A Genetic Search-based Heuristic for a Fleet Size and Periodic Routing Problem with Application to Offshore Supply Planning

Thomas Borthen[1]
Henrik Loennechen[1]
Xin Wang[1]
Kjetil Fagerholt[1,2]
Thibaut Vidal[3]

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

[2]SINTEF Ocean, Trondheim, Norway

[3]Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), Brazil

T. Borthen, H. Loennechen, X. Wang, K. Fagerholt and T. Vidal (2017). 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.*
doi: 10.1007/s13676-017-0111-x

Abstract
This paper introduces a genetic search-based heuristic to solve an offshore supply vessel planning problem (SVPP) faced by the Norwegian oil and gas company Statoil. The aim is to help the company in determining the optimal size of supply vessels to charter in and their corresponding voyages and schedules. We take inspiration from the hybrid genetic search with adaptive diversity control (HGSADC) algorithm of Vidal et al. (2012), which successfully addresses a large class of vehicle routing problems, including the multi-period VRP (PVRP), and adapt it to account for some special features that are recurrent in maritime transportation but scarcely found in classical PVRPs, in particular, the possibility of having voyages spanning over multiple time periods in the planning horizon. Our computational experiments show that the proposed heuristic is scalable and stable, being able to solve industrial SVPPs of realistic size while significantly outperforming the existing approaches.

A Genetic Search-based Heuristic for a Fleet Size and Periodic Routing Problem with Application to Offshore Supply Planning

Thomas Borthen[1], Henrik Loennechen[1], Xin Wang*[1], Kjetil Fagerholt[1,2], and Thibaut Vidal[3]

[1]Department of Industrial Economics and Technology Management, Norwegian University of Science and Technology (NTNU), Trondheim, Norway
[2]SINTEF Ocean, Trondheim, Norway
[3]Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), Brazil

August 2017

Abstract

This paper introduces a genetic search-based heuristic to solve an offshore supply vessel planning problem (SVPP) faced by the Norwegian oil and gas company Statoil. The aim is to help the company in determining the optimal size of supply vessels to charter in and their corresponding voyages and schedules. We take inspiration from the hybrid genetic search with adaptive diversity control (HGSADC) algorithm of Vidal et al. (2012), which successfully addresses a large class of vehicle routing problems, including the multi-period VRP (PVRP), and adapt it to account for some special features that are recurrent in maritime transportation but scarcely found in classical PVRPs, in particular, the possibility of having voyages spanning over multiple time periods in the planning horizon. Our computational experiments show that the proposed heuristic is scalable and stable, being able to solve industrial SVPPs of realistic size while significantly outperforming the existing approaches.

Keywords: offshore supply vessel planning, fleet sizing, periodic vehicle routing, genetic algorithm.

*Corresponding author. *E-mail address:* xin.wang@iot.ntnu.no

1 Introduction

The offshore oil and gas industry is Norway's largest industry in terms of investments and value creation, and is highly important for the Norwegian economy. The Norwegian State Oil Company, Statoil, is the leading operator on the Norwegian continental shelf, with offshore installations at the fields for exploring, extracting and processing oil and gas. The offshore installations require regular delivery of supplies in order to operate. These supplies are transported from onshore supply depots using *Platform Supply Vessels* (PSVs, or simply referred to as *vessels* in this paper) that are specially designed to supply offshore installations. Statoil acquires the PSVs on time charter, and the expenses related to chartering and operating these PSVs represent a major part of the costs in their upstream supply chain.

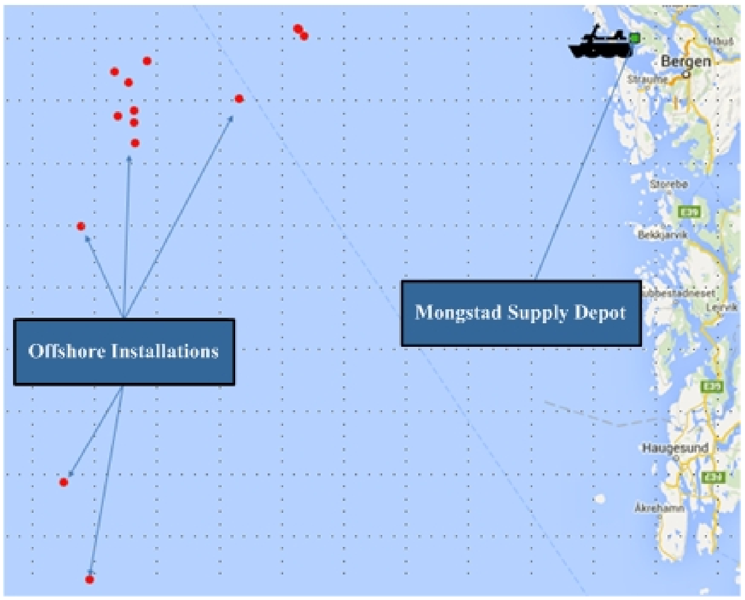


Figure 1: Example of geographic location of the onshore supply depot and offshore installations.

To achieve a cost-efficient service for the offshore installations, Statoil has been using an optimization tool which was developed based on a previous study by Halvorsen-Weare et al. (2012). The study addressed a *Supply Vessel Planning Problem* (SVPP) with several offshore installations, each requiring several services per week, and one onshore supply depot where PSVs load supply cargoes (and discharge back-loads from the installations). See Figure 1 for an example of the geographic location of the supply depot and offshore installations. The aim is to determine the optimal fleet of PSVs to charter in and their corresponding weekly voyages and schedules from the onshore supply depot. The authors proposed a two-stage approach to solve the SVPP, which is an extension of the method used in Fagerholt and Lindstad (2000). In the first stage the shortest sequences for all feasible voyages (where each voyage starts from and ends at the supply depot, and services several installations) are generated, which are then used as input to a voyage-based model, in the second stage, to find the optimal fleet composition and routing decisions using a commercial integer-programming solver. This method has been practically successful in solving problems with up to around 14 installations and 48 weekly services. However, in recent years Statoil merged the operations of several supply depots, leading to a significant increase in problem size, e.g., the number of installations supplied by one depot increased from 14 to 27, and the number of total weekly services from 48 to 80 in a typical case. As a result, the previous two-stage approach can no longer handle the larger problems in reasonable time.

In this paper, we present a new heuristic approach to address this challenge. This approach is inspired by the Hybrid Genetic Search with Adaptive Diversity Control (HGSADC) algorithm (Vidal et al., 2012), which was also generalized into the Unified Hybrid Genetic Search (UHGS) framework in Vidal et al. (2014). HGSADC successfully addresses several Vehicle Routing Problem (VRP) variants including the Multidepot VRP and the Periodic VRP (PVRP). The SVPP considered in this paper shares many similarities with the classic PVRP: a planning horizon over several time units, and multiple possible visits to the customers, over the horizon, subject to some schedule restrictions. These similarities allow us to use a solution representation analogous to the one of HGSADC and to take advantage of its advanced diversity-management mechanisms.

On the other hand, the SVPP presents a distinctive feature which is recurrent in maritime transportation applications but scarcely found in classic PVRPs: each voyage performed by a PSV (vehicle) normally lasts for several days (time units). As a consequence, extra care needs to be taken when modeling the inter-dependencies between voyages, to ensure that a vessel has returned from its previous voyage before embarking on a new one. Therefore, we have made several adaptations and extensions of the framework in order to efficiently address this special feature. We tested the heuristic on instances based on real problems faced by Statoil; a first set of similar size as in Halvorsen-Weare et al. (2012) with up to 14 installations and 48 weekly services, and a second set of larger size with up to 27 installations and 80 weekly services. As highlighted in these experiments, the proposed heuristic significantly outperforms the previous two-stage approach, providing equal or better results in a smaller time, while being more scalable and stable.

The contributions of this paper are twofold. First, we develop a new metaheuristic for the SVPP, which greatly improves upon previous literature and solves industrial supply vessel planning problems of realistic size. Second, by adapting the HGSADC to the SVPP, we demonstrate new possibilities of extensions to account for routes (voyages in SVPP) that span more than one unit of time. This is especially relevant in maritime shipping, which is a a slower transportation mode compared to other sectors. Moreover, we emphasize that the proposed approach is not limited to shipping, and can also be applied to land-based transportation problems with similar features.

The remainder of this paper is organized as follows. We first give a brief literature review in Section 2. The problem description and mathematical model of SVPP are given in Section 3. The proposed heuristic is introduced in Section 4. Its performances are tested and analyzed in Section 5, and we conclude in Section 6.

2 Literature review

Fagerholt and Lindstad (2000) were among the first to study the supply service in the Norwegian Sea, solving a relaxed version of the SVPP. The relaxed version excludes some key constraints that are essential when planning offshore supply vessels in practice, such as the need to spread the departures of supplies from the depot to each installation as evenly as possible throughout the week to provide a steady supply. Later on, Aas et al. (2009) studied the sourcing strategy of PSVs and their role in the offshore logistics. Halvorsen-Weare et al. (2012) presented a voyage-based model using pre-generated voyages, taking into account constraints that ensure evenly spread departures. Based on the voyage-based model, Shyshou et al. (2012) proposed a slightly improved formulation for the SVPP and a large neighborhood search (LNS) heuristic to solve the problem instead of using a commercial solver. For moderate-sized instances (with around 14 installations), the LNS heuristic provides solutions with similar or slightly worse quality and reduced computational time compared with the two-stage approach used by Halvorsen-Weare et al. (2012). In this paper, we use the same voyage-based model to describe the SVPP faced by Statoil, and the proposed genetic search-based heuristic provides consistently better solutions to moderate-sized problems using only a fraction of time required by the previous two-stage or LNS methods. Halvorsen-Weare and Fagerholt (2017) also studied an arc-flow model for the SVPP, but their experiments show that a voyage-based model remains more efficient.

Additional studies have been conducted to integrate other problem attributes into the SVPP, such as uncertain weather conditions, schedule robustness and emission reduction. Halvorsen-Weare and Fagerholt (2011) used simulations to investigate the impact of weather conditions. This study aims to generate more robust PSV voyages that allow for unforeseen events (e.g., adverse weather conditions), to avoid installation shut-downs due to lack of supplies. Norlund and Gribkovskaia (2013) studied the reduction of CO_2-emissions and fuel consumption through speed optimization. Norlund et al. (2015) further evaluated the robustness of the schedules, seeking to identify the trade-offs between low emissions and robust schedules.

The SVPP is a type of fleet composition and vehicle routing problem which involves a simultaneous optimization of fleet size, routing and scheduling decisions. The literature on fleet composition problems is extensive, and we refer to the survey of Hoff et al. (2010) for a presentation of fleet composition problems in both maritime and land-based contexts, as well as a discussion on the industrial aspects related to combined fleet composition and routing. Pantuso et al. (2014) presented another literature survey on fleet size and mix problems in maritime transportation.

As mentioned earlier, the SVPP shares many common features with the Periodic Vehicle Routing Problem (PVRP), surveyed in Francis et al. (2008). Vidal et al. (2012) presented a hybrid algorithm for solving a large class of VRPs, including the PVRP. The authors defined the algorithm as a Hybrid Genetic Search Algorithm with Advanced Diversity Control (HGSADC), where the *"hybrid"* term refers to a greater exploitation of problem-specific knowledge via a local-search technique which is systematically applied on new individuals. Later on, Vidal et al. (2014) generalized the HGSADC into a Unified Hybrid Genetic Search (UHGS) metaheuristic for solving more than 40 VRP variants with various attributes. The framework matches or outperforms the current state-of-the-art algorithms for all classical benchmark instances, including the PVRP instances. In this paper, we extend the HGSADC framework with special adaptations aiming to accommodate voyages that span multiple time periods.

3 The supply vessel planning problem

Problem description. In the SVPP, a fleet of PSVs operating from one common onshore supply depot is used to supply a given number of offshore installations on a periodic (weekly) basis. The available PSVs are currently considered to be identical by Statoil, since they only present small differences. The sailing distances between any pair of installations (and between the depot and each installation) are given, and all PSVs sail at the same constant speed. The goal is to identify the optimal size of PSV fleet to acquire and, at the same time, determine the weekly voyages and schedules for the selected vessels, which are usually valid for months until some changes occur (e.g., the arrival or removal of drilling rigs).

During the planning period (one week) each PSV can sail one or more voyages. All voyages start and end at the supply depot, and should respect a minimum and maximum duration limit, as well as a minimum and maximum limit on the number of installations visited. These constraints are imposed to avoid voyages that are too short or too long, since short voyages lead to unexploited capacity, and long ones involve too much uncertainty. The PSVs are characterized by their deck capacity (in square meters), sailing speed, service speed (when unloading supplies at the installations) and time charter rate. Based on the PSV's capacities and the company's rules, the supply quantity on board upon departure must be within a minimum and maximum bound. Each PSV also needs a given number of hours at the supply depot before departure in order to prepare for a new voyage. Therefore, a limited number of PSVs can be prepared for a new voyage in any given day, depending on the opening hours of the supply depot on that day.

Each installation is characterized by a weekly demand for supplies, expressed in square meters, a service frequency (number of visits during a week), and a service duration. The demand of an installation, at each service, is then computed as its weekly demand divided by its service frequency. In practice, some backhauls are also carried from the offshore installations to the onshore depot. Still, these backhauls can be neglected in the problem definition since their volume is almost always smaller than the demand.

Finally, for each installation, the departures of the voyages servicing this installation from the depot must be evenly spread throughout the week. Consider, for example, an installation that requires two services a week, and a solution in which the PSVs servicing it are scheduled to depart from the depot on two consecutive days. If the installation places a delivery request just after the second vessel has left, the next departure may occur six days later, leading to very significant delays. As such, we require to spread the departures from the depot rather than the actual visits to the installations, to mitigate some of the worst-case situations resulting from bad planning and unforeseen events.

Given these definitions, the objective of the SVPP is to minimize the total cost and satisfy the weekly supply requirements of all installations. The total cost includes the sum of the time charter costs of the selected PSVs and the variable costs for sailing the vessels, proportional to the distance traveled.

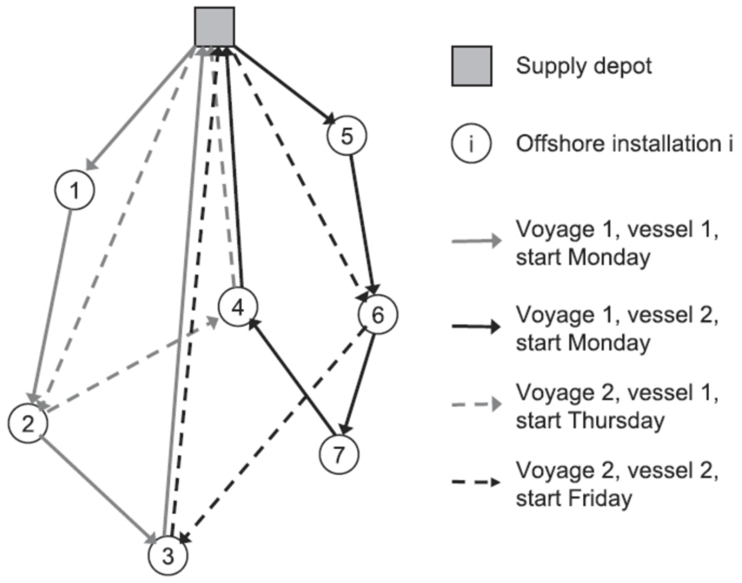

1
2
3
4
5
6
7
Supply depot
i
Offshore installation i
Voyage 1, vessel 1, start Monday
Voyage 1, vessel 2, start Monday
Voyage 2, vessel 1, start Thursday
Voyage 2, vessel 2, start Friday

Figure 2: Example of weekly voyages and schedules for two vessels servicing seven installations with properly spread departures (Halvorsen-Weare et al., 2012).

Figure 2 illustrates an example of solution with well spread departures. In this example, installations 1, 5, and 7 require servicing once a week, and installations 2, 3, 4, and 6 twice a week. Two vessels and in total four voyages are used to supply these seven installations. We observe that the departures towards the installations that require a bi-weekly service are spread, such that the next departure is always four days away at most whenever a supply request is reported.

Mathematical formulation. The current solution approach for the SVPP (Halvorsen-Weare et al., 2012) consists in enumerating the (exponential-sized) set of all feasible voyages, and using it as input of a voyage-based model, described in the following.

Let $\mathcal{R}$ be the set of all candidate voyages, $\mathcal{V}$ the set of PSVs, $\mathcal{N}$ the set of all offshore installations, $\mathcal{T}$ the set of days in the planning horizon, $\mathcal{L}$ the set of all possible voyage duration (in days), and $\mathcal{F}$ the set of all possible visit frequencies (number of weekly visits) for installations. We then denote by $\mathcal{N}_f \subseteq \mathcal{N}$ the set of installations with frequency f, $\mathcal{R}_v \subseteq \mathcal{R}$ the candidate voyages that PSV v may sail, $\mathcal{R}_{vi} \subseteq \mathcal{R}$ the set of candidate voyages of PSV v that visit installation i, and $\mathcal{R}_{vl} \subseteq \mathcal{R}$ the set of candidate voyages of PSV v that has duration l.

Let C_v^{TC} represent the weekly charter cost for using PSV v, and C_{vr}^{S} the sailing cost of PSV v when sailing voyage r. Let S_i be the number of visits required by installation i, F_v the number of days PSV v is available during the planning horizon, and B_t the maximum number of PSVs that may be prepared at the supply depot on day t. To ensure evenly spread departures we use the modeling approach proposed in Shyshou et al. (2012), where $0 \leq h_f \leq |\mathcal{T}|$ is defined to represent the length of an auxiliary sub-horizon for those installations requiring f visits per week. During any sub-horizon h_f, there must be at least $\underline{P_f}$ and no more than $\overline{P_f}$ departures to an installation whose visit frequency is f. For example, to ensure even spread of departures, if the planning horizon is seven days then an installation that requires three visits (i.e., $f = 3$) would need at least one and no more than two departures every three days. This corresponds to $h_3 = 3$, $\underline{P_f} = 1$ and $\overline{P_f} = 2$.

We also define the following decision variables. Let δ_v be a binary decision variable which equals one if PSV v is chartered, and zero otherwise; and x_{vrt} be a binary decision variable which equals one if PSV v sails voyage r starting on day t, and zero otherwise. Based on these definitions, the voyage-based model can be formulated as:

$$\text{minimize} \qquad \sum_{v \in \mathcal{V}} C_v^{TC} \delta_v + \sum_{v \in \mathcal{V}} \sum_{r \in \mathcal{R}_v} \sum_{t \in \mathcal{T}} C_{vr}^{S} x_{vrt}, \tag{1}$$

$$\text{subject to} \qquad \sum_{v\in\mathcal{V}}\sum_{r\in\mathcal{R}_{vi}}\sum_{t\in\mathcal{T}} x_{vrt} \geq S_i \qquad i\in\mathcal{N} \qquad (2)$$

$$\sum_{l\in\mathcal{L}}\sum_{r\in\mathcal{R}_{vl}}\sum_{t\in\mathcal{T}} l x_{vrt} - F_v\delta_v \leq 0 \qquad v\in\mathcal{V} \qquad (3)$$

$$\sum_{v\in\mathcal{V}}\sum_{r\in\mathcal{R}_{v}} x_{vrt} \leq B_t \qquad t\in\mathcal{T} \qquad (4)$$

$$\sum_{r\in\mathcal{R}_{vl}} x_{vrt} + \sum_{r\in\mathcal{R}_{v}}\sum_{\tau=1}^{l-1} x_{vr,(t+\tau)mod|\mathcal{T}|} \leq \delta_v \qquad v\in\mathcal{V}, t\in\mathcal{T}, l\in\mathcal{L} \qquad (5)$$

$$\underline{P_f} \leq \sum_{v\in\mathcal{V}}\sum_{r\in\mathcal{R}_{vi}}\sum_{h=0}^{h_f-1} x_{vr,(t+h)mod|\mathcal{T}|} \leq \overline{P_f} \qquad f\in\mathcal{F}, i\in\mathcal{N}_f, t\in\mathcal{T} \qquad (6)$$

$$\delta_v \in \{0,1\} \qquad v\in\mathcal{V} \qquad (7)$$

$$x_{vrt} \in \{0,1\} \qquad v\in\mathcal{V}, r\in\mathcal{R}_v, t\in\mathcal{T}. \qquad (8)$$

Objective function (1) minimizes the sum of the chartering costs and sailing costs. Constraints (2) ensure the required service frequency for each installation. Constraints (3) ensure that each PSV does not sail more days than allowed. Constraints (4) restrict the number of PSVs prepared at the supply depot on every given day, where *a mod b* denotes the remainder when dividing $a \in \mathbb{N}$ by $b \in \mathbb{N}$. Constraints (5) state that a PSV cannot begin a new voyage before returning from its previous one. Constraints (6) make sure that the departures to each installation are properly spread. The variable domains are given by Constraints (7) and (8).

4 Proposed methodology

A complete solution of the SVPP should specify a suitable number of PSVs to charter in, as well as routing and scheduling decisions for the selected fleet. In the supply vessel planning problem faced by Statoil, however, the PSV charter costs exceed significantly the variable sailing costs. The total sailing costs make up for less than 50% of one PSV's charter cost in the largest problem instances solved by the decision makers. Therefore, fleet size minimization is the primary objective whereas sailing costs minimization comes second.

In accordance with this problem structure, we propose a lexicographic method consisting of two main components. Sections 4.1 to 4.7 describe the first component, a *hybrid genetic search* which works on a *fixed* fleet, and produces near-optimal voyages and schedules for this fleet. Second, Section 4.8 describes our *fleet minimization* approach, which uses the genetic search as a sub-procedure.

Algorithm 1 The Hybrid Genetic Search Procedure

```
1: Initialize population                                                        ▷ Section 4.4
2: while Iterations without improvement $< I^{NI}$ and time $< T^{MAX}$ do
3:     Select parent individuals $s_1$ and $s_2$                                 ▷ Section 4.5
4:     Generate offspring $s_{new}$ from $s_1$ and $s_2$ (crossover)             ▷ Section 4.5
5:     Educate offspring $s_{new}$                                               ▷ Section 4.6
6:     if $s_{new}$ is infeasible then
7:         Repair $s_{new}$ with probability $\rho^{REP}$                        ▷ Section 4.6
8:     end if
9:     if $s_{new}$ is still infeasible then
10:        Insert $s_{new}$ into infeasible subpopulation
11:    else
12:        Insert $s_{new}$ into feasible subpopulation
13:    end if
14:    if maximum subpopulation size $\mu + \lambda$ reached then
15:        Select survivors                                                      ▷ Section 4.7
16:    end if
17:    Adjust penalty parameters for violating feasibility conditions            ▷ Section 4.7
18:    if best individual not improved for $I^{DIV}$ iterations then
19:        Diversify population                                                  ▷ Section 4.7
20:    end if
21:    Return best feasible individual
22: end while
```

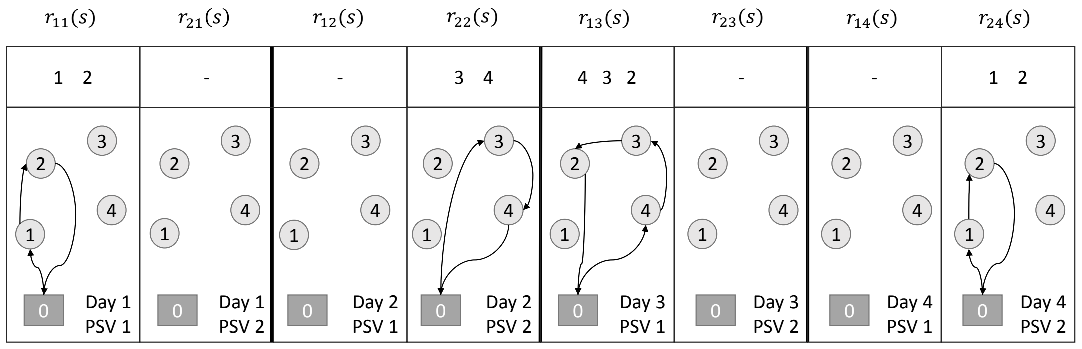
4.1 Overview of the hybrid genetic search procedure

Inspired by the Darwinian principle of the *survival of the fittest* and the natural process of evolution through reproduction, a genetic algorithm (Mitchell, 1998) for an optimization problem evolves a population of individuals, representing solutions, through variation operators (crossover, selection, and mutation) with the goal of producing better ones. To solve the SVPP, we propose a hybrid genetic algorithm based on the Hybrid Genetic Search with Adaptive Diversity Control (HGSADC) metaheuristic introduced in Vidal et al. (2012). The general structure of the method is summarized in Algorithm 1.

Our algorithm works on a *population* of individuals, where each individual corresponds to a (feasible or infeasible) solution of the SVPP with a given fixed fleet, specifying the decisions related to installation-to-PSV assignments, the visit sequences, as well as the departure dates of each PSV. The initialization of the population is described in Section 4.4. It keeps the population $\mathcal{S}$ separated in two disjoint subpopulations: a subpopulation of feasible individuals, and a subpopulation of infeasible individuals. The algorithm keeps creating new individuals (offspring) until there have been I^{NI} iterations without improvement of the best solution, or the maximum running time limit T^{MAX} is attained. One *iteration* here refers to the creation and improvement of one solution. The size of each subpopulation is governed by parameters μ and λ, μ is the minimum size and λ is the generation size (number of offspring), such that the maximum subpopulation size is $\mu + \lambda$. When the maximum size of any subpopulation is reached, its individuals are removed using a *survivor selection* process until there are only μ individuals left in the subpopulation again. We now describe the algorithm's individual representation and evaluation as well as the other search components.

4.2 Individual representation

In the proposed method, each individual is represented as a set of three *chromosomes*: tour chromosome, installation chromosome, and PSV chromosome. Figure 3 illustrates a small example of an individual, representing a solution s of a problem instance with four days, four installations and two PSVs. Each voyage has a duration of at most two days. The top of the figure displays the tour chromosome, which contains an ordered sequence of installations for every combination of day t and PSV v, representing a voyage r_{vt} (an empty sequence would indicate that PSV v does not start a voyage on day t).



(a) Tour chromosome

Inst i	1	2	3	4
Pat $\pi_i(s)$	{1, 4}	{1, 3, 4}	{2, 3}	{2, 3}

(b) Installation chromosome

PSV v	1	2
Pat $\beta_v(s)$	{1, 3}	{2, 4}

(c) PSV chromosome

Figure 3: Illustration of an individual and its tour, installation, and PSV chromosomes.

Then, the installation chromosome represented on the bottom left of Figure 3 provides the *installation pattern* $\pi_i(s)$, i.e., the current departure days to service each installation. For example, pattern {1, 4, 6} on a planning horizon of a week indicates that there are voyages servicing the installation departing every Monday, Thursday and Saturday from the depot. Observe that we can preprocess, for each installation, the set of admissible patterns that satisfy the service frequency and spread departures conditions. Then, checking feasibility with respect to these constraints simply consists in verifying that the pattern of each installation is admissible.

Finally, the PSV chromosome represented in the bottom right of Figure 3 contains, for each PSV v, a *PSV pattern* $\beta_v(s)$ containing the days on which v departs from the depot. The PSV chromosome plays an essential role in the algorithm, as it keeps track of the status of each PSV, to make sure it has returned to the depot before starting a new voyage. This contrasts with the classic PVRP, in which the routes never exceed one time period.

4.3 Evaluation of individuals

In the combinatorial optimization literature, various studies (e.g., Glover and Laguna, 1997; Cordeau et al., 2001; Vidal et al., 2015) have underlined the fact that optimal solutions often lie at the boundary of feasibility, such that allowing a controlled exploration of infeasible solutions in the search process can improve its performance. In line with these observations, we allow infeasible individuals in order to diversify the search, but penalize them proportionally to their amount of infeasibility.

Choosing the right set of constraints to relax is critical for the search performance. We opted to always satisfy the pattern restrictions (i.e., the departure pattern for every installation must be one of its admissible patterns) and maintain a complete solution with visits to all installations. However, the lower and upper limits on voyage duration, supply quantity and number of installations visited during each voyage can be breached with some associated penalty costs.

Penalized solution cost. Let c_{vt} be the sailing cost of a voyage r_{vt} sailed by PSV v departing on day t. Let T_{vt}^{MIN} and T_{vt}^{MAX} be the minimum and maximum duration for the voyage r_{vt}, respectively. The maximum duration T_{vt}^{MAX} depends on the PSV pattern choices for the vessel (Section 4.2), and can be calculated as:

$$T_{vt}^{MAX} = \sum_{p \in \mathcal{P}_v^{PSV}} T_{pt}^{MAX} u_{vp}, \tag{9}$$

where $\mathcal{P}_v^{PSV}$ represents the set of feasible departure patterns for PSV v; and u_{vp} indicates that PSV v uses pattern p when equal to one, and zero otherwise. T_{pt}^{MAX} is the maximum duration of a voyage departing on day t and using PSV pattern p. For example, if a PSV departs on Monday, Wednesday and Saturday, the maximum duration of the voyages sailed by this PSV departing on Monday and Saturday will be of two days in order to be ready for the next departure. The voyage departing on Wednesday, on the other hand, can last three days since there are three days from Wednesday to Saturday.

Furthermore, let Q_v^{MIN} and Q_v^{MAX} represent the minimum and maximum supply quantity carried by PSV v when starting a voyage, and let N^{MIN} and N^{MAX} be the minimum and maximum numbers of installations visited by any voyage. The penalized cost ϕ_{vt} of a voyage r_{vt} is then calculated as follows:

$$\begin{aligned}\phi_{vt} = c_{vt} &+ \omega^D max\{0, T_{vt}^{MIN} - \tau_{vt}, \tau_{vt} - T_{vt}^{MAX}\} \\ &+ \omega^Q max\{0, Q_v^{MIN} - q_{vt}, q_{vt} - Q_v^{MAX}\} \\ &+ \omega^N max\{0, N^{MIN} - n_{vt}, n_{vt} - N^{MAX}\},\end{aligned} \tag{10}$$

where ω^D, ω^Q and ω^N are the *penalty parameters* per unit violation of the constraints on duration, capacity and number of visited installations, respectively; and τ_{vt}, q_{vt} and n_{vt} are the duration, utilized capacity on the PSV and the number of installations visited by voyage r_{vt}, respectively. Finally, the penalized cost ϕ_s of an individual s is the sum of the penalized costs of all its voyages.

Diversity contribution and biased fitness. Relying solely on solution cost for individual evaluations can lead the search towards a premature convergence, where all solutions become identical due to an excessive selection pressure. Hence, as in Vidal et al. (2012), we complement the evaluation of an individual by a measure of its *diversity contribution* $\Delta(s)$, which represents the individual's contribution to the population diversity. Given a distance metric $\delta^H(s_1, s_2)$ between two solutions s_1 and s_2, the diversity contribution $\Delta(s)$ of a solution s is measured as its average distance from its n^{CLO} closest solutions, expressed as:

$$\Delta(s) = \frac{1}{n^{CLO}} \sum_{s' \in \mathcal{N}^{CLO}(s)} \delta^H(s, s'), \tag{11}$$

where $\mathcal{N}^{CLO}(s)$ is the set of the closest solutions to solution s. The distance metric $\delta^H(s_1, s_2)$ used in this equation is a normalized Hamming distance, computed as:

$$\delta^H(s_1, s_2) = \frac{1}{2|\mathcal{N}|} \sum_{i \in \mathcal{N}} (\mathbf{1}(\pi_i(s_1) \neq \pi_i(s_2)) + \mathbf{1}(\mathcal{V}_i(s_1) \neq \mathcal{V}_i(s_2))), \tag{12}$$

where $\mathcal{V}_i(s)$ is the set of PSVs servicing installation i in individual s, and function $\mathbf{1}(\textsc{cond})$ equals one if condition $\textsc{cond}$ is true, and zero otherwise. This distance metric counts the number of installations that have different departure patterns and the number of installations that are serviced by a different set of PSVs. The sum of the two counts is normalized by dividing it by two times the number of installations, giving a value in $[0, 1]$.

The two measures, *penalized solution cost* and *diversity contribution*, help to evaluate the quality of an individual. However, they have different scales and units. Following Vidal et al. (2012), we rely on ranking to avoid any scaling issue: let $Rank^C(s)$ and $Rank^D(s)$ be the ranks of individual s in terms of penalized cost and diversity contribution, respectively (rank 1 being the best). Then, we define the final *biased fitness* of a solution as a weighted sum of both ranks as follows:

$$BF(s) = Rank^C(s) + \left(1 - \frac{n^{ELI}}{|\mathcal{S}|}\right) Rank^D(s). \tag{13}$$

In this equation, the parameter n^{ELI} governs the weight of the diversity contribution in the evaluation. It also represents a number of elite solutions (in terms of penalized solution cost) which are guaranteed to survive to the next generation when selecting out the worst solutions in terms of biased fitness (when a sub-population is full, as described later in Section 4.7).

4.4 Initial Population

The initial population is initialized by creating $K^{INIT} \times \mu$ individuals and allocating them to the appropriate (feasible or infeasible) subpopulation. The number of individuals created should be high enough to create diversity, but not too high to avoid spending extensive computational effort on initial low-quality solutions. Depending on the problem instance and its fleet size, it may be easy or difficult to generate initial feasible solutions. As a consequence, one of the subpopulations may contain less than μ individuals at the end of the initialization phase.

An individual s is created in three steps, one for each chromosome. In the first step, the installation chromosome is created by randomly assigning an installation pattern $\pi_i(s) \in \mathcal{P}_i^{INST}$ to each installation i. The set of days that need departures, denoted $\mathcal{T}^{DEP}(s)$, can then be obtained as the union of all the assigned installation patterns.

The second step consists in creating the PSV chromosome, by randomly assigning each PSV v to a PSV pattern $\beta_v(s)$ that visits at least one day in $\mathcal{T}^{DEP}(s)$, from the set of feasible PSV patterns $\mathcal{P}_v^{PSV}$ for PSV v. When a PSV pattern is assigned, the days in the pattern are removed from $\mathcal{T}^{DEP}(s)$. If $\mathcal{T}^{DEP}(s) \neq \emptyset$ after each PSV has been assigned a PSV pattern, i.e., at least one installation pattern have days with no PSV departing, the second step is restarted. Note that, the PSV patterns could be randomly selected from $\mathcal{P}_v^{PSV}$, but making sure that every selected PSV pattern includes at least one day in $\mathcal{T}^{DEP}(s)$ significantly reduces the number of restarts, as the chance of having at least one PSV departing on every day in $\mathcal{T}^{DEP}(s)$ increases. The second step is also restarted if the depot capacity (maximum number of voyages to be prepared) is violated on any day during the planning horizon.

In the third step, the tour chromosome is created by allocating installations to (PSV, day) couples. The set of installations that have a departure on day t, denoted $\mathcal{N}_t(s)$, can be generated from the installation chromosome. Similarly, the set of PSVs that have a departure on day t, denoted $\mathcal{V}_t(s)$, can be generated from the PSV chromosome. The tour chromosome is created by iterating through all days in the planning horizon, and for each day allocating each installation in $\mathcal{N}_t(s)$ to a randomly-selected PSV in $\mathcal{V}_t(s)$.

Each resulting individual undergoes a local search-based *education* procedure (described later in Section 4.6), and is assigned to the appropriate subpopulation.

4.5 Parent selection and crossover

Crossover is the process where the chromosomes of two parent individuals, s_1 and s_2, are combined into a new offspring individual s_{new}. Each parent is selected by binary tournament, i.e., randomly picking two individuals from the entire population and choosing the one with the best *biased fitness* as the parent. The crossover procedure is described in Algorithm 2 and in the following:

STEP 0. The algorithm begins by selecting which parts of the chromosome should be inherited from which parent. This is done by dividing the set of (PSV, day) couples into three disjoint sets Λ_1, Λ_2 and Λ_{mix}, containing the PSV and days which shall inherit data from s_1, s_2 and both, respectively.

STEP 1. The next step aims to inherit information from s_1. For each (PSV, day) couple in Λ_1, all departures are simply copied from s_1 to s_{new}. For the couples that are in Λ_{mix}, two random cut-off points, α^1_{vt} and α^2_{vt}, are picked, and the installation sequence between α^1_{vt} (inclusive) and α^2_{vt} (exclusive) is copied from s_1 to s_{new}. Note that α^1_{vt} may be larger than α^2_{vt}, in which case the copied sequence is formed by removing the sequence between α^2_{vt} and α^1_{vt}.

STEP 2. This step aims to transmit information from s_2. Since the individuals are not allowed to violate the constraints on the number of visits to each installation, the spread of the departures, and the depot capacity, it is necessary to check their validity when inheriting the installation visits from s_2. These constraints are enforced through the installation patterns and PSV patterns. Line 14 checks that there is not already a departure to installation i on day t, and that the resulting installation pattern is part of at least one of the feasible patterns for installation i. If the condition is not satisfied, installation i cannot be copied to the voyage. If PSV v already departs on day t, there is no change in the PSV pattern and the installation can safely be copied. If PSV v does not depart, Line 17 checks that there is available depot capacity on that day, and that the resulting PSV pattern is part of at least one feasible PSV pattern. Here, $n^{PSV}_t(s_{new})$ equals the number of PSVs departing on day t. If both of these requirements are satisfied, a new voyage is created, and the installation is copied from s_2 to s_{new}.

STEP 3. After all feasible installation departures have been copied from both s_1 and s_2, some visits to installations may still be missing. These visits are subsequently inserted to obtain the offspring individual. To achieve this, we randomly choose an installation missing one or more visits, and insert a visit to it in the cheapest position (in terms of the penalized solution cost) among current voyages. This process is repeated until no missing

Algorithm 2 Crossover operator

```
 1: Given two parent individuals $s_1$ and $s_2$

    STEP 0: INHERITANCE RULE
 2: Pick two random integer numbers between 0 and $|\mathcal{T}| \times |\mathcal{V}|$ according to a uniform
    distribution. Let $n_1$ and $n_2$ be the smallest and the largest of these numbers, respectively
 3: Randomly select $n_1$ (PSV, day) couples to form the set $\Lambda_1$
 4: Randomly select $n_2 - n_1$ remaining couples to form the set $\Lambda_2$
 5: The remaining $|\mathcal{V}| \times |\mathcal{T}| - n_2$ couples make up the set $\Lambda_{mix}$

    STEP 1: INHERIT DATA FROM $s_1$
 6: for each (PSV, day) $(v, t)$ belonging to set $\Lambda_1$ do
 7:     Copy the sequence of installation departures from $r_{vt}(s_1)$ to $r_{vt}(s_{new})$
 8: end for
 9: for each (PSV, day) $(v, t)$ belonging to set $\Lambda_{mix}$ do
10:     Randomly (uniform distribution) select two cut-off points $\alpha^1_{vt}$ and $\alpha^2_{vt}$ and copy the
        $\alpha^1_{vt}$ to $\alpha^2_{vt}$ substring of $r_{vt}(s_1)$ to $r_{vt}(s_{new})$
11: end for

    STEP 2: INHERIT DATA FROM $s_2$
12: for each (PSV, day) $(v, t) \in \Lambda_2 \cup \Lambda_{mix}$ selected in random order do
13:     for each installation departure $i$ in $r_{vt}(s_2)$ do
14:         if $t \notin \pi_i(s_{new})$ and $\exists p \in \mathcal{P}^{INST}_i \mid p \supset (t \cup \pi_i(s_{new}))$ then
15:             if $t \in \beta_v(s_{new})$ then
16:                 Copy installation $i$ at the end of $r_{vt}(s_{new})$
17:             else if $n^{PSV}_t(s_{new}) < B_t$ and $\exists p \in \mathcal{P}^{PSV}_v \mid p \supset (t \cup \beta_v(s_{new}))$ then
18:                 Create new voyage $r_{vt}(s_{new})$ and insert installation $i$
19:             end if
20:         end if
21:     end for
22: end for

    STEP 3: COMPLETE INSTALLATION SERVICES
23: while there are installations with unsatisfied service frequency requirements do
24:     $i \leftarrow$ random installation for which service frequency requirements are not satisfied
25:     Let $\mathcal{F}$ be the set of feasible (PSV, day) combinations $(v, t)$ with respect to the
        feasible installation patterns, feasible PSV patterns and depot capacity based on
        the visits already in $s_{new}$.
26:     if $\mathcal{F} = \emptyset$ then
27:         go to 1                         $\triangleright$ no feasible insertion, restart the procedure
28:     else
29:         Let $\psi(i, v, t)$ be the minimum penalized cost for the insertion of installation $i$
            into the voyage sailed by PSV $v$ on day $t$.
30:         Insert $i$ at least cost position in $r_{vt}$, where $(v, t) = \text{argmin}_{(v,t) \in \mathcal{F}}\ \psi(i, v, t)$.
31:     end if
32: end while

33: Return $s_{new}$ as the new offspring individual.
```

visits remain.

The design of this crossover operator ensures that s_{new} satisfies the required numbers of installation visits, has a feasible installation pattern and does not exceed the depot capacity. Still, s_{new} may contain voyages that violate the minimum and maximum bounds on the number of installations, duration and PSV capacity. These violations will be taken into account in the penalized cost.

4.6 Education and Repair

A local-search based education procedure is used to improve each initial solution and offspring individual generated by the crossover. This procedure is performed in three steps: voyage improvement, pattern improvement, and voyage improvement once again.

The voyage improvement procedure consists in a local search which aims to improve the order of visits in each voyage, but does not change the installation or PSV patterns. The pattern improvement procedure, on the other hand, attempts to improve the individuals by changing these patterns.

4.6.1 Voyage improvement

Let the neighborhood $\Gamma(u)$ of an installation u be defined as the $h \times n$ installations closest to it (according to distance) in the voyage, where n is the number of installations visited on the voyage and $h \in [0, 1]$ is a parameter restricting the size of the neighborhood. The voyage improvement procedure is a local search approach which evaluates, for each installation pair u and $v \in \Gamma(u)$, in random order, the solutions obtained by the following moves, where x and y are the successors of u and v, respectively:

- (M1) Remove u and place it after v;
- (M2) Remove u and x and place u and x after v;
- (M3) Remove u and x and place x and u after v;
- (M4) Swap the position of u and v;
- (M5) Swap the position of u and x with v;
- (M6) Swap the position of u and x with v and y;
- (M7) Swap the position of x and v.

Any improving move, in terms of penalized cost, is directly applied to the solution, and the voyage improvement stops once all moves have been consecutively evaluated without solution improvement.

4.6.2 Pattern improvement

Three sub-procedures are performed in the following order in the pattern improvement procedure: (1) changing installation departure patterns, (2) merging multiple voyages departing on the same day, and (3) moving installation departures from short voyages to other voyages. The last two sub-procedures aim to reduce the total number of voyages sailed. The reason for this is that the distances between the depot and the installations often are longer than those between installations, resulting in a high start-up cost for a voyage. Therefore, fewer voyages usually lead to cheaper schedules.

Changing installation departure patterns. This sub-procedure iterates through all installations once and, for each installation, attempts to improve its departure pattern. This is done by removing all departures to the installation, looping through all feasible departure patterns for the installation and reinserting the departures to the least costly positions on the days given by the installation pattern. The pattern leading to the smallest penalized cost is used if it leads to an improvement of the individual.

Merging voyages. This sub-procedure attempts to merge two voyages departing on the same day into one. This is done by iterating through all days once and, for each day that has more than one voyage, calculating the penalized cost of merging all combinations of two voyages. The merging which reduces the penalized cost the most, if any, is performed on that day.

Reducing the number of voyages. This sub-procedure aims to reduce the number of voyages sailed by moving installation departures from days with short voyages to other days, possibly changing several patterns at the same time to escape from local optima.

Let $n_t^{REM}(s)$ be the smallest number of installation departures that need to be moved to another day in order to reduce the number of voyages sailed on day t. The procedure starts by selecting day t with the lowest $n_t^{REM}(s)$, and changes the installation pattern of those installations having a departure on t. Let $\mathcal{P}_{it}^{MOV}$ be the set of feasible patterns for installation i that do not contain day t. If this set is non-empty, i is *movable* and is included in the set of movable installations with departures on day t, $\mathcal{N}_t^{MOV}(s)$. If $n_t^{REM}(s) > |\mathcal{N}_t^{MOV}(s)|$, there are not enough movable installations to remove a voyage, and the procedure stops. Else, the procedure continues by finding the (installation, pattern)-combination (i, p) that results in the lowest penalized cost, where $i \in \mathcal{N}_t^{MOV}(s), p \in \mathcal{P}_{it}^{MOV}$. After the pattern is changed, the installation is no longer part of $\mathcal{N}_t^{MOV}(s)$, hence $n_t^{REM}(s)$ is decreased by one. The procedure keeps changing the pattern of the installation with the lowest move cost until $n_t^{REM}(s) = 0$. If the penalized cost of the individual is reduced as a result of the procedure, the changes to the individual are performed; if not, it remains as before the procedure started.

4.6.3 Repair

If the resulting individual after education is feasible, it is referred to as *naturally feasible*. If it is infeasible, *repair* takes place with probability ρ^{REP}. The repair procedure consists in applying the education algorithm with $10\times$ higher penalty parameters. If the individual still is infeasible, the education algorithm is run again with $100\times$ higher parameter values.

4.7 Population management

Three different population management mechanisms are used to guide the search. These are the *survivor selection*, *penalty parameter adjustment* and *diversification* mechanisms, all explained below. The aim of these mechanisms is to maintain a good balance between feasible and infeasible individuals, in order to maintain the diversity of the population as well as high-quality individuals. These mechanisms impact the entire population simultaneously, contrary to crossover, education and repair, which only affect one individual at a time.

Survivor selection. Survivor selection discards both bad quality individuals and *clones*, i.e., individuals that have a Hamming distance of zero to another individual. Survivor selection is performed on a subpopulation whenever its size reaches its upper limit $\mu + \lambda$. This mechanism iteratively removes individuals until there are μ individuals left, by first removing clones; when no clones are left, removing the individuals with highest biased fitness.

Penalty parameter adjustment. The penalty parameters (see Section 4.3) are adjusted every 100 iterations, in order to guide the search towards a desired share of *naturally feasible* individuals. Let ξ^D, ξ^Q and ξ^N be the share of naturally feasible individuals among the last 100 generated individuals with respect to voyage duration, PSV capacity and number of installations, respectively. Let ξ^{REF} be a method parameter, which represents a target ratio of naturally feasible individuals. The penalty parameters are individually adjusted as shown in Algorithm 3, where ζ^{UP} and ζ^{DOWN} are the adjustment factors for the penalties.

Algorithm 3 Penalty parameter adjustment

```
1: for $p = D, Q, N$ do                          ▷ For each type of penalty
2:     if $\xi^p \leq \xi^{REF} - 0.05$ then
3:         $\omega^p = \omega^p \times \zeta^{UP}$
4:     else if $\xi^p \geq \xi^{REF} + 0.05$ then
5:         $\omega^p = \omega^p \times \zeta^{DOWN}$
6:     end if
7: end for
```

Diversification. Finally, a diversification procedure is used to recover population diversity when the algorithm becomes trapped in a local optimum. This procedure is called in case no improvement has been made to the best individual the last I^{DIV} iterations. Diversification works by removing all but the best third of each subpopulation, in terms of biased fitness, and generating $K^{DIV} \times \mu$ new individuals as in the population initialization.

4.8 Fleet minimization

The hybrid genetic search algorithm described in the previous sections minimizes the sailing costs using a given fixed fleet. The objective of the SVPP is, however, to minimize the total costs related to both the usage (chartering) and the operation of the PSVs. Hence, finding a solution of the SVPP also requires to determine a good fleet size.

Recall that in the supply vessel planning problem faced by Statoil, the PSV charter costs exceed significantly the variable sailing costs. As such, sailing cost reductions obtained by increasing the fleet size can never offset the related increase in charter costs. This implies that if there exists a feasible fleet of size k, then the size of the optimal fleet is at most k.

The fleet size optimization procedure thus searches for the smallest number of vessels needed to service all installations. It starts from fleet size $|\mathcal{V}|$ and iteratively attempts to reduce the fleet size by one and search for a feasible solution. For instance, at fleet size $|\mathcal{V}| - 1$ the genetic search is run until the first feasible schedule is found; the fleet size is then decremented and the method is run with a fleet size of $|\mathcal{V}| - 2$. This process is repeated until no feasible schedule can be found for a size k_{MIN}. Then, a final complete genetic search is operated with a fleet size of $k_{\text{MIN}} + 1$ to optimize the sailing costs, and the solution is returned.

5 Computational experiments

In this section, we evaluate the performance of the proposed heuristic for realistic SVPPs. The algorithm is implemented in Java, and our tests are run on an Intel 2.4 GHz processor with 8 GB memory and a single thread.

We use a total of 25 problem instances in our computational study, all of which are based on real cases faced by Statoil in the North Sea as of April 2016. The instances consider one supply depot, three to 27 offshore installations and 10 to 80 total weekly visits. The complete list of instances can be visualized in Table 7, in Section 5.3. Each instance is named after its corresponding number of installations and total number of weekly visits. All instances consider a planning horizon of one week and a pool of six identical PSVs from which Statoil selects for time charter. The resulting fleet size, routing and scheduling decisions will be implemented every week for the months to come. The number of visits required by an installation ranges from one to five per week, and the service time at each installation ranges from one and a half to four hours. The supply depot is open for eight hours (08.00-16.00) from Monday to Saturday and is closed on Sunday. Also, a PSV needs to be at the supply depot before 08.00 to start on a new voyage the same day, and all voyages depart from the depot at 16.00. Finally, every voyage sailed by a PSV is constrained to visit between one and eight installations, and can last a maximum of three days.

In Section 5.1 we present the calibration of the parameters used in the metaheuristic. The experimental results on the test instances are then analyzed in Section 5.2.

5.1 Parameter setting

The *hybrid genetic search procedure* is at the core of the proposed heuristic, and thus its parameters must be carefully calibrated to efficiently solve the real test cases. In light with Vidal et al. (2012, 2014), we observed that the values of most parameters do not significantly affect the performance, as long as they are within reasonable ranges. In this paper those parameters were set without calibration, to $K^{INIT} = 4$ (construction heuristic size factor) and $K^{DIV} = 4$ (diversification size factor); $\zeta^{UP} = 1.2$ and $\zeta^{DOWN} = 0.85$ (penalty adjustment factors). The starting values of the penalties were set to higher values of $\omega^{Q} = \omega^{T} = \omega^{N} = 1000$ to quickly reach the first feasible solution. Note that the penalties are adjusted dynamically by the algorithm, and thus their initial values are of secondary importance. However, carefully setting these values according to the scale of the objective function allows to speed up the initial stage of the search.

We have also identified some parameters which have a more visible effect on the performance of the metaheuristic. These parameters, their descriptions and final values are shown in Table 1. The parameters ξ^{REF} and η^{ELI} have been calibrated individually. The four others have been calibrated in pairs, (μ, λ) and (I^{NI}, η^{DIV}), as they are significantly correlated. The problem instances P-12-40, P-20-68 and P-27-80 were used for the calibration, as they are representative of medium and large problems which are challenging for Statoil. When a particular parameter or pair of parameters is calibrated, the others are set to their standard value indicated in Table 1. Finally, since the heuristic is non-deterministic, we report its average run time over five runs, as well as the percentage gap between its average solution value z and the best known solution (BKS) value z_{BKS} for the instance, computed as Gap(%) $= 100 \times (z - z_{\text{BKS}})/z_{\text{BKS}}$.

Calibration of μ and λ. The minimum subpopulation size μ and the generation size λ were calibrated together, since they both impact the diversity of the population. Table 2 displays the average run time and gaps for different combinations of μ and λ. The combination $\mu = 25$ and $\lambda = 75$ was selected, as it leads to the lowest objective gaps for both P-20-68 and P-27-80.

Table 1: Parameters calibrated and their values after calibration.

Param.	Value	Description
μ	25	Minimum subpopulation size
λ	75	Generation size
I^{NI}	5 000	Maximum number of consecutive iterations without improvement
η^{DIV}	0.1	Proportion of I^{NI}, such that $I^{DIV} = \eta^{DIV} \times I^{NI}$
ξ^{REF}	0.6	Target ratio of feasible individuals
η^{ELI}	0.4	Proportion of elite individuals, such that $n^{ELI} = \eta^{ELI} \times \lvert\mathcal{S}\rvert$

Table 2: Calibration of μ and λ

		$\lambda = 25$	$\lambda = 50$	$\lambda = 75$	$\lambda = 100$
Instance	μ	Time (s) | Gap	Time (s) | Gap	Time (s) | Gap	Time (s) | Gap
P-12-40					
	15	77.0 | 0.00%	95.8 | 0.00%	95.9 | 0.00%	120.9 | 0.00%
	25	86.8 | 0.00%	100.6 | 0.00%	113.3 | 0.00%	148.8 | 0.00%
	35	95.4 | 0.00%	110.8 | 0.00%	130.9 | 0.00%	182.3 | 0.00%
P-20-68					
	15	567.2 | 0.13%	743.9 | 0.10%	708.2 | 0.06%	651.4 | 0.08%
	25	960.5 | 0.05%	724.8 | 0.05%	707.0 | 0.03%	814.0 | 0.06%
	35	780.7 | 0.05%	963.2 | 0.04%	845.6 | 0.05%	1089.1 | 0.05%
P-27-80					
	15	818.3 | 0.32%	678.6 | 0.20%	868.0 | 0.19%	852.3 | 0.20%
	25	746.5 | 0.22%	691.2 | 0.21%	867.0 | 0.15%	875.8 | 0.25%
	35	899.2 | 0.17%	1230.9 | 0.23%	781.6 | 0.26%	1655.6 | 0.20%

Calibration of I^{NI} and η^{DIV}. From our preliminary experiments, a termination criterion of at least 4000 iterations without improvement appeared to be necessary to ensure that our algorithm does not terminate prematurely. We therefore set the base configuration to $I^{NI} = 5000$, and compared it with a longer termination criterion of $I^{NI} = 10000$ iterations. The diversity criterion, η^{DIV}, is the percentage value used to calculate $I^{DIV} = \eta^{DIV} \times I^{NI}$. It determines how frequently the diversification is performed.

Table 3 shows the results for different combinations of I^{NI} and η^{DIV}. Increasing I^{NI} beyond 5000 did not improve the solution quality but significantly increased the run time, and thus we maintained this value. Moreover, $\eta^{DIV} = 0.1$ was selected, as it leads to the best average gap for instance P-27-80. Note that the high average gap obtained when using $\eta^{DIV} = 0.7$ and $I^{NI} = 10000$, for instance P-27-80, was caused by one run with a solution containing five PSVs, while the other runs used four. This happened because no feasible

Table 3: Calibration of I^{NI} and η^{DIV}.

		$I^{NI} = 5\ 000$	$I^{NI} = 10\ 000$
Instance	η^{DIV}	Time (s) | Gap	Time (s) | Gap
P-12-40			
	0.1	89.6 | 0.00%	152.1 | 0.00%
	0.4	81.9 | 0.00%	146.5 | 0.00%
	0.7	86.5 | 0.00%	147.3 | 0.00%
P-20-68			
	0.1	753.5 | 0.05%	1 077.7 | 0.07%
	0.4	781.5 | 0.05%	1 196.7 | 0.05%
	0.7	602.2 | 0.09%	1 024.8 | 0.05%
P-27-80			
	0.1	781.3 | 0.18%	1 229.8 | 0.21%
	0.4	1 023.2 | 0.21%	1 061.8 | 0.18%
	0.7	789.9 | 0.22%	1 257.8 | 4.60%

solution was found with four PSVs during this particular run. A poor initial population and a high value of η^{DIV} may be the reason for this behavior, since a high η^{DIV} leads to less frequent diversifications and thus increases the impact of a poorly diversified initial population.

Calibration of ξ^{REF}. This parameter acts as a target for the ratio of feasible individuals in the population, during penalty adaptations (Section 4.7). A high value for ξ^{REF} leads to higher penalties, hence guiding the search towards more feasible solutions. Table 4 reports the average run time and gap obtained with different values of ξ^{REF} on the selected instances. From our experiments, $\xi^{REF} = 0.6$ led to the lowest average objective gap for instances P-20-68 and P-27-80, and was chosen as the final calibrated value.

Table 4: Calibration of ξ^{REF}.

	$\xi^{REF} = 0.2$	$\xi^{REF} = 0.4$	$\xi^{REF} = 0.6$	$\xi^{REF} = 0.8$
Instance	Time (s) | Gap	Time (s) | Gap	Time (s) | Gap	Time (s) | Gap
P-12-40	99.1 | 0.00%	93.1 | 0.00%	85.9 | 0.00%	82.2 | 0.00%
P-20-68	733.6 | 5.65%	779.7 | 0.11%	639.0 | 0.06%	578.6 | 0.08%
P-27-80	902.8 | 0.23%	778.9 | 0.22%	713.9 | 0.18%	949.0 | 0.28%

Calibration of η^{ELI}. Finally, η^{ELI} represents the proportion of elite individuals within the population. It is used in the biased fitness measure described in Section 4.3. A higher value of η^{ELI} increases the importance of the penalized cost and decreases the importance of the diversity contribution, resulting in increased elitism in the search. Table 5 reports the calibration results for η^{ELI}. The best gap was attained with $\eta^{ELI} = 0.4$ for a moderate increase of run time, and thus this value was selected for the algorithm.

Table 5: Calibration of η^{ELI}.

	$\eta^{ELI} = 0.2$	$\eta^{ELI} = 0.4$	$\eta^{ELI} = 0.6$	$\eta^{ELI} = 0.8$
Instance	Time (s) | Gap	Time (s) | Gap	Time (s) | Gap	Time (s) | Gap
P-12-40	141.3 | 0.00%	149.5 | 0.00%	140.5 | 0.00%	131.2 | 0.00%
P-20-68	947.3 | 0.11%	1 079.9 | 0.05%	948.5 | 0.05%	762.6 | 0.11%
P-27-80	829.9 | 0.38%	1 348.5 | 0.20%	1 045.7 | 0.25%	778.3 | 0.27%

5.2 Comparison with the commercial solver

We compare the performance of the proposed metaheuristic with the resolution of the voyage-based model (Section 3 – used by Statoil) by means of the commercial solver Xpress 7.8. We refer to the latter approach as the VBM method.

The generation of candidate voyages is an essential component of the VBM method, and requires to produce, for each feasible subset of installations (with respect to the constraints on the load and number of visits), the voyage with the smallest sailing cost and duration. In Halvorsen-Weare et al. (2012), this task was done by solving a set of TSPs since the number of installations was relatively small. This generation approach, however, becomes time-consuming for larger instances, such that we have developed a more efficient dynamic programming (DP) algorithm for this task.

Dynamic programming solution approaches are frequently used to solve variants of shortest path and traveling salesman problems, in which partial paths are represented by *labels* (Ahuja et al., 1993). The algorithm starts from the trivial path including only the supply depot, and then builds new paths by extending each existing label to a new installation. Dominance rules are used to eliminate labels during the generation. Moreover, in contrast with the approach of Halvorsen-Weare et al. (2012), where the constraints on the load and number of visits were implicitly satisfied due to the prior selection of *feasible* sets of installations, the algorithm must now satisfy those constraints during the extension of the labels. This is done via *resource constraints* (Irnich and Desaulniers, 2005), allowing to further reduce the number of labels at each step of the algorithm.

Table 6 now compares the performance of the metaheuristic with that of the VBM approach. We restrict this comparison to the set of instances with up to 14 installations and 48 weekly visits, since P-14-48 is the largest instance for which the VBM method can find a feasible solution within an allowed time limit of 10,000 seconds. The first group of columns provide the results of the VBM approach: the numbers of voyages generated, the run time taken by the voyage generation phase, the optimality gaps at the end of the

Table 6: Results of solving the instances with up to 14 installations using the VBM method and the proposed heuristic based on 10 runs.

	VBM				Heuristic	
Problem instance	#Voyages Generated	Voy. Gen. Time (s)	Optimality gap	Total Time (s)	Gap from VBM	Time (s)
P-3-10	7	0.4	0.0%	0.5	0.00%	17.2
P-4-13	15	0.4	0.0%	0.5	0.00%	26.5
P-5-16	31	0.5	0.0%	0.6	0.00%	60.4
P-6-17	63	0.6	0.0%	0.7	0.00%	60.6
P-7-22	127	0.7	0.0%	1.6	0.00%	66.4
P-8-26	255	0.7	0.0%	3.8	0.00%	74.1
P-9-29	510	0.8	0.0%	12.9	0.00%	96.4
P-10-32	1012	0.9	0.0%	69.9	0.00%	93.5
P-11-36	1980	1.8	0.0%	197.9	0.00%	118.4
P-12-40	3796	5.8	0.0%	43.1	0.00%	129.6
P-13-44	7098	23.0	22.3%	>10,000	-0.03%	238.5
P-14-48	12,910	77.3	19.4%	>10,000	-0.60%	166.9

method, and the total run time, including both the generation of the voyages and the resolution of the model with Xpress. The remaining columns present the results of the hybrid genetic algorithm: its average gap over ten runs (relative to the best primal solution produced by the VBM method) and average run time.

From Table 6, we observe that the proposed heuristic outperforms the VBM method in terms of solution quality and time. The VBM method can solve to optimality the instances with up to 40 weekly visits, but for the next two larger problems, the optimality gap remains very large when reaching the time limit of 10,000 seconds. To further examine the potential of the VBM approach, we have also tested it on the same instances, but assuming that the optimal fleet size is known (as this may be relevant in some application contexts). For the instance P-14-48, when the fleet size is fixed to the optimal value of three, the VBM approach can solve the problem to optimality in 224 seconds. However, for a slightly larger problem (P-15-52), the time taken quickly increases to 3620 seconds, and the method fails to find a feasible solution in 10,000 seconds for the next instance (P-16-55). This shows that VBM still remains impracticable, on large problems, even with the a-priori knowledge of an optimal fleet size.

In comparison, the hybrid genetic search systematically finds an optimal solution, on every run, for all instances with up to 40 services, and it produces better solutions for the last two cases. It requires more time on small instances than VBM, but the rate of increase of the run time remains limited when the problem size grows. Such a good scalability is essential for the success of the algorithm on larger test cases of Statoil. With this goal in mind, the computational time and solution quality analyses are now extended to the larger instances in the next section.

5.3 Stability and scalability

Table 7 summarizes the results of the proposed metaheuristic over 10 executions for all the 25 test instances, including the larger test cases from Statoil. The columns "#PSVs" and "#Voyages" indicate the numbers of PSVs and voyages used (the initial fleet size being set to six in all cases). We observe that these values were the same over the 10 runs, indicating that the heuristic finds the same fleet size for all runs of each instance. The table also shows the average run time over 10 executions, and the coefficient of variation (CV) of the total cost, sailing cost and run time. The CV is calculated as the standard deviation divided by the mean. The charter costs are not shown here, as they are always identical over the 10 runs, due to the same fleet size found.

The average CV of the total cost and sailing costs are 0.02% and 0.19%, respectively, and thus the heuristic appears to be very stable in terms of solution quality. To further estimate the growth of the run time as the problem instances become bigger, we fitted the run time of the method as a power law $f(n_s) = \alpha \times n_s^{\beta}$ of the number of weekly visits n_s (done as the least-squares regression of an affine function on the log-log graph). This regression is depicted in Figure 4.

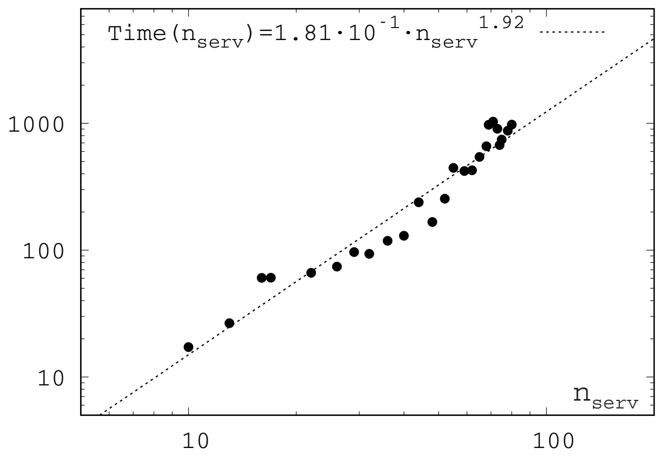


Figure 4: Growth of the run time as a function of the number of weekly visits to installations. Log-log scale.

From these experimental analyses, the observed run time of the method grows in $\mathcal{O}(n_s^{1.92})$, i.e., subquadratically. This good behavior allows the algorithm to stand the test of time as the operations of Statoil become larger and more integrated.

In addition, we remind that the proposed heuristic consists of two components: a fleet optimization routine which checks for feasibility while reducing the fleet size, until

Table 7: Results of the proposed metaheuristic on all problem instances.

Problem instance	#PSVs	#Voyages	Time (s)	Coefficient of variation: Total Cost	Coefficient of variation: Sailing Cost	Coefficient of variation: Run Time
P-3-10	2	4	17.2	0.00%	0.00%	8.18%
P-4-13	2	4	26.5	0.00%	0.00%	7.22%
P-5-16	2	4	60.4	0.00%	0.00%	7.37%
P-6-17	2	4	60.6	0.00%	0.00%	0.91%
P-7-22	2	5	66.4	0.00%	0.00%	0.58%
P-8-26	2	5	74.1	0.00%	0.00%	2.38%
P-9-29	2	5	96.4	0.00%	0.00%	6.85%
P-10-32	2	5	93.5	0.00%	0.00%	5.46%
P-11-36	2	5	118.4	0.00%	0.00%	14.11%
P-12-40	2	6	129.6	0.00%	0.00%	7.60%
P-13-44	3	6	238.5	0.00%	0.00%	4.49%
P-14-48	3	6	166.9	0.00%	0.00%	11.86%
P-15-52	3	7	254.9	0.00%	0.01%	24.51%
P-16-55	3	8	445.7	0.02%	0.18%	27.48%
P-17-59	3	8	421.2	0.01%	0.12%	28.12%
P-18-62	3	8	426.4	0.16%	1.22%	16.66%
P-19-65	3	9	542.3	0.04%	0.31%	16.67%
P-20-68	4	9	661.0	0.04%	0.34%	17.06%
P-21-69	4	9	973.5	0.04%	0.37%	26.09%
P-22-71	4	9	1030.0	0.03%	0.28%	22.73%
P-23-73	4	10	904.1	0.02%	0.21%	31.88%
P-24-74	4	10	672.7	0.03%	0.24%	35.55%
P-25-75	4	10	743.9	0.03%	0.26%	21.54%
P-26-78	4	10	873.8	0.05%	0.43%	36.38%
P-27-80	4	11	976.6	0.08%	0.68%	27.59%
Average				**0.02%**	**0.19%**	**16.37%**

no feasible schedule can be found for a size k_{MIN}, and then a final hybrid genetic search on a fleet of size $k_{\text{MIN}} + 1$. Our experiments have shown that the fleet size optimization procedure consumes around 10 to 30% of the total run time of the heuristic. Most of this time is spent on confirming the infeasibility of the fleet size k_{MIN}. In contrast, checking the feasibility of larger fleet sizes is easier for the algorithm: in most occasions, at least one feasible solution is obtained when building the initial population; otherwise, the first feasible solution is found within 2000 iterations in the worst case.

5.4 Effect of education

In Vidal et al. (2012), the authors presented a sensitivity analysis of the main components of the HGSADC, and reported that the education impacts the objective value the most. In our extension of the method, the education component involves a voyage improvement and pattern improvement procedure, the latter being subdivided into three steps: an improvement via single exchanges of installation departure patterns (similar to the pattern improvement procedure proposed by Vidal et al. 2012), and two steps which aim to reduce the number of voyages. Table 8 analyses the effect of different levels of education on the three selected instances:

- No education means that no education is carried out at all;
- Simple education includes the same education procedure as in Vidal et al. (2012), i.e., the voyage improvement procedure and the first pattern improvement step;
- Full education includes all mentioned education procedures.

Table 8: Impact of three different forms of education on the solution quality of the heuristic

	No education	Simple education	Full education
Instance	Time (s) | Gap	Time (s) | Gap	Time (s) | Gap
P-12-40	84.2 | 14.17%	116.6 | 0.00%	129.1 | 0.00%
P-20-68	140.8 | 1.06%	835.1 | 0.03%	1 125.1 | 0.02%
P-27-80	184.7 | 23.98%	681.8 | 0.24%	824.4 | 0.18%

From these experiments, we observe that the gap, in terms of objective value, is very large when no education is performed. This confirms the well-known fact that the inclusion of a local search-based education in the hybrid genetic algorithm contributes largely to the search. The difference between the simple and full education is less marked. Still, the complete education procedure led to some improvements of solution quality on the largest instance, 0.18% compared to 0.24%, at the price of a 30% increase in terms of run time. These differences are likely to become more significant as the instance size and the number of PSV grow.

6 Conclusions

In this paper, we have studied a real life supply vessel planning problem (SVPP) faced by the oil and gas company Statoil. Up to this date, the company had been using a solution approach based on integer programming over a voyage-based model. This approach, however, was limited to problems of small or medium size, with up to 14 installations and 48 weekly services. As the operations of Statoil become more consolidated, larger problems must be solved and the current approach turns out to be impracticable.

To respond to this challenge, we introduced a new metaheuristic for this practical problem, based on the Hybrid Genetic Search with Adaptive Diversity Control (HGSADC) of Vidal et al. (2012). Our extension of this method has been specifically designed to address additional problem features which are specific to maritime shipping, e.g., voyages that span more than one unit of time, via a problem-tailored solution representation, new crossover and education operators. Our computational experiments on real test instances demonstrate the excellent performance of the new metaheuristic, which retrieves all known optimal solutions on all runs, and improves the best known results for previous instances which were not solved to optimality. The method also scales very well with problem size. Experimentally, the run time of the heuristic appears to grow sub-quadratically, in $\mathcal{O}(n_s^{1.92})$, as a function of the number of installation visits per week. This growth is small enough to envisage the resolution of much larger test cases, which arise when consolidating the operations of more PSVs and onshore supply depots.

Many perspectives remain open with respect to this work. First, the method is not necessarily limited to shipping, and can also be applied to many land-based transportation applications which present similar features (e.g., long-haul routing optimization over multiple time periods). Second, the proposed heuristic was designed to accommodate a pool of vessels of equal characteristics according to the data given by Statoil, but an extension to heterogeneous fleets is also possible via an extension of the PSV chromosome along with dedicated local-search moves. Finally, the considered model is only a simplification of a very complex situation, where various time and service level constraints impact the solution, and different objectives interact. To better capture the trade-offs involved in such decisions, it may be desirable to progress towards multi-objective optimization models to include other practical concerns involved in Statoil's daily operations, such as the robustness and persistence of the planned schedules in case of disruption in some voyages due to bad weather, or if Statoil decides to shut down an installation or open a new one.

Acknowledgments

The authors acknowledge financial support from project *Maritim Offshore Logistikk Optimering* (MOLO) partly funded by the Research Council of Norway. The authors would also like to thank Tor Toftøy and Ellen Karoline Norlund at Statoil.

References

Aas, B., Halskau Sr, Ø., and Wallace, S. W. (2009). The role of supply vessels in offshore logistics. *Maritime Economics & Logistics*, 1(3):302–325.

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993). *Network Flows: Theory, Algorithms, and Applications*. Pearson Prentice-Hall, Upper Saddle River, NJ, USA, 1 edition.

Cordeau, J.-F., Laporte, G., and Mercier, A. (2001). A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society*, 52(8):928–936.

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

Francis, P. M., Smilowitz, K. R., and Tzur, M. (2008). The Period Vehicle Routing Problem and its Extensions. In Golden, B. L., Raghavan, S., and Wasil, E. A., editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 73–102. Springer US, Boston, MA.

Glover, F. and Laguna, M. (1997). *Tabu search.* Kluwer Academic Publishers, Norwell, MA.

Halvorsen-Weare, E. E. and Fagerholt, K. (2011). Robust Supply Vessel Planning. In Pahl, J., Reiners, T., and Voß, S., editors, *Network optimization, INOC 2011, Lecture Notes in Computer Science*, volume 6701, pages 559–573. Springer.

Halvorsen-Weare, E. E. and Fagerholt, K. (2017). Optimization in offshore supply vessel planning. *Optimization and Engineering*, 18(1):317–341.

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

Hoff, A., Andersson, H., Christiansen, M., Hasle, G., and Løkketangen, A. (2010). Industrial aspects and literature survey: Fleet composition and routing. *Computers & Operations Research*, 37(12):2041–2061.

Irnich, S. and Desaulniers, G. (2005). Shortest Path Problems with Resource Constraints. In Desaulniers, G., Desrosiers, J., and Solomon, M. M., editors, *Column Generation*, chapter 2, pages 33–65. Springer US.

Mitchell, M. (1998). *An introduction to genetic algorithms*. MIT press.

Norlund, E. K. and Gribkovskaia, I. (2013). Reducing emissions through speed optimization in supply vessel operations. *Transportation Research Part D: Transport and Environment*, 23:105–113.

Norlund, E. K., Gribkovskaia, I., and Laporte, G. (2015). Supply vessel planning under cost, environment and robustness considerations. *Omega*, 57:271–281.

Pantuso, G., Fagerholt, K., and Hvattum, L. M. (2014). A survey on maritime fleet size and mix problems. *European Journal of Operational Research*, 235(2):341–349.

Shyshou, A., Gribkovskaia, I., Laporte, G., and 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., and Rei, W. (2012). A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, 60(3):611–624.

Vidal, T., Crainic, T. G., Gendreau, M., and Prins, C. (2014). A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operational Research*, 234(3):658–673.

Vidal, T., Crainic, T. G., Gendreau, M., and Prins, C. (2015). Time-window relaxations in vehicle routing heuristics. *Journal of Heuristics*, 21(3):329–358.