11)–(26) define the routing sub-problem of the FFPRP. Constraints (11) ensure that a vessel starts its route by sailing from the dummy origin node to its origin factory, which, by constraints (12), occurs only once. In each time period, a vessel either sails, unloads, loads, or waits, as defined by constraints (13). By constraints (14), each order must either be delivered within its time window, or be considered not delivered. After the vessel has finished unloading or loading, it must sail to its next node, as expressed by constraints (15). Similarly, after having sailed from one order or factory node to another, the vessel can either start loading or unloading, wait or sail to the dummy destination node. If the vessel was waiting in the previous time period, it can either continue waiting, start loading or unloading or sail to the dummy destination node. This is enforced by constraints (16). Constraints (17) ensure that the order node visits meet the recovery time requirements imposed when fish farms are subject to disease outbreaks. Constraints (18) ensure that each order is picked up at a visited factory, whereas the resource synchronization constraints (19) limit the number of simultaneous vessel visits at a factory. Constraints (20) and (21) define a vessel's load in the time period it becomes available and in succeeding time periods, respectively. Constraints (22) limit total vessel load, and ensure that vessels are empty when returning to a factory. Constraints (23) and (24) together restrict the number of different products loaded onto a silo vessel to not exceed the number of silos. By constraints (25), a vessel ends its route by sailing from a factory node to the dummy destination node. Constraints (26) limit the number of vessels ending up at the same factory.

Constraints (27)–(37) impose binary and non-negativity restrictions on the decision variables.

Lastly, note that in order to improve the model's readability, some constraints include variables of non-defined time periods. This is the case for the sums in constraints (5), (7), (13), (16), (17), and (19). Here, the constraints are meant to only sum over variables that are defined. Similarly, in constraints (15), (16), and (17), we only include variables whose t index is defined. Furthermore, the set $\mathcal{A}^W$ may include arcs (i,j) which cannot be traversed by vessel $v \in \mathcal{V}$. In this case, the corresponding variable $w_{vj\tau}$ and parameter T^S_{vij} used in constraints (17) may not be defined. Therefore, we only impose the constraints where all variables and parameters are defined.

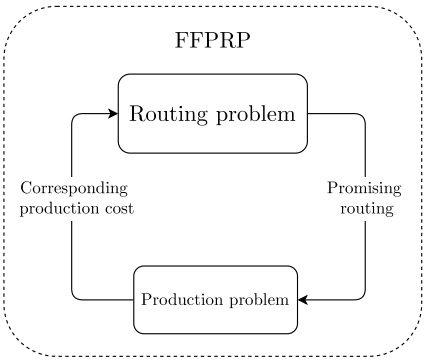


Fig. 3. FFPRP decomposition.

3. A decomposition-based adaptive large neighborhood search heuristic

In order to solve the fish feed production routing problem (FFPRP), we develop and apply a heuristic inspired by the adaptive large neighborhood search (ALNS) heuristic, originally proposed by Røpke and Pisinger (2006). Our proposed method shares several similarities with the ALNS heuristics by Liu et al. (2018) and Lianes et al. (2021), and a similar solution representation is used for the routing part of the problem. However, unlike the aforementioned papers the FFPRP also includes production decisions. Hence, we decompose the problem into a production and routing subproblem, in similar fashion to the ALNS heuristic proposed by Adulyasak et al. (2014a). This section aims to describe the resulting decomposition-based ALNS heuristic, often referred to as simply an *ALNS heuristic*. We start in Section 3.1 by discussing the decomposition of the FFPRP into a production sub-problem and a routing sub-problem. Section 3.2 proceeds to describe the solution representation, whereas Section 3.3 describes the process of constructing an initial solution. Lastly, Section 3.4 presents the ALNS heuristic in more detail and Section 3.5 the derivation of the final solution.

3.1. Problem decomposition

The FFPRP can be decomposed into two problems: the *production sub-problem* and the *routing sub-problem*. The former is concerned with the production scheduling of the factories' production lines as well as inventory management. This sub-problem amounts to constraints (2)–(10) in the model formulation presented in Section 2.3. The routing sub-problem, on the other hand, is concerned with vessel routes and loads, and is defined by constraints (11)–(26).

The two sub-problems are closely connected. From the perspective of the vessels, orders cannot be loaded before the loading quantity is available at the factory. Oppositely, as seen from the factories, the loading quantity must be ready at the time loading starts. For the real-life problem, sailing is the main cost driver, rather than inventory or production setup costs (note that production unit costs are considered sunk). Hence, discovering good *routing* solutions is critical in order to find cost-minimizing solutions to the FFPRP. However, a good routing solution is not necessarily production feasible.

The proposed heuristic therefore focuses on optimizing the sailing routes. Fig. 3 illustrates the interaction between the two sub-problems, where the heuristic searches for promising routing solutions, while the production sub-problem is solved only for these promising routing solutions, in order to estimate the production cost and ensure feasibility.

Algorithm 1 further explains the ALNS heuristic. We use x and y to denote solutions to the routing sub-problem and production sub-problem, respectively. Furthermore, we let $f(x)$ and $g(y)$ denote the

costs of solutions x and y, and $h(x, y)$ the cost of the full FFPRP master problem solution, that is, $h(x, y) = f(x) + g(y)$. The best solution to the FFPRP master problem is denoted $(x^* \quad y^*)$. We start by finding a solution to the routing problem that is also production feasible. We then find a new candidate solution x' to the routing sub-problem, using the ALNS framework. Next, we perform a quick, simplified production feasibility check, and, if needed, make slight adjustments to the routes. The routing cost of the new solution x', $f(x')$, is then compared to the routing cost of the previous best solution, $f(x^*)$. Initially, we only evaluate the routing solution's objective value, in order to save computation time. If the new routing solution x' is promising, meaning the deviation from the routing solution in the currently best solution to the full master problem is less than the *production solve parameter* ϵ, we solve the production sub-problem using a greedy insertion heuristic in order to check for feasibility and obtain an estimate of the production cost $g(y')$. If the production sub-problem is feasible and the new solution to the FFPRP constitutes an improvement, that is, $h(x', y') < h(x^*, y^*)$, we update both x^* and y^*. The algorithm runs until some stopping criterion is met.

Algorithm 1: Outline of ALNS Heuristic for the FFPRP Master Problem

```
Result: solution to the FFPRP
(x*  y*) = best solution to the FFPRP master problem, initially set to solution found by the construction heuristic
x = current routing solution, initially x = x*
while stopping criterion is not met do
    with the current routing solution x as a starting point, generate a new routing solution x' using the ALNS framework
    while solution x' is infeasible by the simplified production feasibility check do
        remove orders from solution x'
    end
    if routing solution x' is a promising routing solution, f(x') < f(x*) · (1 + ε), then
        y' = heuristic production solution corresponding to routing solution x'
        if y' exists and h(x', y') < h(x*, y*) then
            x* = x'
            y* = y'
        end
    end
    if routing solution x' is accepted by some criterion then
        x = x'
    end
end
improve y* with an exact solver
return the best found solution to the FFPRP master problem, (x*  y*)
```

To summarize, we decompose the FFPRP into two sub-problems and solve the problems iteratively using the ALNS framework. Since the routing represents the major cost driver, we primarily evaluate the solutions in terms of routing cost, whereas production costs are evaluated only for promising routing solutions. The integration of the sub-problems is therefore concentrated on the feasibility of the master problem's solution. Note that Algorithm 1 is presented at a high level focusing on the integration of the two sub-problems. Therefore, the following sections aim to present the full heuristic in more detail.

3.2. Solution representation

Each routing solution x consists of a sequence of node visits for each vessel, referred to as *routes*, and a sequence of vessel visits for each factory. We let $i(s)$ denote the sth node visit of a vessel, and $v(u)$ the vessel corresponding to the uth vessel visit at a factory. Furthermore, we keep track of the earliest and latest possible starting times for each node visit, denoted e_s and l_s, respectively. The solution representation for node visits is exemplified in Fig. 4.

Several vessels may be scheduled to visit the same factory, and the visits' respective start time intervals may be overlapping. However, a factory can only serve a limited number of vessels simultaneously. In the case of overlapping start time intervals for visits to the same factory, the order of the vessel visits is not trivial. We therefore also include the vessel visit order at each factory in the solution representation.

Each production solution y contains the production activities performed on each production line for each time step. We let A_l denote the sequence of activities at production line l, and $a_l(t)$ the production activity – that is, the product produced – in time period t. Eventually, $a_l(t)$ is set to *null* if production does not take place at production line l in time period t. Fig. 5 gives an example of production activities at a factory consisting of two production lines.

3.3. Construction of an initial solution

We construct an initial solution by first adding the vessels' initial factories to the routes. Next, we insert order nodes into the routes, based on the *utilities* of the insertions. The process of constructing an initial solution is described in detail in the following sections.

3.3.1. Overview of the construction heuristic

We build an initial solution by iteratively inserting order nodes in the vessels' routes. The prioritization of the insertions are based on 2-regret values, denoted R^2, given by the formula

$$R^2 = U_i^1 - U_i^2, \tag{38}$$

as preliminary tests showed that this procedure yielded satisfactory initial solutions. Here, U_i^h is the insertion utility of inserting node i in its hth best position. For each iteration, we do the insertion corresponding to the greatest R^2 value, whose resulting routes are proven to be feasible. We calculate the utility of each insertion before we evaluate feasibility, as the feasibility check is relatively computationally expensive. The process is continued until all orders are served, or until no routing-feasible insertion exists.

Next, we ensure that the routing solution results in a feasible production schedule, using a greedy insertion heuristic. If the constructed solution is not proven to be production feasible, a small proportion of the most cost-contributing order nodes are removed from voyages starting in factories whose production schedules were regarded infeasible. The production sub-problem is then re-solved in order to determine if the new routing solution has a feasible production counterpart. The process of solving the production sub-problem and removing orders continues until a feasible initial solution to the FFPRP is found. Lastly, we set the best found solution to be the solution generated by the construction heuristic.

3.3.2. Route insertion utility

An *insertion* is defined by an order node j, a vessel v and an insert position s, and compared to other insertions in terms of cost contributions. Specifically, we evaluate the net change in transport cost and penalty for not delivering the order, referred to as the insertion's *utility*. The higher the utility, the more promising is the insertion. As in Section 3.2, we let $i(s)$ denote the node j at position s in a given vessel route. Furthermore, C^T_{vik} denotes vessel v's cost of transportation from node i to node k. The resulting net change in transport cost when inserting order node j in position s of the route of vessel v, ΔC^T_{vsj}, is given by the equation

$$\Delta C^T_{vsj} = \begin{cases} C^T_{vi(s-1)j} + C^T_{vji(s+1)} - 0, \\ \quad \text{if } j \text{ is inserted at the end of the route} \\ C^T_{vi(s-1)j} + C^T_{vji(s+1)} - C^T_{vi(s-1)i(s+1)}, \qquad \text{otherwise.} \end{cases} \tag{39}$$

Note that we in Eq. (39) add two terms also when an order is inserted at the end of a route. The reason why is the requirement that

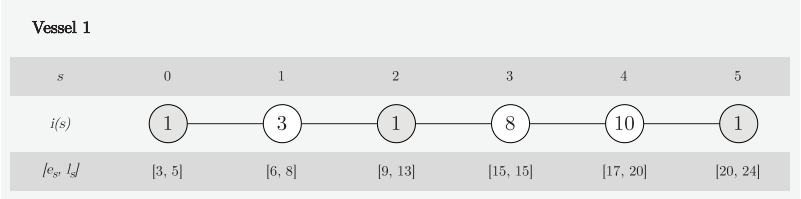


Fig. 4. Example of node visits for vessel 1. White nodes denote order visits and gray nodes factory visits.

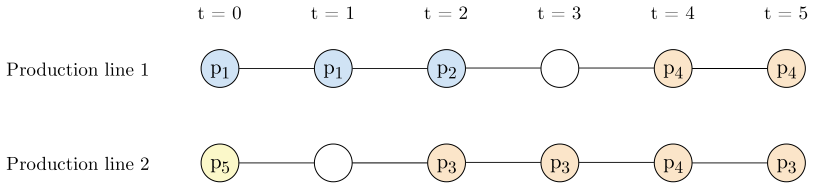
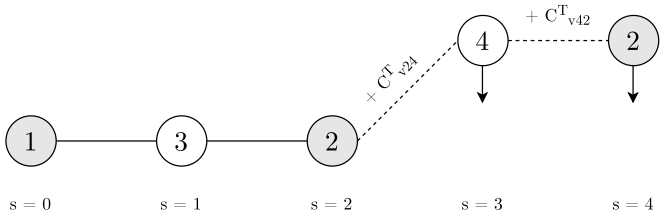
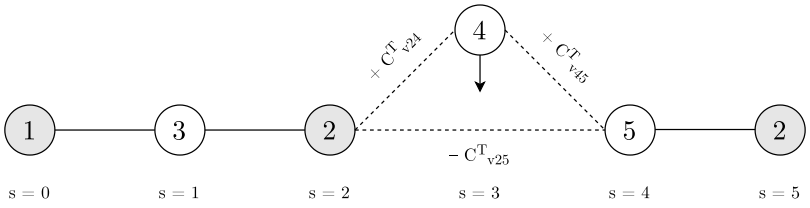


Fig. 5. Example of production activities at a factory. Nodes denote production activities. White nodes represent time periods without production. The remaining colors represent different product groups.



(a) Order node 4 is inserted at the end of the route. A succeeding factory node is also inserted, as all routes are required to have a factory destination.



(b) Order node 4 is inserted in the middle of the route.

Fig. 6. Change in transport cost when inserting an order node at the end of a route 6(a) and in the middle of a route 6(b). White nodes denote order visits and gray nodes factory visits. Dotted lines represent route changes whose costs must be taken into account.

all routes must end in a factory. That is, inserting an order node at the end of a route implies that a factory node must also be inserted. Fig. 6 illustrates the calculation of transport cost change ΔC^T_{vsj} when order node $j = 4$ is inserted at position $s = 3$ in the route of a vessel v.

We define the insertion *utility* as the penalty for not serving the order minus the net change in transport cost when inserting the order node in a given position in a vessel's route. The utility of inserting order node j in position s of vessel v's route, U_{vsj}, is thus given by the equation

$$U_{vsj} = C^E_j - \Delta C^T_{vsj}, \tag{40}$$

where C^E_j is the cost of not serving order j and ΔC^T_{vsj} is the resulting change in transport cost defined by Eq. (39).

3.3.3. Route insertion feasibility

When order nodes are removed from a feasible routing solution, the resulting routing solution will always be feasible. Oppositely, when inserting an order node, as illustrated in Fig. 6, several constraints may cause the routing solution to become infeasible. We, therefore, assert route feasibility before *every* order node insertion, ensuring that only feasible insertions are performed.

As mentioned in Section 1.1, routes are required to start and end in a factory node. A route may also consist of multiple voyages, meaning intermediate factory visits. As the ALNS heuristic only considers insertions of order nodes, factory nodes are automatically inserted when required. We argue that this design allows for a smooth traversal of the search space of multiple voyages and different destination factories. Fig. 6(a) shows an example of an order node being inserted at the end of a route, and thereby forming a new voyage. The order insertion is only allowed if also a new factory node may be placed after the insertion. In other words, the insertion must be time feasible, and must not make more than M^D_i vessels end their routes in factory node i. If several factory nodes meet these criteria, the insertion yielding the highest utility U'_{vsj} according to the definition $U'_{vsj} = -C^T_{vi(s-1)j}$ is chosen. The notation is similar to the notation in Eq. (40), except that j denotes a factory node.

Before we insert an order node, we must ensure that the resulting route is time feasible. Specifically, using the notation presented in

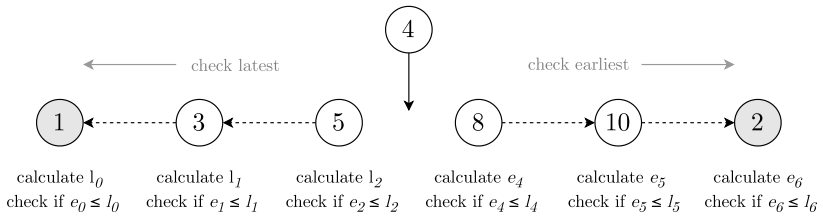


Fig. 7. Intra-route propagation of time feasibility check. The example illustrates the calculations required to check the time feasibility for the insertion of node 4 into the route. Factories are represented by gray nodes.

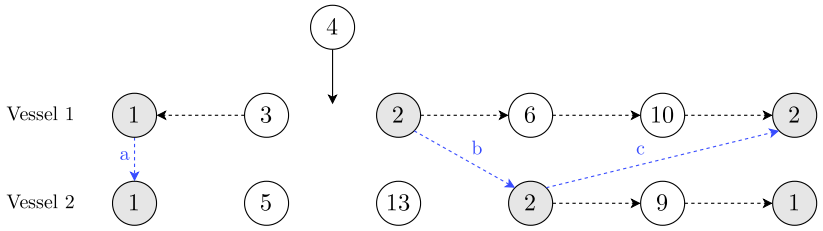


Fig. 8. Inter- and intra-route propagation of time feasibility check. The black dashed arrows denote intra-route propagation, whereas the blue dashed arrows denote inter-route propagation. The labels a, b and c correspond to the respective labels in Fig. 9.

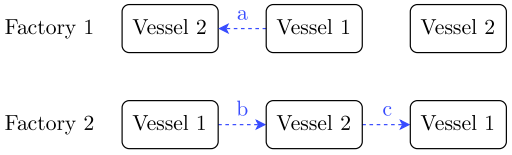


Fig. 9. Vessel visits at two factories, corresponding to the routes in Fig. 8.

Section 3.2, we ensure that there is enough time between the node visit at position $s-1$ and the node visit pushed to position $s+1$ as a result of the insertion into position s. As before, we refer to the order i inserted into a route in position s as $i(s)$. The earliest time for $i(s)$, e_s, is dependent on the earliest time for the previous visit, e_{s-1}, the loading or unloading time for the previous node, $T^L_{vi(s-1)}$, and the sailing time from $i(s-1)$ to $i(s)$, $T^S_{vi(s-1)i(s)}$. The calculation is given by Eq. (41). Similarly, the latest time for $i(s)$, given by Eq. (42), depends on the latest time for the next visit, l_{s+1}, the loading or unloading time for $i(s)$, $T^L_{vi(s)}$, and the sailing time from $i(s)$ to $i(s+1)$, $T^S_{vi(s)i(s+1)}$. Further, if $i(s)$ is an order node, e_s and l_s may also be constrained by the earliest and latest time periods of the order's time window. These are denoted $E_{i(s)}$ and $L_{i(s)}$, respectively.

$$e_s = \max\left\{E_{i(s)}, e_{s-1} + T^L_{vi(s-1)} + T^S_{vi(s-1)i(s)}\right\} \tag{41}$$

$$l_s = \min\left\{L_{i(s)}, l_{s+1} - T^S_{vi(s)i(s+1)} - T^L_{vi(s)}\right\} \tag{42}$$

If $e_s > l_s$, the insertion is infeasible. However, the insertion may still not be feasible even if $e_s \le l_s$, as preceding or succeeding visits s' may have been pushed to have $e_{s'} > l_{s'}$. Also, the insertion may result in changes to the earliest or latest time for a factory visit. These changes may again affect other routes visiting the same factory, possibly causing time infeasibility. These feasibility checks will be elaborated on next.

The intra-route propagation of the time feasibility check is illustrated in Fig. 7. The latest times are recalculated and checked for nodes *preceding* the insertion, while the earliest times are recalculated and checked for nodes *succeeding* the insertion. If the recalculated e or l is equal to its previous value, the propagation is stopped, as no new values will be obtained from further propagation.

The interdependencies between routes visiting the same factories are demonstrated in Fig. 8, where gray nodes denote factories. Fig. 9 shows the corresponding order of the factories' vessel visits. As vessel 1 visits factory 2 before vessel 2, the earliest time for vessel 2 to visit this factory may be affected by the insertion, illustrated by arrow b. If the earliest time for vessel 2 at factory 2 is changed, this would require further recalculations and checks for succeeding visits on vessel 2's route, as well as for vessel 1's last visit at factory 2 (arrow c). As vessel 2 visits factory 1 before vessel 1, vessel 2's latest time for this visit may be affected by the updated latest time for vessel 1's visit at factory 1 (arrow a). Note that such interdependencies may result in a visit getting several updates for e_s or l_s. In Fig. 8, this is the case for vessel 1's last visit at factory 2. In such cases, the most constraining dependency is kept. Formally, this means $e_s = \max\{e^{new}_s, e^{old}_s\}$ and $l_s = \min\{l^{new}_s, l^{old}_s\}$.

As the total number of loading spots – that is, the maximum number of vessels loading simultaneously at a factory – is dynamic and may be higher than one, additional computations are required in order to determine the earliest and latest times for factory node visits. These computations assume a fixed vessel visit order, and iterate forward in time until there is an available loading spot for the vessel. The relevant time periods to check are time periods after the departures and time periods in which the total number of loading spots increases. For the latest time period, a similar procedure is implemented.

Furthermore, for each insertion, we ensure that the inserted order node is compatible with the relevant vessel, and that the total weight and the number of different products loaded onto the vessel respect the vessel's capacity. We assume that the vessels are empty at the start of each voyage. Checking for feasibility therefore simply implies summing the weights and counting the number of different products delivered on the voyage in which the order is inserted, and ensuring that the vessel's capacity is respected.

3.3.4. Production heuristic

We perform a production feasibility check and estimate production cost whenever a promising routing solution is found. That is, the best routing solution x^* is only updated if we find a corresponding feasible production schedule. By greedily inserting production activities into the production lines' respective schedules, the heuristic aims to construct a feasible solution to the production sub-problem.

In order to assert production feasibility at a given factory, we introduce the parameter $I_{pt'}$, denoting the inventory level of a product p at the start of time period $t' \in \mathcal{T}'$. Here, $\mathcal{T}'$ contains the time periods in which one or several vessels load at the given factory. From the perspective of the factory, these time periods represent *production deadlines*, as they denote the time periods in which the demanded orders must be ready for loading.

Algorithm 2 describes the process of greedily constructing production schedules. Here, an insertion is defined by a product p, a production line l, and a production time period t. An insertion into the production schedule of production line l results in a change in the corresponding production activity sequence A_l. Initially, the inventory level $I_{pt'}$ is equal to the initial inventory given as input to the problem, adjusted for the quantity of product p loaded in time periods prior to t'. The resulting values may be negative, as they represent inventory levels when no production occurs. Note that we construct production schedules separately for each factory, as the production sub-problems for the different factories are disjoint. That is, by assuming fixed loading times and quantities, the factories are independent. For this reason, Algorithm 2 omits factory indices.

Each insertion is associated with an *insertion cost*, denoted C_{lpt}. We define insertion cost according to the equation

$$C_{lpt} = \begin{cases} (t' - t) \cdot C_p^I \cdot Q_{lp}^P + C_p^S, & \text{if } a_l(t-1) \neq p \text{ and } a_l(t+1) \neq p \\ (t' - t) \cdot C_p^I \cdot Q_{lp}^P, & \text{otherwise,} \end{cases} \tag{43}$$

where t' is the production deadline of the produced quantity, C_p^I is the inventory holding cost for product p per time period, Q_{lp}^P is the production capacity for product p at production line l and C_p^S is the cost of starting up production of product p.

Algorithm 2: Greedy Production Schedule Construction Heuristic

Result: solution to the production sub-problem, y'
y' = solution to the production sub-problem, initially empty
for *each factory* **do**
  $\mathcal{T}'$ = set of production deadline time periods
  $I_{pt'}$ = inventory of product p at production deadline time period t'
  A_l = sequence of production activities $a_l(t)$ for production line l, initially $a_l(t) = null$ for all time periods t
  for *production deadline time period* t' *in* $\mathcal{T}'$ *in ascending order* **do**
    while *inventory,* $I_{pt'}$*, is less than quantity to be loaded of product* p *in time period* t' **do**
      $\mathcal{N}$ = set of production activity insertion candidates whose corresponding production deadline is time period t'
      if $\mathcal{N}$ *is empty* **then**
        return *null* # no feasible solution found
      end
      from $\mathcal{N}$, pick and perform the insertion, defined by (l, p, t), with the lowest insertion cost
      add production line l's capacity for product p to $I_{pt'}$
      in A_l, set $a_l(t) = p$
    end
  end
  add the production schedule A_l for each production line l to y'
end
return solution y' to the production sub-problem

The set of activity insertion candidates, denoted $\mathcal{N}$ in Algorithm 2 line 8, consists of feasible insertions into the production schedules of the given factory's production lines. We perform five checks to assert feasibility for an insertion. First, the production line must not already be producing in the given time period. Second, the production schedule resulting from the insertion must satisfy the minimum batch size requirement, meaning that additional activity insertions for the same product may have to be performed. Third, the new solution must comply with the requirements regarding factory production stops, and fourth, production stops between production activities of products belonging to different groups. Lastly, the new solution must not violate the maximum inventory constraint.

The greedy production heuristic solves the production sub-problem significantly faster than exact methods. However, it may fail to find feasible solutions in cases where feasible solutions do exist. Furthermore, as proposed by Neves-Moreira et al. (2018) and Farahani et al. (2012), we fix the vessels' loading times found in the routing solution when solving the production sub-problem. This simplification contributes largely to increase computation efficiency of the production sub-problem, but comes with two disadvantages. First, the cost of the estimated production problem may be somewhat higher than necessary, as some products may be held as inventory slightly longer. Second, some production schedules may be regarded as inventory infeasible, although simply advancing the loading would give the opposite result. This would be problematic for inventory-intensive problem instances. However, we argue that our investigated instances never will be inventory-intensive, as inventory capacity is commonly increased when needed, by renting external depots. For this reason, we argue that our choice of using the factory visits' latest loading times to assert production feasibility is justified.

3.4. Decomposition-based adaptive large neighborhood search heuristic

In Section 3.1, we discussed the decomposition of the FFPRP into a routing sub-problem and a production sub-problem and provided an outline of the master problem heuristic. The focus was sub-problem integration, and thus technical aspects of the ALNS framework were disregarded. In this section, however, we focus on the ALNS and how the framework is used to generate solutions.

The ALNS heuristic aims to derive new best solutions by iteratively removing and reinserting order nodes in a solution. The removals and insertions are carried out by *destroy* and *repair* operators. The adaptiveness of the heuristic lies in the choice of operators. At regular intervals, an operator's *weight* is updated according to the quality of the solutions the operator has contributed to generate, referred to as the operator's *score*. These weights are used when selecting operators to apply in order to generate a new solution, and as a result, the successful operators are chosen more frequently. The new solution is accepted according to some acceptance criteria. The algorithm terminates when a stopping criterion is met.

This section describes the ALNS heuristic and is organized as follows. We start in Section 3.4.1 by providing an overview of the heuristic. Next, in Section 3.4.2, we describe the inclusion of noise in the utility function. Sections 3.4.3–3.4.5 are concerned with destroy operators, the permutation of vessel visits at factories, and repair operators, respectively. We provide a description of the operator selection procedure in Section 3.4.6, and introduce a simplified production check complementing the production schedule heuristic in Section 3.4.7. Lastly, in Section 3.4.8, we define the acceptance and stopping criteria.

3.4.1. Overview of the decomposition-based adaptive large neighborhood search heuristic

Algorithm 3 provides an overview of the ALNS heuristic applied to the FFPRP. We start by constructing an initial solution for which we assert production feasibility, as described in Section 3.3. We then perform I^{ALNS} iterations of the ALNS procedure. The iterations are divided into *segments* consisting of I^{SEG} iterations. We start each iteration by choosing destroy and repair operators, d_1 and d_2, based on the weights w_{dm} for each operator d in the current segment m. The operators are subsequently applied to the current routing solution x, which results in a new candidate solution x'. If x' is a promising solution, we obtain a solution y' to the corresponding production sub-problem by applying the greedy production heuristic described in Section 3.3.4. If the production sub-problem is feasible, we accept the routing solution x'. Further, we examine whether $(x' \quad y')$ constitutes a new best solution to the FFPRP. On the other hand, if x' is not considered promising, the routing solution is accepted according to a simulated annealing criterion. We then update the scores π_d of the applied operators. For each I^{SEG} iteration, the weights for the next segment $w_{d(m+1)}$ are also updated, based on the operators' scores π_d from the previous segment. The scores π_d are reset after each segment. After I^{ALNS} iterations we improve the production sub-problem's solution y^*, corresponding to

routing solution x^*, by using an exact solver. Lastly, we return the best found routing solution, x^*, along with the solution to the corresponding production sub-problem, y^*, which together form a feasible solution to the FFPRP.

Algorithm 3: ALNS Heuristic for the FFPRP

Result: solution to the FFPRP
I^{ALNS} = total number of ALNS iterations
I^{SEG} = number of iterations in each segment
construct a current routing solution x, such that a corresponding feasible production solution y exists
set the best solution, $(x^* \quad y^*) = (x \quad y)$
for *iteration* $= 1$ *to* I^{ALNS} **do**
  choose destroy and repair operators d_1, d_2 based on weights w_{dm} for operator d and segment m
  create candidate routing solution x' by applying the operators d_1 and d_2 on the current routing solution x
  while *solution* x' *is infeasible by the simplified production feasibility check* **do**
    remove orders from routing solution x'
  end
  if $f(x') < f(x^*) \cdot (1 + \epsilon)$ **then**
    obtain a production sub-problem solution y', corresponding to routing solution x', using the greedy production heuristic
    if *production sub-problem is feasible* **then**
      update the current routing solution to be the candidate routing solution, $x = x'$
      if $h(x', y') < h(x^*, y^*)$ **then**
        $(x^* \quad y^*) = (x' \quad y')$
      end
    end
  else if $f(x') < f(x)$ *or solution* x' *is accepted according to the simulated annealing-based criterion* **then**
    update the current routing solution to be the candidate routing solution, $x = x'$
  end
  update scores π_{d_1}, π_{d_2} for the chosen destroy and repair operators d_1, d_2
  if I^{SEG} *iterations has passed since the previous weight update* **then**
    update weight $w_{d(m+1)}$ for the next segment $m + 1$ based on the score π_d for each operator d in segment m
  end
end
improve y^* using an exact solver
return best found solution $(x^* \quad y^*)$

3.4.2. Noise in the utility function

The function used to calculate the utility of an insertion is similar to the function in Eq. (40). However, as the insertion methods are myopic, a noise term similar to that of Røpke and Pisinger (2006) is added. In this way, the search is diversified by introducing randomization in the insertion process. The utility U_{vsj} of an insertion defined by a vessel v, a visit number s, and an order node j is thus given by the equation

$$U_{vsj} = C_j^E - \Delta C_{vsj}^T + \text{noise}, \tag{44}$$

where C_j^E is the penalty for not delivering order j and ΔC_{vsj}^T is the net change in transport cost given by Eq. (39). The noise parameter takes a random value in the range $[-N_v^{max}, N_v^{max}]$, where $N_v^{max} = \eta \cdot \max_{i,j \in \mathcal{N}} \left\{ C_{vij}^T \right\}$. Here, the parameter η controls the noise level, C_{vij}^T is the transport cost from node i to node j for the vessel v and $\mathcal{N}$ is the set of all factory and order nodes. As a result, the noise term will be somewhat proportional to the objective function. As mentioned in Section 3.3.3, the choice of new destination factory is also based on utility. Here, noise is calculated and applied in a similar manner as for order nodes, but with a different noise control parameter, η', as a higher level of noise may be needed to avoid myopic insertions. The choice of whether or not to apply noise in the insertion is done in each iteration by an adaptive roulette wheel mechanism, which is described in detail in Section 3.4.6.

3.4.3. Destroy operators

The destroy operators eliminate parts of the current solution by removing node visits from routes. Specifically, all destroy operators remove at least q^{ALNS} order nodes from the solution, or all order nodes if the number is less than q^{ALNS}. These are inserted into the set of unrouted orders, $\mathcal{U}$. We consider five different types of removal operators: worst removal, related removal, voyage removal, route removal and random removal. The *worst removal* operator iteratively removes order nodes associated with a high cost, until q^{ALNS} order nodes are removed or until all order nodes are removed. *Related removal* removes similar order nodes from the routing solution, motivated by the assumption that similar orders are easier to reshuffle while maintaining feasibility. The relatedness R_{ij} between order nodes i and j is defined by Eq. (45). The more related two order nodes are, the lower the value of R_{ij}. We consider three dimensions of relatedness: spatial relatedness, temporal relatedness, and disease class relatedness. The first, spatial relatedness, is defined by the sailing distance d_{ij} between the order nodes. The second, temporal relatedness, is defined as the absolute difference between the orders' time windows. The third, disease class relatedness, denoted R_{ij}^{disease}, is manually defined. The values are provided in Appendix A. To our knowledge, the highly domain-specific operator concerned with relatedness in terms of disease outbreaks, has not previously been applied.

$$R_{ij} = a_0 d_{ij} + a_1 \left(|\underline{T_i} - \underline{T_j}| + |\overline{T_i} - \overline{T_j}| \right) + a_2 R_{ij}^{\text{disease}} \tag{45}$$

In our implementation, we apply two different related removal operators: one is based on spatial and temporal relatedness, whereas the other is based on spatial and disease class relatedness. This amounts to $a_0, a_1 > 0$ and $a_0, a_2 > 0$, respectively.

The *voyage removal* operator iteratively removes voyages until at least q^{ALNS} order nodes have been removed, or until one voyage has been removed from each route. We apply two different voyage removal operators: the first randomly selects which voyage to remove, and the second selects the voyage of highest cost. The *route removal* operator iteratively removes routes until at least q^{ALNS} order nodes are removed. As for voyage removal, we apply two different route removal operators, using random selection and highest cost selection.

Lastly, the *random removal* operator removes q^{ALNS} randomly chosen order nodes from the routes. Similar to Røpke and Pisinger (2006), we apply a deterministic parameter in order to obtain partial randomness in the removal selection. This is applied in the worst removal, related removal, worst voyage removal, and worst route removal operators.

3.4.4. Vessel visit permutation

Generally, the order of vessel visits at factories is permuted by the destroy and repair operators. However, this is not the case for the order of the vessels' *initial* factory visits, which is determined by the vessels' respective start times. Therefore, in order to traverse a larger part of the solution space, we apply a permutation procedure with probability ρ after the solution is destroyed in each iteration. The permutation procedure finds all pairs of consecutive initial vessel visits at factories whose order may be swapped without violating the routing time constraints. After finding all such pairs, a random pair of vessel visits is chosen and permuted.

3.4.5. Repair operators

After the removal of order nodes – and possibly the permutation of vessel visits – we apply a repair operator to reinsert unserved orders into the solution. We consider three repair operators: greedy insertion, 2-regret insertion and 3-regret insertion.

The *greedy repair* operator always makes the insertion with the greatest utility. Order nodes are inserted until no feasible insertions

Table 1
Success types with corresponding score reward criteria.

Success type	Reward criterion
σ_1	The candidate routing solution x' is part of a new best solution to the master problem
σ_2	The candidate solution x' is not part of a new best solution to the master problem, but is better than the current routing solution x; the candidate solution x' has not been accepted before
σ_3	The candidate solution x' is worse than the current solution x, but was accepted by the simulated annealing criterion; the candidate solution x' has not been accepted before

exist. A candidate solution, meaning the resulting solution after a destroy operator has been applied, is given as input. The utility is calculated according to Eq. (44), and is used to sort the insertions. The insertion yielding the greatest objective improvement is chosen greedily, meaning we evaluate the feasibility of the insertions in descending order according to insertion utility. As the feasibility check is computationally expensive compared to the utility calculation, we save computational effort by only checking feasibility for the best insertions until a feasible insertion is found. The algorithm terminates either when all orders have been inserted or when all insertions have been evaluated.

The *k-regret insertion* operator is similar to the construction heuristic presented in Section 3.3.1, as it inserts the order node whose *regret value* is the highest. However, we now include both 2-regret and 3-regret. We define the k-regret value, R^k, according to the formula

$$R^k = \sum_{h=2}^{k} (U_i^1 - U_i^h), \tag{46}$$

where U_i^h is the insertion utility, as defined by Eq. (44), of inserting node i in its hth best position. For each iteration we insert the order corresponding to the greatest R^k value. Again, we calculate the utility of each insertion before we evaluate feasibility.

3.4.6. Choosing a destroy and repair heuristic

The choice of operators to apply to a candidate solution, as well as the choice whether to apply noise in the insertion methods, are based on a roulette wheel mechanism. In each segment m, each operator d is assigned a weight w_{dm}, representing the operator's success in previous segments. The weights are updated in the end of each segment according to the formula

$$w_{dm} = \begin{cases} \max\left\{\kappa, (1-r)\cdot w_{d(m-1)} + r\cdot \frac{\pi_d}{\theta_d}\right\}, & \text{if } \theta_d > 0 \\ w_{d(m-1)}, & \text{otherwise,} \end{cases} \tag{47}$$

where $0 \le r \le 1$ is the *reaction factor* and π_d and θ_d are the score and the number of usages, respectively, of operator d in the passed segment. In order to ensure that all operators may be used, we define a minimum weight κ. If an operator has not been applied in a segment, meaning $\theta_d = 0$, its weight value is left unchanged.

An operator is chosen with a probability $P(d)$, calculated according to the equation

$$P(d) = \frac{w_{dm}}{\sum_{\hat{d}\in\hat{D}} w_{\hat{d}m}}, \tag{48}$$

where $\hat{D}$ denotes the set of destroy operators if d is a destroy operator, and otherwise the set of repair operators. Note that the destroy and repair operators are chosen independently.

The score π_d of operator d is incremented after each iteration, where the value of the increment depends on the success of the operator. We apply a similar score update scheme as that of Røpke and Pisinger (2006), using three levels of success, presented in Table 1. That is, the increment is dependent on success, where higher success causes the operator to be rewarded with a greater score increment. Note that operators used to generate a promising routing solution for which no feasible production solution is found are not rewarded, as the solution is not accepted.

The process of choosing whether to include noise in the insertion methods follows the exact same procedure, and uses the same score values and minimum weight value. Note that the decision whether or not to include noise is independent from the selected operators.

3.4.7. Simplified production feasibility check

In order to save computation time, we apply a simplified production feasibility check as a first evaluation of a routing solution's production feasibility. This feasibility check does not guarantee that the solution is production and inventory feasible, but will in many cases be able to prove infeasibility. More specifically, if the quantity of products to be picked up at a factory is greater than the factory's total production capacity over the relevant time periods plus its initial inventory, the routing solution cannot have an accompanying feasible production solution.

3.4.8. Acceptance and stopping criteria

The acceptance of a routing solution x' can follow two different paths. First, if the candidate routing solution's objective value $f(x')$ is less than the currently best solution's routing objective $f(x^*)$ times $(1+\epsilon)$, we evaluate production feasibility. If the production sub-problem is feasible, routing solution x' is accepted. If, on the other hand, no feasible production solution is found, routing solution x' is rejected. Second, if solution x' is not considered promising, we use a simulated annealing criterion to evaluate acceptance. The heuristic terminates after I^{ALNS} iterations, as seen in Algorithm 3.

3.5. Determining the final solution

After I^{ALNS} iterations, we have obtained a best found solution $\begin{pmatrix} x^* & y^* \end{pmatrix}$ to the FFPRP. However, in order to further improve the solution y^*, we apply an exact solver to the production sub-problem for T^E seconds. The best found production solution y^* and the routing solution x^* constitute the final solution to the FFPRP.

4. Computational study

The computational tests are run using a 2x 2.3 GHz Intel E5 processor with 64 GB memory. Python 3.8.6 is used to implement both the optimization model, solved with Gurobi 9.1, and the ALNS heuristic.

We start in Section 4.1 by describing our test instances. Next, in Section 4.2, we present the ALNS heuristic parameter tuning and the evaluation of the heuristic setup. In Section 4.3, we compare the results of our heuristic to those of a commercial MIP solver. Lastly, in Section 4.4, we demonstrate how the ALNS heuristic can be used to provide managerial insight by examining the impact of the delivery time window length.

4.1. Test instances

Test instances were created by implementing and using a test instance generator. The implementation is based on the structure, operations, and characteristics of two of Norway's largest fish feed producers, BioMar and Mowi, in order to mimic a realistic setting. We consider eight real vessels, presented in Table 2, each used by either of the two companies to distribute fish feed. Capacity and speed data were provided by our industry collaborators, which also provided the sailing distances.

The values for transport cost are estimates calculated using the method reported by Ivarsøy and Solhaug (2014), which uses data on similar vessels provided by Egil Ulvan Rederi. To calculate loading

Table 2
Vessel name, capacity, speed, and transport cost for the vessels used in our computational tests.

Vessel name	Capacity		Average speed	Transport cost
	[ton]	[no. products]	[knots]	[NOK/h]
Borgenfjord	1600	–	11	939
Høydal	2160	28	12	1215
Nyksund	2600	–	10	1215
Ripnes	1000	–	10	739
Vågsund	1000	21	11	939
Pirholm	1200	–	9	569
With harvest	3000	6	13	1580
With marine	3000	6	13	1580

Table 3
Factory data used in our computational tests.

Factory location	Company	Production lines	Inventory capacity [ton]	Vessel loading spots
Karmøy	BioMar	3	5000	1
Myre	BioMar	2	5000	1
Valsneset	Mowi	3	5000	2

and unloading times, we assume a loading rate of 300 ton/h and an unloading rate of 200 ton/h for all vessels (Egil Ulvan Rederi AS, n.d.).

For a given test instance, each vessel in the fleet becomes available at a given start factory, drawn randomly from the set of factories, at a given time period. The time period each vessel becomes available is randomly drawn from a triangular probability distribution, where the probability is highest for the first time period and decreases linearly to zero for the time period in the middle of the planning horizon. As mentioned in Section 1.1, a vessel may not be able to deliver all orders or visit all fish farms for two reasons. First, silo vessels cannot deliver bag orders, which is required by around 25% of the fish farms. Second, around 5% of the fish farms are located in shallow fjords, where the largest vessels cannot sail, or lack the equipment for certain vessels to dock.

Orders are sampled from a set of real orders provided by our industry collaborator Anteo. The original data set consists of 4465 entries, each representing an order for BioMar or Mowi of a certain quantity of some fish feed product to a fish farm, as well as the voyage start factory and date. Information about the actual product types and product groups, order delivery time windows, and the locations' disease classes were not provided, and therefore had to be generated. To account for the fact that diseases easily spread to nearby fish farms, the generation of order zones is based on spatial proximity. Algorithm 4, provided in Appendix B, presents the process of generating the set of order nodes assigned to the red, yellow, and green disease classes. The recovery time was set to 12 h.

We include the three factories belonging to BioMar and Mowi in the test instances. The values of the unique attributes for each factory are stated in Table 3. When generating the test instances, we take into account that some vessel loading spots at the factories can be occupied by incoming ships transporting raw materials for the fish feed production. Not delivering an order is an absolute last resort, which indicates that the corresponding penalty should be relatively high. To make sure that delivering an order always is preferable, we set the penalty for not delivering an order to the sum of the remaining cost components' maximum values.

Based on current practice, we assume that planning is conducted in a rolling horizon-fashion. Hence, it follows that the orders in the beginning of the current planning horizon have been planned for and produced in the previous planning horizon. The initial inventory at each factory is therefore set based on the product quantities demanded at the beginning of the planning horizon. As initial inventory is believed to have a significant effect upon problem complexity, we will vary the share of the total demanded quantity held as initial inventory across the test instances.

Table 4
Production data used in our computational test.

Production capacity per production line	25	[ton/h]
Production setup cost	1500	[NOK]
Inventory cost	0.1	[NOK/ton/h]
Minimum batch size	50	[ton]

In each test instance, each factory is subject to at most one production stop lasting 12 h. The start time of the production stop is chosen randomly from the set of all time periods. The time needed to shift between different product groups is set to one hour. Table 4 presents the other production-related parameters, which are assumed to be equal for all factories.

Five factors are assumed to be particularly linked to problem complexity: the number of vessels $|\mathcal{V}|$, the number of factories $|\mathcal{F}|$, the number of order nodes $|\mathcal{N}^O|$ and the number of time periods $|\mathcal{T}|$, as well as the initial inventory level I^0 at the factories. Tables C.1 and C.2 in Appendix C provide overviews of the 12 test instances used for tuning of the ALNS heuristic and the 60 test instances used for performance testing, respectively. The test instances' IDs are on the form $|\mathcal{V}| - |\mathcal{F}| - |\mathcal{N}^O| - I^0$, where the initial inventory level measured relative the maximum capacity, I^0, is either 0.2, low (l), or 0.5, high (h). The corresponding number of time periods is based on preliminary analyses estimating the minimum time needed for the vessel fleet to deliver all orders. Based on the real sailing distances and times used for loading and unloading, the time period length is set to one hour, except for the smallest instances with less than 20 orders, where it is set to two hours. This gives us 366 time periods if the planning horizon consists of 14 days, which is significantly higher than what is common in the PRP literature. However, a time period in the standard PRP incorporates a lot more than what it does in the FFPRP, so the two numbers cannot easily be compared. The ID of each test instance has the prefix "t" or "p", depending on whether the instance is used for tuning or performance testing, respectively. Furthermore, the test instance set used for performance testing contains several instances for each setting. To distinguish between these, the instance IDs are also given a suffix (0–2). Lastly, we use a time window length of four days and sample products from a set of ten products and product groups from a set of four product groups for all test instances unless otherwise stated.

We also generate an additional set of test instances to analyze the impact of the length of the time windows, shown in Section 4.4. These test instances are presented in Table C.3 in Appendix C. For each vessel, factory, and order setting we generate three sets (suffix 0–2) of five instances for which the only difference is the delivery time window length (marked "d" in the ID), which varies from one to five days. The initial inventory level is set to medium level of 0.4.

4.2. Parameter tuning and heuristic setup evaluation

The process of tuning the parameters presented of the ALNS heuristic corresponds to the procedure proposed by Røpke and Pisinger (2006). We tune the parameters sequentially, varying only one parameter at a time. For each parameter, three to five different values are tested. For each parameter value, we run the heuristic on each of the tuning instances three times and choose the parameter value corresponding to the best performance, amounting to a trade-off between run time and solution quality. For the following parameter to tune, we apply the final values of the already tuned parameters and the initial values of the parameters that have not yet been tuned. The tuning process was performed once for each parameter. Appendix D provides more details of the tuning process as well as the resulting parameter values.

Table 5
Comparison of the ALNS heuristic run for 40 000 iterations three times to an exact solver run for 3600 s and 600 s on the set of smaller instances. The gap values represent the gap between the relevant objective value and the best objective produced in the experiment, as defined by Eq. (49). For the 3600 s run with an exact solver, the MIP gap is also reported.

	Gurobi (3600 s)				Gurobi (600 s)			ALNS		
	Obj. [NOK]	Gap [%]	MIP Gap [%]	Time [s]	Obj. [NOK]	Gap [%]	Time [%]	Obj. [NOK]	Gap [%]	Time [s]
p-3-1-10-*h*-0	148 152	0.00	0.01	3600	148 152	0.00	600	148 800	0.44	236
p-3-1-10-*h*-1	258 868	0.00	0.00	3600	258 868	0.00	600	260 123	0.48	257
p-3-1-10-*h*-2	250 985	0.00	0.00	13	250 985	0.00	13	252 199	0.48	400
p-3-1-10-*l*-0	180 567	0.00	0.35	3600	180 567	0.00	600	182 295	0.96	284
p-3-1-10-*l*-1	424 385	0.00	0.23	3600	424 385	0.00	600	425 556	0.28	348
p-3-1-10-*l*-2	259 913	0.00	0.01	3600	259 913	0.00	600	261 926	0.77	257
p-3-1-15-*h*-0	334 513	0.00	0.01	3600	334 513	0.00	600	336 480	0.59	479
p-3-1-15-*h*-1	155 255	0.00	16.11	3600	155 255	0.00	600	155 925	0.43	420
p-3-1-15-*h*-2	382 181	0.00	0.89	3600	382 181	0.00	600	384 317	0.56	436
p-3-1-15-*l*-0	513 762	0.00	2.71	3600	513 762	0.00	600	516 126	0.46	606
p-3-1-15-*l*-1	165 985	0.00	31.79	3600	166 035	0.03	600	169 562	2.16	314
p-3-1-15-*l*-2	237 889	0.00	0.00	3600	237 889	0.00	600	238 556	0.28	385
Avg. 3-1-(...)	**276 038**	**0.00**	**4.34**	**3301**	**276 042**	**0.00**	**551**	**277 655**	**0.66**	**369**
p-5-2-10-*h*-0	90 065	0.00	0.00	860	90 065	0.00	600	92 358	2.55	255
p-5-2-10-*h*-1	136 060	0.00	1.25	3600	136 060	0.00	600	138 114	1.51	201
p-5-2-10-*h*-2	124 362	0.00	2.21	3600	124 362	0.00	600	126 672	1.86	257
p-5-2-10-*l*-0	107 486	0.00	49.26	3600	107 513	0.02	600	108 087	0.56	544
p-5-2-10-*l*-1	408 457	0.00	1.13	3600	408 464	0.00	600	409 456	0.24	346
p-5-2-10-*l*-2	88 803	0.00	0.00	767	88 803	0.00	600	90 181	1.55	327
p-5-2-15-*h*-0	133 926	0.00	18.21	3600	138 641	3.52	600	144 369	7.80	317
p-5-2-15-*h*-1	166 669	0.00	2.46	3600	166 669	0.00	600	169 356	1.61	363
p-5-2-15-*h*-2	147 474	0.00	10.06	3600	147 518	0.03	600	154 946	5.07	379
p-5-2-15-*l*-0	96 082	0.00	0.97	3600	96 082	0.00	600	98 451	2.47	408
p-5-2-15-*l*-1	144 342	0.00	7.17	3600	144 342	0.00	600	145 097	0.52	494
p-5-2-15-*l*-2	156 899	0.00	9.08	3600	162 936	3.85	600	158 705	1.15	374
Avg. 5-2-(...)	**150 052**	**0.00**	**8.48**	**3136**	**150 955**	**0.62**	**600**	**152 983**	**2.24**	**355**
Avg. (...)-*h*-(...)	**194 043**	**0.00**	**4.27**	**3073**	**194 439**	**0.30**	**551**	**196 972**	**1.95**	**333**
Avg. (...)-*l*-(...)	**232 048**	**0.00**	**8.56**	**3364**	**232 558**	**0.33**	**600**	**233 667**	**0.95**	**391**
Avg. total	**213 045**	**0.00**	**6.41**	**3218**	**213 498**	**0.31**	**576**	**215 319**	**1.45**	**362**

Using the tuned parameter values, we proceed to evaluate the setup of the ALNS heuristic. Specifically, we analyze the effect of using adaptive weights and several different operators, referred to as *configurations* of the large neighborhood search (LNS), and explore different setups related to integration of the production sub-problem and the routing sub-problem. The results are shown in Appendix E. In conclusion, the heuristic performs significantly better when a range of operators is applied, and slightly better than when operators are chosen randomly. As for sub-problem integration, the setup described in Section 3 has the best performance among the investigated setups.

4.3. Computational results

This section presents the computational results. Section 4.3.1 evaluates the performance of the ALNS heuristic as a whole, whereas Section 4.3.2 is dedicated to the production schedule heuristic described in Section 3.3.4.

4.3.1. ALNS heuristic performance

The performance of the ALNS heuristic is evaluated by comparing the results to those produced by the commercial MIP solver Gurobi on the 60 test instances presented in Table C.2. The number of iterations is set with the intention to roughly match the companies' preferable maximum run of ten minutes.

Specifically, we perform 40 000, 20 000, 10 000, and 5000 iterations on the test instances with <20, 20, 40, and 60 orders, respectively. The heuristic is run three times for each test instance. As for the commercial solver, we report the results both after 600 and 3600 s.

First, we examine 24 smaller instances with 10 and 15 orders, three products, and a time period length of two hours. The results are presented in Table 5. As seen from the MIP gap, defined as the absolute difference between Gurobi's primal and dual bounds, divided by the primal bound, Gurobi is able to find optimal or close-to-optimal solutions for several of these instances after 3600 s. The gap column is defined as

$$\text{gap} = \frac{|z' - \underline{z}|}{|\underline{z}|}, \tag{49}$$

where z' and $\underline{z}$ are the objective values of the relevant solution and the best solution for a given test instance produced in the experiment (by either Gurobi or the ALNS heuristic), respectively. For the ALNS heuristic, the average gap across these instances amounts to 1.45%.

Comparing the solutions from Gurobi to those of the heuristic, we observe that for most of the smaller gaps, the routing is similar except for departure and arrival times. This results in different inventory costs. Recall that when we solve the production sub-problem, we fix the production deadlines to the vessels' latest arrival times. Consequently, fish feed that is already available at the factory may be held as inventory for a slightly longer time than necessary. The exact solver, on the other hand, is able to reduce the inventory cost by letting vessels depart as early as possible. In practice, a fish feed planner would probably choose to shift the production to earlier time periods and let the vessels depart from the factory earlier. However, we argue that the deviation is negligible in a practical setting, as the departure time can easily be adjusted for by the fish feed planner, and because the cost difference is marginal.

Furthermore, we observe that the gaps of the solutions increase with the number of vessels and factories. Specifically, comparing the 3-1-(...) setting to the 5-2-(...) setting, the average MIP gap increases from 4.27% to 8.48%, and the gap of the ALNS from 0.66% to 2.24%. These results indicate that the multi-depot attribute has a significant impact on problem complexity.

Next, we investigate the average results for the settings corresponding to high and low initial inventory, denoted (...)-*h*-(...) and (...)-*l*-(...),

Table 6
Comparison of the ALNS heuristic to an exact solver run for 3600 s and 600 s on the set of test instances containing 20, 40, and 60 orders. The ALNS heuristic is run three times for 20 000, 10 000 and 5000 iterations for instances of 20, 40, and 60 orders, respectively. The gap values represent the gap between the relevant objective value and the best objective produced in the experiment, as defined by Eq. (49). Note that for some instances, the exact solver was not able to find a feasible solution within 600 s. These runs are denoted by "–". The corresponding average rows, marked with an asterisk, do not include these missing values.

	Gurobi (3600 s)			Gurobi (600 s)			ALNS		
	Obj. [NOK]	Gap [%]	Time [s]	Obj. [NOK]	Gap [%]	Time [s]	Obj. [NOK]	Gap [%]	Time [s]
p-3-1-20-*h*-0	1 468 888	762.18	3600	2 739 733	1508.12	600	170 374	0.00	611
p-3-1-20-*h*-1	506 675	12.71	3600	1 855 736	312.82	600	450 118	0.13	587
p-3-1-20-*h*-2	1 390 970	693.18	3600	2 508 636	1330.51	600	175 747	0.22	479
p-3-1-20-*l*-0	2 019 899	189.99	3600	4 210 157	504.44	600	696 551	0.00	833
p-3-1-20-*l*-1	661 168	219.68	3600	4 446 078	2049.68	600	206 848	0.01	693
p-3-1-20-*l*-2	1 296 383	461.39	3600	4 426 196	1816.74	600	232 048	0.49	607
p-3-1-40-*h*-0	4 163 952	1177.67	3600	6 157 022	1789.22	600	329 299	1.04	759
p-3-1-40-*h*-1	6 724 128	2614.14	3600	7 247 089	2825.23	600	247 948	0.08	655
p-3-1-40-*h*-2	5 943 428	2260.09	3600	5 943 428	2260.09	600	254 445	1.04	598
p-3-1-40-*l*-0	7 974 148	2560.37	3600	8 675 162	2794.23	600	308 202	2.82	971
p-3-1-40-*l*-1	7 680 881	2305.22	3600	7 878 787	2367.19	600	319 602	0.08	693
p-3-1-40-*l*-2	7 474 128	2589.21	3600	7 474 128	2589.21	600	279 602	0.60	632
p-3-1-60-*h*-0	11 447 168	2789.15	3600	–	––	600	405 657	2.38	658
p-3-1-60-*h*-1	9 188 074	2483.97	3600	11 161 864	3039.06	600	356 268	0.19	618
p-3-1-60-*h*-2	12 852 484	1036.98	3600	14 676 456	1198.34	600	1 133 375	0.26	1194
p-3-1-60-*l*-0	12 531 787	1281.23	3600	14 313 594	1477.62	600	931 711	2.69	1101
p-3-1-60-*l*-1	10 164 539	975.70	3600	14 149 972	1397.47	600	1 063 412	12.54	1233
p-3-1-60-*l*-2	11 649 623	1121.62	3600	14 328 949	1402.59	600	960 646	0.74	1299
Avg. 3-1-(...)	**6 396 574**	**1417.58**	**3600**	*7 776 058	*1803.68	**600**	**473 436**	**1.41**	**790**
p-5-2-20-*h*-0	150 297	0.00	3600	2 490 180	1556.84	600	158 099	5.19	457
p-5-2-20-*h*-1	181 239	3.88	3600	223 224	27.94	600	174 477	0.00	374
p-5-2-20-*h*-2	206 911	2.45	3600	1 743 561	763.30	600	203 334	0.68	418
p-5-2-20-*l*-0	1 114 407	493.74	3600	2 216 200	1080.76	600	188 106	0.22	383
p-5-2-20-*l*-1	221 918	20.03	3600	3 532 248	1810.58	600	184 878	0.00	561
p-5-2-20-*l*-2	151 875	0.87	3600	2 339 026	1453.55	600	150 564	0.00	473
p-5-2-40-*h*-0	6 740 367	2891.31	3600	9 115 794	3945.50	600	226 306	0.43	562
p-5-2-40-*h*-1	6 489 026	2602.52	3600	9 806 050	3983.98	600	242 168	0.86	525
p-5-2-40-*h*-2	3 776 309	1290.34	3600	9 622 765	3442.86	600	271 815	0.08	540
p-5-2-40-*l*-0	6 916 795	2691.34	3600	9 697 981	3813.71	600	248 090	0.12	551
p-5-2-40-*l*-1	7 072 428	2506.80	3600	7 936 992	2825.46	600	274 466	1.16	534
p-5-2-40-*l*-2	6 412 426	2701.02	3600	9 287 675	3956.96	600	237 031	3.54	493
p-5-2-60-*h*-0	12 611 209	3155.18	3600	–	––	600	393 039	1.45	573
p-5-2-60-*h*-1	14 047 903	4590.95	3600	–	––	600	315 784	5.45	635
p-5-2-60-*h*-2	12 682 847	2921.03	3600	–	––	600	446 298	6.31	623
p-5-2-60-*l*-0	14 606 419	4215.51	3600	–	––	600	360 444	6.49	682
p-5-2-60-*l*-1	13 583 492	4717.36	3600	–	––	600	324 695	3.20	606
p-5-2-60-*l*-2	14 113 212	3378.98	3600	–	––	600	420 396	3.63	625
Avg. 5-2-(...)	**6 726 616**	**2099.35**	**3600**	***5 667 641**	***2388.45**	**600**	**267 777**	**2.16**	**534**
Avg. (...)-*h*-(...)	**6 142 882**	**1733.82**	**3600**	***6 092 253**	***1998.84**	**600**	**330 808**	**1.43**	**604**
Avg. (...)-*l*-(...)	**6 980 307**	**1777.38**	**3600**	***7 660 876**	***2089.35**	**600**	**410 405**	**2.13**	**721**
Avg. total	**6 561 595**	**1755.60**	**3600**	***6 721 850**	***2096.07**	**600**	**370 607**	**1.78**	**662**

respectively. A low initial inventory will typically lead to higher costs, as this often imposes stricter requirements on the routing and increases the number of production starts. Inventory costs, however, will be lower in this setting. We see that the results agree with this. In fact, the average objective in the low initial inventory setting is about 20% higher than in the high initial inventory setting. Furthermore, we expect low initial inventories to give more complex problems, as this setting will often require a higher level of coordination between routing and production. The difference in average MIP gap supports this. For the ALNS heuristic, on the other hand, the average gap is significantly lower in the low initial inventory setting than in the high initial inventory setting.

Table 6 presents the results for the larger test instances containing 20, 40, and 60 orders. These results demonstrate the superiority of the ALNS heuristic over Gurobi for more realistically sized test instances of the FFPRP. Running the Gurobi solver for one hour results in an average gap of 1756% across all instances, whereas the average gap is only 1.78% for the ALNS heuristic. The poor performance of Gurobi for instances of only 20 orders illustrates the large complexity of the FFPRP. For the instances with 60 orders, Gurobi is able to assign at most 18 orders to routes, and for most instances, less than ten. On the other hand, the heuristic leaves at most three orders unserved for these test instances.

The gap between Gurobi's dual bound and the average objective value found by the ALNS heuristic is somewhat large. Specifically, for the instances involving 20 orders, the average dual bound is 33% lower than the average objective value found by the ALNS heuristic. For the larger test instances, the difference is even greater. A possible explanation is that the ALNS heuristic fails to find good solutions. However, we argue that the FFPRP may be a difficult problem to provide strong dual bounds for. A similar PS-VRP including five to 15 customers is studied by Belo-Filho et al. (2015). Here, the objective values of the best solutions found by their matheuristic are on average 64.7% worse than the corresponding dual bound. According to the authors, the high level of detail incorporated in the production scheduling problem cause weak dual bounds, and the large gap was therefore expected. Similarly, the instances of the production sub-problems of the FFPRP are highly detailed, due to the large number of time periods considered. We, therefore, suggest that the reason for the large gaps of the ALNS heuristic is related to weak dual bounds rather than poor solution quality.

Table 7
Results when running 5000 iterations of the ALNS heuristic using different time window lengths for otherwise similar test instances. The columns state the average objective value and the average number of unserved orders, respectively, after three runs of the heuristic for each test instance.

	$T^{TW} = 1$		$T^{TW} = 2$		$T^{TW} = 3$		$T^{TW} = 4$		$T^{TW} = 5$	
	Obj. [NOK]	Uns.	Obj. [NOK]	Uns.	Obj. [NOK]	Uns.	Obj. [NOK]	Uns.	Obj. [NOK]	Uns.
tw-3-1-30-T^{TW}d-0	1 099 364	4.0	856 886	3.0	657 324	2.0	439 982	1.0	433 257	1.0
tw-3-1-30-T^{TW}d-1	2 382 375	11.0	2 371 315	11.0	2 153 926	10.0	1 978 530	9.0	1 754 190	8.0
tw-3-1-30-T^{TW}d-2	2 182 778	8.0	1 958 110	7.0	1 697 187	6.0	1 261 208	4.0	1 029 736	3.0
tw-3-1-60-T^{TW}d-0	2 264 045	6.0	1 719 963	4.0	1 420 005	3.0	1 105 793	2.0	796 698	1.0
tw-3-1-60-T^{TW}d-1	1 693 323	4.0	908 949	1.0	619 451	0.0	543 583	0.0	556 317	0.0
tw-3-1-60-T^{TW}d-2	1 615 079	3.0	750 785	0.0	661 280	0.0	646 369	0.0	565 341	0.0
tw-5-2-30-T^{TW}d-0	1 224 390	4.0	1 175 323	4.0	726 176	2.0	481 739	1.0	472 276	1.0
tw-5-2-30-T^{TW}d-1	1 921 063	7.0	1 428 392	5.0	993 987	3.0	539 273	1.0	287 339	0.0
tw-5-2-30-T^{TW}d-2	823 417	2.0	746 173	2.0	751 471	2.0	358 132	0.0	234 571	0.0
tw-5-2-60-T^{TW}d-0	1 890 637	5.0	1 620 544	4.0	1 316 211	3.0	995 252	2.0	511 145	0.0
tw-5-2-60-T^{TW}d-1	1 297 209	3.0	1 220 285	3.0	786 582	1.0	501 577	0.0	477 061	0.0
tw-5-2-60-T^{TW}d-2	1 788 776	5.0	1 490 465	4.0	766 846	1.0	456 818	0.0	430 190	0.0
Average	1 681 871	5.2	1 353 932	4.0	1 045 871	2.8	775 688	1.7	629 010	1.2

4.3.2. Production schedule heuristic performance

To evaluate the performance of the greedy production heuristic presented in Section 3.3.4, we use the instances with 20, 40, and 60 orders (a total of 60 instances). Specifically, we run the ALNS heuristic with 5000 iterations for each of the 60 test instances and record the production sub-problems solved when a promising routing solution is found. From each of these instances, we sample 20 production sub-problems, which gives us a total of 1200 production sub-problems, and solve each of these both by the greedy production heuristic and Gurobi. For the latter, we investigate the results when using time limits of both 30 and 120 s.

The results indicate that the production heuristic constructs sufficiently good solutions within an average run time of 0.03 s. For the commercial solver, run with time limits of 30 and 120 s, the average run times were 16 and 55 s, respectively. Only for six out of the 1200 production sub-problem instances, a feasible production schedule was found by the commercial solver within 120 s but not by the heuristic. This indicates that the heuristic is seldom unable to find feasible solutions when they exist. Furthermore, the average gap is 9.20% for the production heuristic. In comparison, average gaps of 3.18% and 0.00% are obtained for the 30 and 120 s runs with the commercial solver, respectively. We argue that the heuristic's gap is sufficiently small, given that the production cost, on average, contributes with no more than 8.6% of the objective value for the solutions provided by the ALNS heuristic. In conclusion, the production heuristic fulfills its purpose to rapidly evaluate feasibility and estimate the cost of the solution to the production sub-problem.

4.4. Impact of time window length

As mentioned in Section 1, fish feed companies are subject to customer relation power imbalances. Specifically, customers may require an order to be delivered within a short time window, for instance within working hours, or on a specific weekday. These delivery time restrictions limit the planning flexibility. Therefore, extending the delivery time windows may enable significant cost savings for the fish feed companies and for the supply chain as a whole.

In the previous test instances, the time window length was set to four days. However the actual delivery time flexibility experienced by some fish feed producers may be much smaller. In order to analyze the effect of this flexibility, we compare the results for a new set of test instances, where the time window length is varied between one and five days. Specifically, we first generate a test instance with a time window length of one, and next, we make duplicate instances with increasing time window lengths. The added time window length is distributed evenly before and after the original time window. Furthermore, we use sets of three and five vessels, one and two factories, and 30 and 60 orders. In all the test instances, we use a medium initial inventory level of 0.4.

Table 7 presents the results. The reported objective value and the number of unserved orders are averages across three runs for the same test instance. The number of unserved orders has a significant impact on objective value. In fact, the penalty for not delivering an order is set to obtain solutions where the number of served orders is as large as possible. Consequently, the magnitude of the objective values may be somewhat artificial. For these reasons, we argue that the number of unserved orders serves as a more useful metric to evaluate the cost reduction potential of extending the delivery time windows.

As expected, the quality of the solutions improves significantly when increasing the time window length. Specifically, the average number of orders not served decreases from 5.2 to 1.2 when the time window length is increased from one to five days. The fish feed producers may to some extent be able to overcome this issue by suggesting time windows to the customers that fit in their already planned routes and schedules. However, in the case of unforeseen events or the inclusion of a new customer on a route, the lack of routing flexibility may be costly. Hence, obtaining a tighter customer collaboration to increase time windows wherever practically possible represents a significant cost reduction potential for the whole supply chain, which in this experiment is quantified by the number of unserved orders.

5. Concluding remarks

This paper studied the fish feed production routing problem (FFPRP), which is an important planning problem faced by fish feed producers and distributors. The FFPRP is a highly complex and integrated problem consisting of production scheduling and vessel routing. The former incorporates multiple products as well as product groups. The latter exhibits several extensions compared to the traditional VRP, such as multi-depot, multi-trip, time windows, resource synchronization, and heterogeneous vessels. A solution to the FFPRP consists of a set of production schedules, each assigned to a production line, and a set of routes, each assigned to a vessel. In addition, the production schedules and the routes must correspond, meaning that the required product quantities must be available at the factory before they can be loaded onto vessels. Thus, an optimal solution to the FFPRP is one where production costs and routing costs, including penalties for not delivering customer orders, are minimized. To our knowledge, the resulting integrated problem has not previously been studied.

We proposed a discrete time MIP model for the FFPRP. Furthermore, we proposed a decomposition-based adaptive large neighborhood search (ALNS) heuristic, which decomposes the problem into two sub-problems, i.e., the production scheduling and the routing sub-problems. This decomposition is motivated by that the main cost

Table A.1
Disease relatedness values.

diseaseClass(i)	diseaseClass(j)	$R_{ij}^{disease}$
Green	Red	10
Green	Yellow	6
Yellow	Red	4
Green	Green	0
Yellow	Yellow	0
Red	Red	0

reduction potential for the FFPRP lies within the routing sub-problem. We therefore primarily evaluate solutions in terms of routing cost, whereas production feasibility and costs are evaluated only for the promising routing solutions. The routing sub-problem is solved with an ALNS heuristic, which is largely built upon Røpke and Pisinger (2006). However, to account for the special features of the FFPRP, several other extensions are also included. First, we have included a new problem-specific destroy operator, which is concerned with relatedness of fish farms in terms of fish disease outbreaks at the customers' fish farms. Second, we have implemented several new checks to handle the required resource synchronization emerging from the limited number of loading spots at the factories that are shared among the vessels. Third, the proposed ALNS implementation is designed for the multi-trip setting.

We generated a number of test instances based on real data from the industry partners and it was shown that both the proposed heuristic and the MIP solver are able to find high quality solutions in reasonable time for small problem instances with less than 20 orders. However, for the larger instances with 20 to 60 orders, the MIP solver had very high optimality gaps even after one hour of running time. The average cost across the solutions found by the MIP solver was more than 1700% higher than that of the best solutions found by the ALNS heuristic. Furthermore, in terms of run time, the MIP solver spends more than 50 min on each problem instance on average, whereas the average run time of the heuristic is around 11 min. From this, we conclude that the decomposition-based ALNS heuristic generates solutions to realistically sized problem instances of the FFPRP of significantly higher quality than MIP solvers within less time.

CRediT authorship contribution statement

Ivar Brekkå: Conceptualization, Methodology, Software, Writing – original draft. **Solveig Randøy:** Conceptualization, Methodology, Software, Writing – original draft. **Kjetil Fagerholt:** Conceptualization, Supervision, Writing – original draft, Writing - review & editing. **Kristian Thun:** Conceptualization, Supervision. **Simen Tung Vadseth:** Conceptualization, Supervision, Writing – original draft, Writing – review & editing.

Acknowledgments

We are grateful for the insights and data provided by our industry collaborators BioMar, Mowi, and Anteo.

Appendix A. Disease class relatedness

Table A.1 provides the disease class relatedness values. The lower value of $R_{ij}^{disease}$, the higher the relatedness. We assume $R_{ij}^{disease} = R_{ji}^{disease}$, and trivial relations are therefore excluded.

Appendix B. Order zone generation algorithm

Algorithm 4 presents the procedure of assigning order nodes to disease classes.

Algorithm 4: Order Zone Generation

Result: set of order nodes for each disease class
$\mathcal{N}^O$ = set of all order nodes
$\mathcal{N}^R$ = set of order nodes assigned to the red disease class, initially empty
$\mathcal{N}^Y$ = set of order nodes assigned to the yellow disease class, initially empty
$\mathcal{N}^G$ = set of order nodes assigned to the green disease class, initially empty
R = percentage of order nodes to assign to the red disease class
$\mathcal{U}$ = set of disease candidates, initially empty
while $\mathcal{N}^R$ *contains less than a share* R *of the order nodes* **do**
  if $\mathcal{U}$ *is empty* **then**
    randomly select an order node $i \notin \mathcal{N}^R$ from $\mathcal{N}^O$
  else
    randomly select an order node i from $\mathcal{U}$
  end
  add order node i to $\mathcal{N}^R$
  add all order nodes $j \notin \mathcal{N}^R$ located within a radius of 10 kilometers to order node i to $\mathcal{U}$
end
for *order node* i *in* $\mathcal{N}^R$ **do**
  add all order nodes $j \notin \mathcal{N}^R$ which are located within a radius of 30 kilometers to order node i to $\mathcal{N}^Y$
end
$\mathcal{N}^G = \mathcal{N}^O \setminus (\mathcal{N}^R \cup \mathcal{N}^Y)$
return the sets of order nodes corresponding to each disease class, $\mathcal{N}^R, \mathcal{N}^Y$ and $\mathcal{N}^G$

Table C.1
Overview of test instances used for parameter tuning of the ALNS heuristic. The column names denote ID, number of vessels, number of factories, number of orders, initial inventory level, number of time periods, and the corresponding number of days, respectively.

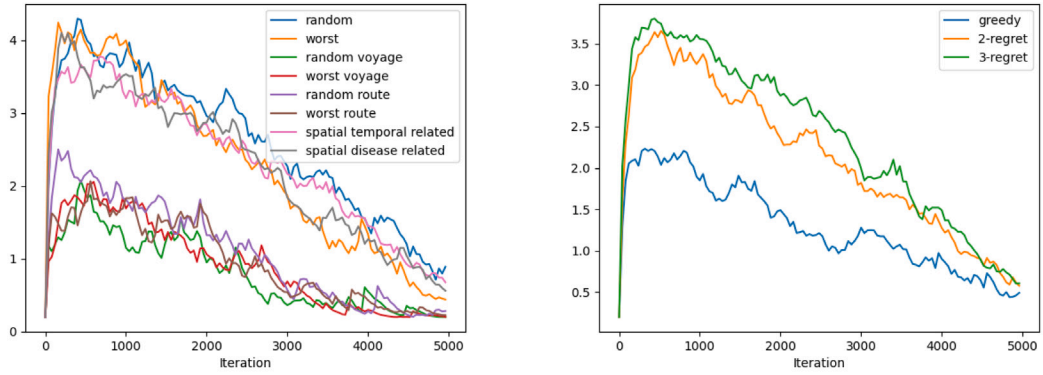
Test instance ID	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{F}\rvert$	$\lvert\mathcal{N}^O\rvert$	I^0	$\lvert\mathcal{T}\rvert$	Days
t-3-1-20-*h*	3	1	20	0.5	72	6
t-3-1-20-*l*	3	1	20	0.2	72	6
t-3-1-40-*h*	3	1	40	0.5	108	9
t-3-1-40-*l*	3	1	40	0.2	108	9
t-3-1-60-*h*	3	1	60	0.5	168	14
t-3-1-60-*l*	3	1	60	0.2	168	14
t-5-2-20-*h*	5	2	20	0.5	60	5
t-5-2-20-*l*	5	2	20	0.2	60	5
t-5-2-40-*h*	5	2	40	0.5	96	8
t-5-2-40-*l*	5	2	40	0.2	96	8
t-5-2-60-*h*	5	2	60	0.5	120	10
t-5-2-60-*l*	5	2	60	0.2	120	10

Appendix C. Test instance overview

Tables C.1–C.3 provide overviews of the test instances used for tuning, performance testing, and managerial insights, respectively.

Appendix D. Parameter tuning

Table D.1 provides an overview of the parameters, along with their respective initial values and final values, in the order of calibration. The initial value of a parameter refers to the value used before the parameter was tuned. These were found through an ad-hoc trial and error phase, while developing the ALNS heuristic. Also, many of the initial values were to some extent inspired by Røpke and Pisinger (2006) and Liu et al. (2018). Oppositely, the final value refers to the value set after tuning. Only the parameters whose value was considered to have a significant impact on the quality of the heuristic were subject to tuning. For the parameters not tuned, we only report the final value. The number of iterations I^{ALNS} was set to 5000. This amounts to run times in the range of one to ten minutes, depending primarily on the number of orders with which the test instance is concerned.



(a) Weights of destroy operators (b) Weights of repair operators

Fig. 10. Destroy operator weights 10(a) and repair operator weights 10(b) for 5000 iterations of the ALNS heuristic with adaptive weights on the parameter tuning instances. The plots show an average across the 12 tuning instances, each instance run once.

Table C.2
Overview of test instances used for performance testing of the ALNS heuristic. The column names denote ID, number of vessels, number of factories, number of orders, initial inventory level, number of time periods, and the corresponding number of days, respectively. The set is split into two subsets: a set of small instances and a set of larger instances. The former set is customized so that the instances are sufficiently small to be handled by exact solvers; these instances incorporate fewer orders to be served, only three different products, and a time period length of two hours. The latter set consists of more realistically sized problem instances, with a time period length of one hour.

Test instance ID	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{F}\rvert$	$\lvert\mathcal{N}^O\rvert$	I^0	$\lvert\mathcal{T}\rvert$	Days
p-3-1-10-*h*-(0-2)	3	1	10	0.5	60	5
p-3-1-10-*l*-(0-2)	3	1	10	0.2	60	5
p-3-1-15-*h*-(0-2)	3	1	15	0.5	60	5
p-3-1-15-*l*-(0-2)	3	1	15	0.2	60	5
p-5-2-10-*h*-(0-2)	5	2	10	0.5	60	5
p-5-2-10-*l*-(0-2)	5	2	10	0.2	60	5
p-5-2-15-*h*-(0-2)	5	2	15	0.5	60	5
p-5-2-15-*l*-(0-2)	5	2	15	0.2	60	5
p-3-1-20-*h*-(0-2)	3	1	20	0.5	144	6
p-3-1-20-*l*-(0-2)	3	1	20	0.2	144	6
p-3-1-40-*h*-(0-2)	3	1	40	0.5	216	9
p-3-1-40-*l*-(0-2)	3	1	40	0.2	216	9
p-3-1-60-*h*-(0-2)	3	1	60	0.5	336	14
p-3-1-60-*l*-(0-2)	3	1	60	0.2	336	14
p-5-2-20-*h*-(0-2)	5	2	20	0.5	120	5
p-5-2-20-*l*-(0-2)	5	2	20	0.2	120	5
p-5-2-40-*h*-(0-2)	5	2	40	0.5	192	8
p-5-2-40-*l*-(0-2)	5	2	40	0.2	192	8
p-5-2-60-*h*-(0-2)	5	2	60	0.5	288	12
p-5-2-60-*l*-(0-2)	5	2	60	0.2	288	12

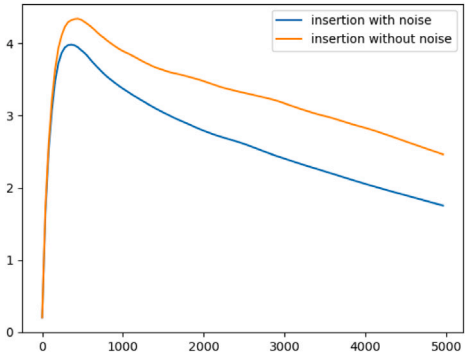


Fig. 11. Weights for the roulette wheel selection of noise application in the insertion methods. The plot shows an average across 12 runs on the tuning instances. Each instance was run once, with 5000 iterations.

Table C.3
Overview of test instances used for analyzing the impact of the length of the time windows. The column names denote ID, number of vessels, number of factories, number of order nodes and length of the delivery time window in number of days, respectively. The ID prefix "tw" refers to the analyses concerned with time window length.

Test instance ID	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{F}\rvert$	$\lvert\mathcal{N}^O\rvert$	T^{TW}
tw-3-1-30-1d-(0-2)	3	1	30	1
tw-3-1-30-2d-(0-2)	3	1	30	2
tw-3-1-30-3d-(0-2)	3	1	30	3
tw-3-1-30-4d-(0-2)	3	1	30	4
tw-3-1-30-5d-(0-2)	3	1	30	5
tw-3-1-60-1d-(0-2)	3	1	60	1
tw-3-1-60-2d-(0-2)	3	1	60	2
tw-3-1-60-3d-(0-2)	3	1	60	3
tw-3-1-60-4d-(0-2)	3	1	60	4
tw-3-1-60-5d-(0-2)	3	1	60	5
tw-5-2-30-1d-(0-2)	5	2	30	1
tw-5-2-30-2d-(0-2)	5	2	30	2
tw-5-2-30-3d-(0-2)	5	2	30	3
tw-5-2-30-4d-(0-2)	5	2	30	4
tw-5-2-30-5d-(0-2)	5	2	30	5
tw-5-2-60-1d-(0-2)	5	2	60	1
tw-5-2-60-2d-(0-2)	5	2	60	2
tw-5-2-60-3d-(0-2)	5	2	60	3
tw-5-2-60-4d-(0-2)	5	2	60	4
tw-5-2-60-5d-(0-2)	5	2	60	5

Appendix E. ALNS setup evaluation

Appendices E.1 and E.2 present the analyses of different LNS configurations and sub-problem integration setups, respectively.

E.1. LNS configurations

Table E.1 reports the results of the analysis of three different LNS configuration setups. We conclude that the heuristic performs significantly better when a range of operators is applied. The benefit of using adaptive weights when selecting operators as opposed to drawing operators randomly is somewhat smaller, but still evident. Fig. 10 illustrates the average operator weights across the tuning instances when applying the 5000 iterations of the ALNS heuristic with adaptive weights. Lastly, we investigate the weights used in the roulette wheel selection of whether or not to apply noise in the insertion methods. The results are illustrated in Fig. 11.

E.2. Sub-problem integration

We investigate the effect of performing the simplified production feasibility check presented in Section 3.4.7 in different phases of the

Table D.1
Overview of parameters used in the ALNS heuristic. For parameters not subject to tuning, only the final values are reported.

Parameter	Initial value	Final value	Description
$\hat{q}$	$[0.1, 0.4]$	$[0.1, 0.3]$	Interval of uniform distribution from which the share of order nodes to remove from the ALNS solution, q^{ALNS}, is randomly drawn
$\begin{pmatrix} a_0^{(1)} & a_1 \end{pmatrix}$	$(0.01 \quad 0.5)$	$(0.005 \quad 1)$	Related removal; spatial and temporal relatedness weights
$\begin{pmatrix} a_0^{(2)} & a_2 \end{pmatrix}$	$(0.01 \quad 0.1)$	$(0.005 \quad 0.2)$	Related removal; spatial and disease-class relatedness weights
σ_1	33	33	Score for finding new globally improving solution
σ_2	9	9	Score for finding a new locally improving solution
σ_3	13	1	Score for finding new solution
r	0.1	0.2	Reaction parameter
η	0.1	0.1	Noise control parameter
p	5	5	Determinism parameter
η'	0.1	1.0	Destination factory noise control parameter
ϱ	0	0.05	Vessel visit permutation parameter
ϵ	0	0.05	Production solve parameter
κ		0.2	Lower threshold for the adaptive weights
μ		$\sqrt[I^{ALNS}]{0.002}$	Simulated annealing cooling rate; set such that the final temperature is $0.2\% \cdot T_{start}$
T_{start}			Simulated annealing start temperature; set such that the probability of accepting a candidate solution is 50% if the candidate solution is less than 10% worse than the current solution
I^S		40	Number of iterations within one ALNS segment
Θ		50	Maximum number of iterations with the same current solution
T^E		30	Maximum time limit (in seconds) for exact solver to improve the production solution in the final step

Table E.1
Results after running the ALNS heuristic for 10 000 iterations using three different setups. In the first setup (left), only one destroy and one repair operator is used (random removal and greedy insertion). In the second setup (middle), operators are chosen randomly from the full set of operators. In the third setup (right), operators, also from the full set of operators, are chosen based on adaptive weights. The gap values are defined as the percentage difference between the average and the best objective value.

	LNS (one operator pair)				LNS (random)				ALNS (adaptive)			
	Obj. [NOK]	Prod. [%]	Time [s]	Gap [%]	Obj. [NOK]	Prod. [%]	Time [s]	Gap [%]	Obj. [NOK]	Prod. [%]	Time [s]	Gap [%]
t-3-1-20-*l*	679 977	2.38	143	25.98	540 043	3.11	172	0.05	539 887	3.11	171	0.02
t-3-1-20-*h*	149 754	9.52	101	4.71	143 090	9.72	187	0.06	143 141	9.75	166	0.09
t-3-1-40-*l*	652 030	4.60	353	19.90	552 878	6.10	814	1.67	552 317	5.53	694	1.56
t-3-1-40-*h*	280 869	10.07	207	6.43	271 186	10.76	597	2.76	269 876	10.46	515	2.27
t-3-1-60-*l*	455 134	8.80	529	11.47	432 746	9.59	1365	5.98	416 564	10.03	1194	2.02
t-3-1-60-*h*	418 618	11.21	453	13.52	398 307	11.08	1322	8.02	386 158	11.23	1123	4.72
t-5-2-20-*l*	201 791	10.79	96	1.71	198 474	10.86	162	0.04	198 399	10.87	168	0.00
t-5-2-20-*h*	205 139	11.67	115	12.88	182 336	12.09	187	0.33	183 778	12.56	178	1.12
t-5-2-40-*l*	299 560	13.52	283	11.82	287 759	14.20	497	7.42	296 921	14.03	464	10.84
t-5-2-40-*h*	216 228	14.94	213	9.37	210 979	14.56	503	6.72	206 325	16.31	441	4.36
t-5-2-60-*l*	490 012	12.71	512	39.58	352 210	17.66	1213	0.32	374 138	16.86	1061	6.57
t-5-2-60-*h*	492 291	12.11	457	28.21	406 589	13.27	1162	5.89	388 550	14.56	1017	1.19
Average	378 450	10.19	289	15.47	331 383	11.08	682	3.27	329 671	11.28	599	2.90

Table E.2
Results after running the ALNS heuristic for 10 000 iterations using three different sub-problem integration setups. In the first setup (left), the simplified production feasibility check is performed for each insertion, whereas in the second setup (middle), we do not perform it at all. In the third setup (right), the simplified production feasibility check is performed after the repair operator.

	Often (each insertion)				Never				Default (after repair)			
	Obj. [NOK]	Prod. [%]	Time [s]	Gap [%]	Obj. [NOK]	Prod. [%]	Time [s]	Gap [%]	Obj. [NOK]	Prod. [%]	Time [s]	Gap [%]
t-3-1-20-*l*	540 017	3.11	368	0.05	539 911	3.12	159	0.03	540 043	3.11	165	0.05
t-3-1-20-*h*	143 090	9.72	459	0.06	143 118	9.74	153	0.07	143 034	9.74	167	0.02
t-3-1-40-*l*	599 319	5.87	1539	64.77	663 593	4.79	625	82.44	493 861	6.76	670	35.78
t-3-1-40-*h*	287 398	9.96	1465	8.91	268 114	10.82	482	1.60	270 038	10.81	451	2.33
t-3-1-60-*l*	422 449	9.85	4150	3.50	412 182	10.44	1146	0.98	413 882	10.33	1160	1.40
t-3-1-60-*h*	384 925	11.01	4217	4.88	394 734	11.03	1235	7.55	378 307	11.45	1205	3.07
t-5-2-20-*l*	198 474	10.86	342	0.04	198 550	10.85	151	0.08	198 477	10.89	162	0.04
t-5-2-20-*h*	182 692	12.57	411	0.98	182 265	12.50	174	0.75	183 962	12.75	186	1.69
t-5-2-40-*l*	294 375	13.75	1314	10.15	283 794	14.48	476	6.19	298 644	14.01	504	11.75
t-5-2-40-*h*	210 549	14.96	1153	7.02	206 204	16.11	421	4.81	209 587	16.52	388	6.53
t-5-2-60-*l*	360 683	17.44	3595	2.91	357 880	17.49	1067	2.11	373 034	17.38	1103	6.43
t-5-2-60-*h*	387 353	13.98	3744	3.75	397 007	14.51	988	6.34	385 112	13.97	998	3.15
Average	334 277	11.09	1897	8.92	337 279	11.32	590	9.41	323 999	11.48	597	6.02

ALNS heuristic. Table E.2 presents the results. In conclusion, the default setup – where the simplified production feasibility check is applied after the repair operator – has the best performance among the three setups investigated.

References

Absi, N., Archetti, C., Dauzère-Pérès, S., Feillet, D., 2015. A two-phase iterative heuristic approach for the production routing problem. Transp. Sci. 49 (4), 784–795.

Adulyasak, Y., Cordeau, J.-F., Jans, R., 2014a. Optimization-based adaptive large neighborhood search for the production routing problem. Transp. Sci. 48, 20–45.
Adulyasak, Y., Cordeau, J.-F., Jans, R., 2014b. The production routing problem: A review of formulations and solution algorithms. Comput. Oper. Res. 55, 141–152.
Agra, A., Christiansen, M., Ivarsøy, K.S., Solhaug, E., Tomasgard, A., 2017. Combined ship routing and inventory management in the salmon farming industry. Ann. Oper. Res. 253, 799–823.
Azi, N., Gendreau, M., Potvin, J.-Y., 2014. An adaptive large neighborhood search for a vehicle routing problem with multiple routes. Comput. Oper. Res. 41, 167–173.
Belo-Filho, M., Amorim, P., Almada-Lobo, B., 2015. An adaptive large neighbourhood search for the operational integrated production and distribution problem of perishable products. Int. J. Prod. Res. 53 (20), 6040–6058.
Braekers, K., Ramaekers, K., Van Nieuwenhuyse, I., 2015. The vehicle routing problem: State of the art classification and review. Comput. Ind. Eng. 99, 300–313.
Cattaruzza, D., Absi, N., Feillet, D., 2018. Vehicle routing problems with multiple trips. Ann. Oper. Res. 271, 127–159.
Chandra, P., Fisher, M.L., 1994. Coordination of production and distribution planning. European J. Oper. Res. 72, 503–517.
Christiansen, M., Fagerholt, K., Rachaniotis, N.P., Stålhane, M., 2017. Operational planning of routes and schedules for a fleet of fuel supply vessels. Transp. Res. 105, 163–175.
Crevier, B., Cordeau, J.-F., Laporte, G., 2007. The multi-depot vehicle routing problem with inter-depot routes. European J. Oper. Res. 176 (2), 756–773.
Drexl, M., 2012. Synchronization in vehicle routing—A survey of VRPs with multiple synchronization constraints. Transp. Sci. 46, 297–316.
Egil Ulvan Rederi AS, Fartoyene, n.d. [Accessed May 31, 2021]. https://ulvan-rederi.no/fartoyene/with-marine/.
Farahani, P., Grunow, M., Günther, H.O., 2012. Integrated production and distribution planning for perishable food products. Flex. Serv. Manuf. J. 24 (1), 28–51.
Food and Agriculture Organization of the United Nations, 2020. The state of world fisheries and aquaculture 2020. [Accessed May 31, 2021]. http://www.fao.org/state-of-fisheries-aquaculture.
Grønhaug, R., Christiansen, M., 2009. Supply chain optimization for the liquefied natural gas business. In: Nunen, J.A., Speranza, M.G., Bertazzi, L. (Eds.), Innovations in Distribution Logistics. Springer Berlin Heidelberg, pp. 195–218.
Grønhaug, R., Christiansen, M., Desaulniers, G., Desrosiers, J., 2010. A branch-and-price method for a liquefied natural gas inventory routing problem. Transp. Sci. 44 (3), 400–415.
Halvorsen-Weare, E.E., Fagerholt, K., 2013. Routing and scheduling in a liquefied natural gas shipping problem with inventory and berth constraints. Ann. Oper. Res. 203, 167–186.
Halvorsen-Weare, E.E., Fagerholt, K., Nonås, L.M., Asbjørnslett, B.E., 2012. Optimal fleet composition and periodic routing of offshore supply vessels. European J. Oper. Res. 223, 508–517.
Ivarsøy, K.S., Solhaug, I.E., 2014. Optimization of Combined Ship Routing and Inventory Management in the Salmon Farming Industry, (tech. rep.) (Master Thesis). Norwegian University of Science and Technology.
Li, Y., Chu, F., Chu, C., Zhu, Z., 2019. An efficient three-level heuristic for the large-scaled multi-product production routing problem with outsourcing. European J. Oper. Res. 272 (3), 914–927.
Li, Y., Chu, F., Côté, J.-F., Coelho, L.C., Chu, C., 2020. The multi-plant perishable food production routing with packaging consideration. Int. J. Prod. Econ. 221, 107472.
Lianes, I.M., Noreng, M.T., Fagerholt, K., Slette, H.T., Meisel, F., 2021. The aquaculture service vessel routing problem with time dependent travel times and synchronization constraints. Comput. Oper. Res. 134, 105316.
Liu, R., Tao, Y., Xie, X., 2018. An adaptive large neighborhood search heuristic for the vehicle routing problem with time windows and synchronized visits. Comput. Oper. Res. 101.
Manousakis, E.G., Kasapidis, G.A., Kiranoudis, C.T., Zachariadis, E.E., 2022. An infeasible space exploring matheuristic for the production routing problem. European J. Oper. Res. 298 (2), 478–495.
Miranda, P.L., Cordeau, J.-F., Ferreira, D., Jans, R., Morabito, R., 2018. A decomposition heuristic for a rich production routing problem. Comput. Oper. Res. 98, 211–230.
Montoya-Torres, J.R., López Franco, J., Nieto Isaza, S., Felizzola Jiménez, H., Herazo-Padilla, N., 2015. A literature review on the vehicle routing problem with multiple depots. Comput. Ind. Eng. 79, 115–129.
Moons, S., Ramaekers, K., Caris, A., Arda, Y., 2017. Integrating production scheduling and vehicle routing decisions at the operational decision level: A review and discussion. Comput. Ind. Eng. 104, 224–245.
Mowi, 2020. Salmon farming industry handbook 2020. [Accessed January 27, 2021]. https://mowi.com/it/wp-content/uploads/sites/16/2020/06/Mowi-Salmon-Farming-Industry-Handbook-2020.pdf.
Mutlu, F., Msakni, M.K., Yildiz, H., Sönmez, E., Pokharel, S., 2016. A comprehensive annual delivery program for upstream liquefied natural gas supply chain. European J. Oper. Res. 250, 120–130.
Neves-Moreira, F., Almada-Lobo, B., Cordeau, J.-F., Guimarães, L., Jansb, R., 2018. Solving a large multi-product production-routing problem with delivery time windows. Omega 86, 154–172.
Our World in Data, 2019. The world now produces more seafood from fish farms than wild catch. [Accessed June 2, 2021]. https://ourworldindata.org/rise-of-aquaculture.
Qiu, Y., Wang, L., Xu, X., Fang, X., Pardalos, P.M., 2018. A variable neighborhood search heuristic algorithm for production routing problems. Appl. Soft Comput. 66, 311–318.
Rakke, J.G., Stålhane, M., Moe, C.R., Christiansen, M., Andersson, H., Fagerholt, K., Norstad, I., 2011. A rolling horizon heuristic for creating a liquefied natural gas annual delivery program. Transp. Res. C 19 (5), 896–911.
Røpke, S., Pisinger, D., 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transp. Sci. 40, 455–472.
Shaw, P., 1998. Using constraint programming and local search methods to solve vehicle routing problems. In: Principles and Practice of Constraint Programming — CP98. pp. 417–431.
Shyshou, A., Gribkovskaia, I., Laporte, G., Fagerholt, K., 2012. A large neighbourhood search heuristic for a periodic supply vessel planning problem arising in offshore oil and gas operations. INFOR: Inf. Syst. Oper. Res. 50 (4), 195–204.
Solyalı, O., Süral, H., 2017. A multi-phase heuristic for the production routing problem. Comput. Oper. Res. 87, 114–124.
Stålhane, M., Rakke, J.G., Moe, C.R., Andersson, H., Christiansen, M., Fagerholt, K., 2012. A construction and improvement heuristic for a liquefied natural gas inventory routing problem. Comput. Ind. Eng. 62 (1), 245–255.
Statistics Norway, 2020. Oppdrettslaks til heile verda. [Accessed May 31, 2021]. https://www.ssb.no/jord-skog-jakt-og-fiskeri/artikler-og-publikasjoner/oppdrettslaks-til-heile-verda.
Ullrich, C.A., 2013. Integrated machine scheduling and vehicle routing with time windows. European J. Oper. Res. 227, 152–165.
Vadseth, S.T., Andersson, H., Stålhane, M., Chitsaz, M., 2022. A Multi-Start Route Improving Matheuristic for the Production Routing Problem. Working Paper.
Vidal, T., Crainic, T.G., Gendreau, M., Prins, C., 2013. Heuristics for multi-attribute vehicle routing problems: A survey and synthesis. European J. Oper. Res. 231, 1–21.
Yağmur, E., Kesen, S.E., 2021. Multi-trip heterogeneous vehicle routing problem coordinated with production scheduling: Memetic algorithm and simulated annealing approaches. Comput. Ind. Eng. 161, 107649.
Zhen, T., Zhang, Q., 2009. A combining heuristic algorithm for the multi-depot vehicle routing problem with inter-depot routes. In: 2009 International Joint Conference on Artificial Intelligence. pp. 436–439.