A Hybrid GRASP-VNS for Ship Routing and Scheduling Problem with Discretized Time Windows

Jesica de Armas[a], Eduardo Lalla-Ruiz[a], Christopher Expósito-Izquierdo[a], Dario Landa-Silva[b], Belén Melián-Batista[a]

[a] *University of La Laguna, Dept. of Computer and Systems Engineering, 38271, Spain*
[b] *School of Computer Science, The University of Nottingham, UK*

Abstract

This paper addresses the Ship Routing and Scheduling Problem with Discretized Time Windows. Being one of the most relevant and challenging problems faced by decision makers from shipping companies, this tramp shipping problem lies in determining the set of contracts that should be served by each ship and the time windows that ships should use to serve each contract, with the aim of minimizing total costs. The use of discretized time windows allows for the consideration of a broad variety of features and practical constraints in a simple way. In order to solve this problem we propose a hybridazation of a Greedy Randomized Adaptive Search Procedure and a Variable Neighborhood Search, which improves previous heuristics results found in literature and requires very short computational time. Moreover, this algorithm is able to achieve the optimal results for many instances, demonstrating its good performance.

Keywords: Ship Routing and Scheduling Problem, Tramp Shipping, GRASP, Variable Neighbourhood Search

1. Introduction

The most important mode of transport for international trade is seaborne shipping. An estimated 80 per cent of world trade is carried by sea [43]. It means that, compared to other modes of freight transportation, ships are far superior for moving large volumes over long distances. Due to the increasing

Email addresses: jdearmas@ull.es (Jesica de Armas), elalla@ull.es (Eduardo Lalla-Ruiz), cexposit@ull.es (Christopher Expósito-Izquierdo), dario.landasilva@nottingham.ac.uk (Dario Landa-Silva), mbmelian@ull.es (Belén Melián-Batista)

development of economies and globalization, the international trade is continuously rising, and as a consequent the sector of maritime transport has also shown an enormous growth.

Container ships require a huge capital investment and very high daily operating costs. Investment in a ship may range in the millions and operating costs in the thousands dollars a day. There are three main types of costs in seaborne shipping: capital and depreciation, running, and operating costs. Capital and depreciation costs are related to the loss of ships market value respect to the initial investment. Running costs are usually fixed and include maintenance, insurance, crew salaries, and overhead costs, among others. Operating costs are directly related to ships daily operations, and include fuel consumption, port and customs expenses, tolls at canals, etc. These latter costs depend on characteristics like travel distance, navigation speed, and maritime routes. Therefore, capital and depreciation costs, and running costs are not usually expenses that can be subject to optimization as a result of improvements in routing, but the operating costs can be optimized through better routing as it is the aim in this work. Accordingly, good scheduling is of economical essence in this increasingly competitive area.

In this regard, Gatica and Miranda [15] focus on optimally solving a ship routing and scheduling problem with a heterogeneous tramp fleet. They propose a network-based model in which discretized time windows for picking and delivering cargoes are defined. Discretized time windows are just time instants in which these picking and delivering cargoes can be carried out. This allows to consider a broad variety of features and practical constraints by simply adding/deleting arcs or modifying the corresponding cost parameters, which has the advantage of preserving the network structure. In particular, they consider problems in which navigation speed can be used to control fuel consumption, which may have a significant impact in the quality of the solution, since fuel consumption follows an approximately cubic function of speed [37]. They solved the model by means of the general-purpose solver, CPLEX[1]. Numerical results show that the model presents a much better trade-off between solution quality and computing time than a similar constant-speed continuous model. Recently, in order to obtain quality results for real-life-sized problems in less computational time, Castillo-Villar et al. [7] develop a Variable Neighborhood Search (VNS) algorithm to solve this

[1]http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/

problem. This VNS is very simple, since it is just a list of neighborhoods sequentially explored. Results reported in that work show that the VNS provides solutions with a gap between 6% and 7% to the optimal solutions.

Due to the fact that this is an operational problem (*i.e.*, it has to be solved daily) and may be integrated with other related problems (Berth Allocation Problem [4], Container Stowage Problem [3], etc.), it is important to solve it quickly and provide results of the highest possible quality. Motivated by that, this work presents a hybridization of a Greedy Randomized Adaptive Search Procedure (GRASP) with a more complex and elaborate VNS to solve the same problem with the aim of reducing the gap and obtaining results in less computational time.

The remainder of this article is organized as follows. At first, Section 2 reviews research efforts from earlier studies that are related to the current study. Then, in Section 3 the formulation of the problem is shortly outlined. In Section 4 the main features of the hybrid GRASP-VNS method is described. Experimental tests are performed and results are discussed in Section 5. Finally, in Section 6 the main conclusions extracted in this work and some future work lines are summarized.

2. Literature Review

In the literature, the first works about ship routing arose in the 70s [1, 2], but the first survey appeared more than ten years later [36]. Recent reviews on ship routing problems can be found in works by Christiansen et al. [8, 9], where literature contributions are classified.

The majority of papers about ship routing and scheduling problems focus on the development of Mixed Integer Programming (MIP) models or heuristics/metaheuristics methods to solve them. Fox and Herden [13] describe a MIP model to schedule ships from ammonia processing plants to eight ports in Australia. The objective is to minimize freight, discharge, and inventory holding cots while taking into account the inventory, minimum discharge tonnage, and ship capacity constraints. Moreover, using a MIP model, an inventory routing problem with multiple products was analyzed by Ronen [38] for liquid bulk oil cargo. Sherali et al. [40] present an aggregate MIP model for the problem of transporting refined-oil products from three ports in Kuwait to customers located in Europe, North America, and Japan. The model considers the existence of alternative maritime routes. The authors develop a reformulation of the MIP model and a set of valid inequalities that allow them to design several algorithmic solution strategies.

In relation to heuristics and metaheuristics, Korsvik et al. [20] use a Tabu Search algorithm which allows infeasible solutions with respect to ship capacity and time. Other works in the literature introduce specific concepts in ship routing problems. Romero et al. [35] propose a GRASP and discuss aspects related to data gathering and updating, which are particularly difficult in the context of ship routing. Lin and Liu [22] combine ship allocation, routing and freight assignment in a particular kind of ship routing. Kosmas and Vlachos [21] consider a cost function that depends on the wind speed and its direction, as well as on the wave height and its direction, and solve the problem using a Simulated Annealing algorithm.

Moreover, in the related literature, depending on the operation mode, three kinds of ship routing problems can be distinguished: liner, industrial, and tramp shipping. Liners [19, 25, 42] operate according to an agreed itinerary and schedule similar to a bus line. In industrial shipping, the cargo owner or shipper controls the ships. Industrial operators strive to minimize the costs of shipping their cargoes. Tramp fleets engage in contracts to transport specified (usually large) volumes of cargo between two ports within a period of time. They engage in contracts to make one or several trips, each trip having specified origin and destination ports and time windows for picking and delivering the cargo. Tramp is usually the operation mode selected to transport liquid and dry commodities, or cargo involving a large number of units. During the last few decades there has been a shift from industrial to tramp shipping [8, 9].

Particularly, the literature on tramp shipping problems is quite sparse and only a few papers tackle such problems. The reason for this lack of research interest in this shipping sector is attributed to the historic existence of a large number of small tramp shipping companies operating in the market. However, more recently, increased demand and the tendency of larger companies to outsource shipping of their cargoes has led to the growth of small companies, the growth of the associated scheduling problems, and a corresponding increase interest from researchers in this type of problems. A tramp routing and scheduling problem was solved by Brønmo et al. [5], where a multistart Local Search heuristic is developed. The proposed unified Tabu Search heuristic by Korsvik et al. [20] also solves the specific tramp shipping. In contrast to the procedure followed by Brønmo et al. [5], Malliappi et al. [23] present a VNS heuristic, and the results show that this procedure outperforms the previous heuristics. Norstad et al. [29] address this problem considering speed optimization and develop a multistart Local Search heuristic to solve it

Additionally, as introduced above, Gatica and Miranda [15] develop a network-based model for the Ship Routing and Scheduling Problem with Discretized Time Windows with a heterogeneous tramp fleet. The objective is to minimize the total operating cost of serving a set of trip cargo contracts, considering time window constraints at both the origin and destination of cargoes. A distinctive aspect of their methodology is that time windows for picking and delivering cargoes are discretized. This leaves room for including a broad variety of features and practical constraints, such as navigation speed to control fuel consumption. More specifically, they assume that pick-up and delivery may start only at a finite set of time instances within the corresponding time windows. In general (*i.e.* urban) vehicle routing, the only known application of time discretization is for modelling time-dependent travel times (time to traverse an arc depends on the time instance the travel starts). That approach is followed, for example, by Ichoua et al. [18], Woensel et al. [44], and Donati et al. [10]. Gatica and Miranda [15] demonstrate that numerical results considering discretized time windows presents a much better trade-off between solution quality and computational time than a similar constant speed continuous model. Recently, Castillo-Villar et al. [7] developed a VNS algorithm to solve this specific tramp shipping problem with discretized time windows, obtaining quality results in less computational time than the necessary for the initial MIP model implemented in CPLEX.

The main contribution of this paper is to propose a hybrid GRASP-VNS algorithm that improves upon the results from Castillo-Villar et al. [7]. We have developed a more elaborated VNS with a more complex structure for better exploration of the search space, which jointly with the proposed GRASP as start method, allows to obtain a better performance. The improvement is on achieving smaller gap values than those reported by Castillo-Villar et al. [7]. Our technique achieves results of higher quality in short computation times. In this particular problem, the quality of results is very important, since, as stated before, minimizing operating cost is of high relevance in the competitive area of ship routing. Moreover, the faster the results are obtained, the more agility will have the rest of processes that depend on the pickup or delivery of ships cargoes, so that it is another challenge to achieve.

3. Ship Routing and Scheduling Problem with Discretized Time Windows

This section presents the description of the Ship Routing and Scheduling Problem with Discretized Time Windows (hereinafter SRSPDTW). Firstly, Section 3.1 introduces the details of the discretized modelling approach and the characteristics of the problem. Secondly, Section 3.2 presents the mathematical model proposed by Gatica and Miranda [15] and used by Castillo-Villar et al. [7], which is considered in this paper.

3.1. Problem Description

We consider the routing and scheduling problem for tramp shipping which is composed of: (i) a fleet of ships; (ii) a set of cargo contracts that need to be served; (iii) a set of time instants or discretized time windows at which each contract can be served; and (iv) a set of links or arcs between time instants of different contracts for each ship. Each of these arcs represents a ship serving a contract at a time instant and then serving another contract at a different time instant, all this with an associated cost. Each contract is a single trip from one port to another, picking-up and delivering a cargo. A contract must be served at one of the possible time instants that are also called nodes. Therefore, the problem here presented consists of deciding the set of contracts to be served by each ship and the chosen time instants, *i.e.* selecting a set of arcs, with the aim of serving all contracts while minimizing total relevant costs.

The fleet of ships is heterogeneous due to differences in capacity, speeds, fuel consumption, etc. Although each ship can serve only one contract at a time, there are incompatibilities between cargoes and ships or between ships and ports, so that not all ships can serve all contracts. Furthermore, two contracts may be incompatible with each other, *i.e.*, the corresponding trips cannot be done consecutively by the same ship, unless a time delay (*e.g.* for cleaning) or a third trip is placed between them.

It is important to notice that given a sequence of contracts to be served by a single ship, an empty trip must take place from the delivery port of each contract to the origin of the next contract in the sequence, unless these two ports coincide. These empty trips represent a significant portion of total avoidable cost and they are taken into account in this problem. Relevant costs are mainly operating costs, but may also include other kinds of costs as long as they can be associated with individual trips.

One of the most important costs corresponds to the fuel consumption expenses. Since fuel consumption depends on navigation speed, controlling the speed impacts not only on the travel time, but also on the travel costs. In this work, a network-based model is used, and it allows for the consideration of navigation speed and a broad variety of features and practical constraints by simply adding/deleting arcs between contracts or modifying the corresponding cost parameters, which has the advantage of preserving the network structure. This flexibility arises from the discretization of the time windows, which allows for both, the feasibility (existence) and the cost parameter of each potential arc, to be determined outside the model.

3.2. Mathematical Formulation

As stated above, the discretized modeling approach used in Gatica and Miranda [15] and Castillo-Villar et al. [7] has been adopted. In order to make this work self-contained, the model is explained below.

The SRSPDTW can be defined as follows. Let $G = (V, A)$ be a directed graph, where V is the node set and A is the arc set. Each node $i \in V$ represent a time instant and the contract associated with that node is represented by $n(i)$. On the other hand, each contract $n(i)$ has a set of associate nodes $D_{n(i)}$, *i.e.*, the set of possible starting times for trip $n(i)$. In SRSPDTW, the ships are indexed by means of $k = 1, 2, \ldots, B$, where B is the number of available ships. Each $arc(i, j, k)$ represents the service of contracts $n(i)$ and $n(j)$ consecutively by ship k. The arc is included in the network if both, the trips and the ship, are compatible, and if it is feasible for ship k to begin service of contract $n(i)$ at the time instance represented by node i, make the empty trip from the destination port of contract $n(i)$ to the origin of contract $n(j)$, and be available to begin service of contract $n(j)$ at the time instance associated with node j.

For each arc, the cost parameter c_{ijk} represents the total minimal cost when the ship delivers contract $n(i)$ immediately followed by contract $n(j)$. To complete the network, a fictitious node 0 is created to represent the source and destination of all ships (ports that can be different). For each ship k and node i, if contract $n(i)$ is compatible with ship k, both an $arc(0, i, k)$ and an $arc(i, 0, k)$ also exist. Cost c_{0ik} is calculated based on the real initial position of ship k, and cost c_{i0k} represents the cost incurred if ship k serves contract $n(i)$ and must go to a final destination port.

The mathematical formulation of the problem from Gatica and Miranda [15] is as follows:

$$minimize \sum_{(i,j,k)\in A} c_{ijk} \cdot x_{ijk} \quad (1)$$

$$s.t.: \sum_{i\in V/(0,i,k)\in A} x_{0ik} \leq 1 \quad k = 1, 2, \dots, B \quad (2)$$

$$\sum_{(i,j,k)\in A/j\in D_n} x_{ijk} = 1 \quad n = 1, 2, \dots, N \quad (3)$$

$$\sum_{i\in V/(i,j,k)\in A} x_{ijk} = \sum_{l\in V/(j,l,k)\in A} x_{jlk} \quad j \in V, \quad k = 1, 2, \dots, B \quad (4)$$

$$x_{ijk} \in \{0, 1\} \quad (i, j, k) \in A \quad (5)$$

where N is the number of contracts to be served, V is the set of nodes in the network, D_n is the set of nodes associated with contract n (i.e., set of possible starting times for trip n), A is the set of arcs in the network, c_{ijk} is the cost of $arc(i, j, k)$, and:

$$x_{ijk} = \begin{cases} 1 & \text{if } arc(i,j,k) \text{ is part of the solution} \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

Selecting $arc(i, j, k)$ as part of the solution ($x_{ijk} = 1$) implies that ship k will serve contract $n(i)$ and will serve contract $n(j)$ immediately afterwards. Selecting $arc(0, i, k)$ implies that $n(i)$ is the first contract to be served by ship k, and selecting $arc(i, 0, k)$ implies that $n(i)$ is the last contract to be served by ship k.

The objective function (1) represents the total solution cost. Constraints (2) ensure that each ship is employed in at most one route. A route is defined as a sequence of contracts to be served. Constraints (3) ensure that, for each contract n, exactly one arc entering set D_n is selected, establishing that each contract must be served exactly once, by exactly one ship, which begins service at exactly one of the nodes or time instants in the discretized time window for cargo pick up. For nodes different to the central fictitious node, constraints (4) state that if an entering arc is selected, a leaving arc must also be selected and that both arcs must be associated with the same ship. For the fictitious node, these constraints (4) state that if a leaving arc associated with ship k is selected, then an entering arc associated with the same ship must also be selected (i.e., if a ship exits the node), and then it

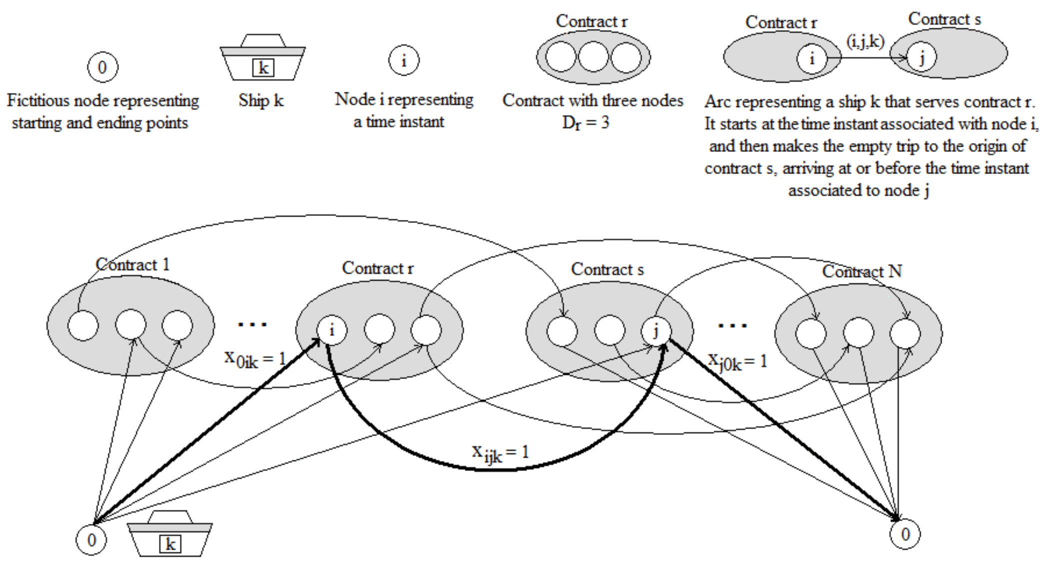

Figure 1: Example of a partial graph of the model for a ship k

must return to it. Arcs leaving the fictitious node represent the ships that are, in fact, used in the solution.

Figure 1 shows an illustrative partial graph of the model, where the nodes of the network represent discrete and feasible starting times for each contract. The ovals group all nodes related to the same contract. Notation is on the top of the figure. As stated before, all ships are supposed to start and end at a fictitious node 0. Some arcs for a single ship k are drawn, based on the feasible trips that can be selected. The chosen route is marked with bold lines. If ship k can serve contract r at time instant of node i, the $arc(0, i, k)$ will be selected and x_{0ik} will be equal to 1. Then, if the ship k can serve contract r and go to serve contract s at time instant of node j, the $arc(i, j, k)$ will be selected and x_{ijk} will be equal to 1. Finally, if ship k does not serve more contracts, it is supposed to go to the fictitious node 0, so that $arc(j, 0, k)$ is used and x_{j0k} will be equal to 1.

4. Hybrid GRASP-VNS Methodology

On the one hand, Greedy Randomized Adaptive Search Procedure (GRASP) is a metaheuristic algorithm commonly applied to combinatorial optimization problems, and consists of iterations made up from successive constructions of a greedy randomized solution and subsequent iterative improvements of it. The greedy randomized solutions are generated by adding elements to the problem solution set from a list of elements ranked by a greedy function according to the quality of the solution they will achieve. To obtain variability in the candidate set of greedy solutions, well-ranked candidate elements are often placed in a Restricted Candidate List (also known as RCL), and chosen at random when building up the solution. GRASP was first introduced in Feo and Resende [12], and some survey papers are Feo and Resende [11] and Resende and Ribeiro [34].

On the other hand, VNS, proposed by Mladenović and Hansen [26], is another metaheuristic for solving combinatorial optimization problems. VNS systematically changes different neighborhoods within a local search, unlike many metaheuristics where only a single neighborhood is employed. The basic idea is that a local optimum defined by one neighborhood structure is not necessary the local optimum of another neighborhood structure, thus the search can systematically traverse different search spaces which are defined by different neighborhood structures. This makes the search much more flexible within the solution space of the problem, and potentially leads to better solutions which are difficult to obtain by using single-neighborhood-based local search algorithms. Many extensions of VNS have been made, mainly to be able to solve large problem instances [16, 17, 24, 28, 30].

This paper proposes the use of a hybrid GRASP-VNS algorithm providing a solution to the SRSPDTW. Results are obtained in less computational time than previous approaches [7, 15], and solutions are of high quality, as it is shown in Section 5. Both aspects are specially important when dealing with large-scale instances. The proposed hybrid algorithm incorporates two powerful features, the effective constructive and improving ability of GRASP and the flexibility of VNS to explore different search spaces for the problem.

It is important to notice that a solution to the problem consists of a route for each ship, so that a route is defined as the set of contracts to be performed by the corresponding ship as well as the chosen discretized time windows. The general algorithm (Algorithm 1) proposed in order to obtain these kind of solutions is based on the repetition (L times) of two main steps: the construction of an initial feasible solution using a GRASP, and the improving of this solution applying a VNS algorithm.

Algorithm 1: General Algorithm

// *Initialization*.
Initialize $BestSol \leftarrow \emptyset$.
while *(the stopping condition is not met (L is not reached))* **do**
  Generate an initial solution s using GRASP algorithm.
  // *VNS*.
  while *(the stopping condition is not met (M is not reached))* **do**
    (1) Set $k \leftarrow 1$;
    (2) Repeat the following steps (a), (b), and (c) until $k = k_{max}$:
      (a) *Shaking*. Generate a point s' at random from the k^{th} neighborhood of s:
      (b)*Local Search*.
        while *(improvement is achieved)* **do**
          $s'' \leftarrow$ swapInter(s');
          $s'' \leftarrow$ improveRoutes(s'');
        while *(improvement is achieved)* **do**
          $s'' \leftarrow$ relocation(s');
          $s'' \leftarrow$ improveRoutes(s'');
        while *(improvement is achieved)* **do**
          $s'' \leftarrow$ 2-opt(s');
          $s'' \leftarrow$ improveRoutes(s'');
        while *(improvement is achieved)* **do**
          $s'' \leftarrow$ relocationChanging(s');
          $s'' \leftarrow$ improveRoutes(s'');
      (c) *Move or not*. If this local optimum is better than the incumbent, move there ($s \leftarrow s''$), and continue the search ($k \leftarrow 1$); otherwise, set $k \leftarrow k + 1$.
  Update $BestSol$.

4.1. GRASP for an Initial Feasible Solution

In order to obtain an initial solution, a GRASP has been developed. This algorithm operates as follows. Firstly, a list composed of ships, contracts, and costs is created, as shown in Figure 2. If the problem is composed of B ships, the first B contracts are assigned to each ship with their corresponding costs, as long as the ships can go to perform the contracts. This cost is the least possible cost so that each ship performs the contract using a time node, and taking into account that ships are supposed to be in fictitious node 0 at the beginning. The list is sorted in non-increasing order of cost,

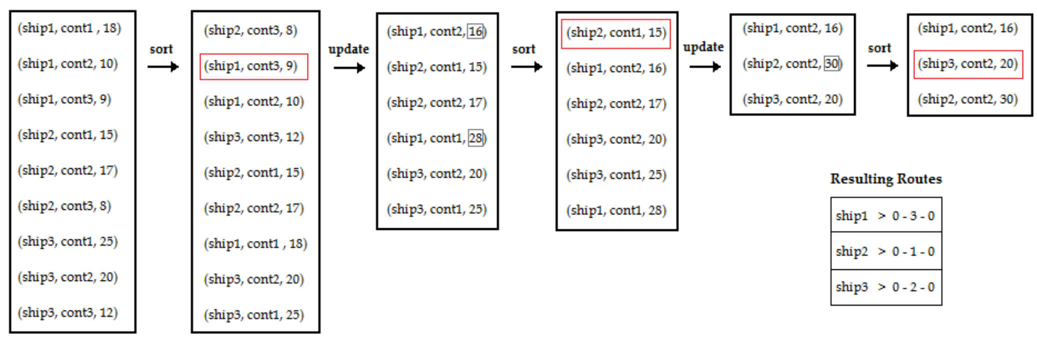

Figure 2: GRASP behaviour example

and the chosen element is randomly selected from the RCL. The RCL is formed by the first three elements of the sorted list, parameter that has been adjusted in order to obtain variable quality solutions. Once the element is chosen, the corresponding contract is assigned to the ship. Then, the element is deleted from the list, as every other element that contains the selected contract. Moreover, for every element containing the selected ship, the cost is updated taking into account that the ship is previously performing the selected contract. This process is repeated until the list is empty, point at which a new list is created with the following B contracts. When there are no more contracts, the process finishes.

At this point, we have the route that each ship will perform, but it could happen that some contracts have not been assigned due to arc restrictions. In that case, a new process starts, trying to insert these contracts at some point within the routes. If that is not possible, *swapInter* and *relocation* movements (which are explained in next section) are tried repeatedly seeking to achieve the new contract insertion. After carrying out a certain number of iterations of these movements without achieving the insertion, the process is suspended, as it can happen that it is not possible to obtain an initial feasible solution.

Figure 2 shows an example where the problem is composed of three ships and three contracts. The first three contracts are assigned to each ship supposing that they are at the fictitious node at the beginning. Then, the list is sorted by cost and one of the elements of the RCL is selected. This selection is depicted in the figure using a framing red rectangle The list is updated deleting the elements with the selected contract and changing the cost corresponding to the selected ship, since now it is performing the selected contract. This process continues until the list is empty, so that the first three contracts are just in a ship route. Similarly, the next three contracts will be assigned to the three ships repeating the process until there are no more contracts.

4.2. Variable Neighborhood Seach Algorithm

As shown in Algorithm 1, unlike the VNS composed of a list of neighborhoods sequentially explored in Castillo-Villar et al. [7], the VNS algorithm applied in this work is composed of three phases: shaking, local search, and move decision. At the beginning, for M times, variable k is set to 1 (line 5), and then the iteration of the three phases starts.

In the shaking phase a solution is randomly generated applying the corresponding neighborhood structure, *i.e.*, the k^{th} neighborhood structure (line 7), with k_{max} representing the total number of neighborhood structures. The sequence of neighborhood structures has been chosen following the ideas described by Repoussis et al. [32] which provided high quality results for a vehicle routing problem that presents many similarities with our problem. The sequence is defined as follows: $CROSS$, $2-opt^*$, $relocation$, $relocationChanging$, and $swapInter$. This sequential selection is applied based on cardinality, which implies moving from relatively poor to richer neighborhood structures, and significantly increases the possibilities of finding higher quality solutions. The neighborhood structures $GENI$ and $Or-opt$ used by Repoussis et al. [32] have been discarded because they are not applicable to this particular kind of routes. On the other hand, the $relocationChanging$ structure, similar to $relocation$, has been added to the sequence. Each operator works randomly, so that the corresponding operator in each iteration of the VNS is performed a limited number of times in order to try obtaining a feasible solution. If it is not possible, the VNS proceeds to next iteration increasing k. The way they work is the following:

- The $CROSS$ operator [41] selects a subsequence of contracts from a route, other subsequence of contracts from other route, and exchanges both subsequences ($O(P^2n^2)$ being P the maximum length of the subsequences).
- The $2-opt^*$ operator [31] chooses two routes and exchange the last part of both routes after two selected point, one from each route ($O(n^2)$).

- The *relocation* operator [6] deletes a contract from a route and inserts it into another route ($\mathrm{O}(n^2)$).

- The *relocationChanging* operator is a modification of the *relocation* one, where the nodes from contracts between the new one is going to be inserted can change to another node belonging to these contracts, in order to accommodate the new one ($\mathrm{O}(n^2)$).

- The *swapInter* operator selects a contract from a route, other contract from other route, and swaps them ($\mathrm{O}(n^2)$).

In the local search phase (lines 8-20), four different neighborhood structures are sequentially applied at each iteration: $swapInter$, $relocation$, $2-opt^*$, and $relocationChanging$. These structures are similar to the ones applied in the shaking phase, but instead of working randomly, they search the movement that involves the highest reduction of cost, *i.e.*, the best solution of the neighborhood. This way, each structure is applied until no improvement is achieved (lines 9-20). An improvement method is always applied after performing a neighborhood movement. This method explores all feasible combinations of arcs that connect two contracts in the route of a ship, selecting the pair of arcs with lowest cost. It means that this method tries to find an improvement of the solutions based on an analysis of the time windows of each contract, respecting the contracts already assigned to the route of a ship.

The order of neighborhood structures exploration in the local search phase has been established by means of the following study. Firstly, each structure has been individually applied in the local search phase of the VNS, in order to asses its contribution during the search process. A representative subset of instances has been used in this analysis,

A subset of representative instances - one instance of each group of 15 instances explained in Section 5 - has been used in this analysis. In Figure 3 the neighborhood structures $swapInter$, $relocation$, $2-Opt^*$, and $relocationChanging$ are identified by $N1$, $N2$, $N3$, and $N4$, respectively. The first graph shows that the $N4$ provides the lowest average value of the minimization objective function when used individually. However, applying this neighborhood structure is computationally expensive, so that obtaining results involves large times. For this reason, using $N4$ as first or second neighborhood structure has been discarded. Thereupon, secondly, each combination of two structures without $N4$ has been applied as shown in the

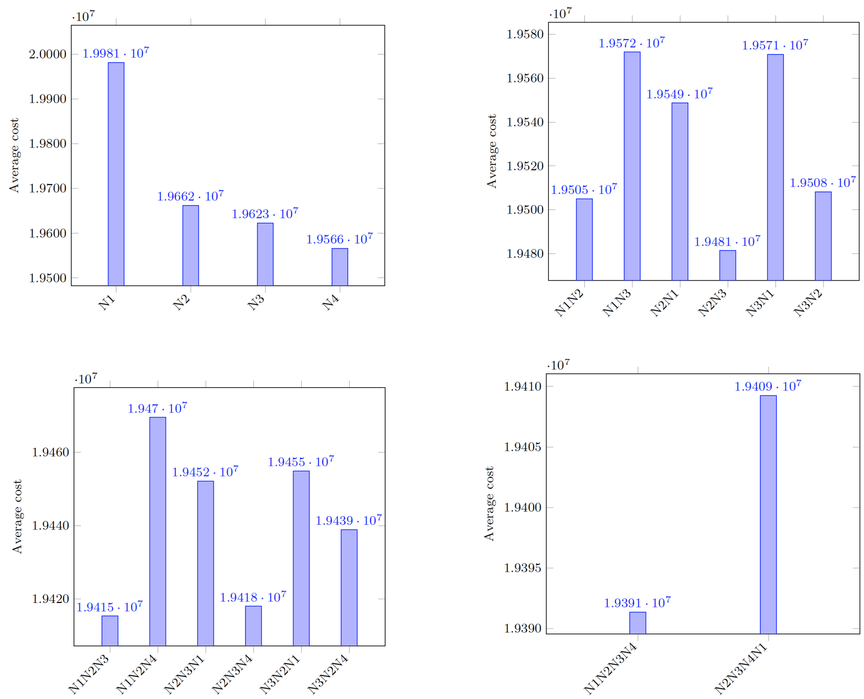

Figure 3: Evaluation of the combination of neighborhood structures

second graph, and the three combinations which involve better results have been selected ($N1N2$, $N2N3$, $N3N2$). Then, each combination of three structures starting from these better ones has been applied as shown in the third graph, and the two combinations which involve better results have been selected ($N1N2N3$, $N2N3N4$). As last step, every combination of four structures starting from these better ones has been applied as shown in the fourth graph, and the best one has been selected ($N1N2N3N4$).

Finally, in the move decision phase (line 21) the new solution is compared to the initial one, and if the new one is better, then the solution is updated and the search starts again setting k to 1. Otherwise, k is increasing by 1 and the next neighborhood in the shaking phase is used.

5. Computational Experiments

This section is devoted to analyze the performance of the hybrid GRASP-VNS algorithm introduced in Section 4 for solving the SRSPDTW. Results produced by the proposed algorithm have been compared to exact solutions and previous results reported in the literature [7]. For more clarity, hereinafter the whole heuristic algorithm proposed by Castillo-Villar et al. [7] is referenced as CVH, and its greedy start method is referenced as Greedy. Our algorithm has been implemented using Java Standard Edition 7 and computational experiments have been performed using a 3.00 GHz Intel Core i-5 processor with 6 GB of RAM running under Ubuntu 12.10.

The set of instances used in this work is the same set generated by Castillo-Villar et al. [7]. There are eighteen groups of instances, each on considering a different combination of ships, time window nodes, and contracts. The set of discrete time windows (*i.e.* number of possible starting times for each contract) consists of 3, 6, or 15 nodes. The number of ships varies over the values of 4, 5, 7, and 9. The number of contracts varies over the values of 30, 40, and 50. Each group contains 15 different instances. In total, the benchmark is composed of $3 \cdot 4 \cdot 3 \cdot 15 = 540$ instances.

In order to obtain the best results using the proposed GRASP-VNS algorithm (Algorithm 1), a parameter setting experimental study has been conducted. Applying the Friedman test [14], M has been fixed to 10, L to 10, and k_{max} to 5. Moreover, the maximum number of times that each neighborhood structure in the shaking phase is tried until a feasible movement is obtained has been fixed to 30. This is because not all movements corresponding to a neighborhood are feasible due to time windows.

With the aim of demonstrating not only the benefits of our whole proposal, GRASP-VNS, but also the benefits of both sides, GRASP and VNS, firstly, 20 executions have been made for each instance using our VNS together with the Greedy, based on prioritizing the earliest due date contracts and seeking to assign each contract at the earliest possible instant to a ship. This combination is referenced as Greedy-VNS. The average results have been compared to the ones provided by Castillo-Villar et al. [7], where stopping rules consider a limit time of 7,200 seconds for CPLEX (sometimes its solution does not correspond to an optimal solution) and a maximum number of iterations without improvement in the solution for their whole heuristic method CVH. These results, produced by CPLEX and the heuristic method, have been kindly provided by the authors. This way, it is possible to compare the performance of our VNS algorithm with the performance of the VNS by Castillo-Villar et al. [7]. Secondly, we have made 20 executions using our hybrid GRASP-VNS proposal (see Section 4) in order to show the improvements provided by our VNS, the improvements provided by the GRASP regarding the Greedy, and the general improvements of the hybrid GRASP-VNS regarding the CVH. We have use the same formula than Castillo-Villar et al. (2014) to calculate the gap values:

$$gap = \frac{Z - CPLEX}{CPLEX} \cdot 100 \quad (7)$$

where Z corresponds to the value obtained by the corresponding heuristic. Positive gaps are obtained when CPLEX finds better solutions.

Tables 1, 2, and 3 show a summary of all results obtained for these instances with 30, 40, and 50 contracts, respectively. Each instance type, consisting of 15 instances, is indicated according to its combination of contracts ($Cn.$), ships ($Sh.$), and nodes ($Nd.$) in columns 1, 2, and 3. The next four columns are related to solutions obtained using CPLEX. Column 4 is the the number of instances from which an optimal solution is found ($Opt.\ found$). Column 5 is the number of instances for which an optimal solution is not found, but a feasible solution is found ($Only\ feas.\ sol.$). Column 6 is the number of instances for which no solution is found ($Sol.\ not\ found$). Finally, column 7 is the average time spent to obtain a solution ($Avg.\ time(s.)$). It is important to note that sometimes CPLEX is not able to achieve the optimal solution within the limit time, but it might provide a feasible one or not at all. Thus, the value of this column corresponds to the average of the time needed to obtain the optimal or a feasible solution for the 15 instances of each group, so that if there is not solution for an instance this instance is not taken into account to calculate the average. This last consideration is always keeping in mind to calculate the average values in this work.

The next three columns present the CVH results: the number of instances for which solution is not found ($Sol.\ not\ found$), the average time needed by the algorithm to provide a solution taking into account that it is executed considering a maximum number of iterations without any improvement in the solution as stopping criterion ($Avg.\ time(s.)$), and the gap between CPLEX and CVH objective function values ($Gap_1(\%)$). Results of Greedy together with our VNS algorithm, *i.e.* Greedy-VNS, are shown in next five columns: the number of instances for which feasible solution is not found by the algorithm ($Sol.\ not\ found$), the average number of executions (of the 20 executions made for each instance) for which an optimal objective value is reached ($Avg.\ opt\ found$), the average time needed to provide a solution considering that the algorithm finishes when loops finish, and the loops are controlled by fix parameters L and M ($Avg.\ time(s.)$), the gap between

CPLEX and Greedy-VNS objective function values ($Gap_2(\%)$), and the gap between CVH and Greedy-VNS objective function values ($Gap_3(\%)$). Finally, results for the GRASP-VNS algorithm proposed here are shown in the six right-most columns. Column $Sol.\ not\ found$ gives the number of instances for which a solution is not found by the algorithm. Column $Avg.\ opt\ found$ shows the average number of executions for which the optimal objective value is reached. Column $Avg.\ time(s.)$ shows the average time spent by the algorithm to obtain solutions. Columns $Gap_4(\%)$, $Gap_5(\%)$, and $Gap_6(\%)$ present the gap between CPLEX and GRASP-VNS objective function values, the gap between the CVH and GRASP-VNS objective function values, and the gap between Greedy-VNS and GRASP-VNS objective function values, respectively.

Table 1 shows results for instances of 30 contracts corresponding to the smallest size instances. In terms of the quality of solutions, using our VNS instead of the VNS by Castillo-Villar et al. [7], *i.e.* the Greedy-VNS algorithm, the gap with regard to CPLEX solutions is considerably reduced from 5.17 to 0.27% on average, and with regard to CVH, solutions are improved on an average of 4.63%. This behavior is repeated with the larger instances as shown in next two tables. This way, we have demonstrated the better performance of our VNS. Additionally, if GRASP replaces Greedy obtaining the GRASP-VNS proposal of this paper, then the results are even better, so that the gap goes from 0.27 to 0.18% regarding CPLEX, and from -4.63 to -4.70% regarding CVH. Once again, this behavior is repeated with the larger instances as shown in next two tables. Although the improvement introduced by GRASP could seem not very high, an important advantage of using it is that this is able to find more feasible solutions than the other proposals, as shows column 15 ($Sol.\ not\ found$) of each table, and another remarkable aspect is that more than half of the times (11.46 out of 20) that the GRASP-VNS is executed using these instances, an optimal solution is found, demonstrating the robustness of the algorithm. This ratio decreases when the number of contracts increases due to instances complexity, as can be seen in the following tables. However, the approach provided by Castillo-Villar et al. [7] did not produce optimal solutions according to their paper. In regard to computation time, results of GRASP-VNS are far better than CPLEX and CVH. It is noteworthy that although execution machines are different, the magnitude of values is not only due to the difference between machines, but also because of the efficiency of the proposed algorithm.

Instances of 40 contracts are medium size instances and results for them are shown in Table 2. For this set of instances CPLEX and CVH need between 4 and 8 minutes on average to obtain solutions, whereas GRASP-VNS requires about 20 seconds on average. The average gap value between CPLEX solutions and GRASP-VNS solutions increase a bit due to the magnitude of instances, to a value of 0.46%. However, the gap between CVH and GRASP-VNS is even better than the gap obtained with 30-contract instances (-6.06%). This demonstrates that the performance of CVH worsens when the complexity of instances increase, while this is not the case for the proposed GRASP-VNS algorithm.

Table 3 shows instances of 50 contracts corresponding to the largest size instances. The behaviour of the GRASP-VNS algorithm is very similar to the one corresponding to 40-contract instances. One more time, the average gap values in respect to CPLEX is low, 0.58%, and CVH results are improved on an average of 5.14%. Notice that average computation times for these largest instances are shorter than the average times for 40-contract instances, but it is due to a particular 40-contract instance with 5 ships and 15 nodes that consumes particularly longer computation time.

An important point that can be highlighted from Tables 1, 2, and 3 is that our GRASP-VNS algorithm always finds a solution if CPLEX has found a solution, and even sometimes GRASP-VNS is able to find a solution when CPLEX has not found a feasible one, as can be seen for 50-contract instances. In contrast, CVH always solves less number of instances than CPLEX.

			CPLEX				*CVH*			*Greedy - VNS*					*GRASP - VNS*					
Cn.	Sh.	Nd.	Opt. found	Only feas. sol.	Sol. not found	Avg. time (s.)	Sol. not found	Avg. time (s.)	Gap_1 (%)	Sol. not found	Avg. opt. found	Avg. time (s.)	Gap_2 (%)	Gap_3 (%)	Sol. not found	Avg. opt. found	Avg. time (s.)	Gap_4 (%)	Gap_5 (%)	Gap_6 (%)
30	4	3	13	0	2	1.00	3	4.83	4.79	3	10.58	0.55	0.22	-4.32	2	11.84	0.61	0.12	-4.41	-0.10
30	4	6	13	0	2	4.84	2	15.92	5.36	2	7.38	1.17	0.36	-4.71	2	9.92	1.33	0.21	-4.84	-0.14
30	4	15	13	0	2	171.84	2	93.15	5.53	2	5.46	6.42	0.42	-4.81	2	9.38	7.26	0.22	-4.99	-0.19
30	5	3	15	0	0	1.13	0	7.06	4.34	0	13.00	0.60	0.22	-3.94	0	13.80	0.62	0.18	-3.97	-0.03
30	5	6	15	0	0	12.73	0	26.53	5.49	0	11.33	1.30	0.20	-4.98	0	12.53	1.31	0.17	-4.99	-0.02
30	5	15	15	0	0	140.60	0	151.40	5.49	0	10.33	7.18	0.19	-4.99	0	11.26	7.34	0.20	-4.97	0.01
			14.00	0.00	1.00	55.36	1.17	49.82	5.17	1.17	9.68	2.87	0.27	-4.63	1.00	11.46	3.08	0.18	-4.70	-0.08

Table 1: Summary of results for instances with 30 contracts

			CPLEX				*CVH*			*Greedy - VNS*					*GRASP - VNS*					
Cn.	Sh.	Nd.	Opt. found	Only feas. sol.	Sol. not found	Avg. time (s.)	Sol. not found	Avg. time (s.)	Gap_1 (%)	Sol. not found	Avg. opt. found	Avg. time (s.)	Gap_2 (%)	Gap_3 (%)	Sol. not found	Avg. opt. found	Avg. time (s.)	Gap_4 (%)	Gap_5 (%)	Gap_6 (%)
40	5	3	14	0	1	3.85	6	21.55	5.00	6	5.67	0.82	0.50	-4.24	1	5.53	2.63	0.35	-4.39	-0.16
40	5	6	15	0	0	42.00	3	82.50	8.11	3	4.75	2.09	0.67	-6.76	0	3.13	11.72	0.61	-6.97	-0.23
40	5	15	15	0	0	719.40	3	442.58	7.49	3	1.42	14.08	0.72	-6.20	0	3.53	87.50	0.54	-6.40	-0.21
40	7	3	15	0	0	8.66	0	33.93	6.27	0	1.93	69.03	0.40	-5.47	0	1.73	1.00	0.34	-5.52	-0.05
40	7	6	15	0	0	447.53	0	137.13	7.38	0	2.13	2.61	0.49	-6.37	0	3.20	2.64	0.41	-6.44	-0.08
40	7	15	12	3	0	1870.86	0	645.53	7.76	0	2.07	16.63	0.54	-6.63	0	1.66	16.97	0.49	-6.66	-0.04
			14.33	0.50	0.17	515.38	2.00	227.20	7.00	2.00	2.99	17.54	0.55	-5.95	0.17	3.13	20.41	0.46	-6.06	-0.13

Table 2: Summary of results for instances with 40 contracts

			CPLEX				*CVH*			*Greedy - VNS*					*GRASP - VNS*					
Cn.	Sh.	Nd.	Opt. found	Only feas. sol.	Sol. not found	Avg. time (s.)	Sol. not found	Avg. time (s.)	Gap_1 (%)	Sol. not found	Avg. opt. found	Avg. time (s.)	Gap_2 (%)	Gap_3 (%)	Sol. not found	Avg. opt. found	Avg. time (s.)	Gap_4 (%)	Gap_5 (%)	Gap_6 (%)
50	7	3	14	0	1	18.71	2	109.30	5.95	2	0.69	1.41	0.76	-4.85	1	0.66	1.61	0.58	-5.02	-0.18
50	7	6	13	0	2	183.53	4	385.45	5.35	4	0.00	4.35	0.70	-4.39	1	1.53	4.82	0.57	-4.46	-0.08
50	7	15	7	7	1	4291.85	2	1551.38	5.87	2	0.14	30.67	0.67	-4.87	1	0.36	34.14	0.64	-4.91	-0.04
50	9	3	15	0	0	23.80	0	134.06	6.69	0	1.00	1.60	0.59	-5.68	0	0.73	1.61	0.54	-5.71	-0.04
50	9	6	15	0	0	371.53	0	558.40	6.53	0	0.73	4.70	0.86	-5.30	0	0.86	4.70	0.51	-5.59	-0.31
50	9	15	7	8	0	4450.20	0	2795.53	6.09	0	0.13	32.71	0.94	-4.83	0	0.06	33.09	0.61	-5.13	-0.30
			11.83	2.50	0.67	1556.60	1.33	922.35	6.08	1.33	0.45	12.57	0.75	-4.99	0.50	0.70	13.33	0.58	-5.14	-0.16

Table 3: Summary of results for instances with 50 contracts

			Greedy		*GRASP*	
Cn.	Sh.	Nd.	Gap (%)	Avg. time (s.)	Gap (%)	Avg. time (s.)
30	4	3	44.29	0.10	23.59	0.20
30	4	6	50.56	0.12	27.81	0.31
30	4	15	54.62	0.25	34.75	1.07
30	5	3	54.19	0.10	23.80	0.13
30	5	6	54.56	0.13	26.67	0.19
30	5	15	55.60	0.28	30.57	0.44
			52.30	0.16	27.87	0.39

Table 4: Summary of Greedy and GRASP results for instances with 30 contracts

			Greedy		*GRASP*	
Cn.	Sh.	Nd.	Gap (%)	Avg. time (s.)	Gap (%)	Avg. time (s.)
40	5	3	59.54	0.11	28.20	1.67
40	5	6	66.01	0.17	33.09	7.01
40	5	15	59.35	0.41	40.01	8.03
40	7	3	66.01	0.12	25.84	0.17
40	7	6	58.75	0.19	30.34	0.27
40	7	15	59.35	0.54	34.21	0.70
			61.50	0.26	31.95	2.98

Table 5: Summary of Greedy and GRASP results for instances with 40 contracts

			Greedy		*GRASP*	
Cn.	Sh.	Nd.	Gap (%)	Avg. time (s.)	Gap (%)	Avg. time (s.)
50	7	3	67.63	0.13	31.70	0.36
50	7	6	73.10	0.23	36.97	0.85
50	7	15	75.02	0.76	43.74	4.66
50	9	3	67.76	0.14	27.96	0.20
50	9	6	68.60	0.27	30.28	0.36
50	9	15	69.48	0.97	33.49	1.35
			70.26	0.42	34.02	1.30

Table 6: Summary of Greedy and GRASP results for instances with 50 contracts

In order to better clarify the impact of the GRASP on the solution of the problem, Tables 4, 5 and 6 provide a comparison of gaps - regarding the CPLEX solutions - when applying the Greedy and the GRASP individually. As can be seen, the GRASP always provides much more quality results than the Greedy in very short time. Therefore, it is reflected once again the improvement made by the GRASP.

Gaps/Ships	4	5	7	9
Gap_4	0.18	0.34	0.51	0.55
Gap_5	-4.75	-5.28	-5.50	-5.48

Table 7: Gaps per number of ships

Gaps/Nodes	3	6	15
Gap_4	0.35	0.41	0.45
Gap_5	-4.84	-5.55	-5.51

Table 8: Gaps per number of nodes in contracts

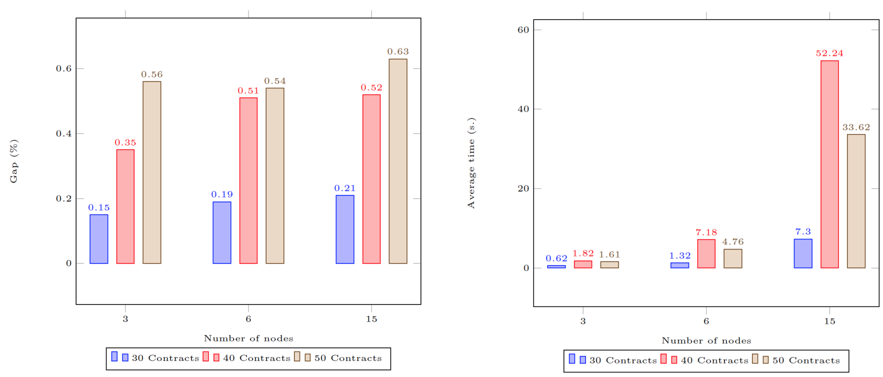

Figure 4: Average time and gaps per number of nodes

From previous tables, it can be deduced that the number of contracts affects the solutions quality, since the problem complexity increases. In order to know how other instances features influence the quality, Tables 7 and 8 report gaps classified according to the number of ships and the number of nodes per contract. As before, $Gap_4(\%)$ and $Gap_5(\%)$ present the gap between CPLEX and GRASP-VNS objective function values, and between the CVH and GRASP-VNS objective function values, respectively. In Table 7, it can be seen that an increment in the number of ships slightly increases the gap between CPLEX and GRASP-VNS solutions due to the rise in instances complexity. Nevertheless, solutions provided by CVH are continuously improved by the GRASP-VNS algorithm, giving an indication that the quality of CVH solutions is clearly much more influenced by the increasing complexity. In Table 8, it can be seen that the increment in the number of nodes also results in an increase of the gap between CPLEX and GRASP-VNS solutions. Moreover, once again, CVH solutions are widely improved by the proposed GRASP-VNS algorithm, as shown by the negative gaps.

With the aim of analysing the behaviour of the GRASP-VNS algorithm when time windows are more discretized, Gap_4 from Table 8 has been split by number of contracts requested in the instances, obtaining the first chart of Figure 4. This way, it can be seen that the highest gap is always presented for 50-contract instances and the lowest gap for the 30-contract instances as expected due to the rise in complexity. A slightly tendency of the gap to increase appears for each number of contracts when the number of nodes increases. The same comparison has been made taking into account time instead of gap. However, in this case it is evident that 40-contract instances present more difficulties to be solved than the other ones, since times are always the highest when solving these instances. Additionally, the sharp increase of time going from 6 to 15 nodes is quite clear, which means that the more discretization is used, the higher will increase the time.

6. Conclusions and Further Research

In this paper, a hybrid GRASP-VNS algorithm for solving a SRSPDTW has been proposed. This problem belongs to the tramp shipping category, which is increasingly present in the field of maritime cargo transport. The objective considered is to minimize the total cost of serving a set of trip cargo contracts, discretized time windows for picking and delivering cargoes. This allows for a broad variety of features and practical constraints, such as navigation speed to control fuel consumption. Moreover, previous works in literature demonstrated that numerical results considering discretized time windows presents a much better trade-off between solution quality and computing time than a similar constant speed continuous model. Even taking into account discretized time windows, using exact algorithm to obtain the optimal results involves large computational times. The hybrid GRASP-VNS algorithm proposed here achieves high-quality solutions in less computational time, and it has been demonstrated that both parts of the algorithm, GRASP and VNS, contribute to this good behaviour.

It is noticeable from the computational experiments that results obtained do not only improve previous approximated solutions in the literature, but they are much closer to the optimal ones, presenting an average gap between 0.18 and 0.58 %. Actually, optimal solutions are obtained for many instances. Additionally, this GRASP-VNS algorithm finds solutions even when CPLEX is not able to find a feasible one in two hours.

On the other hand, an analysis of the proposed hybrid algorithm behaviour was conducted in order to understand how the number of time windows influences the quality of results. It has been shown that the quality of solutions is slightly affected by the level of discretization, so that the more number of nodes, the higher the gap in respect to optimal solutions but still within an acceptable level. However, the computational time shows a sharp increase when the number of nodes goes from 6 up to 15. This means that although the quality of solutions is acceptable, when the number of nodes increases, the computational effort rises quickly. Hence, implementing the right degree of discretization of the problems instances in hand is a key aspect when solving this problem.

On the basis of the contributions presented in this paper, the next stage of the research will be focused on the analysis of how the consideration of the Container Stowage Problem impacts in the selection of contract nodes. The more time containers spend in maritime terminal, the more money should be paid, so this cost should be taken into account. Another open line for future research is applying the proposed hybrid GRASP-VNS algorithm to other tramp shipping or even to other kind of ship routing problems.

7. Acknowledgments

This work has been partially funded by the European Regional Development Fund, the Spanish Ministry of Economy and Competitiveness (project TIN2012-32608) and the Tri-continental Campus of Excellence. Eduardo Lalla-Ruiz and Christopher Expósito-Izquierdo thank the Canary Government the financial support they receive through their doctoral grants.

References

[1] Appelgren, L.H., 1969. A column generation algorithm for a ship scheduling problem. Transportation Science 3, 53–68.

[2] Appelgren, L.H., 1971. Integer programming methods for a vessel scheduling problem. Transportation Science 5, 64–78.

[3] Avriel, M., Penn, M., Shpirer, N., 2000. Container ship stowage problem: complexity and connection to the coloring of circle graphs. Discrete Applied Mathematics 103, 271 – 279.

[4] Bierwirth, C., Meisel, F., 2010. A survey of berth allocation and quay crane scheduling problems in container terminals. European Journal of Operational Research 202, 615–627.

[5] Brønmo, G., Christiansen, M., Fagerholt, K., Nygreen, B., 2007. A multi-start local search heuristic for ship scheduling-a computational study. Computers and Operations Research 34, 900–917.

[6] Cassani, L., Righini, G., 2004. Heuristic algorithms for the tsp with rear-loading, in: 35th Annual Conference of the Italian Operations Research Society (AIRO XXXV), Lecce, Italy.

[7] Castillo-Villar, K., González-Ramírez, R., Miranda González, P., Smith, N., 2014. A heuristic procedure for a ship routing and scheduling problem with variable speed and discretized time windows. Mathematical Problems in Engineering 2014, 24–36.

[8] Christiansen, M., Fagerholt, K., Nygreen, B., Ronen, D., 2013. Ship routing and scheduling in the new millennium. European Journal of Operational Research 228, 467–483.

[9] Christiansen, M., Fagerholt, K., Ronen, D., 2004. Ship routing and scheduling: Status and perspectives. Transportation Science 38, 1–18.

[10] Donati, A.V., Montemanni, R., Casagrande, N., Rizzoli, A.E., Gambardella, L.M., 2008. Time dependent vehicle routing problem with a multi ant colony system. European Journal of Operational Research 185, 1174 – 1191.

[11] Feo, T., Resende, M., 1995. Greedy randomized adaptive search procedures. Journal of Global Optimization 6, 109–133. doi:10.1007/BF01096763.

[12] Feo, T.A., Resende, M.G., 1989. A probabilistic heuristic for a computationally difficult set covering problem. Operations Research Letters 8, 67 – 71.

[13] Fox, M., Herden, D., 1999. Ship scheduling of fertilizer products. ORI 12, 21–28.

[14] Friedman, M., 1937. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. Journal of the American Statistical Association 32, 675–701.

[15] Gatica, R., Miranda, P., 2011. Special issue on latin-american research: A time based discretization approach for ship routing and scheduling with variable speed. Networks and Spatial Economics 11, 465–485.

[16] Hansen, P., Mladenovic, N., Brimberg, J., Moreno Pérez, J., 2010. Handbook of Metaheuristics. Springer. chapter Variable Neighborhood Search. pp. 61–86.

[17] Hoeller, H., Melian, B., Voss, S., 2008. Applying the pilot method to improve VNS and GRASP metaheuristics for the design of SDH/WDM networks. European Journal of Operational Research 191, 691–704.

[18] Ichoua, S., Gendreau, M., Potvin, J.Y., 2003. Vehicle dispatching with time-dependent travel times. European Journal of Operational Research 144, 379 – 396.

[19] Kjeldsen, K., 2011. Classification of ship routing and scheduling problems in liner shipping. INFOR 49, 139–152.

[20] Korsvik, J., Fagerholt, K., Laporte, G., 2010. A tabu search heuristic for ship routing and scheduling. Journal of the Operational Research Society 61, 594–603.

[21] Kosmas, O., Vlachos, D., 2012. Simulated annealing for optimal ship routing. Computers and Operations Research 39, 576–581.

[22] Lin, D.Y., Liu, H.Y., 2011. Combined ship allocation, routing and freight assignment in tramp shipping. Transportation Research Part E: Logistics and Transportation Review 47, 414–431.

[23] Malliappi, F., Bennell, J., Potts, C., 2011. A variable neighborhood search heuristic for tramp ship scheduling. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 6971 LNCS, 273–285.

[24] Melian, B., 2006. Using memory to improve the VNS metaheuristic for the design of SDH/WDM networks, in: Almeida, F and Aguilera, MJB

and Blum, C and Vega, JMM and Perez, MP and Roli, A and Sampels, M (Ed.), Hybrid Metaheuristics, Proceedings, pp. 82–93. 3rd International Workshop on Hybrid Metaheuristics, Gran Canaria, SPAIN, OCT 13-14, 2006.

[25] Meng, Q., Wang, S., Andersson, H., Thun, K., 2014. Containership routing and scheduling in liner shipping: Overview and future research directions. Transportation Science 48, 265–280.

[26] Mladenović, N., Hansen, P., 1997. Variable neighborhood search. Computers & Operations Research 24, 1097–1100.

[27] Moon, I., Qiu, Z., Wang, J., 2014. A combined tramp ship routing, fleet deployment, and network design problem. Maritime Policy and Management 42, 68–91.

[28] Moreno-Vega, J.M., Melian, B., 2008. Introduction to the special issue on variable neighborhood search. Journal of Heuristics 14, 403–404.

[29] Norstad, I., Fagerholt, K., Laporte, G., 2011. Tramp ship routing and scheduling with speed optimization. Transportation Research Part C: Emerging Technologies 19, 853–865.

[30] P. Hansen, N.M., Perez, J.M., 2008. Variable neighborhood search. European Journal of Operational Research 191, 593–595.

[31] Potvin, J., Rousseau, J., 1995. J Oper Res Soc 46, 1433–1446.

[32] Repoussis, P.P., Paraskevopoulos, D.C., Tarantilis, C.D., Ioannou, G., 2006. A reactive greedy randomized variable neighborhood tabu search for the vehicle routing problem with time windows, in: Almeida, F and Aguilera, MJB and Blum, C and Vega, JMM and Perez, MP and Roli, A and Sampels, M (Ed.), Hybrid Metaheuristics, Proceedings, pp. 124–138. 3rd International Workshop on Hybrid Metaheuristics, Gran Canaria, Spain, Oct 13-14, 2006.

[33] Resende, M., Ribeiro, C., 2002. Greedy Randomized Adaptive Search Procedures.

[34] Resende, M., Ribeiro, C., 2003. Greedy randomized adaptive search procedures, in: Glover, F., Kochenberger, G. (Eds.), Handbook of Metaheuristics. Kluwer Academic Publishers, pp. 219–249.

[35] Romero, G., Durn, G., Marenco, J., Weintraub, A., 2013. An approach for efficient ship routing. International Transactions in Operational Research 20, 767–794.

[36] Ronen, D., 1983. Cargo ships routing and scheduling: Survey of models and problems. European Journal of Operational Research 12, 119–126.

[37] Ronen, D., 1993. Ship scheduling: The last decade. European Journal of Operational Research 71, 325 – 333.

[38] Ronen, D., 2002. Marine inventory routing: shipments planning. J Oper Res Soc 53, 108–11.

[39] Santhanakrishnan, S., Narendran, T., Ganesh, K., Anbuudayasankar, S., 2012. Comparison of meta-heuristics for container ship routing problem. International Journal of Services and Operations Management 12, 348–367.

[40] Sherali, H., Al-Yakoob, S., Hassan, M., 1999. Fleet management models and algorithms for an oil-tanker routing and scheduling problem. IIE Transactions 31, 395–406.

[41] Taillard, É., Badeau, P., Gendreau, M., Guertin, F., Potvin, J.Y., 1997. A tabu search heuristic for the vehicle routing problem with soft time windows. Transportation science 31, 170–186.

[42] Tran, N., Haasis, H.D., 2013. Literature survey of network optimization in container liner shipping. Flexible Services and Manufacturing Journal, 1–41.

[43] UNCTAD, 2013. Review of maritime transport.

[44] Woensel, T.V., Kerbache, L., Peremans, H., Vandaele, N., 2008. Vehicle routing with dynamic travel times: A queueing approach. European Journal of Operational Research 186, 990 – 1007.