Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 750232, 13 pages
http://dx.doi.org/10.1155/2014/750232

Research Article

A Heuristic Procedure for a Ship Routing and Scheduling Problem with Variable Speed and Discretized Time Windows

Krystel K. Castillo-Villar,[1] Rosa G. González-Ramírez,[2] Pablo Miranda González,[2] and Neale R. Smith[3]

[1] *Department of Mechanical Engineering, The University of Texas at San Antonio, One UTSA Circle, San Antonio, TX 78249, USA*
[2] *Industrial Engineering School, Pontificia Universidad Católica de Valparaíso, Valparaíso, Chile*
[3] *Quality and Manufacturing Center, Tecnológico de Monterrey, Campus Monterrey, Mexico*

Correspondence should be addressed to Rosa G. González-Ramírez; rosa.gonzalez@ucv.cl

Received 22 November 2013; Revised 5 February 2014; Accepted 11 February 2014; Published 8 April 2014

Academic Editor: Sergii V. Kavun

This paper develops a heuristic algorithm for solving a routing and scheduling problem for tramp shipping with discretized time windows. The problem consists of determining the set of cargoes that should be served by each ship, the arrival, departure, and waiting times at each port, while minimizing total costs. The heuristic proposed is based on a variable neighborhood search, considering a number of neighborhood structures to find a solution to the problem. We present computational results, and, for comparison purposes, we consider instances that can be solved directly by CPLEX to test the performance of the proposed heuristic. The heuristics achieves good solution quality with reasonable computational times. Our computational results are encouraging and establish that our heuristic can be utilized to solve large real-size instances.

1. Introduction

In light of the phenomenon of globalization, rapid growth of Asian economies, and the increasing volumes of international trade, global logistics management in business operations has become more important than ever. In this regard, transportation is becoming a more strategic business function because transport costs account for a larger percentage of the cost of goods sold. There is an increasing interest in reducing transportation costs and increasing route efficiency. Maritime transportation plays a key role in international trade as it represents a low cost transportation mode for high volume and long-distance shipments, being far less expensive than airplane transportation. Hence, maritime transportation is responsible for the majority of long-distance shipments in terms of volume. According to the review of maritime transport by UNCTAD [1], more than seven million tons of goods are carried by ship annually. Some illustrative statistics are provided in [2]. General shipping industry statistics are available in publications by the Institute of Shipping Economics and Logistics (http://www.isl.org/) and the Astrup Fearnley Group (http://www.isl.orgwww.fearnley.com/).

Optimizing maritime transportation systems involves several types of decisions (strategic, tactical, and operational). The strategic decisions include network design (configuration of the routes and their frequencies) and fleet and ship size determination. Tactical decisions include routing and scheduling of ships either for liner, tramp, or industrial shipping. Operational decisions refer to day-to-day decisions which may be aided by the design of on-board advisory systems that increase a vessel's operability and performance.

In this work, we consider a tactical problem consisting of routing and scheduling a heterogeneous tramp fleet. Gatica and Miranda [3] propose a network based model in which time windows for picking and delivering cargoes are discretized and the model is solved directly by using CPLEX. Authors showed that the loss of optimality due to the discretization approach was not significant. Size of the instances solved by the authors considered up to 50 cargo contracts, a fleet size of up to 9 ships, and a level of discretization of 15 time nodes. As reported in the results section, there were several instances that cannot be solved to optimality, which is proportional to the level of discretization. For the continuous case, only 50% of the instances could be solved, and as long as the level of discretization increases, higher number of instances can be solved to optimality.

Difficulty in solving real-life sized problems motivated us to propose, as an extension of this previous work, a heuristic procedure based on a variable neighborhood search to efficiently solve the problem for larger size instances. In order to evaluate the performance of the procedure, we create a set of instances using the same instance generation procedure used in [3] and compare the performance of the heuristic procedures against the results obtained by using CPLEX.

The rest of the paper is organized as follows. Section 1 provides a brief review of related literature and Section 2 presents background of the problem. Section 3 presents the problem description and introduces the notation and the mathematical formulation provided in [3]. Section 4 provides algorithmic and implementation details. Section 5 presents computational experimentation and finally conclusions and recommendations for further research are provided in Section 6.

2. Background

The problem addressed in this paper is related to the general traveling salesman problem (TSP), the vehicle routing problem (VRP), and, especially, to a variation of the VRP: the vehicle routing problem with time windows (VRPTW). TSP is a problem based on a salesman who must visit *n* clients and return to the initial place of departure. The objective is to visit all clients without passing through the ones previously visited [4]. VRP is considered as a generalization of TSP where the clients request either delivery or pick up of an amount of cargo. The VRP differs from the TSP problem in the fact that more than one vehicle is needed to deliver the cargoes with an associated cost [5]. We refer the interested reader to the work of Laporte and Osman [6], for a comprehensive review on VRP and VRPTW, and to the work of Ando and Taniguchi [7], which discusses recent issues arising on city logistics and urban freight transport.

Christiansen et al. [8] discuss several differences between ship routing and other vehicle routing problems and justifies the need for research specifically focused on ship routing and scheduling. The most recent review of ship routing and scheduling is presented in [9], in which research on ship routing and scheduling problems during the new millennium is reviewed. Some of the highlights are that the number of papers doubles every decade and that research on liner shipping, marine inventory routing, and optimal speed is leading the research efforts. In [8], a literature review for these problems is provided and perspectives for further research as well as other optimization and decision-support techniques within the shipping industry are discussed. For earlier literature reviews, the reader is referred to [10, 11].

Three general modes of operation for shipping companies can be distinguished: industrial, tramp, and liner [8, 11]. In liner shipping, the ships follow a published schedule with regular itineraries and predetermined routes, frequencies, and port arrivals/departures; it is very similar to a bus line. The tramp shipping company follows the available cargoes similarly to a taxi. A tramp shipping company usually has a fixed amount of contract cargoes that it is committed to carry and tries to maximize the profit from optional cargoes. In industrial shipping, the cargo owner usually controls the ships and aims to ship cargo at a minimal cost. In the simplest cases, industrial fleet operation is similar to tramp shipping. In more general cases, the industrial fleet operation becomes more complex, especially when trips are not prespecified and a supply network has to be determined based on time-dependent supply chain demand functions. The main difference is that industrial shipping is commonly used for a specific type of cargo related to a certain type of industry. Tramp is usually the operation mode to transport liquid and dry commodities or cargo involving a large number of units (e.g., vehicles) and liner shipping is the selected mode to transport containerized cargo which represents the major segment of liner shipping [3].

The main costs in ocean shipping are (1) capital and depreciation costs which are related to the loss of a ship's market value with respect to the initial investment, (2) running costs which are fixed costs such as maintenance, insurance, crew salaries, and overhead costs, among others, and (3) operating costs which are associated with day-to-day operations such as fuel consumption, port and customs expenses, and tolls paid at canals, among others. Fuel consumption has been a relevant subject in the maritime industry as well as for the world's largest navies due to oil price variability and environmental considerations which drive the effort for fuel-efficient navigation. Fuel consumption can be, to a large extent, controlled by navigation speed since it is approximately a cubic function of speed [11].

We base our discussion on the recent works on ship routing and scheduling of tramp fleets. Even when maritime transportation is a part of a supply chain, Christiansen et al. [8] found that little work has been done to integrate the whole supply chain. Later, Flatberg et al. [12] developed another solution approach for solving the problem proposed in [11]; they use an iterative improvement heuristic combined with an LP solver. Fox and Herden [13] describe a MIP model to schedule ships from ammonia processing plants (which convert ammonia into different fertilizer products) 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. The MIP model is solved by using commercial optimization software. An inventory routing problem similar to [11] but with multiple products was analyzed by Ronen [14] for liquid bulk oil cargo. Considering multiple products adds complexity to the model since it requires separating the shipments planning stage from the ship scheduling stage. The methods used were MIP and heuristics.

Christiansen et al. [8] commented on the lack of research on tramp shipping as compared to industrial shipping. One main reason could be the large number of small operations in the tramp market. The first work to introduce a typical tramp ship scheduling problem was presented by Appelgren [15, 16]. DW decomposition was employed to solve it. Instead of minimizing costs, the model maximizes the actual marginal contribution (excluding fixed costs). Kim and Lee [17] developed a prototype decision-support system for ship scheduling in the bulk trade where the scheduling problem is formulated as a set packing problem with similar constraints as in Appelgren's model. Several authors have developed decision-support systems for companies operating both in tramp and industrial modes [18–20]. A tramp routing and scheduling problem that maximizes the profit from operating the fleet was solved by [21, 22]. Brønmo et al. [23] develop a multistart local search heuristic. Korsvik et al. [24] propose a unified tabu search heuristic, which allows infeasible solutions with respect to ship capacity and time. In contrast to the procedure followed by Brønmo et al. [23], Malliappi et al. [25] present a variable neighborhood search heuristic; the results show that this procedure outperforms the previous heuristics. Recent research efforts [26, 27] were conducted in solving problems where cargo may be split among several ships.

This paper revisits the model in [3] and proposes heuristic procedures for solving large-scale problems. Gatica and Miranda [3] developed a network based model for the routing and scheduling of a heterogeneous tramp fleet that aims 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. To the best of our knowledge, this is the first paper that proposes a discretization of time windows for picking up and delivering cargoes. This characteristic allows for a broad variety of features and practical constraints to be considered such as navigation speed to control fuel consumption.

3. Discretized Time-Window Approach for Solving a Ship Scheduling and Routing Problem

In this section, we present the description of the ship scheduling and routing problem with discretized time windows. Section 3.1 presents the details of the discretized modeling approach and the characteristics of the problem. Section 3.2 presents the mathematical model proposed by Gatica and Miranda [3] which is considered in this paper.

3.1. Problem Description. We consider the routing and scheduling problem for tramp shipping which has been addressed by Gatica and Miranda [3]. This is one of the most relevant and challenging problems faced by decision makers at shipping companies along with planning and operation of the liner fleets. An important difference between tramp and liner operations is that liner shipping allows a more static and long-run operation planning than tramp shipping. Liner shipping operates under fixed routes. In contrast to liner case, tramp shipping faces a more dynamic demand and changes in contracts and routes. This emphasizes the importance to address this problem and to provide optimization models for designing the routes and schedules of the ships as well as the need of developing algorithms that allows efficiently solving this problem repeatedly and within reasonable computational times due to the dynamism of the tramp shipping.

As is typical in tramp shipping, contracts and their required schedules are known beforehand by ship contractors. Contracts correspond to a single shipment between two ports, both with time windows for loading and unloading the cargo. Each ship can serve one contract at a time. The fleet of ships is nonhomogeneous in terms of capacity, speed, fuel consumption, and, hence, costs. Not all ships can serve all contracts because the specific characteristics required per contract (cargo-ship or port-ship incompatibilities), which conditions the arc set of the model's underlying network. Incompatible cargoes may make it infeasible for the corresponding trips to be done consecutively by the same ship which also conditions the network.

For each sequence of cargoes to be served by a single ship, a ballast or empty trip must take place from the delivery port of each cargo to the origin of the next cargo in the sequence, unless the delivery port and next origin happen to coincide. Costs for the trips are computed based on fuel consumption and the distance travelled as well as some other operational variables and may vary from ship to ship due to the heterogeneity of the fleet. Furthermore, the speed of the ship is a key factor that affects cost. On the other hand, income is fixed for each contract and hence profit is maximized by minimizing total costs. Then, the problem consists of defining the set of cargoes to be served by each ship as well as the times of arrival and departure and waiting times at each port, with the aim of serving all cargoes at minimal total cost.

3.2. Mathematical Formulation. We adopt the discretized modeling approach used in [3] for the time windows for picking up and delivering cargoes, assuming that the arrival time to the origin port must occur only at discrete times. The same applies for cargo delivery times. Thus, each contract consists of a discrete set of possible time instants in which cargo may be picked up and/or delivered.

The nodes, indexed by $i = 1, \ldots, N$, of the network represent discrete and feasible starting times for each cargo. For each node i, the cargo associated with that node is represented by $n(i)$. The set of nodes for cargo $n(i)$ is represented by D_i. Ships are indexed by $k = 1, \ldots, B$. Each arc(i, j, k) represents the service of cargoes $n(i)$ and $n(j)$ consecutively by ship k and is included in the network if both the trips and the ship are compatible. Arc(i, j, k) is included in the network if it is feasible for ship k to begin service of cargo $n(i)$ at the time instance represented by node i, which can deliver the cargo within the corresponding time windows, make the ballast trip from the destination port of cargo $n(i)$ to the origin of cargo $n(j)$, and be available to begin service of cargo $n(j)$ at the time instance associated with node j.

For each arc, the cost parameter c_{ijk} represents the total minimal cost incurred when the ship delivers cargo $n(i)$ immediately followed by cargo $n(j)$. Costs include travel costs for trip between the ports associated with $n(i)$, waiting times, and the ballast trip to the origin port of $n(j)$. To complete the network, a fictitious node 0 is created to represent the source of all ships. For each ship k and node i, if cargo $n(i)$ is compatible with ship k, there are an arc$(0, j, k)$ and an arc$(i, 0, k)$. Cost c_{0ik} is calculated based on the real initial position of ship k, and cost c_{0ik} represents the minimum total

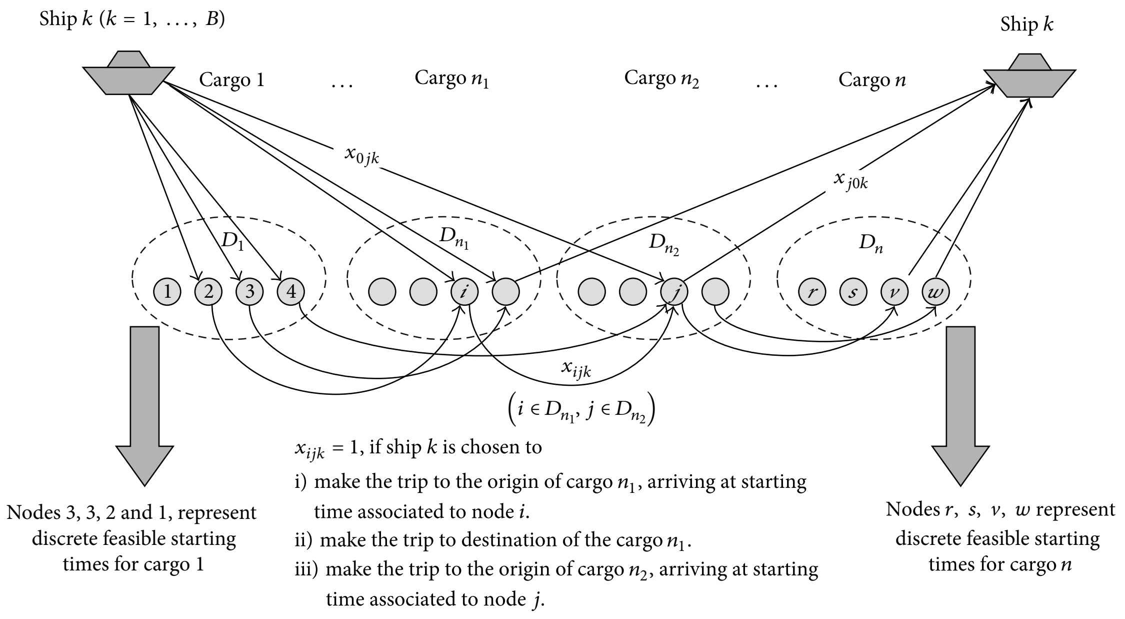

Ship k $(k = 1, \ldots, B)$
Ship k
Cargo 1
...
Cargo n_1
Cargo n_2
...
Cargo n
x_{0jk}
x_{j0k}
D_1
D_{n_1}
D_{n_2}
D_n
1
2
3
4
i
j
r
s
v
w
x_{ijk}
$(i \in D_{n_1}, j \in D_{n_2})$
$x_{ijk} = 1$, if ship k is chosen to
i) make the trip to the origin of cargo n_1, arriving at starting time associated to node i.
ii) make the trip to destination of the cargo n_1.
iii) make the trip to the origin of cargo n_2, arriving at starting time associated to node j.
Nodes 3, 3, 2 and 1, represent discrete feasible starting times for cargo 1
Nodes r, s, v, w represent discrete feasible starting times for cargo n

FIGURE 1: Graph representation of the problem. Source: Gatica and Miranda [3].

cost incurred if ship k serves cargo $n(i)$. It is assumed that ship k will stay at the port of destination of its last assigned cargo.

We can observe a single-ship view of the graph in Figure 1. The segmented ovals group all nodes (feasible starting times) related with the same cargo. Some arcs for a single ship k are drawn, based on the feasible trips that can be selected. Each arc, as previously mentioned, takes into account the ballast trip between destination port of cargo $n(i)$ and the origin of cargo $n(j)$.

The mathematical formulation of the problem from [3] is as follows:

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

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

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

$$\sum_{i\in V/(i,j,k)\in A} x_{ijk} = \sum_{l\in V/(j,l,k)\in A} x_{ijk} \qquad j \in V,\ k = 1, \ldots, B \tag{4}$$

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

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

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

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

The objective function (1) represents the total solution cost. Constraints (2) ensure that each ship is employed at most in one route. A route is defined as a sequence of cargoes to be served. Constraints (3) ensure that, for each cargo n, exactly one arc entering set D_n is selected, establishing that each cargo 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 other than 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, this constraint states 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 must return to it. Arcs leaving the fictitious node represent the ships that are, in fact, used in the solution.

4. Proposed Methodology

There are several contributions related to ship routing and scheduling problems in the literature and several mathematical models have been proposed to optimize related decisions. In addition, diverse solution approaches based on either metaheuristics or mathematical programming methods have been developed. Heuristic procedures are frequently used when exhaustive enumeration and/or optimal solution methods are impractical.

The difficulty of the ship routing and scheduling problem with discretized time windows motivated us to propose a heuristic procedure based on a Variable Neighborhood Search (VNS) metaheuristic structure. In VNS, the basic idea is to explore different vicinities in a systematic way. The majority of local search algorithms use a single neighborhood. VNS is based on three basic concepts: (1) a local minimum with respect to a structure of vicinity is not related to another local minimum with respect to another structure of vicinity; (2) a global minimum is a local minimum with respect to all possible structures of vicinity; and (3) local minima with respect to one or more structures of vicinity are close to each other [21].

Originality of the proposed algorithm relies on the adaptation of a local search heuristic to a routing and scheduling problem that is very challenging and recent, that is, the routing and scheduling of a tramp fleet with variable speed and discretized time windows (Gatica and Miranda [3]) which has not been extensively addressed in the literature (see Christiansen et al. [8]). The proposed heuristic procedure provides a solution to the routing and scheduling problem of a tramp fleet with discretized time windows and variable speed in a more reasonable computational effort which enhances the usability of the proposed method to address large scale instances faster than using CPLEX (as numerical results in Chapter 5 display).

The proposed algorithm consists of two main stages. The first stage searches for a feasible solution that defines a route for each ship. The second stage seeks to improve the initial solution by a local search procedure similar to a VNS mechanism. For this, different neighborhood structures were defined which are sequentially applied in a steepest descent fashion. A solution to the problem consists of a route for each ship in which the route is defined as the set of contracts to be performed by the ship as well as the departure and arrival time at each port (as allowed by the discretized times). Pseudocode 1 presents the pseudocode of the general procedure.

4.1. Stage 1. Construction of an Initial Feasible Solution. As shown in Pseudocode 1, the first stage refers to the construction of an initial feasible solution. For this, we employ a greedy procedure. In order to describe this procedure, we introduce the nomenclature described in Table 1, in which each node represents an allowable and discrete time instant in which cargo can be picked up or delivered. Each contract consists of a group of nodes in which the cargo is able to be picked up at an origin and delivered at a destination. Arcs joining nodes represent the associated cost of the trip, based on the distance between the origin-destination pair and the speed required to arrive at the correct time instant at the destination.

The *Construction()* procedure analyzes each contract with the aim of assigning it to a ship. For this, a sorted list of contracts based on their due dates is formed with the earliest due date contract at the top of the list. The ships are also sorted on a list. Initially, the ships are in a random order. The first iteration of the procedure begins by selecting the first contract of the list and the first ship on the list in order to analyze if it is possible to assign the contract to the ship at the earliest time instant in the discrete set of time instances in the time window of the contract. If this is possible, the contract is assigned to the ship and the ship is placed at the last position of the list of ships. Otherwise, we select the next ship of the list and repeat the same procedure until the contract is assigned to a ship in the earliest possible time instant. In each iteration, the ship to which the contract is assigned is placed in the last position of the list of ships. Therefore, the greedy function of this procedure is based on prioritizing the earliest due date contracts and seeking to assign each contract at the earliest possible instant to a ship.

Table 1: Overview of graphical nomenclature.

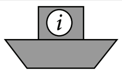
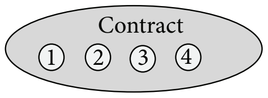
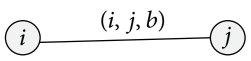
Graphical nomenclature:	Represents
	Ship i.
①	Node or instant of time.
	Contract with the corresponding discrete feasible starting times for cargo of the trip. Consider the illustration that the node ① is the earliest time instant and ④ is the latest time instant.
	Arc representing a ship b that departs at time instant i from the origin port and arrives at time instant j at the destination port.

Figure 2 illustrates the procedure. In the example, we consider four ships and twelve contracts. The first contract is assigned in its earliest time window to the first ship. The process is repeated for contracts 2 to 5. However, when we analyze contract 6 with the corresponding sequential ship 2, we realize that it is not possible to assign the contract in its earliest time instant to ship 2, so ship 3 must be considered. Given that it is possible to assign the contract to ship 3, this is assigned. Then, we consider contract 7. At the top of the list of ships we still have ship 2, so we explore the possibility to assign it to contract 7 in its earliest time instant. Given that this is possible, we assign the contract. The procedure is repeated until all contracts are assigned to ship at some time instants, always striving to assign the contracts as early as possible. As can be seen in Figure 2, there are some contracts that get assigned to later time instants because it was not possible to assign them to any ship at earlier time instants.

4.2. Stage 2. Local Search Procedure. Once a feasible initial solution has been constructed, a local search procedure is applied in order to improve the solution. For this, we propose to explore iteratively and in a steepest descent fashion, using four neighborhood structures. If no improvement of the solution is attained, then we consider two additional alternative neighborhood structures that will be also explored in a steepest descent fashion. Pseudocode 2 presents the pseudocode of the general procedure.

4.2.1. ImproveRoute(). This neighborhood of solution x consists of the set of solutions that results from exploring all feasible combinations of arcs that connect two contracts in

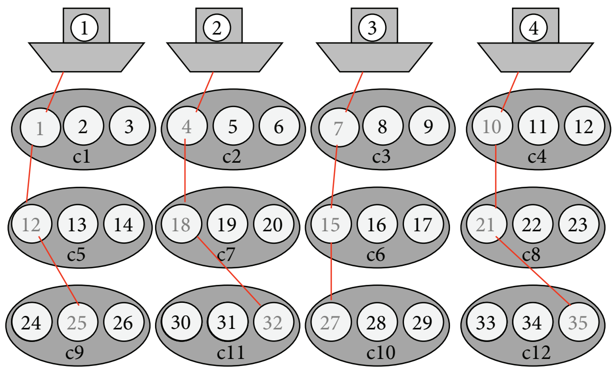


Figure 2: Example of *Construction()* procedure.

```
Ship route Procedure
Input: $P$ := a problem instance ($N, V, D_n, B, A, c_{ijk}$)
Output: $x^*$ := Route for each ship $k$,
or empty set with no feasible solution.
        $x \leftarrow$ Construction();
Set $f^* = f(x)$; $x^* = x$
  Do until a stopping condition is met
    $x \leftarrow$ Local-Search();
    If $f(x) < f^*$ then
      $f^* = f(x)$ and $x^* = x$;
  End Do
Return $x^*$
```

Pseudocode 1: Pseudocode of the solution procedure.

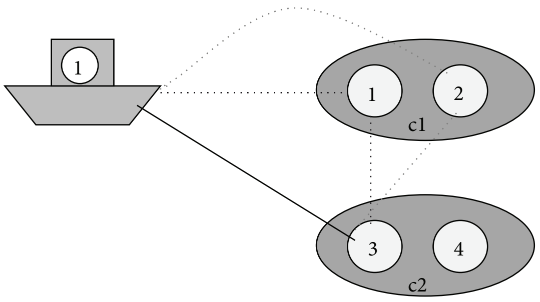


Figure 3: Illustration of an iteration of the *ImproveRoute()* procedure.

the route of a ship, selecting the pair of arcs with lowest cost. The procedure aims to improve solutions based on an analysis of the time windows of each contract, respecting the contracts assigned to the route of a ship. Figure 3 shows different options for a ship (each dotted line color corresponds to an option).

4.2.2. InsertionContractsN(). This neighborhood structure consists of the set of solutions that results from moving a contract from a route to insert it into another route. For this, the procedure considers a pair of ship routes and evaluates both the active nodes and nonactive nodes (not included in the initial solution). If moving a contract from one route to another improves the solution reducing total costs, then the move is performed. The order of the contracts is maintained at all times.

Figure 4 illustrates the procedure. The initial solution (black lines) consists of the routes of ship 1 (contract 1 at time instant 2, followed by contract 2 at time instant 3) and ship 2 (contract 3 at time instant 6). The procedure then analyzes the insertion of a contract from the route of ship 1 into the route of ship 2. The yellow arcs are an example of an option that includes nonactive nodes of the initial solution.

As observed in Figure 4, analyzing the insertion of contract 2 into the route of ship 2, it turns out that costs are reduced if contract 2 is inserted into the route of ship 2 as shown by the red lines. The resulting route for ship 2 selects contract 2 at time instant 3 followed by contract 3 at time instant 6.

4.2.3. InterchangeContractsN(). This neighborhood structure consists of the set of solutions that results from interchanging contracts between two routes of ships considering both the active nodes and nonactive nodes (those nodes that are part of the current solution as well as those that are not) and respecting the initial order of contracts in the routes. The exchange is performed only if the new configuration provides lower costs.

Figure 5 illustrates this method. Consider the initial solution presented in part (a) of the figure and assume that we will evaluate the exchange of contracts 1 and 3 (c1 and c3). As can be seen in part (b) of the figure, we add arcs to get a new solution (red, blue, pink, and yellow dotted lines). Black dotted lines correspond to the initial solution and the black line indicates that contract 2 should be performed by ship 1. If the new configuration provides lower costs, the contracts are interchanged and a new solution is obtained. Suppose this is the case for solution found with pink and yellow dotted arcs, as shown in (c).

4.2.4. TwoOptUpward(). This neighborhood of solution x consists of the set of solutions that results from crossing over a pair of routes, based on a variation of the 2 Opt local search algorithm proposed by [22]. The procedure considers a pair of arcs to be crossed and the upward part of the each

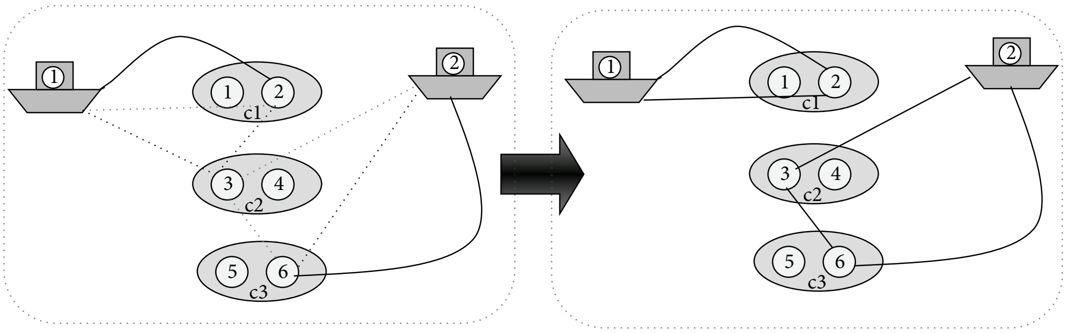
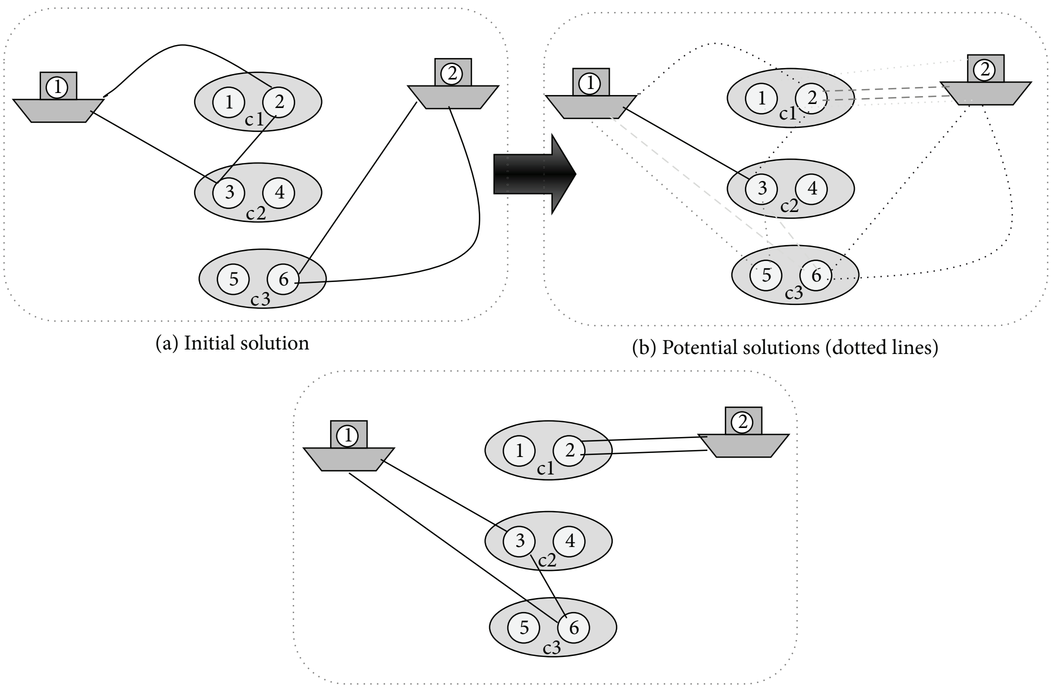


Figure 4: Illustration of *InsertionContractsN()* procedure.



(a) Initial solution

(b) Potential solutions (dotted lines)

(c) Best solution

Figure 5: Illustration of *InterchangeContractsN()* procedure.

route is swapped. The procedure differs with respect to the 2 Opt procedure in that segments of several contracts are exchanged and not only the individual route is improved. Figure 6 illustrates the procedure.

4.2.5. InsertionContractsN2(). This neighborhood of solution x is a variant of the *InsertionContractsN()* but differs in that when a contract is inserted between a pair of contracts of another route, for the previous contract to the one inserted, we select the earliest possible node for its departure and for the following contract in the route to the one inserted, we select the latest possible node for arriving. The insertion is performed only if lower costs are obtained. The procedure is illustrated in Figure 7.

4.2.6. InterchangeContracts_S(). This neighborhood structure is a simplification of the *InterchangeContractsN()* procedure in which solutions that result from all possible pair of contracts to be exchanged between two ship routes are evaluated, considering only active nodes (those that are currently part of the solution). The exchange is performed only if the new configuration provides lower costs.

4.2.7. InterchangeContracts_C(). This neighborhood structure is a variant of previous one (Interchange Contracts-S) in which a pair of contracts is exchanged but instead of searching among a pair of ships, the search is performed on a contracts sequence.

5. Computational Experimentation

This section describes the test instances generation and the computational results. The heuristic procedure was implemented in JAVA SE 6 and numerical experimentation was performed to test its performance. We generate a set of instances of different sizes. Previous instances solved in [3]

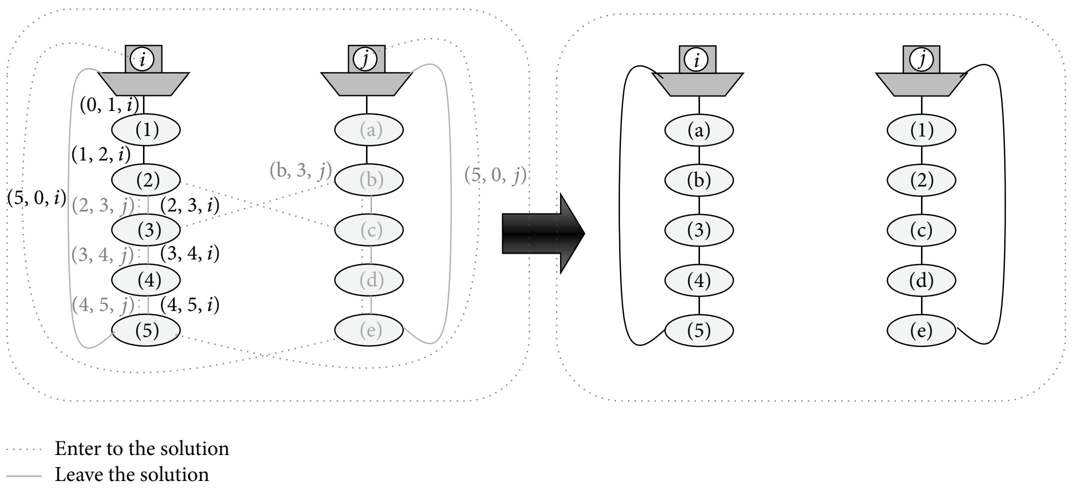


Figure 6: Illustration of an iteration of the *TwoOptUpward()* procedure.

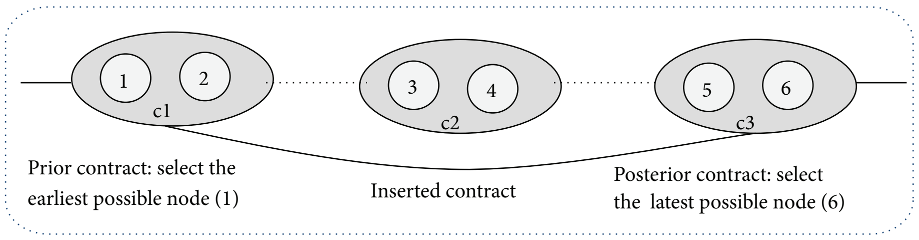

c1
c2
c3
Prior contract: select the earliest possible node (1)
Inserted contract
Posterior contract: select the latest possible node (6)

Figure 7: Pseudocode of the local search procedure.

were not available, so that new instances were generated using the test instances generator code developed by [3]. For comparison purposes, all instances were also solved by CPLEX 11.1. Numerical experiments were performed using a 2.00 GHz Pentium processor with 2 GB of RAM running under Windows XP.

5.1. Instance Generation. An instance of the problem consists of a list of contracts and the corresponding set of nodes for arrival and departure arrivals, as well as a set of arcs for all the feasible trips that could be performed. Each arc possesses an initial node, a final node, a ship number, and an associated cost. As it was previously mentioned, we employed the test instances generator developed by [3] which is composed of a real based database containing potential contracts in a port-port matrix, including distances of trip estimated to each potential contract.

The database contains 400 potential cargo contracts, a port-port matrix which contains distances of trips estimated for ach potential cargoes among 87 different ports. For each potential cargo, a pair of ports was chosen with real place of arrival which is frequently used in practice. Information regarding the necessity of going through a channel for each trip is used as well as the channel fees. In order to generate an instance, a subset of the cargoes from the database is randomly selected, according to the instance size. For the generation of random numbers, 45 different seeds are employed. The application is coded in JAVA SE 6. It is important to recall that the instances generated should guarantee that it is possible to obtain a feasible solution.

Eighteen groups of instances were generated, each group composed of a combination of ships, time window nodes, and contracts. The set of discrete time window consists of 3, 6, or 15 nodes. The number of ships was varied over the values of 4, 5, 7, and 9. The number of contracts was varied over the values of 30, 40, and 50. Each group contains 15 different instances. Instance sizes consider between 14,000 and 2,000,000 arcs. In total, we generated $3 \times 4 \times 3 \times 15 = 540$ instances.

5.2. Results and Discussion. In this section, a comparison of the results obtained by using CPLEX and the heuristic is presented. Tables 1 and 2 present the results according to the instance groups (18 groups generated with 15 instances each). Stopping rules consider a limit time of 7200 seconds for CPLEX. For the heuristic no limit time was set, considering only a maximum number of iterations without any improvement in the solution as stopping criterion based on an epsilon which was defined in terms of the instance size. For comparison purposes, instance sizes in which CPLEX could find at least a feasible solution are considered which are of similar size as those solved in [3]. Hence, for each instance solved, we compare the results obtained by CPLEX and the heuristic and estimate a gap with respect to the best integer solution reported by CPLEX (which for some

Table 2: Summary of results of instances of 30 contracts.

Instance			CPLEX				Heuristic		
Contracts	Ships	Nodes per window	Optimum found	Only feasible solution	No solution	Average time (sec)	Solution not found	Average time (sec)	GAP O.F. (%) average
30	4	3	13	0	2	1	3	4.80	4.79
30	4	6	13	0	2	4.84	2	15.92	5.36
30	4	15	13	0	2	171.84	2	93.15	5.53
30	5	3	15	0	0	1.13	0	7.06	4.34
30	5	6	15	0	0	12.73	0	26.53	5.49
30	5	15	15	0	0	140.60	0	151.40	5.49

```
Local-Search()
Input: $x$ := Initial Feasible solution (Route for each ship $k$)
Output: $x^*$ := Best Solution found (Route for each ship $k$)
  Do until no further improvement is achieved
    InsertionContractsN()
    ImproveRoute()
  End do
  Do until no further improvement is achieved
    InterchangeContractsN()
    ImproveRoute()
  End do
  Do until no further improvement is achieved
    2OptUpward()
    ImproveRoute()
  End do
  Do until no further improvement is achieved
    InsertionContractsN2()
    ImproveRoute()
  End do
  If no improvement of the initial solution was found, then
    InterchangeContracts_S()
    If no improvement was previously found, then
      InterchangeContracts_C()
    End if
  End If
  Return $x^*$
```

Pseudocode 2: Pseudocode of the Local Search Procedure.

instances corresponds to the optimal solution) as described by (7) in which Z corresponds to the value obtained by the heuristic. Positive gaps are obtained when CPLEX finds better solutions. Consider

$$\text{variation} = \frac{Z - \text{CPLEX}}{\text{CPLEX}} * 100. \quad (7)$$

Tables 2, 3, and 4 show a summary of all results obtained for those instances of 30, 40, and 50 contracts, respectively. Each instance type is indicated according to its combination of contracts, ships, and nodes which account for 15 replicates of each instance type. The averages of 15 replicates are shown for the execution times of CPLEX and the execution times of the heuristic. Furthermore, average gaps computed according to (7) are also presented, considering only those cases in which at least a feasible solution was obtained by CPLEX or by the heuristic. The tables indicate, for each instance type, the number of instances in which an optimum solution was found, the instances in which a feasible (nonoptimal) solution was found, and also those cases in which no feasible solution was obtained by CPLEX. Similarly, for the heuristic procedure, the tables present the number of instances in which an initial solution could not be found and, consequently, could not apply the methods of local search.

Table 2 shows instances of 30 contracts corresponding to the smaller size instances in which the difference on computational times between CPLEX and the heuristic resulted no significant. In terms of the quality of solutions, average gaps of the solution found by the heuristic with respect to CPLEX is less than 5.5%.

Instances of 40 contracts are medium size instances and we can observe from Table 3 a more significant difference between computational times of CPLEX and the heuristic. In terms of the gaps found by the heuristic solutions with respect to CPLEX, the maximum average gap corresponds to 8.11%. In most of the instances, CPLEX found an optimal solution or at least a feasible solution.

Table 4 shows instances of 50 contracts correspond to the biggest size instances, and computational times of CPLEX and the heuristic present more significant differences. In terms of average gaps, we observe even lower gaps with respect to the instances of 40 contracts, where the maximum average gap corresponds to 6.69%. It is noteworthy that the heuristic improves the execution times for large problems, which indicates that it can be employed for solving larger size instances. On the other hand, we can observe a good performance of the procedure with relatively small gaps with respect to the optimum solutions found by CPLEX, with a maximum value of 8.11%, which corresponds to a medium size instance (40 contracts) and does not increment proportionally to instance size.

Tables 5, 6, and 7 present results classified according to the number of ships and nodes per window for the instances of 30, 40, and 50 contracts, respectively. The tables present the percentage of instances in which CPLEX found an optimum solution, a feasible solution, and no solution, based on the number of instances ran for each type. Results found by the heuristic procedure are also presented in terms of the percentage of instances in which no solution was found. Average computational times are presented for both CPLEX

Table 3: Summary of results of instances of 40 contracts.

Instance			CPLEX				Heuristic		
Contracts	Ships	Nodes per window	Optimum found	Only feasible solution	No solution	Average time (sec)	Solution not found	Average time (sec)	GAP O.F. (%) average
40	5	3	14	0	1	3.85	6	21.55	5.00
40	5	6	15	0	0	42	3	82.50	8.11
40	5	15	15	0	0	719.40	3	442.58	7.49
40	7	3	15	0	0	8.66	0	33.93	6.27
40	7	6	15	0	0	447.53	0	137.13	7.38
40	7	15	12	3	0	1870.86	0	645.53	7.76

Table 4: Summary of results of instances of 50 Contracts.

Instance			CPLEX				Heuristic		
Contracts	Ships	Nodes per window	Optimum Found	Only Feasible Solution	No Solution	Average Time (sec)	Solution not Found	Average Time (sec)	GAP O.F. (%) Average
50	7	3	14	0	1	18.71	2	109.30	5.95
50	7	6	13	0	2	183.53	4	358.45	5.35
50	7	15	8	7	1	4291.85	2	1551.38	5.87
50	9	3	15	0	0	23.80	0	134.06	6.69
50	9	6	15	0	0	371.53	0	558.40	6.53
50	9	15	7	8	0	4450.20	0	2795.53	6.09

Table 5: Results per contracts.

Type of instance	CPLEX				Heuristic			
Contracts	Optimum found	Only feasible solution	No solution	Average time (sec)	Solution not found	Average time (sec)	GAP % time average	GAP % OF average
30	93.33%	0.00%	6.67%	55.36	7.78%	49.81	−0.1001	5.17
40	95.56%	3.33%	1.11%	515.38	13.33%	227.20	−0.5591	7.00
50	80.00%	16.67%	4.44%	1556.60	8.89%	917.85	−0.4103	6.08

and the heuristic. Average gaps of the heuristic with respect to CPLEX are also presented for each type of instance.

As observed in Table 5, computational times for CPLEX and the heuristic increase as long as the number of contracts also increase. However, not necessarily the number of instances in which CPLEX cannot find an optimal solution increases proportionally as the number of contracts increases, given that for 40 contracts the percentage of optimal solutions found by CPLEX is higher than for 30 contracts, but for 50 contracts it is observed significant reduction on the number of optimal solutions found by CPLEX. In the case of the heuristic, similar results are found in which the percentage of cases in which no feasible solution is found decreases for the 50 contracts instances. For the 50 contracts instances, computational times of the heuristic procedure are significantly lower than CPLEX times.

As observed in Table 6, the number of ships does not significantly increase the difficulty of the instance. Similar results in terms of the number of optimal solutions are found by CPLEX for most of the cases. For the heuristic procedure, very similar gaps are obtained for all the instances which, on average, are about 6%. Average times increase for both CPLEX and the heuristic for the instances with more ships which was expected.

As shown in Table 7, as the number of nodes increases, it becomes more difficult to obtain exact solutions by CPLEX and computational times increase significantly and, in this case, the heuristic performs better. On the other hand, gaps of the heuristic with respect to CPLEX do not increase proportionally with respect to the difficulty of the problem and are about 6%.

Provided that in some instances CPLEX found more efficient solutions than the heuristic procedure; Tables 8 and 9 present an analysis of the results to determine which the cases are in which the proposed heuristic is efficient. Table 8 presents a comparison of computational times between the heuristic and CPLEX considering the number of nodes per window and the number of contracts. An index is computed as indicated at (8). When the index is less than 1, the heuristic achieves better results. Consider

$$\text{Index} = \frac{\text{TimeHeuristic}}{\text{TimeCPLEX}}. \quad (8)$$

Table 6: Results per ships.

Type of instance	CPLEX				Heuristic			
Ships	Optimum found	Only feasible solution	No solution	Average time (sec)	Solution not found	Average time (sec)	GAP % time average	GAP % OF average
4	86.67%	0.00%	13.33%	59.23	15.56%	37.96	−0.3591	5.23
5	98.89%	0.00%	1.11%	153.29	13.33%	121.94	−0.2045	5.99
7	85.56%	11.11%	4.44%	1136.86	8.89%	472.62	−0.5842	6.43
9	82.22%	17.78%	0.00%	1615.18	0.00%	1162.66	−0.2801	6.44

Table 7: Results per nodes per window.

Type of instance	CPLEX				Heuristic			
Nodes per window	Optimum found	Only feasible solution	No solution	Average time (sec)	Solution not found	Average time (sec)	GAP % time average	GAP % of average
3	97.33%	0.00%	2.67%	9.525	10.67%	51.78	4.4365	5.50
6	95.56%	0.00%	4.44%	177.03	10.00%	196.49	0.1099	6.370
15	77.78%	20.00%	3.33%	1940.79	7.78%	946.60	−0.5122	6.371

Table 8: Analysis of computational times.

Contracts/nodes	3	6	15
30	5.568075117	2.416050085	0.78271028
40	4.434852118	0.448654832	0.420077521
50	5.724770642	1.651803409	0.497241494

Table 9: Analysis of quality of the heuristic.

Contracts/nodes	3	6	15
30	4.565	5.425	5.51
40	5.635	7.745	7.625
50	6.32	5.94	5.98

As observed in Table 8, the heuristic does not present an advantage for the smallest size instances (those with 3 nodes and for the 3 variations of contracts). For the medium size instances (those with 6 number of nodes), results are not conclusive (for some cases the heuristic performs better and for other it does not). However, for the bigger size instances (those with 15 nodes), the heuristic presents better results independent of the number of contracts. Hence, the heuristic is more effective (in terms of computational effort) when the instance contains more nodes. As observed in Table 8, the number of contracts does not increase significantly the complexity of the instance. This can be attributed to the fact that when the number of contracts is increased, the number of ships also increases. Based on the results, when the number of nodes (and consequently the complexity of the problem) is increased, the heuristic is a more attractive option. It is worth noting that the discretization of time windows was the approach followed to tackle a continuous time problem in this paper. As more precision is demanded (i.e., more time windows or nodes are needed), the heuristic results in a more attractive option than using CPLEX.

Table 9 presents an analysis of the quality of the heuristic in terms of the GAP obtained with respect to CPLEX considering the number of contracts and nodes per window. The average gaps of each type of instance are shown. As observed in the table, gaps do not increase with respect to the size of the instance and are relatively similar for all the instance types, with an average of about 6% which is a good feature of the proposed heuristic.

6. Conclusions and Further Research

We present a heuristic based on variable neighborhood search heuristic to solve a routing and scheduling problem for tramp shipping operations that are modeled adopting a time based discretization as proposed in [3]. This heuristic approach is an alternative method for solving large instances that CPLEX cannot solve efficiently or even find a feasible solution in reasonable times. Several special features such as navigation speed and time windows are introduced as network parameters in the instances, without increasing the complexity of the heuristic.

Numerical results show that the proposed solution approach requires reasonable computational times with reasonable gaps with respect to the solution found by CPLEX which in general were less than 8% and were about 6% in average. Instances tested in this work represent the size of real instances for a medium size shipping company for tramp mode.

As observed in Section 5, numerical results show that the number of nodes per window is the main parameter that affects the difficulty of the instance. It was observed that the heuristic procedure works better for bigger size instances; hence, if more precision is demanded for the problem (more nodes), then the heuristic represents an efficient method. Furthermore, numerical results indicate that the heuristic presents a similar performance for all the instances and gaps do not increase with respect to the instance size, which is an important element of the heuristic and it may be expected a good and robust performance for bigger size instances in which no comparison with respect to CPLEX is possible.

The usability and applicability that the proposed heuristic provides to solve the routing and scheduling problem of a tramp fleet with discretized time windows and variable speed allow addressing instances of bigger size with less computational times than when using CPLEX. Due to comparison purposes, only instances in which CPLEX is able to find a solution were solved but it is expected that bigger size instances may be able to be solved in reasonable times without decreasing the quality of the solutions as it was observed in the numerical results where the quality of the heuristic did not decrease when the size of the instance increased.

Other features that can be easily incorporated into the heuristic are, for instance, congestion at the ports or priorities of the cargoes, which for the case of the mathematical model may increase its complexity. In particular, for the case of variable navigation speed, a very complex problem is generated even for median size instances, resulting in high computational times even for instances or reduced size. This can be observed in the discretization approach. An instance of 50 cargoes and 15 window times generates $50 \times 15 = 750$ nodes.

The approach model and heuristic presented in this paper provide a decision-support tool for helping shipping companies in the design of the routes and schedules of their tramp fleets. This problem is frequently faced by the industry of international trade which presents an increasing trend in the current global environment and maritime shipping represents the bigger participation among the different transportation modes. Maritime industry presents a high dynamism and variability on its operations; thus, decision-support tools to improve their operations are extremely important. This model and solution procedure provide a mechanism to efficiently plan the routes of the fleet in order to minimize costs associated with the consumption of fuel (which can be, to a large extent, controlled by navigation speed) and to reduce lead times. In this regard, both governments and private industry may potentially benefit from the reduction of logistics costs in which the maritime fleet cost represents an important percentage of the total logistical costs.

For further research we recommend generating instances in which the navigation speeds and the discretized time windows may be determined through the local search instead of including fixed values within the network model. Additionally, mathematical programming techniques for solving the algorithm could be explored. For instance, Branch and Cut algorithms have been widely used for solving vehicle routing problems and may prove to be useful for this problem as well. We also propose applying the model and solution procedure in some real-world case studies in order to measure real improvements with respect to the current operations. Another avenue of research could involve attempting to apply similar models with discretized time windows to liner shipping operations, where the main difference is that usually no ballast or empty trips are required as in tramp shipping.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This research was supported by the Office of the Vice President for Research of the University of Texas at San Antonio through the Tenure-Track Research Grant. The authors would like to thank Oliver Riquelme and Axel Wilson, who worked on the problem addressed in this paper to obtain their bachelor degree in Industrial Engineering at the Catholic University of Valparaiso, making a valuable contribution to this research.

References

[1] UNCTAD, "Review of Maritime Transport," U.N. Publication, 2009.

[2] R. Agarwal and Ö. Ergun, "Ship scheduling and network design for cargo routing in liner shipping," *Transportation Science*, vol. 42, no. 2, pp. 175–196, 2008.

[3] R. A. Gatica and P. A. Miranda, "Special issue on Latin-American research: a time based discretization approach for ship routing and scheduling with variable speed," *Networks and Spatial Economics*, vol. 11, no. 3, pp. 465–485, 2011.

[4] H. P. Williams, "Models for solving the travelling salesman problem," in *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, R. Barták and M. Milano, Eds., vol. 3524, pp. 17–18, Springer, Berlin, Germany, 2005.

[5] G. Laporte, "The vehicle routing problem: an overview of exact and approximate algorithms," *European Journal of Operational Research*, vol. 59, no. 3, pp. 345–358, 1992.

[6] G. Laporte and I. H. Osman, "Routing problems: a bibliography," *Annals of Operations Research*, vol. 61, no. 1, pp. 227–262, 1995.

[7] N. Ando and E. Taniguchi, "Travel time reliability in vehicle routing and scheduling with time windows," *Networks and Spatial Economics*, vol. 6, no. 3-4, pp. 293–311, 2006.

[8] M. Christiansen, K. Fagerholt, and D. Ronen, "Ship routing and scheduling: status and perspectives," *Transportation Science*, vol. 38, no. 1, pp. 1–18, 2004.

[9] M. Christiansen, K. Fagerholt, B. Nygreen, and D. Ronen, "Ship routing and scheduling in the new millennium," *European Journal of Operational Research*, vol. 228, no. 3, pp. 467–483, 2012.

[10] D. Ronen, "Cargo ships routing and scheduling: survey of models and problems," *European Journal of Operational Research*, vol. 12, no. 2, pp. 119–126, 1983.

[11] D. Ronen, "Ship scheduling: the last decade," *European Journal of Operational Research*, vol. 71, no. 3, pp. 325–333, 1993.

[12] T. Flatberg, J. Havardtun, O. Kloster, and A. Lokketangen, "Combining exact and heuristic methods for solving a Vessel Routing Problem with inventory constraint and time windows," *Ricerca Operativa*, vol. 29, no. 91, pp. 55–68, 2000.

[13] M. Fox and D. Herden, "Ship scheduling of fertilizer products," *OR Insight*, vol. 12, no. 2, pp. 21–28, 1999.

[14] D. Ronen, "Marine inventory routing: shipments planning," *Journal of the Operational Research Society*, vol. 53, no. 1, pp. 108–114, 2002.

[15] L. H. Appelgren, "A column generation algorithm for a ship scheduling problem," *Transportation Science*, vol. 3, no. 1, pp. 53–68, 1969.

[16] L. H. Appelgren, "Integer programming methods for a vessel scheduling problem," *Transportation Science*, vol. 5, no. 1, pp. 64–78, 1971.

[17] S.-H. Kim and K.-K. Lee, "An optimization-based decision support system for ship scheduling," *Computers and Industrial Engineering*, vol. 33, no. 3-4, pp. 689–692, 1997.

[18] D. O. Bausch, G. G. Brown, and D. Ronen, "Scheduling short-term marine tansport of bulk products," *Maritime Policy and Management*, vol. 25, no. 4, pp. 335–348, 1998.

[19] H. D. Sherali, S. M. Al-Yakoob, and M. M. Hassan, "Fleet management models and algorithms for an oil-tanker routing and scheduling problem," *IIE Transactions*, vol. 31, no. 5, pp. 395–406, 1999.

[20] K. Fagerholt, "A computer-based decision support system for vessel fleet scheduling—experience and future research," *Decision Support Systems*, vol. 37, no. 1, pp. 35–47, 2004.

[21] N. Mladenović and P. Hansen, "Variable neighborhood search," *Computers and Operations Research*, vol. 24, no. 11, pp. 1097–1100, 1997.

[22] G. A. Croes, "A method for solving traveling salesman problems," *Operations Research*, vol. 6, pp. 791–812, 1958.

[23] G. Brønmo, M. Christiansen, K. Fagerholt, and B. Nygreen, "A multi-start local search heuristic for ship scheduling-a computational study," *Computers & Operations Research*, vol. 34, no. 3, pp. 900–917, 2007.

[24] J. E. Korsvik, K. Fagerholt, and G. Laporte, "A tabu search heuristic for ship routing and scheduling," *Journal of the Operational Research Society*, vol. 61, no. 4, pp. 594–603, 2010.

[25] F. Malliappi, J. A. Bennell, and C. N. Potts, "A variable neighborhood search heuristic for tramp ship scheduling," in *Computational Logistics*, vol. 6971 of *Lecture Notes in Computer Science*, pp. 273–285, 2011.

[26] H. Andersson, M. Christiansen, and K. Fagerholt, "Maritime pickup and delivery problems with split loads," *INFOR*, vol. 49, no. 2, pp. 79–91, 2011.

[27] J. E. Korsvik, K. Fagerholt, and G. Laporte, "A large neighbourhood search heuristic for ship routing and scheduling with split loads," *Computers & Operations Research*, vol. 38, no. 2, pp. 474–483, 2011.