European Journal of Operational Research 295 (2021) 908–923

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Production, Manufacturing, Transportation and Logistics

# Exact and heuristic algorithms for the fleet composition and periodic routing problem of offshore supply vessels with berth allocation decisions

Bruno S. Vieira [a,*], Glaydston M. Ribeiro [a], Laura Bahiense [a,b], Roberto Cruz [c], André B. Mendes [c], Gilbert Laporte [d,e]

[a] Transportation Engineering Program Alberto Luiz Coimbra Institute - Graduate School and Research in Engineering Federal University of Rio de Janeiro, Rio de Janeiro, Brazil
[b] Systems Engineering and Computer Science Program Alberto Luiz Coimbra Institute - Graduate School and Research in Engineering Federal University of Rio de Janeiro, Rio de Janeiro, Brazil
[c] Department of Naval Architecture and Ocean Engineering University of São Paulo, São Paulo, Brazil
[d] Canada Research Chair in Distribuition Management HEC Montréal, Montréal, Canada
[e] School of Management, University of Bath, Bath, United Kingdom

ARTICLE INFO

*Article history:*
Received 16 July 2020
Accepted 13 March 2021
Available online 24 March 2021

*Keywords:*
OR in maritime industry
Offshore supply vessel planning
Periodic VRP
Heterogeneous fleet sizing
Berth allocation
B&C algorithm
ALNS heuristic

ABSTRACT

This paper presents a branch-and-cut algorithm and an adaptive large neighborhood search (ALNS) heuristic for the periodic supply vessel planning problem (PSVPP) arising in the upstream offshore petroleum logistics chain. Platform supply vessels support the offshore oil and gas exploration and production activities by transporting all the necessary material and equipment back and forth between offshore units and an onshore supply base according to a delivery schedule. The PSVPP consists of solving a periodic vehicle routing problem and simultaneously determining an optimal fleet size and mix of heterogeneous offshore supply vessels, their weekly routes and schedules for servicing the offshore oil and gas installations, and the berth allocations at the supply base. The branch-and-cut algorithm considers a reduced formulation for the problem which performs much better than the complete one, and easily finds optimal solutions for the smaller and most of the clustered instances. The ALNS heuristic contains new features which include multiple starts and spaced local searches. These algorithms were tested on instances with up to 79 offshore units, providing better results than the best available.



## 1. Introduction

The aim of this paper is to solve a periodic supply vessel planning problem (PSVPP) arising in the upstream offshore petroleum logistics chain of oil and gas exploration and production. The PSVPP variant tackled in this paper models a real-life problem faced by a Brazilian oil and gas exploration and production (E&P) operator. As described in Gribkovskaia, Laporte, and Shlopak (2008) and in Aas, Halskau, and Wallace (2009), the PSVPP is a key problem in this context since it ensures the regular replenishment of equipment and other supplies to offshore units, as well as the return of waste and depleted equipment back to the onshore base. This is usually a high-cost operation, essential to avoid interruptions in the continuous production of oil.

Unifying Halvorsen-Weare and Fagerholt (2017), Kisialiou et al. (2018a) and Cruz et al. (2019), we consider (*i*) the simultaneous determination of the heterogeneous fleet composition, the vessel schedules and the berth allocations; (*ii*) continuous and flexible departures from the base; (*iii*) historical data to model the berth allocations and departures; and (*iv*) the solution for the largest real-world offshore instances ever presented, with up to 79 installations.

We present a branch-and-cut algorithm and an adaptive large neighborhood search (ALNS) heuristic with multiple starts and spaced local searches to solve this PSVPP. We also use a replicable one-week planning horizon both for the offshore units and the vessels, and solve the largest available benchmark instances without dividing them into clusters.

* Corresponding author.
*E-mail addresses:* bruno.vieira@pet.coppe.ufrj.br (B.S. Vieira), glaydston@pet.coppe.ufrj.br (G.M. Ribeiro), laura@cos.ufrj.br (L. Bahiense), redwacruz@yahoo.com.br (R. Cruz), andbergs@usp.br (A.B. Mendes), gilbert.laporte@cirrelt.net (G. Laporte).

https://doi.org/10.1016/j.ejor.2021.03.022
0377-2217/

### 1.1. Problem description

The periodic supply vessel planning problem with berth allocations is defined according to the description provided by a Brazilian oil and gas operator, and unifies the characteristics described in Cruz et al. (2019), Kisialiou et al. (2018a), and Halvorsen-Weare and Fagerholt (2017):

- *Non-stop operations*: No opening hours are considered for either the onshore base or the offshore unit. The onshore base, the offshore units and the vessels operate continuously, seven days a week, without any interruption, and no time windows are considered at the offshore units.
- *Periodic service*: Each offshore unit has its frequency of service, commonly called "frequency of visits", depending mainly on its own demand, which is limited by the required deck area. The production units have a more stable demand, implying a lower frequency of service, typically once or twice a week. In the case of drilling rigs, the demand is more unpredictable, implying a higher frequency of service, typically three or more times per week.
- *Spread departures from the base*: The frequency of attendance to each maritime unit is linked to the vessel departures. In fact, the departures from the onshore base to each offshore unit should be spread as evenly as possible over the planning horizon in order to avoid delays in meeting the delivery requests from the installations. This characteristic and "Periodic service" are satisfied through the service frequency patterns (mentioned before), creating what we call *delivery patterns*.
- *Heterogeneous and common resource fleet*: Each type of vessel has its own deck capacity, a fixed cost over the planning horizon, and a variable cost per travelled distance. In addition, the vessels are not associated to routes, but are considered a common resource to be used whenever requested and with the objective of reducing costs.
- *Route duration*: The vessel capacity limits the maximum number of offshore units that can be visited on the same route, and longer routes are more subject to delays and adverse weather conditions. Hence, the maximum number of installations serviced in the same route cannot exceed eight. As a result, the time required for the routes vary between one and four days, approximately.
- *Two daily berth departures*: The loading times at the onshore base are calculated by considering a fixed setup time (vessel berthing time) added to the variable loading time, which is proportional to the voyage's total demand. The maximum number of departures from each berth on each day is limited to two, based on the actual loading rates and on the historical average amount of cargo transported. Therefore, there is one physical berth position that has to be shared for a maximum of two unloading/loading operations, and each departure from each berth's queue position is called a *departing position*.
- *Berth departures' tolerance*: In order to bring flexibility to the berth planning process, a vessel can begin the loading process on a day and finish it on the next day, given a tolerance of up to 12 h to leave the onshore base. The inclusion of this flexibility in hours on the departure day in the model is important to deal with cases where the port is overloaded.
- *Static routing*: All parameters are known in advance, so no stochastic data are used and the weather impact on the voyage duration is not considered.

In short, the PSVPP tackled here consists of solving a periodic vehicle routing problem by simultaneously determining the optimal fleet size and mix of heterogeneous offshore supply vessels, their weekly routes and schedules for servicing the offshore oil and gas installations, as well as the berth allocations (BA) at the supply base. Hence we will call this problem the PSVPP-BA.

### 1.2. Literature review

The PSVPP belongs to the family of periodic vehicle routing problems (PVRP), an important branch of the classical vehicle routing problem since many practical applications impose periodic visits to the customers during the planning horizon, as shown in Baptista, Oliveira, and Zúquete (2002).

The best available exact and metaheuristic algorithms for the PVRP can be found in Baldacci, Bartolini, Mingozzi, and Valletta (2011) and in Vidal, Crainic, Gendreau, Lahrichi, and Rei (2012). The PSVPP is significantly more complicated than the classical PVRP since the supply vessels make multiple-day voyages, some of the offshore unit have time windows for service, and the vessels departures from the onshore base should be spread as evenly as possible. The literature devoted to the supply vessels planning problems is rather limited. The seminal paper of Fagerholt and Lindstad (2000) dealt with a simplified version of the PSVPP that did not consider the assignment of voyages to the days of the week, and did not directly address the required spread of departures from the base. The candidate routes were generated by considering the night closures of some offshore units. This paper was followed by the works of Aas, Gribkovskaia, Halskau, and Shlopak (2007) and Gribkovskaia et al. (2008), which tackled the pickup and delivery problem involving one single vessel under limited storage capacity at the offshore units and on the vessels. Gribkovskaia et al. (2008) developed a tabu search metaheuristic to solve large instances.

Halvorsen-Weare, Fagerholt, Nonås, and Asbjørnslett (2012) incorporated periodicity in route planning, including the departing day in the route selection decision variable. These authors developed a two-phase algorithm which first generates feasible candidate voyages, and then uses these voyages as an input for a set covering model. Shyshou, Gribkovskaia, Laporte, and Fagerholt (2012) pursued this research by introducing a large neighborhood search heuristic (LNS) to solve large instances of the PSVPP, which was quite efficient compared with the two-phase approach. Halvorsen-Weare and Fagerholt (2017) introduced an alternative arc-flow model which did not perform well as the voyage-based model.

Cuesta, Andersson, Fagerholt, and Laporte (2017) also studied a pickup and delivery problem considering a single and multiple vessels. They added the possibility of not servicing all offshore units by introducing a penalty associated with the losses due to the unattended demand. They also considered the transportation of all cargo to be compulsory, which led to the need of expanding the vessel fleet. Borthen, Loennechen, Wang, Fagerholt, and Vidal (2018) adapted the genetic search heuristic of Vidal et al. (2012) to a simplified version of the PSVPP with a single and fixed departure time, homogeneous fleet of vessels and no time windows. This was followed by Borthen, Loennechen, Fagerholt, Wang, and Vidal (2019) who introduced a bi-objective approach, this time trying to minimize changes from the current solution whenever offshore units are added or removed from the current plan. Kisialiou et al. (2018a) extended the work of Halvorsen-Weare et al. (2012) by allowing flexible departure times from the onshore base within discrete departure slots and considering coupled vessels. They proposed an ALNS heuristic to solve this problem and compared its results with a voyage-based formulation solved by CPLEX.

Bierwirth and Meisel (2015) presented a comprehensive overview of berth allocation problems. An ALNS implementation was made in Mauri, Ribeiro, Lorena, and Laporte (2016) in order to assign ships to berthing positions along a quay in a port. This was followed by Iris, Pacino, and Ropke (2017) who presented

**Table 1**
Scientific works and PSVPP's tackled attributes.

| Work | Attribute | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MV | PR | HF | FS | PD | TW | FD | BA | VC | CU | CP | LI | MO | WS |
| Fagerholt and Lindstad (2000) | X | | | | | X | | | | | | | | |
| Aas et al. (2007) | | | | | X | | | | | X | | | | |
| Gribkovskaia et al. (2008) | | | | | X | | | | | X | | | | |
| Halvorsen-Weare et al. (2012) | X | X | X | X | | X | | | | | | | | |
| Shyshou et al. (2012) | X | X | | X | | X | X | | | X | | | | |
| Halvorsen-Weare & Fagerholt (2017) | X | X | X | X | | X | | | | | | | | X |
| Cuesta et al. (2017) | X | | | | X | | | | | X | | | | |
| Borthen et al. (2018) | X | X | | X | | | | | | | | | | |
| Borthen et al. (2019) | X | X | | X | | | | | | | | | X | |
| Kisialiou et al. (2018a) | X | X | X | X | | X | X | | X | | X | | | |
| Kisialiou et al. (2018b) | X | X | | X | | X | X | | | | | | | X |
| Cruz et al. (2019) | X | X | X | X | | | X | X | | | | | | |
| This Paper | X | X | X | X | | | X | X | | | X | X | | |

improved formulations. Cruz et al. (2019) were the first to simultaneously integrate berth allocation decisions with the determination of the heterogeneous fleet composition and the vessels schedules in the PSVPP, considering continuous and flexible departures from the base and historical data to model the berth allocations and departures. They proposed a multi-step model to solve exactly real-world large instances comprising up to 79 offshore units. Kisialiou, Gribkovskaia, and Laporte (2018b) developed an ALNS heuristic to generate robust supply vessel schedules under stochastic weather conditions, calculating the expected level of service for different sets of vessels.

Table 1 presents a brief summary about the PSVPP's tackled attributes currently available in literature, with the attributes been Multiple vessels (MV), Periodic routing (PR), Heterogeneous fleet (HF), Fleet sizing (FS), Pickup and delivery (PD), Time windows (TW), Flexible departures (FD), Berth allocation (BA), Vessel coupling (VC), Constrained offshore units (CU), Explicit cyclical planning (CP), Non-clustered large real-world instances (LI), Multiple objectives (MO) and Weather simulation (WS).

### 1.3. Scientific contributions

Although the problem tackled in this paper is the one considered by Cruz et al. (2019), our modelling of the PSVPP-BA is new in two important aspects: (*i*) we consider *non-clustered large real-world instances*, expanding the search space, even knowing that the oil company suggest these clusters, which allows to handle a wider range of instances; and (*ii*) we consider a *explicit cyclical planning horizon*, constraining the search space, which makes the generated solutions easier to replicate indefinitely due to the problem's "non-stop operations" characteristic. Both features make the problem more general and harder to solve.

In Cruz et al. (2019), the offshore units are grouped into clusters. A cluster consists of a group of nearby platforms that are served together, as a separate entity from the others. They belong to the same oilfield and are operated by the same onshore base. This segregation is common in the Brazilian offshore E&P, and allows a better management of the logistics process. In this paper we expand the problem to a more broad and challenging one, since the instances are non-clustered. As a consequence, all possible combinations for serving the offshore units are considered.

We present two mathematical formulations, an efficient branch-and-cut algorithm and an ALNS heuristic to solve the largest available benchmark instances. This ALNS implementation constitutes an interesting contribution since it is enriched with multiple starts, oscillating feasibility penalties and spaced local search procedures, all used together.

The remainder of this paper is organized as follows. Section 2 presents the proposed mathematical modelling. Section 3 describes the branch-and-cut algorithm. Section 4 describes the ALNS heuristic. Section 5 details the computational experiments. Lastly, Section 6 presents our main conclusions as well as suggestions for future work.

## 2. Mathematical modelling

This section presents our mathematical models for the PSVPP-BA which, differently from Cruz et al. (2019), couples individual vessels to voyages (routes). Let $G = (N, E)$ be an undirected graph in which the set of nodes $N = N^B \cup N^{OI}$ is made up of the set of berths $N^B$ and the set of offshore units $N^{OI}$, and the set $E = \{(i, j) | i, j \in N, i < j\}$ contains the edges between the nodes in $N$. Every offshore unit $i \in N^{OI}$ is periodically served by a set $V$ of vessels of types in $V'$, and each vessel departs from a berth $b \in N^B$ in a time horizon consisting of a set $T$ of periods that can be infinitely replicated.

Each offshore unit $i \in N^{OI}$ has a demand $q_i$ expressed in square meters ($m^2$), a service time $t_i$ expressed in periods, and must be served $f_i$ times within the time horizon. Thus, a set of delivery patterns $P_i$ is associated to each offshore unit $i \in N^{OI}$, corresponding to $f_i$ services. As mentioned in Section 1.1, the production units are serviced typically once or twice a week, while drilling rigs are commonly serviced at least three times a week. Each vessel type $v \in V'$ has a capacity $U_v$, a travel speed $S_v$, a fixed cost $C_v^f$, and a variable cost $C_v^d$ per distance unit traveled in the time horizon. Since $T$ usually consists of the seven days of the week, each vessel type $v \in V'$ can perform two or three voyages per week. Each berth $b \in N^B$ has a fixed maneuvering and startup time per vessel $L_b^f$, a variable loading rate $L_b^q$ (in terms of period/$m^2$) and a queue capacity $K_b$ with two departing positions $k \in K_b = \{1, 2\}$, due to problem attribute "Two daily berth departures". As mentioned in the Section 1.1, under "Berth departures tolerance", a vessel may start loading in one period $t \in T$ and keep loading during period $t + 1$ with a departure time never exceeding half of that period. Such a tolerance, for each period and berth, is controlled in the model by the constant $TOL = 1.5$.

### 2.1. Complete model

The first proposed mathematical formulation is a voyage-based model in which every voyage $r$ belongs to the set of all non-dominated feasible voyages $R$, composed of a set of sequentially served offshore units $r = \{i, i \in N^{OI}\}$, with total load $Q_r = \sum_{i \in r} q_i$, total distance $D_r$, and total service time $T_r = \sum_{i \in r} t_i$. The voyages are generated by enumerating all combinations of offshore units

with feasible load for the largest vessel and following the "Route duration" problem attribute (Section 1.1), and then a Traveling Salesman Problem (Applegate, Bixby, Chvatál, & Cook, 2006) is solved and the generated route is added to $R$. For each $r \in R$ there is a vessel type $v \in V'$ such that $Q_r \leq U_v$. Each vessel type in $v \in V'$ is associated to a subset $R_v = \{v \in V' \mid Q_r \leq U_v\}$ containing feasible voyages for $v \in V'$.

The constant $M$ is a big positive number and $|T|$ is the total number of periods. The following binary matrices are used in the mathematical formulation: $M^1_{ir}$ indicates whether offshore unit $i \in N^{OI}$ belongs or not to voyage $r \in R$; $M^2_{pt}$ indicates whether delivery pattern $p \in P_i$ allows or not offshore unit $i \in N^{OI}$ to be serviced for some voyage starting in period $t \in T$, and $M^3_{ltvr}$ indicates whether voyage $r \in R_v$ started in period $t \in T$ with vessel type $v \in V'$ is active or not in period $l \in T$. Due to the periodic propriety, $M^3_{ltvr} = M^3_{(l+|T|)tvr}$.

The binary variables are the following: $x_{rtvnbk} = 1$ indicates that route $r \in R_v$ starts at period $t \in T$, with vessel $n \in V$ of type $v \in V'$, on berth $b \in N^B$ and position $k \in K_b$, otherwise, $x_{rtvnbk} = 0$; $y_{vn} = 1$ indicates that vessel $n \in V$ of type $v \in V'$ is used, otherwise, $y_{vn} = 0$; $z_{ip} = 1$ determines that delivery pattern $p \in P_i$ is selected for offshore unit $i \in N^{OI}$, otherwise $z_{ip} = 0$.

The integer variable $t_{vnt}$ represents the number of periods occupied by vessel $n \in V$ of type $v \in V'$ starting at period $t \in T$. The continuous variables are as follows: $t^{WB}_{bt}$ is the waiting time for berth $b \in N^B$ at period $t \in T$ imposed by the previous period; $t^{LB}_{btk}$ represents the loading time for berth $b \in N^B$ in position $k \in K_b$ and period $t \in T$; and finally $t^{WV}_{vnt}$, $t^{LV}_{vnt}$ and $t^{RV}_{vnt}$ respectively define, the wait, the load and the route time for vessel $n \in V$ of type $v \in V'$ starting at period $t \in T$.

The complete mathematical formulation is as follows:

$$\text{Minimize} \quad \sum_{v \in V'} \sum_{n \in V} (C^f_v y_{vn} + \sum_{r \in R_v} \sum_{t \in T} \sum_{b \in N^B} \sum_{k \in K_b} C^d_v D_r x_{rtvnbk}) \tag{1}$$

subject to

$$\sum_{p \in P_i} z_{ip} = 1 \qquad i \in N^{OI} \tag{2}$$

$$\sum_{v \in V'} \sum_{n \in V} \sum_{r \in R_v} \sum_{b \in N^B} \sum_{k \in K_b} M^1_{ir} x_{rtvnbk} = \sum_{p \in P_i} M^2_{pt} z_{ip} \qquad i \in N, t \in T \tag{3}$$

$$\sum_{l \in T} \sum_{r \in R_v} \sum_{b \in N^B} \sum_{k \in K_b} M^3_{ltvr} x_{rtvnbk} \leq y_{vn} \qquad v \in V', n \in V, t \in T \tag{4}$$

$$\sum_{v \in V'} \sum_{n \in V} \sum_{r \in R_v} x_{rtvnbk} \leq 1 \qquad b \in N^B, k \in K_b, t \in T \tag{5}$$

$$t^{LB}_{btk} = \sum_{v \in V'} \sum_{n \in V} \sum_{r \in R_v} (L^f_b + Q_r L^q_b) x_{rtvnbk} \qquad b \in N^B, k \in K_b, t \in T \tag{6}$$

$$t^{WB}_{bt} + \sum_{k \in K_b} t^{LB}_{btk} \leq TOL \qquad b \in N^B, t \in T \tag{7}$$

$$t^{WB}_{b(t+1)} \geq t^{WB}_{bt} + \sum_{k \in K_b} t^{L}_{btk} - 1 \qquad b \in N^B, t \in T \tag{8}$$

$$t^{WB}_{b(|T|+X)} = t^{WB}_{bX} \qquad X \in \mathbb{Z}, b \in N^B, t \in T \tag{9}$$

$$t^{WV}_{vnt} \geq t^{WB}_{bt} + \sum_{\alpha < k} t^{LB}_{bt\alpha} + \left( \sum_{r \in R_v} x_{rtvnbk} - 1 \right) M \qquad v \in V', n \in V, t \in T, b \in N^B, k \in K_b \tag{10}$$

$$t^{LV}_{vnt} = \sum_{r \in R_v} \sum_{b \in N^B} \sum_{k \in K_b} (L^f_b + Q_r L^q_b) x_{rtvnbk} \qquad v \in V', n \in V, t \in T \tag{11}$$

$$t^{RV}_{vnt} = \sum_{r \in R_v} \sum_{b \in N^B} \sum_{k \in K_b} \left( T_r + \frac{D_r}{S_v} \right) x_{rtvnbk} \qquad v \in V', n \in V, t \in T \tag{12}$$

$$t_{vnt} \geq t^{WV}_{vnt} + t^{LV}_{vnt} + t^{RV}_{vnt} \qquad v \in V', n \in V, t \in T \tag{13}$$

$$\sum_{t \in T} t_{vnt} \leq |T| \qquad v \in V', n \in V \tag{14}$$

$$y_{vn} \in \{0, 1\} \qquad v \in V', n \in V \tag{15}$$

$$t^{WB}_{bt} \in [0, TOL - 1] \qquad b \in N^B, t \in T \tag{16}$$

$$t^{LB}_{btk} \geq 0 \qquad b \in N^B, k \in K_b, t \in T \tag{17}$$

$$t^{WV}_{vnt} \geq 0, t^{LV}_{vnt} \geq 0, t^{RV}_{vnt} \geq 0 \qquad v \in V', n \in V, t \in T \tag{18}$$

$$t_{vnt} \in \mathbb{Z}^+ \qquad v \in V', n \in V, t \in T \tag{19}$$

$$z_{ip} \in \{0, 1\} \qquad i \in N^{OI}, p \in P_i \tag{20}$$

$$x_{rtvnbk} \in \{0, 1\} \qquad r \in R_v, t \in T, v \in V', n \in V, b \in N^B, k \in K_b. \tag{21}$$

The objective function (1) minimizes the sum of fixed and variable sailing costs. Constraints (2) and (3) ensures that any selected set of routes fulfills the periodic service for each offshore unit, servicing them with spread departures enforcing the "Periodic service" and "Spread departures from the base" problem attributes (Section 1.1). Constraints (4) guarantee that the active routes in each period $t \in T$ for each vessel $n \in V$ of type $v \in V'$ do not overlap, managing the heterogeneous fleet. Constraints (5) ensure that at most one route $r \in R_v$ with vessel $n \in V$ of type $v \in V'$ using berth $b \in N^B$ and position $k \in K_b$ in period $t \in T$ can exist. Constraints (6) manage the berth loading times. Constraints (7) guarantee that the berth loading and waiting times do not exceed the operational tolerance $TOL$ in any period. Since $TOL = 1.5$, these constraints enforce that the current period cannot take more than half (12h) of the next period, according to the "Berth departures' tolerance" problem attribute (Section 1.1). Constraints (8) and (9) state that if the loading time of some berth exceeds one period, then the exceeding amount of time is carried over the next period as waiting time. Constraints (10)–(14) manage the loading, waiting and total routing times for the vessels, restricting them to the planning horizon. Lastly, Constraints (15)–(21) define the domains of the variables.

In order to illustrate some aspects of this model, Fig. 1 shows a case with routes $R = \{A, B, C, D\}$, one berth $N^B = \{1\}$, using two vessels of type 1, so $V = \{1, 2\}$. Route A loads and departures at period 2, Route B loads at period 6 but departures at period 7, route C loads at period 7 after route B and departures at period 1 and route D loads and departures at period 4. Since no pair of routes starts at the same period, all routes use only the first berth position. Thus, the set of variables equal to one is $X = \{x_{A,2,1,1,1,1,1}, x_{B,6,1,1,1,1,1}, x_{C,7,1,2,1,1,1}, x_{D,4,1,2,1,1,1}\}$. Let the total loading time to routes A, B, C and D be, respectively, 0.6, 1.3, 0.9 and 0.5 periods, so $t^{LB}_{1,2,1} = 0.6$, $t^{LB}_{1,4,1} = 0.5$, $t^{LB}_{1,6,1} = 1.3$ and $t^{LB}_{1,7,1} = 0.9$,

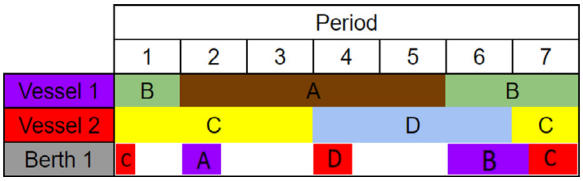


**Fig. 1.** Example of a vessel and berth cyclical scheduling.

following Constraints (7). Considering Constraints (8) and (7), for period 6, $t^{WB}_{1,7} \geq 1.3 - 1$ and $1.3 \leq 1.5$. For period 7, the exceeding loading time from period 6 (0.3) is carried as the waiting berth time $t^{WB}_{1,7}$, so following Constraints (8) and (7), for period 6, $t^{WB}_{1,8} \geq 0.3 + 0.8 - 1$ and $0.3 + 0.9 \leq 1.5$ and they are all satisfied, with Constraints (9) forcing the exceeding loading time from period 7 (0.2) to be carried to period 1 with $t^{WB}_{1,7+1} = t^{WB}_{1,1}$.

Finally, Constraints (4) verify in each time period $t \in T$, for each vessel $n \in V$ of type $v \in V'$, all the routes that have started in any previous period and are still active in period $t$. Therefore, they are necessary restrictions to avoid collisions when starting routes for each vessel, but they are not sufficient since they do not consider the berth dependant loading times. Let $s^{comp}$ be an integer solution of model (1)–(21). Let $S^{comp}_x$ be the set composed by the "$x_{rtvnbk}$ part" of $s^{comp}$ such that $x_{rtvnbk} = 1$. Then, $S^{comp}_x$ is a set of vectors representing only the selected routes. In order to illustrate the non-sufficiency of Constraints (4), suppose that $S^{comp}_x = \{x_{37,2,3,1,1,2}, x_{24,2,3,2,1,1}, x_{82,5,3,1,1,1}, x_{56,6,3,2,1,1}\}$ and, in period 2, vessel 1 of type 3 has a waiting time equal to 0.25 day, and a joint loading and routing time equal to 2.80 days. Then, we have one berth with two positions, and two vessels of type 3 performing two routes each. Vessel 1 departs at $t = 2$ in route 37 and at $t = 4$ in route 82, and vessel 2 departures at $t = 2$ in route 24 and at $t = 6$ in route 56. This solution respects Constraints (4), but is infeasible in practice. In fact, the total route time for vessel 1 of type 3 is equal to $\lceil 2.80 + 0.25 \rceil = 4$, thus this vessel will not be able to perform route 82 at $t = 4$. For this example, constraint $x_{37,2,3,1,1,2} + x_{24,2,3,2,1,1} + x_{82,5,3,1,1,1} + x_{56,6,3,2,1,1} \leq 3$ could be included in (1) - (21) to cut $s^{comp}$. However, only the set $\{x_{37,2,3,1,1,2}, x_{24,2,3,2,1,1}, x_{82,5,3,1,1,1}\}$ of variables is required to avoid the overlap, then it should be sufficient to use constraint $x_{37,2,3,1,1,2} + x_{24,2,3,2,1,1} + x_{82,5,3,1,1,1} \leq 2$.

Constraint (22), derived from $s^{comp}$, cuts this solution if it contains any overlapping route not considered by Constraints (4) in which we consider only the variables responsible for each overlap.

$$\sum_{(r,t,v,n,b,k) \in S^{comp}_x} x_{rtvnbk} \leq |S^{comp}_x| - 1. \tag{22}$$

Consequently, in a branch-and-bound tree, whenever a new integer solution $s^{comp}$ of model (1)–(21) is found, a new Constraint (22), derived from $S^{comp}_x$, is added to (1)–(21) within a branch-and-cut framework.

### 2.2. Reduced model

The complete model (1)–(21) suffers from the curse of symmetry: it is possible to obtain different solutions, of the same objective value, by changing vessels of the same type or swapping berth schedules. The solutions obtained are different, but the routes, berth scheduling, fleets and related costs are the same. Another weakness of this model is the presence of the "bigM" coefficient in Constraints (10). These characteristics make the model (1)–(21) hard to solve.

In order to overcome these difficulties, we propose a branch-and-cut framework based on the better-conditioned reduced formulation (23)–(31). This formulation considers the binary variables $x_{rtv}$ and $z_{ip}$. If $x_{rtv} = 1$, route $r \in R_v$ starts at period $t \in T$, with vessel type $v \in V'$; otherwise, $x_{rtv} = 0$. If $z_{ip} = 1$, delivery pattern $p \in P_i$ is selected for offshore unit $i \in N^{OI}$, as in (1)–(21); otherwise, $z_{ip} = 0$. It also considers the integer variables $y_v$ which indicates how many vessels of type $v \in V'$ are used. Finally, if $c_{vtl} = 1$ indicates that some route with vessel type $v \in V'$, starting at period $t \in T$ and active for $l \in T$ periods is used; otherwise, $c_{vtl} = 0$. The reduced mathematical formulation is as follows:

$$\text{Minimize} \quad \sum_{v \in V'} C^f_v y_v + \sum_{t \in T} \sum_{v \in V'} \sum_{r \in R_v} C^d_v D_r x_{rtv} \tag{23}$$

subject to

$$\sum_{p \in P_i} z_{ip} = 1 \qquad i \in N^{OI} \tag{24}$$

$$\sum_{v \in V'} \sum_{r \in R_v} M^1_{ir} x_{rtv} = \sum_{p \in P_i} M^2_{pt} z_{ip} \qquad i \in N, t \in T \tag{25}$$

$$\sum_{l \in T} \sum_{r \in R_v} M^3_{ltvr} x_{rlv} \leq y_v \qquad t \in T, v \in V' \tag{26}$$

$$\sum_{v \in V'} \sum_{r \in R_v} x_{rtv} \leq D \qquad t \in T \tag{27}$$

$$y_v \in \mathbb{Z}^+ \qquad v \in V' \tag{28}$$

$$z_{ip} \in \{0, 1\} \qquad i \in N^{OI}, p \in P_i \tag{29}$$

$$c_{vtl} \in \{0, 1\} \qquad v \in V', t \in T, l \in T \tag{30}$$

$$x_{rtv} \in \{0, 1\} \qquad r \in R_v, t \in T, v \in V'. \tag{31}$$

The objective function (23) minimizes the sum of fixed and variable sailing costs. Constraints (24) and (25) ensure the periodic service for each offshore unit. Constraints (26) allocate the fleet according to the maximum number of active routes per period and per vessel type. Constraints (27) ensure that no period exceeds the maximum number of departures $D$, which is twice the number of available berths. Constraints (28)–(31) define the domain of the variables. The sets of indexes and parameters are the same as defined in (1)–(21).

Although being similar to the models of Halvorsen-Weare and Fagerholt (2017) and Kisialiou et al. (2018a), the reduced formulation (23) - (31) is a voyage (route) based model that performs the route scheduling and handles the fleet sizing while having a smaller number of variables and no symmetry or "BigM" coefficients.

Analogously to Constraints (4), Constraints (26) are also necessary but not sufficient to avoid collisions when starting routes for each vessel in model (23)–(31). Let $s^{red}$ be an integer solution of (23)–(31), obtained in some node of its branch-and-bound tree. Let $S^{red}_x$ be the "$x_{rtv}$ part" of $s^{red}$ such that $x_{rtv} = 1$. Constraints (32), derived from $s^{red}$, are generated through a separation procedure for each new incumbent solution in order to bring the solution space from (23)–(32) to (1)–(22) as detailed below.

$$\sum_{(r,t,v) \in S^{red}_x} x_{rtv} \leq |S^{red}_x| - 1. \tag{32}$$

Finally, as Constraints (26) have a larger search space than Constraints (4), the difference between them is fixed by Constraints (33)–(34), dynamically generated through a separation procedure as well, with $S^{red}_{xtv}$ being the variables from $S^{red}_x$ with vessel of

type $v$, $F_v$ is the correct fleet size and $l_{rv}$ is the amount of periods route $r \in R$ remains active with vessel type $v \in V'$, i.e. $l_{rv} := \lceil LoadTime(rv) + RouteTime(rv) \rceil$. Constraints (34) can all be statically generated, however in out experiments, dynamically generating them showed better results for non-clustered instances. The details of how the search spaces differ are explained in Section 3.2.

$$y_v - \sum_{(r,t,v) \in S^{red}_{xv}} c_{vtl_{rv}} \geq F_v - |S^{red}_{xv}| \tag{33}$$

$$c_{vtl_{rv}} \geq x_{rtv} \qquad (r,t,v) \in S^{red}_{xv}. \tag{34}$$

## 3. Branch-and-cut algorithm

The complete model (1)–(21) contains a berth scheduling subproblem which is in fact a feasibility problem, so it does not interfere on the objective function, but adds two indices, various constraints and variables for managing these constraints. This was one of the motivations for proposing the reduced model (23)–(31).

So, whenever an incumbent solution is generated for the reduced model, we can try to generate a feasible berth schedule from it. If such a solution is found, it is accepted; otherwise, it can be used to generate a suitable cut to reinforce the reduced model in order to approximate its solution space to that associated with the complete formulation. As mentioned in Codato and Fischetti (2006), when cuts based on infeasible sets are minimal and generated within low computational times, this approach generally produces positive results.

Therefore, this process of transforming an incumbent reduced solution into a feasible solution for the complete formulation consists in solving a feasibility subproblem that will be called *SubProblem*, and will be detailed in Section 3.1. However, this process is not simple, and requires a study of the reasons for infeasibility, detailed in Section 3.2, in order to guarantee the generation of valid cuts for the separation method, which will be detailed in Section 3.3.

### 3.1. Feasibility subproblem

The feasibility subproblem *SubProblem* receives an incumbent solution $s^{red}$ of the reduced problem (24)–(32) comprising a heterogeneous fleet and a set of selected routes with fixed vehicle types and periods. To transform this incumbent solution into a feasible solution for the complete problem, it is necessary to determine, for each route a particular vessel $n \in V$, a berth $b \in N^B$, and a berth position $k \in K_b$, while verifying constraints (2)–(22).

In this way, *SubProblem* can be formally described as [(4)$^\star$–(21)$^\star$, (35)] where $^\star$ means that $r, t$ and $v$ are fixed in the values obtained from the "$x_{rtv}$ part" of $s^{red}$ in these constraints.

$$\sum_{n \in V} \sum_{b \in N^B} \sum_{k \in K_b} x_{rtvnbk} = 1 \qquad (r,t,v) \in S^{red}_{x}, \tag{35}$$

If *SubProblem* returns a feasible solution, it will be feasible solution for the complete problem with a heterogeneous fleet, and a set of routes so that each route has a departure period, type and vessel identification, berth and berth position, while respecting all berth scheduling constraints; otherwise, the causes of infeasibility must be analyzed to generate an appropriated cut, as detailed in Section 3.2.

### 3.2. Causes of infeasibility

There are two cases in which an incumbent solution $s^{red}$ for the reduced model cannot be feasible for the complete model: the impossibility of allocating routes to vessels, and the impossibility of allocating vessels to berths.

In the reduced formulation, Constraints (26) control the size and mix of the fleet, according to the number of active routes per period, and the solution $s^{red}$ satisfies these constraints. However, this fleet may not be sufficient to satisfy Constraints (4) of the complete problem, leading to the first cause of infeasibility – the impossibility of allocating routes to vessels.

To illustrate this situation, Fig. 2(a) shows a possible solution of the reduced formulation that comprises a set of routes with defined periods for a given vessel type. Constraints (26) define a fleet of $y_v = 3$ vessels, since in each period there is a maximum of three active routes. Therefore, in order to transform the $s^{red}$ solution of Fig. 2(a) into a feasible solution to the complete formulation, these routes must be served by three vessels and simultaneously verify Constraints (4). However, Fig. 2(b) shows a case that, at best, there will always be a route that will not be allocated to a vessel.

There is a pair of well-known combinatorial problems that explains what happens in this first case. Let $G = (V, E)$ be a graph, representing Fig. 2, in which every vertex $i \in V$ is a route with fixed departure time and every edge $(i, j) \in E$ means that routes $i$ and $j$ are active at some period. So, finding the right-hand side $y_v$ of Constraints (26) is equivalent to determining the maximum clique size of $G$, known as $\omega(G)$. On the other hand, transforming $y_v$ into $y_{vn}$ of Constraints (4) is equivalent to solving the minimum vertex colouring (MVC) problem over $G$. There is a well-known relation in graph theory stating that $\omega(G) \leq MVC(G)$ for any graph $G$ (Bondy & Murty, 1976). Therefore, as shown in Fig. 2, and from the previous graph relation, we may need more vessels when transforming a feasible solution $s^{red}$ into a feasible solution $s^{comp}$.

The inconvenience is that solving the minimum vertex colouring problem over a generic graph $G$ is NP-hard, since its decision version is NP-complete (Karp, 1972). In Kisialiou et al. (2018a), a voyage-based model based on flexible departures from the base and the possibility of coupling vessels by swapping their schedules was proposed. Their model was solved exactly on small- and medium-size instances. Generalizing these concepts, Cruz et al. (2019) introduced the integration of berth allocation decisions to the fleet composition and periodic routing problem, solving the largest available instances. Both considered a weekly planning horizon for the installations and a two-week planning horizon for the vessels.

Conversely, we decided to use a weekly cyclical horizon for both vessels and installations, and to solve the reduced problem within a branch-and-cut framework in such a way that whenever an incumbent solution $s^{red}$ is infeasible only due to fleet allocation, as illustrated in Fig. 2, a cut of type (33) based on $s^{red}$, for each vessel type $v$ with problem allocating routes to vessel, with $F_v$ being the correct fleet size (minimal colouration) for the vessel type $v$, is added to the reduced formulation aiming to approximate its solution space to that associated with the complete formulation. This approach works in our context since the colouring problems that we will have to solve are not very large.

The second cause of infeasibility occurs when an incumbent solution $s^{red}$ for the reduced model cannot be transformed into a feasible solution for the complete model due to the impossibility of allocating vessels to berths. To illustrate this situation, Fig. 3(a) shows a possible solution for the reduced formulation that is composed of one berth, five routes allocated to a vessel of type 1, and two routes allocated to a vessel of type 2. We can see that for the vessel of type 1, there are at most two simultaneously active routes per period, while for the vessel of type 2, this number is reduced to at most one active route per period. Therefore, Constraints (26) define a fleet of two vessels of type 1 and one vessel of type 2. In Fig. 3, "VT1" and "VT2" denote vessel types; "L + R

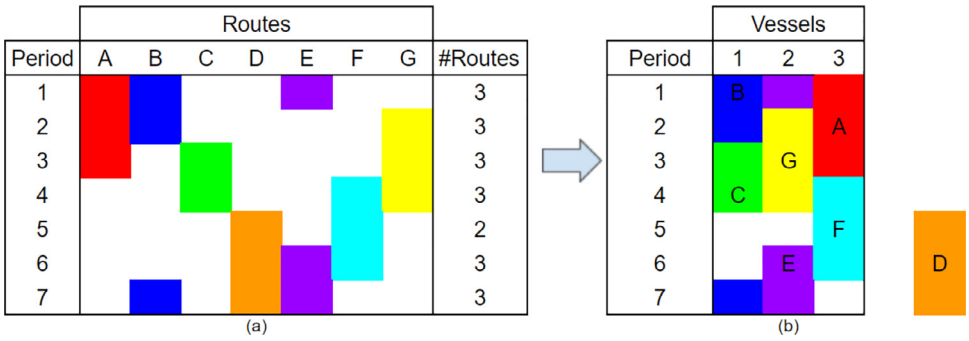


**Fig. 2.** First cause of infeasibility: the impossibility of allocating routes to vessels.

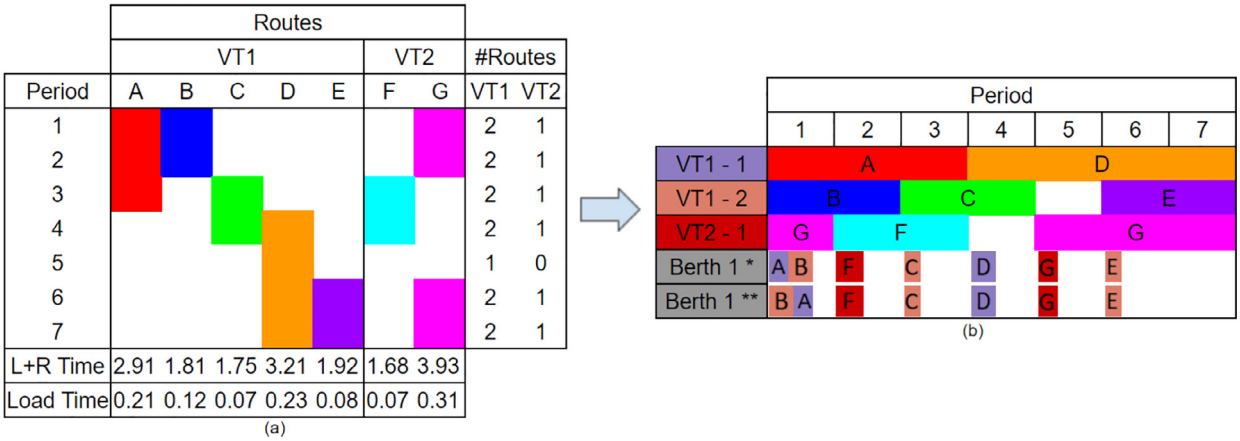


**Fig. 3.** Second cause of infeasibility: the impossibility of allocating vessels to berths.

Time" represents the loading time plus the route time; and "Load time" indicates only the loading time. Fig. 3(b) shows the only feasible vessel allocation to routes shown in Fig. 3(a), represented by VT1 - 1, VT1 - 2 and VT2 - 1, and the two possibles berth allocations, indicated by * and **.

As previously mentioned, when analyzing only the routes and vessel types, the fleet composed of two vessels of type 1 and one vessel of type 2 is sufficient to handle the five routes of Fig. 3(a). For example, Routes A and D can be allocated to the first vessel of type 1, Routes B, C and E to the second vessel of type 1, and finally Routes F and G can be allocated to the vessel of type 2 as illustrated by Fig. 3(b), VT1 - 1, VT1 - 2 and VT2 - 1.

However, since there is a single berth, an infeasibility is generated when considering the loading and route times of each route. For instance, if route A is the first to be executed, illustrated by Fig. 3(b), Berth 1 *, the waiting time for loading route B will be 0.21, and this route will have a total duration of 0.21 + 1.81 = 2.02. As a consequence, route B will be active in period 3, overlapping route C. This infeasibility also occurs if route B is the first to be executed illustrated by Fig. 3(b), Berth 1 **. In this case, the waiting time for loading route A will be 0.12, and this route will have a total duration of 0.12 + 2.91 = 3.03. Subsequently, route A will be active in period 4, overlapping route D. Since there is no other way to allocate the five routes to the two types of vessels and there is only one berth, it is impossible to transform the incumbent solution $s^{red}$ of the reduced model into a feasible solution $s^{comp}$ for the complete formulation. Nonetheless, note that if route C did not exist, it would be possible to allocate Routes A and E to a vessel of type 1 and routes B and D to another vessel of type 1, without generating overlaps.

### 3.3. Separation procedure

The separation procedure presented in Algorithm 1 generates Constraints (32) dynamically, in order to approximate the search space of the reduced model to that of the complete model. This procedure receives an incumbent solution $s^{red}$ of the reduced model and then solves a minimum colouring problem for each type of vessel, as shown in Section 3.2; as a result, we obtain the necessary fleet to serve the routes of $s^{red}$ (Line 1). If the fleet deter-

**Algorithm 1:** Separation procedure.

**Input**: Incumbent $s^{red}$
1 $Fleet \leftarrow MinimalColouring(s^{red})$;
2 **if** $Fleet = Fleet(s^{red})$ **then**
3  **if** $SubProblem(s^{red})$ *is infeasible* **then**
4  $s^{red} \leftarrow Reduce(s^{red})$;
5  Add cut (32) to (23)-(31) using $s^{red}$;
6 **else**
7  $Fleet(s^{red}) \leftarrow Fleet$;
8  **if** $SubProblem(s^{red})$ *is infeasible* **then**
9  $s^{red} \leftarrow Reduce(s^{red})$;
10  Add cut (32) to (23)-(31) using $s^{red}$;
11  **else**
12  Add cut (33) to (23)-(31) for each $v$ with conflicting minimal colouring, using $s^{red}$;

mined by the colouring problem is the same one used in $s^{red}$, the feasibility *SubProblem*, is solved (Line 2). If this problem is infeasible (Line 3), as in the example shown in Fig. 3, then the solution $s^{red}$ is reduced, according to Algorithm 2 (Line 4), and a cut (32) is

**Algorithm 2:** Reduce.

**Input**: Solution $s^{red}$
**Output**: Minimal Solution $s^{red}$
1 **forall the** $(r, t, v) \in S_x^{red}$ **do**
2  Remove $x_{rtv}$ from $s^{red}$;
3  **if** *SubProblem*$(s^{red})$ *is feasible* **then**
4  Insert $x_{rtv}$ into $s^{red}$;
5 **return** $s^{red}$;

added to the reduced problem (Line 5). Conversely, if the fleet determined by the colouring problem is different from that indicated by $s^{red}$ (Line 6), as in the example shown in Fig. 2, the $s^{red}$ fleet is replaced with the one generated by the colouring problem (Line 7). If the feasibility *SubProblem* is infeasible (Line 8), as in the example shown in Fig. 3, then the solution $s^{red}$ is reduced, according to Algorithm 2 (Line 9), and a cut (32) is added to the reduced problem (Line 10). If the feasibility *SubProblem* is feasible, a cut (33) is added to the reduced problem (Line 12). Algorithm 2 is used in Lines 4 and 9 of the Algorithm 1. It tries to reduce the number of variables $x_{rtv}$ from $s^{red}$ (Line 2) which allows a stronger cut (32) for the reduced problem.

## 4. Adaptive large neighborhood search heuristic

Introduced by Ropke and Pisinger (2006a), ALNS extends the metaheuristic large neighborhood search (LNS) proposed by Shaw (1998), which is based on the principle of destruction and reconstruction. At each iteration, ALNS applies an operator to *destroy* a solution $s$ and *reconstruct* it in a different way, thus generating a new solution $s'$. This new solution is accepted according to the simulated annealing (SA) acceptance criterion (Hwang, 1988): if $s'$ is better than $s$, the search continues from $s'$, otherwise the search continues from $s'$ with a given probability. What differentiates ALNS from LNS is that in LNS the destruction (removal) and construction (insertion) operators are chosen with the same probabilities, whereas in ALNS this selection is made according to an adaptive mechanism by which, at each iteration, the probability of selecting a method depends on its past performance by the adaptive layer.

The ALNS effectiveness was tested on various vehicle routing problems due to its ease of adaptation to many problems and yielded good results. This work couples the ALNS well tested routing decisions with multiple starts in order to explore a larger set of fleet distributions and spaced local searches to speed up the search.

### 4.1. Metaheuristic overview

Already implemented in some works (see Bräysy, Hasle, & Dullaert, 2004; Koç, Jabali, & Laporte, 2018; Palomo-Martínez, Salazar-Aguilar, & Laporte, 2017), the multi-start scheme works with three parameters which are the number of starts $\alpha$, the number of iterations per start $\beta$, and the total number of iterations $\gamma$. Algorithm 3 presents the main procedure. It receives the general problem data, then generates $\alpha$ initial solutions and applies an ALNS (Algorithm 4 ) limited to $\beta$ iterations. The best generated solution $s^*$ over the $\alpha$ starts is then given to one more ALNS as an initial solution, with a limit of $\gamma - \alpha\beta$ iterations.

**Algorithm 3:** Multi-start metaheuristic.

**Input**: Graph $G$, set of periods $T$, set of vessels $V$ and set of global parameters
**Output**: Best feasible solution found $s^*$
1 $S \leftarrow \emptyset$;
2 **for** $k = 1$ *to* $\alpha$ **do**
3  Build an initial solution $s$;
4  $s \leftarrow ALNS(s, \beta)$;
5  $S \leftarrow S \cup \{s\}$;
6 $s^* \leftarrow s_l \in S | \forall s \in S \rightarrow C(s_l) \leq C(s)$;
7 $s^* \leftarrow ALNS(s^*, \gamma - \alpha\beta)$;
8 **return** $s^*$.

**Algorithm 4:** ALNS($s_{best}$, $N$).

**Input**: Initial Solution $s_{best}$ and maximum number of iterations $N$
**Output**: Best feasible solution found $s_{feasible}$
1 Initializes the weights for the insertion and removal operators;
2 $s_{curr} \leftarrow s_{best}$; $\theta \leftarrow C(s_{best})\theta_0'$; $C(s_{feasible}) \leftarrow \infty$; $C(s_{prom}) \leftarrow \infty$;
3 **for** *iteration* $= 1$ *to* $N$ **do**
4  $s_{aux} \leftarrow s_{curr}$;
5  Select a removal operator $o^-$ and apply it to $s_{aux}$;
6  Select a insertion operator $o^+$ and apply it to $s_{aux}$;
7  **if** $C(s_{aux}) \leq C(s_{prom})$ **then**
8  $s_{prom} \leftarrow s_{aux}$;
9  **if** *iteration is multiple of* $\delta$ **then**
10  $s_{aux} \leftarrow LocalSearch(s_{prom})$;
11  $C(s_{prom}) \leftarrow \infty$;
12  **if** $s_{aux}$ *is feasible and* $C(s_{aux}) \leq C(s_{feasible})$ **then**
13  $s_{feasible} \leftarrow s_{aux}$;
14  **if** $C(s_{aux}) \leq C(s_{best})$ **then**
15  $s_{best} \leftarrow s_{aux}$; $s_{curr} \leftarrow s_{aux}$;
16  **else**
17  **if** $C(s_{aux}) \leq C(s_{curr})$ **then**
18  $s_{curr} \leftarrow s_{aux}$;
19  **else**
20  **if** $Accept_{SA}(C(s_{aux}), C(s_{curr}), \theta)$ **then**
21  $s_{curr} \leftarrow s_{aux}$;
22  Update scores $\pi_{o^-}$ and $\pi_{o^+}$;
23  **if** *iteration is multiple of* $\rho$ **then**
24  Update penalties' weights;
25  Update insertion and removal operators' weights;
26  $\theta \leftarrow \mu\theta$;
27 **return** $s_{feasible}$.

Algorithm 4 presents the general framework of the ALNS implemented in this work. It receives an initial solution as the best known solution $s_{best}$ and the maximum number of iterations $N$. It then initializes the weights for the insertion and removal operators, the current solution $s_{curr}$, the temperature $\theta$, the best feasible solution found $s_{feasible}$, which is stored, and the most promising solution $s_{prom}$ identified in the last $\delta$ iterations, the one to which the local searches are applied.

Lines 3–26 constitute the main loop. An auxiliary solution $s_{aux}$ is first defined, and removal and insertion operators are applied. From Lines 7–11, the most promising solution $s_{prom}$ is updated. If the current iteration is multiple of $\delta$, a local search is applied to $s_{prom}$, the solution found is kept as $s_{aux}$, and $s_{prom}$ is reset. Lines 12–13 update the best known feasible solution found $s_{feasible}$. Lines 14–26 describe the usual SA mechanism and the ALNS acceptance flow which includes updating the scores and the weights used in the selection of the insertion and removal operators when the current iteration is multiple of $\rho$. We use a penalized objective function with adjustable weights which are updated in Line 24. The related update process is described in the next section. At the end of the main loop, the best feasible solution found $s_{feasible}$ is returned.

### 4.2. Search space and feasibility

The search space is computed as [(2)–(21)] and by two types of violation: (i) load violations associated with vessels exceeding their capacities; and (ii) time violations associated with vessels performing voyages concurrently in the same period (Constraints (4)), or berths occupying too much loading time from one period to the next (Constraints (8)) or exceeding the total loading time in the planning horizon (Constraints (7)).

Let $O_v^t$ and $E_v^q$ be the total overlapping time and exceeding load, respectively, for vessel $v$; let $E_{ct}^t$ be the total loading time exceeded (greater than $TOL$) for berth $b \in N^B$ in period $t \in T$; and let $E_b^{tt}$ be the total loading time exceeded in the planning horizon for berth $b \in N^B$. Each type of violation has an associated weight: $\omega^q > 0$ and $\omega^t > 0$ are the penalty weights for load and time violations, respectively. Hence, a cost for violations is computed as:

$$CV(s, \omega^q, \omega^t) = \omega^q \sum_{v \in V} E_v^q(s) + \omega^t \left( \sum_{v \in V} O_v^t(s) + \sum_{b \in N^B} \left( E_b^{tt}(s) + \sum_{t \in T} E_{ct}^t(s) \right) \right). \quad (36)$$

Considering that the terms of Eq. (36) are non-negative, solution $s$ is feasible if and only if $CV(s, \omega^q, \omega^t) = 0$. Finally, the penalized objective function cost for a solution $s$ is given by $C(s) = C'(s) + CV(s, \omega^q, \omega^t)$ where $C'(s)$ is given by objective function (1). The penalty weights have equal initial values $\omega_0$ which are adjusted for each segment of $\rho$ iterations. If the best solution for the ALNS ($s_{best}$) is infeasible for the load constraints, $\omega^q \leftarrow \tau^+ \omega^q$ with $\tau^+ > 1$, otherwise $\omega^q \leftarrow \tau^- \omega^q$ with $\tau^- \leq 1$. This procedure is also used for the weight $\omega^t$ when $s_{best}$ violates time constraints.

### 4.3. Initial solution

The general strategy is to diversify the initial fleet with poor routing sequences, so that the ALNS heuristic can iteratively reduce the fleet size and improve the routing. Every initial solution is generated in two phases called Fix and Split. The Fix phase assigns to each offshore unit $u \in N^{OI}$ a randomly selected delivery pattern $p \in P_u$. Thus, each period $t \in T$ has a list $L_t$ of offshore units to be split into voyages made by the available vessels. In the Split phase, for each period $t \in T$ a vessel $v \in V$ is randomly selected and installations $u \in L_t$ are randomly chosen to be serviced by vessel $v$ respecting its capacity. When the capacity of vessel $v$ is reached, a berth is assigned to it. While list $L_t$ is not empty, a new vessel is randomly selected and the process is reiterated. When all periods have been evaluated, the initial solution is obtained.

### 4.4. Removal and insertion operators

At each iteration of the ALNS heuristic, $n^- \in [n_1^- |N^{OI}|, n_2^- |N^{OI}|]$ offshore units are removed from $s_{aux}$ by a removal operator $o^-$ and reinserted by an insertion operator $o^+$. The parameters $n_1^-$ and $n_2^-$ define the range of the search around the current solution $s_{curr}$ (Ropke & Pisinger, 2006a).

Our ALNS uses five removal and three insertion operators based on algorithms proposed by Ropke and Pisinger (2006b), Ribeiro and Laporte (2012) and Koç et al. (2018). We also tested the removal operators 'average cost per unit' from Paraskevopoulos, Repoussis, Tarantilis, Ioannou, and Prastacos (2008), 'historical node-pair' and 'cluster removal' from Pisinger and Ropke (2007), as well as other Shaw (1998) variants, but after extensive computational experiments, we realized that the best results were obtained with the five removal and the three insertion operators described as follows.

For the removal operators, we use three Shaw variants (RO1, RO2 and RO3), which are based on Shaw (1998), a random removal (RO4) and a worst-removal procedure (RO5). The Shaw variants try to remove similar offshore units. The similarity between offshore units $u, v \in N^{OI}$ is given by Eq. (37), where $d_{uv}$ is the distance between the installations, $D$ is the maximum distance between any two offshore units, $q_u$ and $q_v$ are demands of the offshore units, $Q$ is the maximum demand found for all offshore units, $f_u$ and $f_v$ are the frequencies of the offshore units, and $|T|$ is the number of periods; $\phi_1$, $\phi_2$ and $\phi_3$ are positive weights:

$$Shaw(u, v) = \phi_1 \frac{d_{uv}}{D} + \phi_2 \frac{|q_u - q_v|}{Q} + \phi_3 \frac{|f_u - f_v|}{|T|}. \quad (37)$$

Let $L^-$ be the list of removed offshore units. Initially, the Shaw variants start by randomly selecting one offshore unit $u \in N^{OI}$ to be removed, thus $L^- \leftarrow \{u\}$. The remaining offshore units $v \in N^{OI} \setminus L^-$ are sorted in non-decreasing order according to Eq. (37). Then, the operator selects a random number $a \in (0, 1]$ and the offshore unit $v \in N^{OI} \setminus L^-$ with index $\lceil a^p |N^{OI} \setminus L^-| \rceil$ in $N^{OI} \setminus L^-$, is removed. The parameter $p \geq 1$ must be tuned. Higher values of $p$ increases the probabilities to select offshore units of $N^{OI} \setminus L^-$ with lower Shaw distances. The list of removed offshore units is updated: $L^- \leftarrow L^- + \{v\}$. This process is reiterated until the number of removed installations is reached ($n^-$). At each iteration, the initial offshore unit $u$ is selected from $L^-$.

On this basis, with the $\phi$ values indicated by Ropke and Pisinger (2006b) and Koç et al. (2018), the following Shaw variants are:

- *Shaw removal (RO1)* uses $\phi_1 = 0.4$, $\phi_2 = 0.3$ and $\phi_3 = 0.3$;
- *Shaw neighbours removal (RO2)* uses $\phi_1 = 1$, $\phi_2 = 0$ and $\phi_3 = 0$; and
- *Shaw frequency removal (RO3)* uses $\phi_1 = 0.2$, $\phi_2 = 0$ and $\phi_3 = 0.8$.

The *Random removal (RO4)* operator removes randomly offshore units until reaches $n^-$. This operator generates a poor set of removed offshore units, but this helps diversify the search.

The *Worst removal (RO5)* operator also uses the removal strategy used in the Shaw operator. In this case, each non-removed offshore units $v$ is assigned a cost which is the difference of the solution with and without it. All non-removed offshore units are sorted in non-decreasing order based on this cost and the removal operator used (that which uses $a \in (0, 1]$ and $p \geq 1$) and the Shaw operator is applied. This process is reiterated until $n^-$ offshore units have been reached. Conversely, every insertion operator $o^+$ starts with the set of removed offshore units $L^-$ and inserts them back in solution $s$. The insertion operator $\Phi(u, s)$ inserts the offshore unit $u$ into the best position of solution $s$. That is, $\Phi(u, s)$ finds the best position to insert $u$ taking into account the incremental cost for solution $s$. All delivery patterns for $u$ are analysed to define this best position. Thus, let $C(\Phi(u, s))$ be the incremental cost for solution $s$ when $u$ is inserted into the best position.

The first insertion operator, called *Deep greedy insertion (IO1)*, finds $u \in L^- | \forall v \in L^- \rightarrow C(\Phi(u, s)) \leq C(\Phi(v, s))$ and inserts it into the best position $\Phi(u, s)$. This process is reiterated until all removed offshore units have been inserted back. The second insertion operator, called *Greed insertion (IO2)*, randomly selects an offshore unit $u \in L^-$ and inserts it into the best position $\Phi(u, s)$. This process ends when all removed offshore units have been inserted back. Finally, the last insertion operator is the *k-regret insertion (IO3)* which is based on a $k$-regret criterion, as in Ropke and Pisinger (2006b) and Ribeiro and Laporte (2012). Given a set of removed offshore units $L^-$, for each offshore units $u \in L^-$, it computes a regret value which is based on the delivery patterns available for $u$. For each offshore unit $u \in L^-$, the $k$-best insertion positions per period are obtained. Each position has a cost increment, and therefore a regret cost can be found for each period.

Thus, for each delivery pattern available for $u$, we compute a regret cost for it which is obtained by adding the regrets generated for each period defined for the corresponding delivery pattern. We then choose the offshore unit $u \in L^-$ with the largest value of regret per delivery pattern to be inserted into solution $s$.

### 4.5. Local search

The local search mechanism was inspired by Vidal et al. (2012) and is described in Algorithm 5 which contains two main steps: voyage improvement (VI) and delivery pattern improvement (PI). These steps are sequentially applied on the current solution $s$, with the first one selected randomly. The algorithm ends when no improvement can be found for solution $s$. Step VI tries to improve the total distance travelled by all vessels per period by applying swap moves, and Step PI tries to find a better delivery pattern distribution for the offshore units.

**Algorithm 5:** Local Search.

```
Input: Solution s
Output: Improved solution s
k0 ← TRUE; k1 ← TRUE;
b ← Rand(TRUE, FALSE);
repeat
    if b then
        k0 ← PI(s);
    else
        k1 ← VI(s);
    b ← ¬b;
until k0 ∨ k1;
return s.
```

Step VI first randomly selects a period $t \in T$. Two offshore units $u$ and $v$, both serviced in period $t$, are then randomly chosen. Let $k_u, k_v \in \{0, 1, 2, 3\}$ be indicators of size. For each combination of $k_u$ and $k_v$, the $k_u$ installations serviced after $u$ are swapped with the $k_v$ installations serviced after $v$. If this move reduces the cost, Step VI ends and returns true. Otherwise, the swap is undone and a new combination is tested. When all combinations are evaluated, two new installations not yet selected are chosen and the combinations are tested again. If all installations have been tested, a new period $t$ not yet selected is randomly chosen and the entire process is repeated. Finally, if an improvement move is not found, Step VI returns false.

For Step PI, an installation $u$ is randomly selected from $N^{OI}$. This installation is removed from $s$ and reinserted according to position $\Phi(u, s)$, generating solution $s'$. If the cost of solution $s'$ is better than that of solution $s$, Step PI returns true and ends. Otherwise, a new installation not yet evaluated is randomly selected from $N^{OI}$. If all installations have been evaluated and no improvement has been found, Step PI ends, returning false.

It is very important to consider berth allocation decisions in the periodic routing of offshore supply vessels since an infeasible allocation of a berth yields a high cost. The incorporation of berth allocation decisions in the periodic planning of offshore supply vessels makes the problem more difficult to solve by exact approaches, but this was not the case for our ALNS heuristic. Here, the berth allocation decisions were integrated to the insertion and removal processes. We tested some specific heuristics for swapping routes or reallocating berths, as in Borthen et al. (2018) and Kisialiou et al. (2018a), but they only increased the complexity of the heuristic with only a negligible improvement in solution quality.

### 4.6. Adaptive layer

Every iteration, the ALNS chooses a pair of insertion and removal operators through a roulette wheel mechanism. Initially, all operators have the same weight. The search is divided into segments of $\rho$ iterations each. When a segment ends, the weights are updated based on the score obtained for the last segment, taking into account the number of times each operator has been used during that segment.

The score of an operator is increased by a parameter equal to $\sigma_1$, $\sigma_2$ or $\sigma_3$ when it identifies a new solution. If a pair of removal-insertion operator finds a new best solution, their scores are increased by $\sigma_1$, if it finds a solution better than the current one, their scores are increased by $\sigma_2$, and if it finds a non-improving solution which is accepted, their scores are increased by $\sigma_3$. Thus, the weight $\phi_i$ of operator $i$ is updated by Eq. (38), where $\pi_i$ is the resulting score and $\xi_i$ is the number of times operator $i$ has been used in the last segment, where the parameter $\tau_A$ is called *reaction factor*:

$$\phi_i = (1 - \tau_A)\phi_i + \tau_A \frac{\pi_i}{\xi_i}. \tag{38}$$

### 4.7. Simulated annealing

We use the SA acceptance criterion, i.e., given a current solution $s$, a neighbor solution $s'$ is accepted if it provides a better cost, and it is accepted with probability $\exp[(C(s') - C(s))/\theta]$ otherwise, where $\theta \geq 0$ is the current temperature and $C(s)$ is the penalized solution cost defined in Section 4.2. The temperature starts at $\theta_0$ and is multiplied by a cooling rate $\mu$ at every iteration.

It is expected that $\theta$ will have values proportional to $C(s)$ throughout the search. The choice of the initial temperature $\theta_0$ and the final temperature $\theta_F$ are therefore related to the cost of the initial solution $s_{initial}$ for a consistent acceptance criterion throughout the search process. Thus, the values of $\theta_0$ and $\theta_F$ are chosen according to parameters $\theta'_0$ and $\theta'_F$, explicitly defined by Eqs. (39) and (40). This leads to the cooling rate $\mu$ which is dependent of the maximum number of iterations $N$, as expressed by Eq. (41):

$$\theta_0 = C(s_{initial})\theta'_0 \tag{39}$$

$$\theta_F = C(s_{initial})\theta'_F \tag{40}$$

$$\mu = \sqrt[N-1]{\frac{\theta'_F}{\theta'_0}}. \tag{41}$$

The values of $\theta'_0$ e $\theta'_F$ must be tuned. The results for the PSVPP-BA are shown in Section 5.2.

## 5. Computational experiments

This section presents the results of our computational experiments. From the E&P operators' perspective, or even from the perspective of offshore logistics providers, cost savings can be achieved through a better planning of the PSV fleet. This planning must take into consideration the berth capacity constraints in a 24/7 continuous operation, and the berth time dependents on the amount of cargo loaded rather than being a fixed time. These operational characteristics make the problem hard to solve for large instances.

The ALNS based heuristic and exact the methods was implemented in the C programming language, using the gcc 10.2 compiler with -O3 option. The computer used in all experiments was an AMD Threadripper 9 3960x 24c/48t with static clocks @ 4.0Ghz processor, 128GB DDR4 of RAM and 130GB of SSD reserved expanded memory, running Ubuntu 20.04 x64 operating system. The commercial solver used for the TSP solving, complete model, reduced model and subproblem was Gurobi 9.0.3. All the reported

heuristic and TSP solving results are from single thread processing, and the complete and reduced models results are the total 24 thread processing time, and all computational times are expressed in seconds. Before presenting the results, we introduce the set of instances, and the details of the parameters tuning processing.

### 5.1. Set of instances

The fleet-sizing and periodic routing problem with berth allocation decisions tackled in this paper is difficult to solve, due to its combinatorial nature and the size of the considered instances. These are the largest available benchmark instances for this problem. In order to achieve a good-quality solution or even an optimal solution, Cruz et al. (2019) solved the problem in steps, with a Multi Step Model Approach (MSMA), aiming to reduce the processing time. The instances solved in this paper are real-based cases obtained from a Brazilian oil company, from its operations at Campos Basin. Four cases are presented: C10, C15, C41 and C79, having 10, 15, 41 and 79 offshore units, respectively. In cases C10 and C15, routes are generated considering all possible combinations for all units. In cases C41 and C79, Cruz et al. (2019) generated routes considering all possible combinations of the units belonging to their clusters (i.e., groups of offshore units), while we solved these instances without clustering, which makes them even harder to solve by mathematical programming algorithms.

The largest available real-world instance, C79 Cruz et al. (2019), is made up to 79 offshore units operating in the Campos Basin, while C10, C15 and C41 are cut-down versions of C79. Four berths are available at the onshore base and the maximum number of departures from each berth in each day is limited to two. The heterogeneous fleet is composed of three types of vessels: PSV4500 with deck capacity of 900 $m^2$, PSV3000 deck capacity of 600 $m^2$ and PSV1500 with deck capacity of 300 $m^2$. All vessels share approximately the same travel speed of 10 knots ($\approx$ 18.52 km/h).

Cruz et al. (2019), also defined segregated instances with an "S" suffix. Their motivation to segregate the instances came from the fact that the production installations usually remain fixed in the same position for many years, while the drilling rigs and maintenance platforms are constantly moved from one oilfield to another. In this work, there is no need to consider these instances, since they would be the same as those without the suffix. The authors also tested C41 and C79 with different numbers of berths. Therefore, in order to standardize the tests, each instance is named with the number of offshore units, weekly visits and available berths: for example C79-112-5 instance has 79 offshore units to be served, 112 weekly visits and five available berths.

For more extensive testing, we created instances C21-39-1, C30-48-2, C50-70-3, C60-89-3 and C69-99-4 with the following procedure: starting with instance C79-112-4, randomly select and remove offshore units until the targeted number is reached, than each offshore unit has its coordinates randomized within the boundary box defined by the offshore units presented in instance C79-112-4. In order to evaluate the isolated impact of clustering, for each new instance and for instance C15-1-1, we created a manual cluster arrangement based on location of the offshore units. The instances with 41 and 79 offshore units already had cluster arrangements from Cruz et al. (2019) and they were kept. Finally, two extra instances, C66-94-4 and C40-62-2, were created just for the heuristic parameter tuning, applying the same method used to create the new instances.

Table 2 shows, for each test instance and cluster configuration, the total number of feasible routes and the total CPU time to solve all respective Traveling Salesman Problems. The CPU time ranges from 0.05 seconds and approximately 3.7 h. It is important to highlight that these times are included in the computational experiments performed for the exact methods.

**Table 2**
Number of offshore units, cluster arrangements and feasible non dominated routes.

| #Offshore Units | #Clusters | #Routes | CPU(s) |
|---|---|---|---|
| 10 | 1 | 912 | 0.22 |
| 15 | 1 | 10,021 | 2.79 |
| 15 | 2 | 370 | 0.05 |
| 21 | 1 | 28,929 | 8.79 |
| 21 | 2 | 1103 | 0.18 |
| 30 | 1 | 91,689 | 20.69 |
| 30 | 4 | 581 | 0.08 |
| 41 | 1 | 782,289 | 209.55 |
| 41 | 5 | 1373 | 0.22 |
| 41 | 7 | 438 | 0.06 |
| 50 | 1 | 1,298,168 | 373.71 |
| 50 | 7 | 591 | 0.07 |
| 60 | 1 | 4,071,416 | 1529.14 |
| 60 | 8 | 1178 | 0.15 |
| 69 | 1 | 14,930,022 | 6402.72 |
| 69 | 8 | 1169 | 0.16 |
| 79 | 1 | 29,013,740 | 13317.54 |
| 79 | 9 | 2168 | 0.32 |
| 79 | 12 | 1107 | 0.11 |

### 5.2. Parameters tuning

All parameters, except for $\alpha$, $\beta$ and $\gamma$, were calibrated through single executions of the ALNS heuristic. The initial parameters values were estimated based on related articles, such as Kisialiou et al. (2018b) and Koç et al. (2018), or simply by trial and error.

As mentioned in Section 1, the fixed costs are much higher than the variable costs for the PSVPP-BA. Therefore, in order to attenuate the influence of the variance of the initial random solutions on the calibration process, a random seed and an initial solution were fixed in each run (within a total of 30 fixed initial solutions), so that the difference between the rounds was only due the parameters to be calibrated.

The tuning of the multi-start parameters $\alpha$, $\beta$ and $\gamma$ was based on the same principle with a sequence of 15 random seeds and a sequence of 100 initial solutions were fixed for each tuning run, having a total of $100 \times 15 = 1,500$ initial solutions, per instance. In this way, for the same values of $\alpha$ and $\beta$, a larger $\gamma$ implies the same best solution after the $\alpha$ starts, but a better final solution is not guaranteed since the temperature values per iteration in the final ALNS execution are different.

The tuning of the ALNS' parameters occurred in three different ways: in isolation ($\rho$, $p$, $\delta$, and $\tau_A$), and in triplets (($\sigma_1, \sigma_2, \sigma_3$), and ($\alpha, \beta, \gamma$)), according to the types of correlations between them. The results were normalized based on the initial parameter values, and the average CPU times were omitted when the variation of the parameters did not change the CPU times by more than 5%.

As a result of the tuning process, Table 3 presents the initial values, the tested values and the best achieved values for each parameter of the ALNS based heuristic developed in this work. The details about the tunning process are reported in the Appendix.

### 5.3. Computational results and comparative analysis

Considering the parameters presented in Table 3, the heuristic was executed 15 times, with random seeds and the exact methods were executed once per instance. Table 4 presents, for each instance, the results of the branch-and-cut algorithm described in Section 3 to the reduced model and the results of the complete formulation, including the value of the best incumbent solution found (UB), the lower bound (LB) and CPU time (CPU, in seconds). It also shows the results of the ALNS comprising the best solution cost (Best), the average solution cost (Average), the standard deviation (Dev) in percentage based on average, and the average CPU time

**Table 3**
Parameters settings.

| Parameter | Description | Initial value | Tested values | Best value |
|---|---|---|---|---|
| $\tau^-$ | Infeasible update weight | 1.15 | 1.05, 1.10, 1.15, 1.25 | 1.15 |
| $\tau^+$ | Feasible update weight | 0.9 | 0.6, 0.75, 0.9, 1.00 | 0.9 |
| $n_1^-$ | Lower removal rate | 0.1 | 0.05, 0.10, 0.15, 0.20 | 0.1 |
| $n_2^-$ | Upper removal rate | 0.3 | 0.20, 0.25, 0.30, 0.40 | 0.25 |
| $\theta_0'$ | Initial relative temperature | 1/3 | 1, 1/3, 1/10, 1/50 | 1/50 |
| $\theta_F'$ | Final relative temperature | 1/300 | 1/50, 1/100, 1/300, 1/500 | 1/500 |
| $\rho$ | Segment size | 50 | 25, 50, 100, 200, 300 | 50 |
| $p$ | Removal's Level of determinism | 4 | 1, 2, 4, 6, 10 | 4 |
| $\delta$ | Local search interval | 50 | 10, 25, 50, 200, 500 | 50 |
| $\tau_A$ | Adeptness rate | 0.1 | 0.01, 0.05, 0.1, 0.2, 0.3 | 0.05 |
| $\sigma_1$ | Best score | 10 | 0, 5, 10 | 5 |
| $\sigma_2$ | Improvement score | 5 | 0, 5, 10 | 0 |
| $\sigma_3$ | Worst score | 0 | 0, 5, 10 | 10 |
| $\gamma$ | Total number of iterations | 50,000 | (15, 50, 150, 300)x$10^3$ | 300,000 |
| $\alpha$ | Number of starts | – | 1, 15, 35, 50, 75, 100 | 50 |
| $\beta$ | Iterations per start | – | 100, 300, 500, 1000 | 100 |

**Table 4**
Heuristic and exact results for all instances.

| Instance | Complete Model | | | Branch and Cut | | | ALNS | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | UB | LB | CPU(s) | UB | LB | CPU(s) | Best | Average | Dev(%) | CPU(s) |
| C10-15-1 | 53.77 | 53.77 | 85.6 | 53.77 | 53.77 | 1.2 | 53.77 | 53.77 | 0.00 | 13.4 |
| C15-24-1 | 80.37 | 77.11 | 86400.0 | 80.37 | 80.37 | 82.5 | 80.37 | 80.37 | 0.00 | 21.6 |
| C21-39-1 | 202.55 | 147.9 | 86400.0 | 176.33 | 151.08 | 86400.0 | 160.75 | 160.95 | 0.08 | 48.4 |
| C30-48-2 | – | 205.56 | 18873.5 | 221.84 | 211.68 | 86400.0 | 221.84 | 225.05 | 0.54 | 99.2 |
| C41-59-2 | – | 233.59 | 5100.4 | 259.91 | 234.54 | 86400.0 | 256.23 | 257.22 | 0.91 | 150.8 |
| C41-59-3 | – | 233.59 | 5042.4 | 252.64 | 233.95 | 86400.0 | 252.01 | 254.88 | 0.29 | 149.4 |
| C50-70-3 | – | – | 623.4 | 385.64 | 343.68 | 86400.0 | 356.97 | 359.47 | 0.98 | 220.6 |
| C60-89-3 | – | – | 1730.7 | 482.40 | 464.00 | 57981.2 | 464.19 | 470.38 | 1.27 | 291.6 |
| C69-99-4 | – | – | 6612.6 | – | 455.94 | 26540.7 | 480.84 | 495.84 | 1.01 | 351.7 |
| C79-112-4 | – | – | 13417.7 | – | – | 14562.3 | 559.59 | 578.45 | 1.99 | 583.3 |
| C79-112-5 | – | – | 13356.9 | – | – | 14562.4 | 554.75 | 562.76 | 0.75 | 549.0 |
| C79-112-6 | – | – | 13303.4 | – | – | 14562.9 | 554.89 | 557.21 | 0.45 | 547.7 |

**Table 5**
Results with clustered instances and exact methods.

| Instance | #Clusters | Complete Model | | | Branch and Cut | | |
|---|---|---|---|---|---|---|---|
| | | UB | LB | CPU(s) | UB | LB | CPU(s) |
| C15-24-1 | 2 | 90.18 | 90.18 | 2785.4 | 90.18 | 90.18 | 10.2 |
| C21-39-1 | 2 | 177.16 | 165.15 | 86400.0 | 177.14 | 170.17 | 86400.0 |
| C30-48-2 | 4 | 246.68 | 222.14 | 86400.0 | 226.28 | 226.28 | 185.4 |
| C41-59-2 | 5 | – | 252.10 | 86400.0 | 259.56 | 259.56 | 77400.1 |
| C41-59-3 | 5 | 302.02 | 246.79 | 86400.0 | 259.52 | 259.52 | 1750.8 |
| C41-59-2 | 7 | 262.09 | 261.84 | 86400.0 | 262.09 | 262.09 | 33500.7 |
| C41-59-3 | 7 | 262.09 | 261.88 | 86400.0 | 262.09 | 262.09 | 1065.4 |
| C50-70-3 | 7 | 392.82 | 366.44 | 86400.0 | 375.07 | 375.07 | 57488.6 |
| C60-89-3 | 8 | – | 448.90 | 86400.0 | 483.03 | 470.14 | 86400.0 |
| C69-99-4 | 8 | – | 460.58 | 86400.0 | 492.43 | 492.43 | 65781.6 |
| C79-112-4 | 9 | – | 536.16 | 86400.0 | 583.78 | 559.42 | 86400.0 |
| C79-112-5 | 9 | – | 536.16 | 86400.0 | 566.85 | 557.56 | 86400.0 |
| C79-112-6 | 9 | – | 536.16 | 86400.0 | 566.85 | 557.56 | 86400.0 |
| C79-112-4 | 12 | – | 547.42 | 86400.0 | 577.22 | 574.78 | 86400.0 |
| C79-112-5 | 12 | – | 548.10 | 86400.0 | 577.22 | 572.74 | 86400.0 |
| C79-112-6 | 12 | – | 548.10 | 86400.0 | 577.22 | 572.74 | 86400.0 |

(CPU, in seconds). A "-" value means that no result was found. Table 5 similarly presents the results for the exact methods with the different cluster arrangements for each instance in order to further analyse the performance of both exact methods and compare the results with the non-clustered instances, evaluating the impact of clustering.

Analyzing the results for the exact methods for Tables 4 and 5, it is important to highlight that either an optimal solution was found or we stopped due to lack of memory or we stopped due to a time limit of 86400 seconds. The branch-and-cut (B&C) method applied to the reduced model showed a clear advantage over the complete formulation, providing better bounds, with significantly reduced CPU times.

The ALNS, when compared to the exact methods, presented the best performance, both in terms of solution quality and computational time. The percentile standard deviation values demonstrate the robustness of the ALNS heuristic, remaining below 1% for the smaller instances, and between 0.5% and up to 2% for the larger instances, which correspond to the real-world cases derived from the Brazilian oil industry. The CPU times scaled well between the instances, with a maximum time of 583.3 s for the largest instance, which is quite reasonable for a difficult problem such as

**Table 6**
Comparison between MSMA (Cruz et al., 2019) and ALNS for the C41 group of instances.

| C41 Instance | MSMA C41S(2B) | MSMA C41(2B) | ALNS C41-2 | MSMA C41(3B) | ALNS C41-3 |
|---|---|---|---|---|---|
| Clusters | 7 | 5 | 1 | 5 | 1 |
| Total Cost | 261.99 | 270.20 | 256.23 | 256.59 | 252.01 |
| Fleet Cost | 248 | 256 | 243 | 243 | 238 |
| Routing Cost | 13.99 | 14.20 | 13.23 | 13.59 | 14.01 |
| PSV4500 | 5 | 5 | 4 | 4 | 3 |
| PSV3000 | 1 | 2 | 2 | 2 | 3 |
| PSV1500 | 1 | 0 | 1 | 1 | 1 |
| Routes | 17 | 19 | 17 | 19 | 19 |

the PSVPP-BA. Finally, Table 4 shows that the increase in the number of berths per group of instances does not have a measurable impact on the CPU times.

Table 6 shows the comparison between the MSMA (Cruz et al., 2019) and the ALNS results for the C41 group of instances. Columns 2 –4 and 5–6 show the results for the two-berth instances and the three-berth instances, respectively. For the two-berth scenario, MSMA found a solution of total cost 261.99, and the ALNS was able to find a better total cost solution of 256.23, with both lower fleet and routing costs. In the three-berth scenario, MSMA found a solution of total cost 256.59, and the ALNS was able to find a better total cost solution of 252.01, with lower fleet cost and higher routing costs.

Table 7 provides a comparison between MSMA and the ALNS results for the C79 group of instances. Columns 2–4, 5–6 and 7–8 show the results for the four-berth instances, the five-berth instances, and the six-berth instances, respectively. The four-berth scenario is the most restrictive and closest to the real case faced by the Brazilian oil industry. The MSMA found a solution of total cost 570.37 for the C79S(4B) instance, and the ALNS was able to find a better total cost solution of 559.59, with both lower fleet and routing costs, although concentrating the fleet of vessels only on the PSV4500 type, with deck capacity of 900 $m^2$. For the five-berth scenario, MSMA was not able to find an optimal solution, but found an incumbent solution of total cost 591.55 and 5.58% of optimality gap for the C79(5B) instance, the same solution found for the C79(4B) four-berth instance. The ALNS was able to find a better total cost solution of cost 554.75 for the C79-5 instance, with both lower fleet and higher routing costs, although again the fleet concentrates mainly on the PSV4500 vessel. Finally, in the six-berth scenario, MSMA found a solution of total cost 558.55, and the ALNS was able to find a better total cost solution of 554.89, with lower routing costs, and the same fleet cost, although mainly concentrated on the PSV4500 vessel. Once again, it is important to emphasize that we are not considering clustered installations.

To conclude, Tables 6 and 7 show that the proposed ALNS-based heuristic was able to find better solutions than those of Cruz et al. (2019) with both a smaller fleet and smaller routing costs, with well balanced fleets for the C41 instances and fleets concentrated mainly on the PSV4500 vessel for the C79 instances. Lastly, the good heuristic results were uphold by the good exact results of the branch-and-cut method applied to the reduced model.

## 6. Conclusions

We have presented an exact branch-and-cut method and an adaptive large neighborhood search (ALNS) heuristic with multiple starts and spaced local searches to solve the periodic supply vessel planning problem (PSVPP) arising in the upstream offshore petroleum logistics chain. The PSVPP tackled in this work consists of a periodic vehicle routing problem while simultaneously determining the optimal fleet size and a mix of heterogeneous offshore supply vessels, their one week routes and schedules for servicing the offshore oil and gas installations, besides the berth allocations at the supply base.

We have extended the previous works of Halvorsen-Weare and Fagerholt (2017), Kisialiou et al. (2018a) and Cruz et al. (2019) since we used a replicable one-week planning horizon both for the offshore units and the vessels. We solved the largest available real-world instances, without dividing them into clusters, and we achieved good solutions relatively fast, performing significantly better than the branch-and-cut algorithm.

## Acknowledgements

This work was conducted during a scholarship supported by the International Cooperation Program CAPES/COFECUB at the Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), and financed by CAPES (grant 88881.186960/2018-01) - Brazilian Federal Agency for Support and Evaluation of Graduate Education within the Ministry of Education of Brazil. The support of the Canadian Natural Sciences and Engineering Research Council (grant 2015-06189) is gratefully acknowledged. This work was partially supported by the National Council for Scientific and Technological Development - CNPq, under grants #309661/2019-6 and #307835/2017-0. Thanks are due to the editor and to the referees for their valuable comments.

## Appendix A. Detailed parameters tuning

According to Section 5.2, this appendix presents all details about the tuning phase. So, Tables A.8 to A.14 show the average percentage deviation (Dev (%)) of the solution values from the initial set of parameters caused by the variation of these parameters.

### *A1. Impact of each operator*

In order to validate the performance of each operator, Table A.8 presents the tuning results for the ALNS without each one of its heuristics. Between the removal heuristics, the ones with higher impact was the Shaw frequency removal (RO3) and Worst removal (RO5) operators and looking to the insertion operators, the

**Table 7**
Comparison between MSMA (Cruz et al., 2019) and ALNS for the C79 group of instances.

| C79 | MSMA | MSMA | ALNS | MSMA | ALNS | MSMA | ALNS |
|---|---|---|---|---|---|---|---|
| **Name** | C79S(4B) | C79(4B) | C79-4 | C79(5B) | C79-5 | C79(6B) | C79-6 |
| **Clusters** | 12 | 9 | 1 | 9 | 1 | 9 | 1 |
| **Total Cost** | 570.37 | 591.55 | 559.59 | 591.55 | 554.75 | 558.55 | 554.89 |
| **Fleet Cost** | 540 | 560 | 532 | 560 | 527 | 527 | 527 |
| **Routing Cost** | 30.37 | 31.55 | 27.59 | 27.75 | 28.49 | 31.55 | 27.89 |
| **PSV4500** | 9 | 8 | 14 | 8 | 13 | 8 | 13 |
| **PSV3000** | 6 | 7 | 0 | 7 | 1 | 6 | 1 |
| **PSV1500** | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| **Routes** | 34 | 38 | 28 | 38 | 29 | 38 | 29 |

**Table A.8**
Average percentage solution deviation without each operator compared with no operator removed (Base).

| Operator | RO1 | RO2 | RO3 | RO4 | RO5 | IO1 | IO2 | IO3 | Base |
|---|---|---|---|---|---|---|---|---|---|
| Dev (%) | 0.70 | 0.79 | 1.74 | 0.53 | 1.86 | 3.10 | 2.84 | 1.33 | 0.00 |

ones with higher impact was the Deep greedy insertion (IO1) and Greed insertion (IO2) operators. As the removal of any operator implies a lower algorithmic performance, all proposed operators were kept.

### *A2. Single tested set of parameters*

Table A.9 presents the tuning results for the single tested ALNS' parameters segment size $\rho$, removal level of determinism $p$, local search interval $\delta$, reaction factor $\tau_A$ and adaptive scores $(\sigma_1, \sigma_2, \sigma_3)$. The segment size parameter $\rho$ controls the frequency of updates of the adaptive weights and the penalty weight. It was tested for the values 25, 50, 100, 200 and 300, and the best result was reached at the initial value $\rho = 50$. The removal level of the determinism parameter $p$ was tested in a range from totally random $p = 1$ to extremely deterministic $p = 10$, passing through the values $p = 2$, $p = 4$ and $p = 6$. The worst results were observed at both extremes $p = 1$ and $p = 10$, and the best result was achieved with the initial value $p = 4$. The local search interval parameter $\delta$ was tested for the values 10, 25, 50, 200 and 500. The best result was reached at the initial value $\delta = 50$, and both the most spaced and least spaced search intervals had a negative impact at the end of the ALNS search. In addition to the average percentage deviation on the objective function caused by the variation of $\delta$, Table A.9 shows the average CPU times in seconds, spent in the computation considering each of its tested values. Taking into account the behavior of the CPU times, it is expected that more frequent executions of local searches (what is computationally expensive) will increase the total search time, but this was not the case. This behavior is justified by the fact that we have applied the local search to the most promising solution instead of the current one. We tested the impact of having no local searches as well ($\delta = \infty$) and we obtained in average percentage solution deviation of 4.215% within approximately the same CPU time.

The adaptive parameters have a marginal overall impact. The reaction factor parameter $\tau_A$ was tested for the values $0.01, 0.05, 0.10, 0.20$ and $0.30$, and the best performance was obtained for $\tau_A = 0.05$. The adaptive scores $(\sigma_1, \sigma_2, \sigma_3)$ were tested for the values $(0, 10, 5)$, $(10, 0, 5)$, $(10, 5, 0)$, $(5, 0, 10)$ and $(0, 5, 10)$, and the best performance was achieved for $(5, 0, 10)$, which may not be in accordance with the initial proposition of Ropke and Pisinger (2006a), but is in line with the recent works of Koç et al. (2018) and Kisialiou et al. (2018a).

**Table A.9**
Average percentage solution deviation with varying $\rho$, $p$, $\tau_A$ and $\delta$.

| | | | | | | |
|---|---|---|---|---|---|---|
| $\rho$ | Values | 25 | **50** | 100 | 200 | 300 |
| | Dev (%) | 2.39 | **0.000** | 2.42 | 2.55 | 3.15 |
| $p$ | Values | 1 | 2 | **4** | 6 | 10 |
| | Dev (%) | 1.05 | 0.93 | **0.000** | 2.45 | 2.87 |
| $\delta$ | Values | 10 | 25 | **50** | 200 | 500 |
| | Dev (%) | 2.51 | 0.96 | **0.000** | 2.08 | 3.38 |
| | CPU(s) | 51.1 | 46.7 | 47.5 | 48.0 | 49.2 |
| $\tau_A$ | Values | 0.01 | **0.05** | 0.10 | 0.20 | 0.30 |
| | Dev (%) | 0.284 | **-0.131** | 0.000 | 0.261 | 0.142 |
| $(\sigma_1, \sigma_2, \sigma_3)$ | Values | (0,10,5) | (10,0,5) | (10,5,0) | **(5,0,10)** | (0,5,10) |
| | Dev (%) | 0.384 | 0.415 | 0.000 | **-0.195** | 0.541 |

**Table A.10**
Average solution deviation (%) with varying feasibility weights update values.

| $\tau^- \setminus \tau^+$ | 0.60 | 0.75 | **0.90** | 1.00 |
|---|---|---|---|---|
| 1.05 | 1.39 | 0.49 | 0.17 | 2.93 |
| 1.10 | 1.16 | 0.66 | 0.07 | 2.90 |
| **1.15** | 1.27 | 0.07 | **0.00** | 2.07 |
| 1.25 | 1.50 | 1.40 | 0.27 | 2.88 |

### *A3. Feasibility weights updating*

Table A.10 presents the tuning results for the pair of parameters $(\tau^-, \tau^+)$. The infeasible update weight $\tau^-$ was tested for the values 1.05, 1.10, 1.15 and 1.25, while the feasible update weight $\tau^+$ was tested for the values 0.60, 0.75, 0.90, and 1.00. The best result was achieved for the pair of initial values $(\tau^- = 1.15, \tau^+ = 0.90)$. It is interesting to emphasize that the last column $\tau^+ = 1.00$ exhibits the worst results for all tested $\tau^-$ values. This means that once the best solution is feasible, the penalty weights do not decrease, discouraging the exploration of infeasible solutions.

### *A4. Removal range*

Table A.11 presents the tuning results for the pair of parameters $(n_1^-, n_2^-)$. Besides the average percentage deviation of the solution values caused by the variation of these parameters, Table A.11 shows the average CPU times spent in the computation, considering each pair of values. The lower removal rate $n_1^-$ was tested for the values 0.05, 0.10, 0.15 and 0.20, while the upper removal rate $n_2^-$ was tested for the values 0.20, 0.25, 0.30, and 0.40. The best result was achieved for the values $n_1^- = 0.10$ and $n_2^- = 0.25$. The average CPU times behaved as expected since a larger number of removals and insertions per iteration implied higher CPU times.

### *A5. Temperature range*

Table A.12 presents the tuning results for the pair of parameters $(\theta_0', \theta_F')$ with the average CPU times, spent in the computation considering each pair of values. The initial relative temperature $\theta_0'$ was tested for the values 1, 1/3, 1/10 and 1/50, while the final relative temperature $\theta_F'$ was tested for the values 1/50, 1/100, 1/300, and 1/500. The best result was achieved for the pair of values $(\theta_0' = 1/50, \theta_F' = 1/500)$, with an improvement of 1.68% in comparison to baseline tests. This pair of parameters does not have a major impact on the quality of the solutions, but the CPU times grow from the lower right corner (lower temperature values) to the upper right corner (higher temperature values).

### *A6. Multi-start performance*

Tables A.13 and A.14 present the average percentage deviation of the solution values related to the best average solution found for all tested combination of $\alpha$, $\beta$ and $\gamma$. Starting with Table A.13 we relate the number of starts $\alpha$ and the number of iterations per start $\beta$, while keeping the total number of iterations fixed at $\gamma = 150,000$. The parameter $\alpha$ was tested for the values 1, 15, 35, 50, 75 and 100, and $\beta$ was tested for the values 100, 300, 500, and

**Table A.11**
Average solution deviation (%) and CPU time, in seconds, with varying removal range.

| $n_1^- \setminus n_2^-$ | Dev (%) | CPU(s) | Dev (%) | CPU(s) | Dev (%) | CPU(s) | Dev (%) | CPU(s) |
|---|---|---|---|---|---|---|---|---|
| | 0.20 | | **0.25** | | 0.30 | | 0.40 | |
| 0.05 | 0.99 | 32.4 | 0.01 | 36.7 | 0.27 | 41.7 | 0.37 | 52.8 |
| **0.10** | −0.12 | 36.7 | **−0.55** | 41.5 | 0.00 | 46.7 | 0.43 | 57.7 |
| 0.15 | 0.53 | 41.8 | 0.24 | 46.7 | 1.03 | 51.9 | −0.03 | 62.3 |
| 0.20 | 0.34 | 48.0 | −0.13 | 52.4 | 0.30 | 57.5 | 0.51 | 68.0 |

**Table A.12**
Average solution deviation (%) and CPU time, in seconds, with varying temperature range.

| $\theta_0' \setminus \theta_F'$ | Dev (%) | CPU(s) | Dev (%) | CPU(s) | Dev (%) | CPU(s) | Dev (%) | CPU(s) |
|---|---|---|---|---|---|---|---|---|
| | 1/50 | | 1/100 | | 1/300 | | **1/500** | |
| 1 | 1.02 | 48.9 | 1.39 | 47.9 | 0.14 | 47.2 | 0.03 | 47.4 |
| 1/3 | 1.05 | 48.5 | 1.41 | 47.8 | 0.00 | 46.7 | −0.26 | 46.3 |
| 1/10 | 0.30 | 47.3 | 0.43 | 46.9 | −0.01 | 45.4 | −0.76 | 45.2 |
| **1/50** | 0.73 | 45.5 | −0.41 | 44.6 | 0.43 | 44.2 | **−1.68** | 43.6 |

**Table A.13**
Average solution deviation (%) with $\gamma = 150,000$ and varying $\alpha$ and $\beta$.

| $\alpha \setminus \beta$ | **100** | 300 | 500 | 1000 |
|---|---|---|---|---|
| 1 | 2.73 | 1.66 | 2.14 | 1.61 |
| 15 | 1.65 | 0.36 | 0.91 | 2.58 |
| 35 | 0.86 | 0.77 | 1.31 | 1.55 |
| 50 | 0.52 | 0.55 | 1.30 | 1.94 |
| 75 | 0.79 | 1.09 | 1.10 | – |
| 100 | 0.95 | 1.01 | 1.82 | – |

**Table A.14**
Average solution deviation (%) and CPU time, in seconds, with $\beta = 100$ and varying $\alpha$ and $\gamma$.

| $\alpha \setminus \gamma$ | 15,000 | 50,000 | 150,000 | **300,000** |
|---|---|---|---|---|
| 1 | 6.75 | 3.23 | 1.43 | 2.37 |
| 15 | 5.39 | 3.76 | 1.65 | 0.26 |
| 35 | 6.92 | 2.38 | 0.86 | 0.47 |
| **50** | 7.19 | 1.68 | 0.52 | **0.00** |
| 75 | 7.47 | 3.20 | 0.79 | 0.38 |
| 100 | 7.93 | 2.68 | 1.65 | 0.43 |
| (s) | 10.2 | 33.6 | 99.4 | 201.3 |

1,000. According to the results of Table A.13, line $\alpha = 1$, i.e. "no-multi-start", yielded the worst results for all values of $\beta$, and no other value of $\alpha$ consistently presented better solutions. In addition, discarding the line $\alpha = 1$, columns $\beta = 500$ and $\beta = 1,000$ consistently presented the worst results with the increase of $\alpha$. Finally, column $\beta = 100$ achieved the best results for all values of $\alpha$, without, however, showing consistency with the increasing or decreasing of $\alpha$.

Based on the results of Table A.13, the number of iterations per start was fixed at $\beta = 100$, and new tests were performed varying the parameters $\alpha$ (number of starts) and $\gamma$ (total number of iterations). The parameter $\alpha$ was tested for the same values as in Table A.13, and $\gamma$ was tested for the values 15,000, 50,000, 150,000, and 300,000. According to the results of Table A.14, the best results are achieved for $\gamma = 300,000$, as the total number of iterations $\gamma$ increases and once again line $\alpha = 1$ yielded the worst results for all $\gamma$ values. Finally, Table A.14 shows that the best result was reached for $\alpha = 50$, $\beta = 100$ and $\gamma = 300,000$.

## References


Aas, B., Gribkovskaia, I., Halskau, O., & Shlopak, A. (2007). Routing of supply vessels to petroleum installations. *International Journal of Physical Distribution & Logistics Management, 37*(2), 164–179.

Aas, B., Halskau, O., & Wallace, S. W. (2009). The role of supply vessels in offshore logistics. *Maritime Economics & Logistics, 11*(3), 302–325.

Applegate, D. L., Bixby, R. E., Chvatál, V., & Cook, W. J. (2006). *The traveling salesman problem: A computational study*. Princeton University Press. http://www.jstor.org/stable/j.ctt7s8xg

Baldacci, R., Bartolini, E., Mingozzi, A., & Valletta, A. (2011). An exact algorithm for the period routing problem. *Operations Research, 59*(1), 228–241.

Baptista, S., Oliveira, R. C., & Zúquete, E. (2002). A period vehicle routing case study. *European Journal of Operational Research, 139*(2), 220–229.

Bierwirth, C., & Meisel, F. (2015). A follow-up survey of berth allocation and quay crane scheduling problems in container terminals. *European Journal of Operational Research, 244*(3), 675–689.

Bondy, J. A., & Murty, U. S. R. (1976). *Graph theory with applications*: 290. London: Macmillan.

Borthen, T., Loennechen, H., Fagerholt, K., Wang, X., & Vidal, T. (2019). Bi-objective offshore supply vessel planning with costs and persistence objectives. *Computers & Operations Research, 111*, 285–296.

Borthen, T., Loennechen, H., Wang, X., Fagerholt, K., & Vidal, T. (2018). A genetic search-based heuristic for a fleet size and periodic routing problem with application to offshore supply planning. *EURO Journal on Transportation and Logistics, 7*(2), 121–150.

Bräysy, O., Hasle, G., & Dullaert, W. (2004). A multi-start local search algorithm for the vehicle routing problem with time windows. *European Journal of Operational Research, 159*(3), 586–605.

Codato, G., & Fischetti, M. (2006). Combinatorial benders' cuts for mixed-integer linear programming. *Operations Research, 54*(4), 756–766. https://doi.org/10.1287/opre.1060.0286.

Cruz, R., Mendes, A. B., Bahiense, L., & Wu, Y. (2019). Integrating berth allocation decisions in a fleet composition and periodic routing problem of platform supply vessels. *European Journal of Operational Research, 275*, 334–346.

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

Fagerholt, K., & Lindstad, H. (2000). Optimal policies for maintaining a supply service in the Norwegian Sea. *Omega, 28*(3), 269–275.

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

Halvorsen-Weare, E. E., & Fagerholt, K. (2017). *Optimization and Engineering, 18*(1), 317–341.

Halvorsen-Weare, E. E., Fagerholt, K., Nonås, L. M., & Asbjørnslett, B. E. (2012). Optimal fleet composition and periodic routing of offshore supply vessels. *European Journal of Operational Research, 223*(2), 508–517.

Hwang, C.-R. (1988). Simulated annealing: Theory and applications. *Acta Applicandae Mathematicae, 12*(1), 108–111.

Iris, c., Pacino, D., & Ropke, S. (2017). Improved formulations and an adaptive large neighborhood search heuristic for the integrated berth allocation and quay crane assignment problem. *Transportation Research Part E: Logistics and Transportation Review, 105*, 123–147.

Karp, R. M. (1972). *Reducibility among combinatorial problems* (pp. 85–103)). Boston: Springer.

Kisialiou, Y., Gribkovskaia, I., & Laporte, G. (2018a). The periodic supply vessel planning problem with flexible departure times and coupled vessels. *Computers & Operations Research, 94*, 52–64.

Kisialiou, Y., Gribkovskaia, I., & Laporte, G. (2018b). Robust supply vessel routing and scheduling. *Transportation Research Part C: Emerging Technologies, 90*, 366–378.

Koç, c., Jabali, O., & Laporte, G. (2018). Long-haul vehicle routing and scheduling with idling options. *Journal of the Operational Research Society, 69*(2), 235–246.

Mauri, G. R., Ribeiro, G. M., Lorena, L. A. N., & Laporte, G. (2016). An adaptive large neighborhood search for the discrete and continuous berth allocation problem. *Computers & Operations Research, 70*, 140–154.

Palomo-Martínez, P. J., Salazar-Aguilar, M. A., & Laporte, G. (2017). Planning a selective delivery schedule through adaptive large neighborhood search. *Computers & Industrial Engineering, 112*, 368–378.

Paraskevopoulos, D. C., Repoussis, P. P., Tarantilis, C. D., Ioannou, G., & Prastacos, G. P. (2008). A reactive variable neighborhood tabu search for the heterogeneous fleet vehicle routing problem with time windows. *Journal of Heuristics, 14*(5), 425–455.

Pisinger, D., & Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & Operations Research, 34*(8), 2403–2435.

Ribeiro, G. M., & Laporte, G. (2012). An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. *Computers & Operations Research, 39*(3), 728–735.

Ropke, S., & Pisinger, D. (2006a). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science, 40*(4), 455–472.

Ropke, S., & Pisinger, D. (2006b). A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research, 171*(3), 750–775.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In M. Maher, & J.-F. Puget (Eds.), *Principles and practice of constraint programming – CP98* (pp. 417–431). Berlin Heidelberg: Springer.

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: Information Systems and Operational Research, 50*(4), 195–204.

Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N., & Rei, W. (2012). A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research, 60*(3), 611–624.