An effective heuristic for solving a combined cargo and inventory routing problem in tramp shipping

Ahmad Hemmati[*], Magnus Stålhane[*], Lars Magnus Hvattum[*], Henrik Andersson[*]

[*] Department of Industrial Economics and Technology Management, Norwegian University of Science and Technology, Alfred Getz veg 3, NO-7491 Trondheim, Norway
{ahmad.hemmati}{magnus.staalhane}{lars.m.hvattum}{henrik.andersson}@iot.ntnu.no

Abstract

In this paper a vendor managed inventory (VMI) service in tramp shipping is considered. VMI takes advantage of introducing flexibility in delivery time and cargo quantities by transferring inventory management and ordering responsibilities to the vendor which in this case is a shipping company. A two-phase heuristic is proposed to determine routes and schedules for the shipping company. The heuristic first converts inventories into cargoes, thus turning the problem into a classic ship routing and scheduling problem. It then uses adaptive large neighborhood search to solve the resulting cargo routing and scheduling problem. The heuristic iteratively changes the cargoes generated to handle the customer's inventories, based on the information obtained from an initial solution. Computational results are presented, discussed and compared with exact solutions on large realistic instances. The results reveal the potential savings from converting traditional contracts of affreightment to an integrated VMI service. The factors that influence the benefits obtainable through VMI are also analyzed.

Keywords: maritime transportation; vendor managed inventory; supply chain management; adaptive large neighborhood search; ship routing; maritime inventory routing;

1 Introduction

Maritime transportation plays a major role in international trade today. More than nine billion tons of goods are carried by ships annually (UNCTAD 2013), having an estimated range from 65% to 85% of the total weight transported in international trade (Christiansen et al., 2007). Maritime transportation is the obvious choice for heavy industrial activities where large volumes are transported over long

distances since it has the lowest per unit cost of all transport modes (Christiansen et al., 2013). This is exploited by industries related to a wide range of products from oil and chemicals to cars and foods.

There are many commercial shipping routes in the world. Even though the main routes are between continents, there are also major domestic routes along shorelines and between islands, for example in Greece, Indonesia, Japan, Norway, Philippines, and the USA.

Among the three basic modes of operation in maritime transportation (liner, industrial and tramp shipping) distinguished by Lawrence (1972), we are interested in tramp shipping which is comparable to a taxi service and follows the available cargoes (a mix of mandatory contract cargoes and optional spot cargoes). Tramp shipping companies often engage in contracts of affreightment (COA) which describe obligations to carry specified quantities of cargo between specified ports within a given time frame for an agreed payment per ton. Some COAs have flexibility in the size of each cargo or shipment. Contracts between the shipping company and the cargo owner can be split into two main categories; *More-Or-Less-Owner's-Option* (MOLOO) and *More-Or-Less-Charterer's-Option* (MOLCO) (Brønmo et al. 2007b). In a typical contract, the quantity transported can vary up to $\pm 10\%$ or $\pm 20\%$ of the expected cargo quantity. In MOLCO contracts, the cargo owner has flexibility to determine the quantity, which is often not decided until the ship enters the loading port. In these situations, the shipping company planner must ensure that the ship assigned to this particular cargo has enough free capacity to carry the upper limit of the cargo quantity. However, in MOLOO contracts the ship owner decides the cargo quantity within the given interval. Sometimes, the planner can utilize this flexibility to achieve better fleet schedules and higher profits (Fagerholt and Lindstad, 2007).

There are mainly two types of problems that have been considered in ship routing and scheduling problems; cargo routing and inventory routing (Al-Khayyal and Hwang, 2007). In this paper we focus on a mix of these two which is described in detail in Section 2. Different variants of cargo routing problems have been targeted in the literature and various solution methods have been employed to solve them. For instance, Brønmo et al. (2007a) presented a multi-start local search heuristic for a ship routing and scheduling problem. Afterwards Korsvik et al. (2010) proposed a tabu search heuristic and Malliappi et al. (2011) developed a variable neighborhood search heuristic to solve almost the same problem. Recently Hemmati et al. (2014) presented a wide range of benchmark instances for the ship routing and scheduling problem. They proposed an adaptive large neighborhood search heuristic to solve the problem.

The maritime inventory routing problem is also considered by many researchers (Christiansen, 1999; Song and Furman, 2010; Engineer et al., 2012; Papageorgiou 2014). Some of the works are based on real-life problems, for example Grønhaug et al. (2010), who worked on a maritime inventory routing problem in the liquefied natural gas business and Christiansen et al. (2011) who presented a case study from the cement industry. Recently, Stålhane et al. (2014) introduced a new problem that combines traditional tramp shipping with a vendor managed inventory (VMI) service. They presented an arc-flow model describing the problem, a path-flow model which is solved using a hybrid approach that combines branch-and-price with a priori path-generation, and a heuristic path-generation algorithm. Their results show that replacing the traditional COAs with VMI services increases supply chain profit and efficiency. However, even with the heuristic path-generation algorithm, they are only able to solve small size instances in which only a few COAs have been replaced by VMI services. In addition, running times presented in their paper drastically increase from seconds to days by small increases in the problem size or even by increasing the number of replaced COAs. The method in (Stålhane et al., 2014) can only solve small instances, and it is therefore interesting to develop a method to solve

realistically sized instances and to evaluate the contribution of introducing a VMI service for such instances.

Our contributions in this paper lies in 1) presenting a powerful novel heuristic to solve the problem introduced in (Stålhane et al., 2014) which enables the solution of realistically sized instances in reasonable time. In particular, we propose a heuristic algorithm which transforms the inventory management problem into a pickup and delivery problem with time windows; 2) introducing new realistically sized benchmark instances for a tramp shipping company offering VMI services; 3) presenting a computational study illustrating the performance of the heuristic; 4) analyzing the economic impact for the shipping company, and the supply chain, from replacing some of the traditional COAs with VMI services for realistically sized instances and also analyzing the factors that influence the benefits obtainable through VMI.

The rest of the paper is organized as follows. Section 2 describes the routing and scheduling problem of a tramp shipping company which offers VMI services. In Section 3 an *iterative cargo generating and routing* (ICGR) heuristic is presented. Numerical experiments appear in Section 4. Finally, conclusions are drawn in Section 5.

2 Problem description

A regular tramp ship routing and scheduling problem is usually classified as a maritime version of the pickup and delivery problem with time windows and consists of routing a given fleet of ships to service a set of cargoes. Each cargo consists of a given quantity and has to be transported from a pickup port to a delivery port within given time windows. In maritime transportation there is usually a mix between contracted cargoes that the shipping company must transport, and a set of optional (or spot) cargoes that the company may transport if it is profitable and there is sufficient capacity in the fleet. The fleet of ships is usually heterogeneous with the ships having different cargo capacities, speeds, and cost structures. In addition, the ships are located at different positions, either in a port or somewhere at sea, at the beginning of the planning horizon.

Often the contracted cargoes of shipping companies are based on COAs, however, as argued in (Stålhane et al., 2014), these COAs may, in some cases, be replaced by a VMI service. VMI services are based on a business model where a third party logistics provider, here a shipping company, has taken on the responsibility of inventory management at both supplier and customer sites. This service permits the customer to focus on its core business and outsource the inventory management processes. The use of VMI has several positive effects on the supply chain performance, such as decreasing the inventory holding costs, reducing stock outs, increasing competition and improving the service level.

In this paper, we consider a combined cargo routing and inventory management problem faced by a shipping company that offers VMI services to some of its customers, i.e. some of the COAs are replaced by VMI services. Thus, the company is engaged in transporting the remaining COAs and optional spot cargoes, as well as offering VMI services to some of its customers. Each VMI service consists of transporting a single product from one storage where the product is produced to another one where the product is consumed, and the shipping company must ensure that the inventory level of both storages stay within their limits throughout the planning horizon.

The problem consists of a set of transportation tasks, $C = \{1, \dots, N\}$, that may be partitioned into three disjoint sets: the set of inventory pairs C^I, which represents COAs that are replaced with VMI services, the set of mandatory cargoes C^M, which represents the remaining COAs, and the set of optional cargoes C^O. It may be defined on a graph $G = (\mathcal{N}, \mathcal{A})$ where $\mathcal{N} = \{1,2, \dots, 2N\}$ is the set of

nodes, and $\mathcal{A}$ is the set of arcs. With each transportation task $i \in C$ there are associated two nodes, the pickup node i and the delivery node $(N + i)$. The nodes may also be divided into three disjoint sets: the set of nodes which are associated with inventory pairs $\mathcal{N}^I$, the set of nodes associated with mandatory cargoes $\mathcal{N}^M$, and the set of nodes associated with optional cargoes $\mathcal{N}^O$. The set of arcs, $\mathcal{A} \subseteq \mathcal{N} \times \mathcal{N}$, consists of the feasible movements of a ship between the nodes.

During a given planning horizon, going from time 0 to T, the inventory level of the storage associated with inventory node $i \in \mathcal{N}^I$ must lie between an upper and a lower limit ($\underline{S_i}$ and $\overline{S_i}$). The initial inventory level is denoted S_i, and a production (if positive) or consumption (if negative) rate R_i, is also associated with each inventory node. Inventory pairs may be serviced at most $\overline{M_i}$ times during the planning horizon to keep the inventories within their limits. It is assumed that there are lower and upper limits ($\underline{Q_i}$ and $\overline{Q_i}$) on the loaded and unloaded quantities and that the associated handling time in port is quantity-dependent. The cargo quantity associated with node $i \in \mathcal{N}^C \cup \mathcal{N}^O$ that must be transported between the pickup and delivery port is denoted by Q_i.

The combined cargo routing and inventory management problem consists of designing routes and schedules for the vessels. The problem also aims at determining the quantities handled at each inventory node without exceeding the storage's inventory limits. The objective of the problem is to maximize the profit of the shipping operations which is the difference between revenues and costs. Revenues come from servicing the inventory and mandatory cargoes and also from the optional cargoes that are serviced, while the costs are associated with the sailing performed by the fleet of ships. A feasible solution considers the ship capacities, time windows for service at ports, and precedence and pairing of visits to the pickup and delivery ports of a given transportation task for each ship. In addition, a feasible solution must ensure feasible upper and lower limits on the cargo quantity loaded or unloaded at each visit to an inventory node, and that all inventory levels stay between their limits. For a mathematical model of the problem, we refer the readers to (Stålhane et al. 2014).

3 Iterative cargo generating and routing heuristic

To solve the routing and scheduling problem of a tramp shipping company offering VMI services an iterative two-phase heuristic is proposed. The basic idea of the heuristic is to first convert the cargo and inventory routing problem into a pure cargo routing problem by transforming the inventories into sets of cargoes. This problem is solved by an adaptive large neighborhood search. The solution is analyzed and the sets of cargoes are updated. The heuristic then iterate between solving the cargo routing problem for a given set of cargo sizes and updating the cargo sizes to obtain a new cargo routing problem. A pure cargo routing problem consists of cargoes with predefined origin and destination nodes, quantity and time windows for both origin and destination nodes. When the cargo and inventory routing problem is converted to a pure cargo routing problem the inventory balance in inventory pairs is taken into account.

In the first step, the iterative cargo generating and routing (ICGR) heuristic finds the amount of cargo that has to be transported on each trip between two nodes associated with a paired inventory by considering the production and consumption rates, the initial stock levels, the inventory limits, and a predefined number of trips. For each trip required between two nodes of a paired inventory, we define a mandatory cargo. The next step of the heuristic is to consider all inventory management constraints to calculate time windows for mandatory cargoes arising from paired inventories. These time windows guarantee that the inventory levels at the inventories remain within their limits if the cargoes are picked up and delivered within the time windows.

The heuristic then checks the feasibility of the problem. To accomplish this, the optional cargoes are ignored and the heuristic tries to find a feasible set of cargoes which replace the inventory management for each inventory pair. Having a feasible set of cargoes along with the mandatory and optional cargoes and time windows related to each cargo, the initial cargo routing problem is prepared and solved by the ALNS heuristic.

After the cargo routing problem is solved, the heuristic creates new alternative cargo quantities for each visit to each inventory and adds them to a list. Then, it generates all possible combinations of cargo quantities for all inventory pairs, where each combination corresponds to a set of cargoes and represents a new cargo routing problem. The number of these combinations grows exponentially with the number of inventories and the number of visits to each inventory. To reduce solution times, the ALNS is not applied to all possible cargo combinations. Instead, a K-means clustering algorithm is used to partition the combinations into clusters of similar combinations. An iterative process is then used to select new cargo quantity combinations that are used as input to the ALNS. The pseudo-code of the heuristic is presented in Algorithm 3.1.

Algorithm 3.1 ICGR heuristic

Inputs: time horizon, number of cargoes, number of inventory pairs, number of vessels, net production-consumption rates, lower stock limits, upper stock limits, initial stocks, vessels capacity, vessels compatibility, travel timetable, travel cost table, minimum and maximum limits on the (un)loaded quantity, (un)loading time, revenue of the cargoes;
Generate initial cargo quantities to replace the inventory pairs (Algorithm 3.2);
Calculate time windows for the generated cargoes (Algorithm 3.3);
Find feasible quantities for inventory cargoes (Algorithm 3.4);
Add mandatory and optional cargoes together with the feasible set of inventory cargoes into the cargo list and set (S^*) as the best solution and $f(S^*) = -Inf$ as the objective value;
For (Initial_iteration < Max Initial_iteration)
    Solve the cargo routing problem by **ALNS** considering all cargoes and get the solution (S) and the objective value $f(S)$ (Algorithm 3.5);
    Update S^* if ($f(S) > f(S^*)$)
    For each inventory pair
        For each visit
            Create pool of **New cargo quantities** (Algorithm 3.6);
        End-for
    End-for
End-for
Generate all possible combinations of cargo quantities;
Use **K-means clustering algorithm** to cluster the combinations (Algorithm 3.7);
For (Iteration < Max Iteration)
    Cluster Selection to select potentially improving clusters (Algorithm 3.8);
    Pick a random combination from the selected cluster which is not explored yet;
    Update the quantities of the inventory cargoes using the combination picked;
    Calculate time windows for new cargoes (Algorithm 3.3);
    Assign the (un)loading time to the inventory cargoes based on their new quantities;
    Solve the cargo routing problem by **ALNS** considering all cargoes and get the solution (S') and $f(S')$ (Algorithm 3.5);
    Update S^* if ($f(S') > f(S^*)$)
    Adjust cluster weights (Algorithm 3.8);
End-for

Algorithm 3.2 Generate initial cargo quantities

The purpose of this algorithm is to generate initial cargo quantities for all inventory pairs. To this end, for each inventory pair $i \in N^I$, we define the minimum total quantity Q_i^T that should be loaded at node i and unloaded at node $N+i$ during the planning horizon. This is to ensure that at the end of the planning horizon (T), the inventory level at nodes i and $N+i$ will remain within the limits.

$$Q_i^T = \max\{(S_i + TR_i - \bar{S}_i), (\underline{S}_{N+i} - S_{N+i} - TR_{N+i})\} \quad (1)$$

In formula (1) we take the maximum of two terms where the first is related to the loading node i and the second to the unloading node $N+i$. At the end of the planning horizon (T), the loading node i with a production rate of R_i, produces TR_i of the product. Considering the initial inventory level (S_i), the total amount that must be transported to ensure that the inventory level at node i does not exceed the upper inventory limit ($\bar{S}_i$), is equal to the first term. The second term is calculated with the same concept for node $N+i$. Because node $N+i$ is a consumption node, the lower inventory limit ($\underline{S}_{N+i}$) is considered in this case.

Inventory pairs may be visited up to $\overline{M}_i$ times to keep the inventories within their limits. It is assumed that the quantities of the initial cargoes are equal for all visits but they may change during the main algorithm. Thus, the quantity of the m^{th} visit to the inventory node $i \in N^I$, Q_{im}, which is initiated in this section is calculated as in formula (2).

$$Q_{im} = \frac{Q_i^T}{\overline{M}_i} \quad (2)$$

It is to note that the sum of quantities on all the visits for node i is constant and is equal to Q_i^T.

Algorithm 3.3 Calculate time windows

For each new generated cargo (i, m), we need to define time windows for both the pickup node and the delivery node taking the inventory levels into account.

$$\underline{T}_{im}^I = \begin{cases} 0 & , S_i \geq \sum_{j=1}^{m} Q_{ij} \\ \left\lceil \frac{\sum_{j=1}^{m} Q_{ij} - S_i + \underline{S}_i}{R_i} \right\rceil & , S_i < \sum_{j=1}^{m} Q_{ij} \end{cases} \quad (3)$$

$$\overline{T}_{im}^I = \begin{cases} 0 & , S_i = \bar{S}_i \\ \left\lfloor \frac{\sum_{j=1}^{m-1} Q_{ij} - S_i + \bar{S}_i}{R_i} \right\rfloor & , S_i < \bar{S}_i \end{cases} \quad (4)$$

Formulas (3) and (4) calculate the time windows for each generated cargo at its pickup node. $\underline{T}_{im}^I$ and $\overline{T}_{im}^I$ are the earliest and latest time that we can visit inventory node i to pickup for the m^{th} time. Figure 1 illustrates the above formulas of calculate time windows for a pickup node with 3 visits with different quantities and it is the same concept for delivery nodes.

$$\underline{T}^{I}_{N+i,m} = \begin{cases} 0 & ,\bar{S}_{N+i} - S_{N+i} \geq \sum_{j=1}^{m} Q_{ij} \\ \left\lceil \frac{\sum_{j=1}^{m} Q_{ij} + S_{N+i} - \bar{S}_{N+i}}{-R_{N+i}} \right\rceil & ,\bar{S}_{N+i} - S_{N+i} < \sum_{j=1}^{m} Q_{ij} \end{cases} \quad (5)$$

$$\overline{T}^{I}_{N+i,m} = \begin{cases} 0 & ,S_{N+i} = \underline{S}_{N+i} \\ \left\lfloor \frac{\sum_{j=1}^{m-1} Q_{ij} + S_{N+i} - \underline{S}_{N+i}}{-R_{N+i}} \right\rfloor & ,S_{N+i} > \underline{S}_{N+i} \end{cases} \quad (6)$$

Formulas (5) and (6) calculate the time windows for each generated cargo at its delivery node. $\underline{T}^{I}_{N+i,m}$ and $\overline{T}^{I}_{N+i,m}$ are the earliest and latest time that we can visit inventory node $N+i$ to deliver for the m^{th} time.

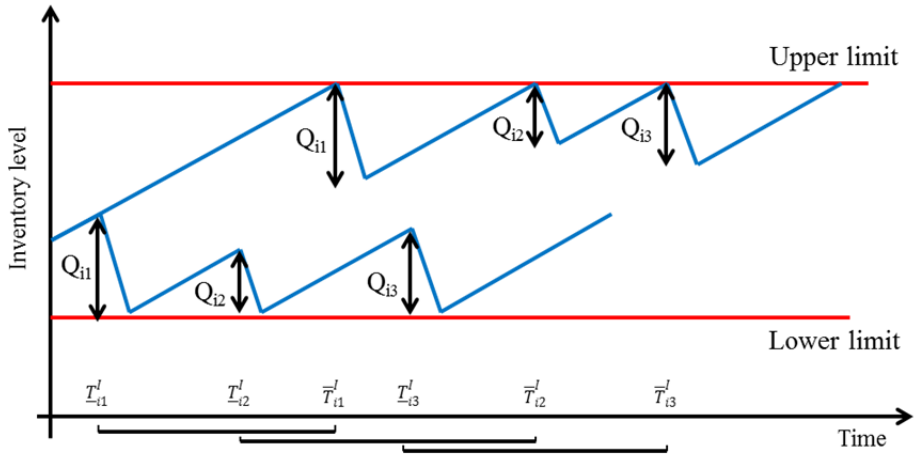

Figure 1: Inventory level during the planning horizon for a production inventory and time windows for 3 visits.

Algorithm 3.4 Find feasible combination of Q_{im}-values

The aim of this algorithm is to find a feasible set of Q_{im} and related time windows which satisfy the inventory limits. To accomplish this, the optional cargoes are ignored and mandatory cargoes are considered as well as inventory cargoes. We start with the mandatory cargoes plus initial cargo quantities for inventory pairs (Algorithm 3.2). If there exist no feasible routing and scheduling solution where all of these cargoes are transported, we start a semi-full search inside the minimum and maximum feasible amounts for Q_{im}. The term semi-full search is used because we divide the feasible interval for each Q_{im} into 100 values and we do not consider all the values inside the interval. Thus for each Q_{im} we have certain number of values which are uniformly distributed within the feasible interval for that Q_{im}. We then consider all the combinations of these Q_{im} and for each combination we find the time window (Algorithm 3.3). The next step is to solve each combination as a cargo routing

problem by ALNS (Algorithm 3.5) and if we get a feasible solution the algorithm stops and the incumbent combination of Q_{im}-values, is kept as feasible set of cargoes.

Algorithm 3.5 Adaptive large neighborhood search

The ALNS heuristic was first introduced by Ropke and Pisinger (2006) and is an extension of the large neighborhood search (LNS) heuristic of Shaw (1997). ALNS provides the possibility of having multiple destroy and repair methods within the same search process as in LNS (Ribeiro and Laporte 2012). In Algorithm 3.5 a general pseudo-code for our ALNS implementation based on Hemmati et al. (2014) to solve the cargo routing problem is presented.

Algorithm 3.5 ALNS heuristic for cargo routing problems

```
1:  Function ALNS
2:    generate initial solution s
3:    s_best = s
4:    Repeat
5:      s' = s
6:      select removal and insertion heuristics based on search parameters
7:      select the number of cargoes to remove and reinsert, q
8:      remove q cargoes from s'
9:      reinsert removed cargoes into s'
10:     If (f(s') > f(s_best)) then
11:       s_best = s'
12:     If accept (s' , s ) then
13:       s = s'
14:     update search parameters
15:   Until stop-criterion met
16:  return s_best
```

In our ALNS implementation for the cargo routing problem, a solution is represented by one vector for each vessel. Each vector consists of a sequence of nodes representing the order in which the nodes are visited along the route of the ship. The ALNS heuristic generates an initial solution based on the feasible combination of Q_{im}-values found in Algorithm 3.4, and in each iteration q requests are removed from the current solution and then reinserted in new positions. The value of q is chosen randomly from the interval $[4, \min(100, \xi n)]$ where n is the number of cargoes and ξ is a constant parameter. Thus, the number of removed requests varies during the search to provide different neighborhood size. This algorithm uses three different removal heuristics: the Shaw removal heuristic, the random removal, and the worst removal. The Shaw removal heuristic is based on removing a set of similar cargoes, while the random removal heuristic selects q cargoes at random and removes them from the solution. Finally, the worst removal heuristic removes cargoes that are placed in high cost positions. Two insertion heuristics is used in this this algorithm: Basic greedy and regret-*k*. A regret-*k* heuristic calculates the regret value which is equal to the sum of the cost differences when a cargo is inserted in its best position and when it is inserted in its *k*-th best position, where in this paper, *k* varies randomly from 2 to 4. In each iteration the regret-*k* heuristic chooses to insert the request that maximizes regret value and insert it at its minimum cost position. This continues until all requests are

inserted. Ties are broken by selecting the insertion with lowest cost. The basic greedy heuristic is equivalent to the regret-1 heuristic. We use an adaptive weight adjustment in line 14 of Algorithm 3.5 to define and adjust the weight of each removal and insertion heuristic by using statistics from earlier iterations. To this end, the entire search is divided into segments with 100 iterations in each. We then calculate a score for each of the heuristics during the search in the current segment and at the end of the segment new weights will be calculated based on their performances.

After weighting the heuristics, we select one of them using a roulette wheel mechanism. The removal and insertion heuristic are selected independently. We have implemented the acceptance criterion of simulated annealing, and the ALNS heuristics stops after 1000 iterations. For more details we refer the readers to Hemmati et al. (2014).

Algorithm 3.6 New cargo quantity generator

This algorithm generates new cargo quantities for the inventory cargoes. The total quantity transported between pairs of inventories is held constant. Thus, when an inventory cargo quantity related to a port visit changes, the quantities of the rest of the visits to this pair must change as well. New quantities for each inventory cargo are defined based on six different criteria, defined later in this section. Limits on the quantity (un)loaded in ports and the sum of the cargo quantities in an inventory pair are considered when a new value is generated for an individual cargo. To ensure a constant total quantity transported, new cargo quantities are generated for the $m-1$ first visits, while the remaining quantity is assigned to the last visit. If the quantity for the last visit violates the minimum or maximum quantity limit, the quantities assigned to the $m-1$ first visits are impossible and the combination is discarded.

The inputs for this algorithm are the current solution and the onboard quantities during the planning horizon. The algorithm also needs the capacity of the vessels and the minimum and maximum limits on the quantity (un)loaded in the ports. We now explain the criteria through an example. Let us assume that $C_i^+$and C_i^- represent the pickup and delivery visits of the i^{th} non-inventory cargo and that I_{im}^+ and I_{im}^- represent the m^{th} pickup and delivery visits of the i^{th} inventory pair. Table 1 presents a feasible solution to an instance with 4 vessels, 2 inventory pairs (each of them has an upper limit of two visits) and 4 contracted cargoes. The 4 contracted cargoes have different sizes: 184, 620, 860, and 210.

$V\#1$	Capacity: 900	Route:	I_{22}^+	I_{22}^-					
		Onboard:	690	0					
$V\#2$	Capacity: 1200	Route:	I_{12}^+	I_{12}^-					
		Onboard:	690	0					
$V\#3$	Capacity: 1200	Route:	C_2^+	C_1^+	C_1^-	C_2^-	C_4^+	I_{21}^+	C_4^-	I_{21}^-
		Onboard:	184	804	184	0	210	900	690	0
$V\#4$	Capacity: 900	Route:	I_{11}^+	I_{11}^-					
		Onboard:	690	0					

Table 1: Feasible solution to an instance with 4 vessels, 2 inventory pairs and 4 contracted cargoes

The solution presented contains the vessel capacity, visit sequence, and the onboard quantity after each visit. For example, vessel number 3 has a capacity of 1200 units. It starts its route by picking up 184 units of the contracted cargo number 2 and then picks up the contracted cargo number 1 with the quantity of 620. Thus, the onboard quantity is equal to 804. The vessel continues its route to deliver them, first cargo number 1 and then cargo number 2 and after delivering both the onboard quantity is zero. So, these four visits make a *segment*. The vessel then picks up the last contracted cargo and the inventory cargo related to the first visit of the second inventory pair and delivers them respectively, creating a second segment. A segment is defined as the list of successive cargoes in a route from the vessel is empty until it is empty the next time. Thus, a route consists of one or more segments. In the example above, all cargoes are serviced with the exception of cargo 3, which is an optional cargo that the fleet does not have sufficient capacity to service in this example.

In the following we generate new quantities based on the criteria. In the example, we have two inventory pairs and each has two visits. As mentioned, new quantities should be defined for all inventory cargoes but in the following we explain the criteria for only the inventory pair I_{21}^{+} and I_{21}^{-} which is serviced by vessel 3. Criteria 1 to 6 should be applied similarly to the remaining inventory pairs.

1. Minimum and maximum possible quantities. The idea behind this criterion is that the extreme points sometimes give the opportunity to handle more cargoes in a route and exploit the vessel capacity efficiently. It mostly happens when the capacity of two vessels are different. Minimum and maximum possible amounts ($\underline{Q_i}, \overline{Q_i}$) are input parameters, here for the given example they are (250, 1000).
2. Equal quantities. Sometimes equal quantities for all cargoes of an inventory pair give the best route. For example where the capacities of two vessels are the same or when extreme points lead to inefficient routes. The initial values for the cargoes are based on this criterion. Here the equal quantity is 690.
3. Vessel capacities. The idea is to use the full capacity of a vessel. In the given example the only vessel capacity between 250 and 1000 is 900, which is for vessels 1 and 4.
4. Vessel capacity subtracted by the maximum load of segments in the same vessel. Using the onboard quantities of the given route, segments are defined. The idea of this criterion is to exploit the vessel capacity in a segment as much as possible. For cargoes which are not in the segment, a new cargo quantity is defined to use the whole capacity of the vessel in the segment. This criterion is implemented for all vessels and segments. In the example, this criterion can be applied to the first segment of the third route since there are no inventory cargoes in this segment. The vessel capacity, 1200, minus the maximum onboard quantity of the first segment 804 is equal to 396.
5. Vessel capacity plus the initial cargo quantity minus the maximum value of the segment. This criterion is used for the segment which includes the given inventory cargo which is the second segments in this example. The idea is to increase the initial quantity of the cargo and use the whole capacity of the vessel. This criterion, in addition to exploit the whole vessel capacity in a segment, gives the opportunity of having cargoes (the rest of cargoes in that inventory pair) of smaller size that are able to fit in other routes. The vessel capacity, 1200, plus initial quantity 690 minus the maximum onboard quantity of the second segment, 900, is equal to 990.

6. Vessel capacities subtracted by the quantity of each mandatory or optional cargo. The idea behind this criterion is to use the whole capacity of the vessel by matching a cargo related to an inventory pair with a mandatory or optional cargo. This criterion considers the quantity of mandatory or optional cargoes, here (620, 184, 860 and 210), and subtracts these amounts from all vessel capacities and keep those between 250 and 1000. This gives potential cargo size of: 280, 716, 580 and 340.

By bringing all new quantities of each criterion together, the inventory cargo related to the first visit of the second inventory pair I_{21}^{+} and I_{21}^{-}, has a list of possible quantities: 250, 280, 340, 396, 580, 690, 716, 900, 990, and 1000. These are new quantities for the inventory pair I_{21}^{+} and I_{21}^{-}. The same criteria are applied for the remaining inventory pairs which may generate additional new different quantities. Then, in step 15 of Algorithm 3.1, we generate all feasible combinations of these new quantities.
Criterion 4 and 5 create quantities that depend on which vessel is handling the cargo in the initial solution. While this may be seen counterintuitive, computational testing revealed that all six criteria are useful.

Algorithm 3.7 K-means clustering algorithm

All feasible combinations of new quantities (generated in Algorithm 3.6) are generated at step 15 of Algorithm 3.1. Each combination represents a cargo routing problem that is solved by ALNS in this paper. Since the number of these combinations is enormous, the ALNS is not applied to all possible cargo combinations. Thus, a systematic selection among combinations is proposed here as well as in Algorithm 3.8. The aim of the algorithm is to select diverse combinations. At the same time we need intensification to select more from a set of similar combinations which lead us to better results. Therefore K-means clustering is used to cluster the combinations based on a similarity measure and then in Algorithm 3.8 an adaptive selection method handles the selection.

K-means is a heuristic that solves the well-known clustering problem. The algorithm partitions objects into a pre-defined (k) number of clusters. The similarity measure in the classic form of K-means clustering is the Euclidean norm which is an intuitive notion of the length of a vector. In our implementation, each combination is a vector of quantities for inventory pairs. The K-means clustering algorithm first chooses k cluster centers that are preferred to be as far as possible from each other. After this, each combination is assigned to its closest cluster center, based on the similarity measure. When this assignment process is over, a new center is calculated for each cluster using the points assigned to it. The assignment is then restarted using the new cluster centers. As a result of this loop we may notice that the k centers change their locations between iterations. When the centers do not move any more, the clustering has reached a local minimum. The algorithm is expressed in the following pseudo-code (MacQueen, 1967):

Algorithm 3.7	**K-means clustering algorithm**
1:	Choose k random points to set as cluster centers
2:	Assign each object to the closest cluster center
3:	When all objects have been assigned, recalculate the positions of the centers
4:	go back to Step 2 unless the centers are not changing

Algorithm 3.8 Cluster Selection and cluster weight adjustment

This algorithm aims at selecting a cluster at each iteration and updating the cluster weights at the end of each segment. The idea is to choose a cluster which may yield a better combination and consequently a better solution. At each iteration a cluster is selected based on its weight. Higher weight means the cluster potentially contains better combinations. Then, an unexplored combination is selected randomly from the selected cluster. This combination is used during the current iteration and at the end of this iteration according to its performance, the selected cluster gets a score. The cluster weights are updated automatically by the scores they obtain during each segment.

The algorithm starts with equal weights for all clusters and then automatically adjusts the weights using statistics from earlier iterations in accordance with the idea of adaptive weight adjustment in (Ropke and Pisinger, 2006). To accomplish this, the entire search is divided into segments with 20 iterations in each segment. Clusters get new weights according to the scores they obtain during the search in the current segment. The score of a cluster is increased based on the quality of the selected combination. When a combination from the selected cluster is evaluated, if it improves the best solution of that cluster then the score of that specific cluster is increased. The cluster gets double score if the combination selected from that cluster improves the global best solution.

Let w_{is} be the weight of cluster i in segment s. The new weights which are considered during the segment $s+1$ are calculated as following:

$$w_{i(s+1)} = w_{is}(1-r) + r\frac{\pi_i}{\theta_i}, \tag{7}$$

where π_i is the score of cluster i, θ_i is the number of times cluster i have been selected, and r is a predefined parameter to balance the earlier weights and the new normalized scores. The values of π_i and θ_i are re-set between segments. In the current segment, s',after weighting the heuristics, we have k clusters with weights $w_{is'}$, $i \in \{1,2,\dots,k\}$. We then select cluster j with probability $\frac{w_{js'}}{\sum_{i=1}^{k} w_{is'}}$ by using a roulette wheel selection principle.

As mentioned in the main algorithm, when a cluster and a combination from the cluster is selected, the time windows for the new combination are calculated using Algorithm 3.3.

4 Computational results

To evaluate the ICGR heuristic, we have tested it on the instances proposed by Stålhane et al. (2014), and also generated a set of new and larger instances using the same instance generator. The instances have been generated to reflect the operations of a deep-sea liquid bulk shipping company operating in the Atlantic basin. The ports are divided into two regions, and 80 % of the loading/unloading pairs have ports in different regions. The distance between the two regions is considerably longer than the distances within each region. The quantities of the optional cargoes are between 0.1 and 0.8 times the capacity of the smallest ship, and the revenue obtained from transporting optional cargoes is proportional to the quantity and the distance between the loading and unloading ports. As in (Stålhane et al., 2014), the revenues obtained from transporting mandatory cargoes and inventory cargoes have been set to zero.

The difference between the lower ($\underline{S_i}$) and upper limits ($\overline{S_i}$) on the inventory levels is about three times the capacity of the smallest ship, and all inventories are half-full at the beginning of the planning horizon. The production and consumption rates (R_i) have been set so that roughly two full ship loads of the smallest ship must be transported during the planning horizon. The fleet of ships is heterogeneous regarding load capacity and cost structure. The largest ship has roughly 50 % larger cargo capacity than the smallest ship, and the sailing costs are proportional to the cargo capacity of the ship. The initial position of each ship is drawn randomly from the set of ports in the instance.

The new instances have been generated by varying the number of ships, cargoes, and inventory pairs. The number of ships is between 4 and 8, the number of cargoes between 10 and 30, the number of inventory pairs between 1 and 4, and the maximum number of visits for each port is set to 4. Since some combinations of ships and cargoes are unrealistic in practice, e.g. 8 ships and 10 cargoes, we have limited the testing to a subset of the possible combinations.

The results on the original instances from (Stålhane et al., 2014) are summarized in Table 2. Stålhane et al. (2014) presented both an exact column generation method and a heuristic alternative. Out of the 27 instances in Table 2, the exact method failed to find solutions in 5 cases and the results are not available for them. Our proposed heuristic is much faster than the heuristic column generation method when the number of inventory pairs is large, and in general finds better solutions.

The first three columns in Table 2 describe the instance size, with respect to the total number of cargoes and inventory pairs ($|C|$), the number of inventory pairs, and the maximum number of visits to each inventory. The next two columns report the total running time (consisting of route generation time and solution time) for the exact and the heuristic column generation, respectively. The sixth column shows the optimality gap of the heuristic route, whereas the next two columns report the optimality gap and the running time for the ICGR heuristic. Finally, the last column shows the effect of Algorithm 3.6 – Algorithm 3.8, giving the improvement from the initial solution obtained at Step 7 in Algorithm 3.1 in the first iteration, and the best solution found after the complete search.

The ICGR heuristic usually finds the same solution as the exact method, but there are three exceptions. First, one of the instances has a negative gap, meaning that the ICGR heuristic finds a solution that is better than the solution found by the exact column generation approach. The explanation lies in a small difference in the assumptions used: the column generation does not allow a vessel to visit an inventory pickup node twice without visiting the corresponding delivery node in between. However, our heuristic can exploit this additional flexibility which is allowed by the arc-flow model in Stålhane et al. (2014), but that was not exploited by the column generation methods. Second, for two instances the proposed heuristic has relatively large gaps. It turns out that the optimal routes from the column generation are not possible to find for our proposed heuristic, since we have assumed that arrival times must be integer (the time windows are rounded to integers). In very tight instances, this assumption may remove some good solutions that would otherwise be feasible.

Table 3 shows profit loss incurred when comparing the COA solution of MOLCO and MOLOO models presented in (Stålhane et al., 2014) with VMI solution. Out of 40 instances introduced in this paper, 2 instances were infeasible, and were discarded. The exact column generation fails for most of these instances, and we focus on comparing the ICGR heuristic for the VMI version of the problem with optimal solutions to MOLCO and MOLOO versions of the instances.

Original Instances			Exact route generation (seconds)	Heuristic route generation (seconds)	Opt. gap Heuristic route generation (%)	Opt. gap ICGR heuristic (%)	ICGR heuristic. (seconds)	Improvement from initial solution (%)
$\|C\|$	$\|C^I\|$	$\overline{M}_i$						
6	1	2	1	1	0.01	0.00	1	0.00
6	1	3	1	1	0.01	0.00	6	0.00
6	1	4	1	1	0.01	0.00	12	0.00
6	2	2	13	2	1.30	0.00	4	2365.60
6	2	3	52	4	1.24	0.00	64	2365.60
6	2	4	257	6	1.24	0.00	83	486.11
6	3	2	872	12	0.07	0.00	39	1279.07
6	3	3	29,516	51	0.25	0.00	83	1249.36
6	3	4	NA	286	NA	NA	236	526.50
10	1	2	7	2	0.00	0.00	2	16.20
10	1	3	4	1	0.00	8.13	21	7.79
10	1	4	7	1	0.00	0.00	92	7.18
10	2	2	98	15	0.00	0.01	15	79.61
10	2	3	1,946	92	0.00	0.05	159	81.78
10	2	4	14,970	223	0.00	0.00	236	82.67
10	3	2	586	27	0.00	0.00	158	0.00
10	3	3	NA	4,639	NA	NA	520	86.90
10	3	4	NA	32,101	NA	NA	236	NA
15	1	2	440	156	3.10	0.00	11	20.34
15	1	3	2,654	177	2.69	0.13	102	12.87
15	1	4	5,752	220	2.69	-2.45	268	16.56
15	2	2	5,181	265	0.00	0.00	183	141.75
15	2	3	89,422	237	0.00	9.80	303	65.59
15	2	4	438,729	369	0.00	0.00	372	60.23
15	3	2	63,829	320	0.00	0.00	278	146.28
15	3	3	NA	1,200	NA	NA	363	63.57
15	3	4	NA	2,423	NA	NA	517	59.62

Table 2: Summary of results on the original test instances, comparing the ICGR heuristic to both an exact and a heuristic column generation method.

Both MOLCO and MOLOO have been solved with different upper bounds on the number of visit ($\overline{M}_i = 2$, 3 and 4), and Table 3 is based on the best of these solutions. For some instances, the optimal solution was not found for specific values of $\overline{M}_i$, and in this case the best of the available solutions is considered. If no optimal solution was found for any value of $\overline{M}_i$, the corresponding entry has been marked with "NA". The profit loss is calculated as the difference between the objective value of the ICGR heuristic and the MOLCO/MOLOO model, divided by the objective value of the ICGR heuristic. Even though our heuristic does not guarantee the optimal value, it is much better than what can be obtained in the best case within the MOLOO or MOLCO framework.

New instances			MOLCO ±10%	MOLCO ±20%	MOLOO ±10%	MOLOO ±20%
$\|C\|$	$\|C^I\|$	$\|V\|$				
10	1	4	16.23	16.56	8.28	10.69
10	2	4	43.70	56.90	28.06	41.07
15	1	4	9.98	11.21	8.74	8.74
15	2	4	27.60	28.25	25.89	25.91
15	3	4	28.55	43.38	26.76	26.76
15	4	4	42.27	56.06	44.49	44.50
20	1	4	0.03	0.03	0.01	0.01
20	2	4	18.04	18.99	0.17	8.23
20	3	4	43.94	79.23	3.32	31.67
20	4	4	73.72	80.20	37.72	37.72
15	1	6	4.82	7.28	4.65	4.76
15	2	6	4.82	7.93	4.82	6.48
15	3	6	17.55	26.28	14.09	17.79
15	4	6	17.71	31.22	27.42	27.46
20	1	6	1.08	3.26	0.21	0.80
20	2	6	5.61	10.99	6.96	7.67
20	3	6	15.45	36.23	5.42	13.08
20	4	6	38.23	81.23	20.93	37.70
25	1	6	0.01	0.01	0.30	0.30
25	2	6	2.70	4.00	5.68	5.68
25	3	6	12.27	21.57	13.71	13.71
25	4	6	30.90	45.96	26.45	29.85
30	1	6	4.00	4.00	NA	NA
30	2	6	0.57	9.00	NA	NA
30	3	6	11.01	14.52	NA	NA
30	4	6	12.77	16.97	NA	NA
20	1	8	0.66	2.34	0.48	0.54
20	2	8	5.74	9.27	0.41	3.32
20	3	8	14.12	28.74	2.65	9.04
20	4	8	29.13	NA	9.41	13.19
25	1	8	0.02	0.02	0.01	0.01
25	2	8	1.30	1.94	0.19	1.73
25	3	8	4.53	9.85	9.19	9.24
25	4	8	12.42	22.21	15.11	16.89
30	1	8	1.31	3.66	1.69	1.74
30	2	8	4.89	6.33	3.51	4.19
30	3	8	5.96	7.40	NA	NA
30	4	8	3.53	15.04	NA	NA

Table 3: The reduction in profit when using the best results obtained through MOLCO and MOLOO, compared with the results obtained by the ICGR heuristic.

We are interested in analyzing the factors that influence the benefits obtainable through VMI. To this end, we have used the results of Table 3 in Figures 2–4. In Figure 2, for each number of inventory pairs and each number of cargoes a point is defined and the average profit loss over different number of vessels is depicted. In Figure 3, each point shows the average profit loss over different number of cargoes for each pair of parameter settings. In Figures 2–4, the numbers are based on the best results of MOLCO 20%. However, the trends are similar for the other variants in Table 3. First, Figures 2 and 3 show that the effect of replacing COAs with VMI increases with the number of inventory pairs $|C^I|$. When the number of inventory pairs increases there is more flexibility, since defining VMI services for each inventory pair brings its own flexibility. Figure 2 also indicates that the effect is smaller for instances with many cargoes: with many optional cargoes to choose from, it is possible to utilize the fleet relatively efficiently even without the added flexibility resulting from the introduction of VMI. Figure 3 indicates that the effect is also smaller with a larger fleet $|V|$. Larger fleets make the problem looser, and even without VMI there is enough flexibility to accommodate optional cargoes.

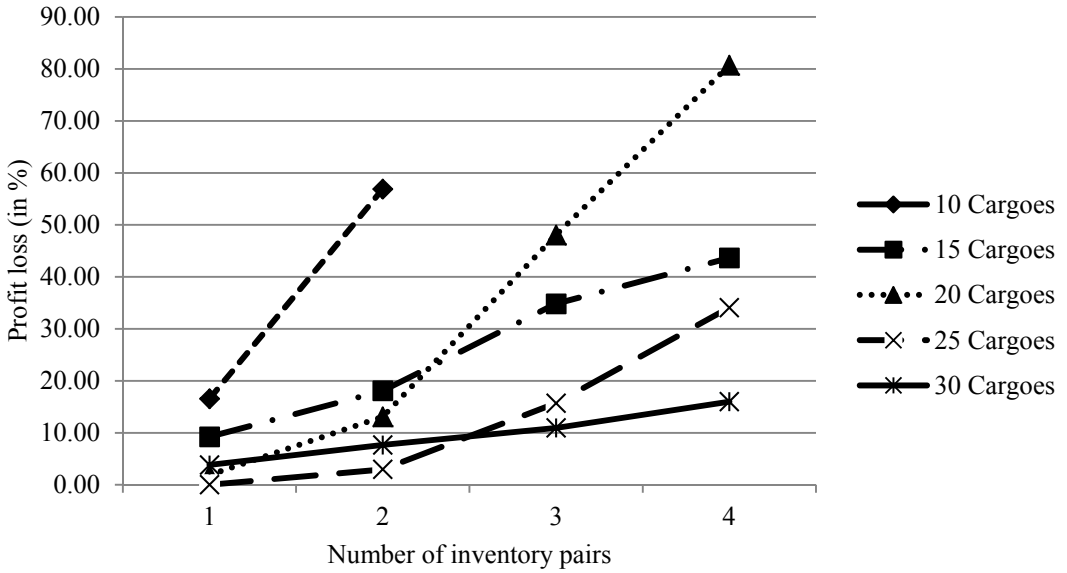

Figure 2: Showing the average reduction in profits when using MOLCO 20%, compared to results obtained using the ICGR heuristic, focusing on the effect of the number of inventory pairs included and the number of cargoes.

The picture is completed by Figure 4, which shows that the importance of VMI is greater when there are a large number of inventory pairs $|C^I|$ compared with the total number of cargoes $|C|$. This figure clearly illustrates the twofold source of flexibility. More inventory pairs and less contracted cargoes give more flexibility to shipping company since they are able to choose the proper size for each transportation task. In Figure 4, each point shows the profit loss for different ratios of inventory pairs to the total number of cargoes and wherever the ratio is equal for two or more instances we have taken the average.

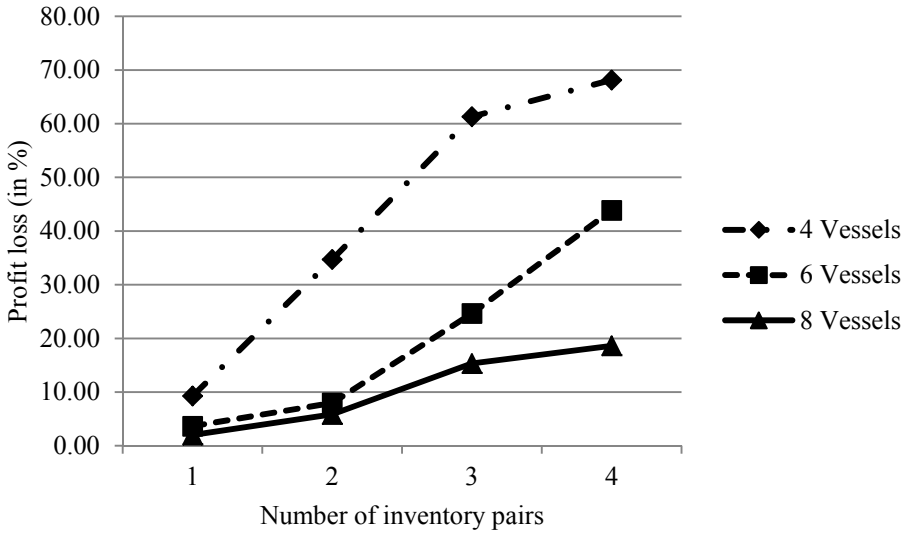

Figure 3: Showing the average reduction in profits when using MOLCO 20%, compared to results obtained using the ICGR heuristic, focusing on the effect of the number of inventory pairs included and the number of vessels.

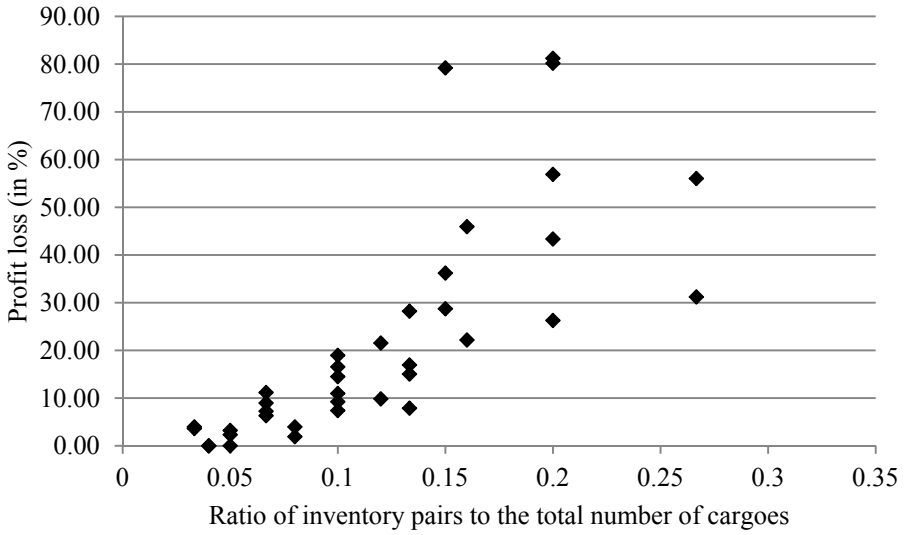

Figure 4: Showing the average reduction in profits when using MOLCO 20%, compared to results obtained using the ICGR heuristic, focusing on the effect of the number of spot cargoes included in the problem instances.

The results hence seem to verify the usefulness of introducing VMI, while emphasizing that the value of VMI increases with the number of inventory pairs introduced. At the same time, the value of VMI is lower when there are many spot cargoes available.

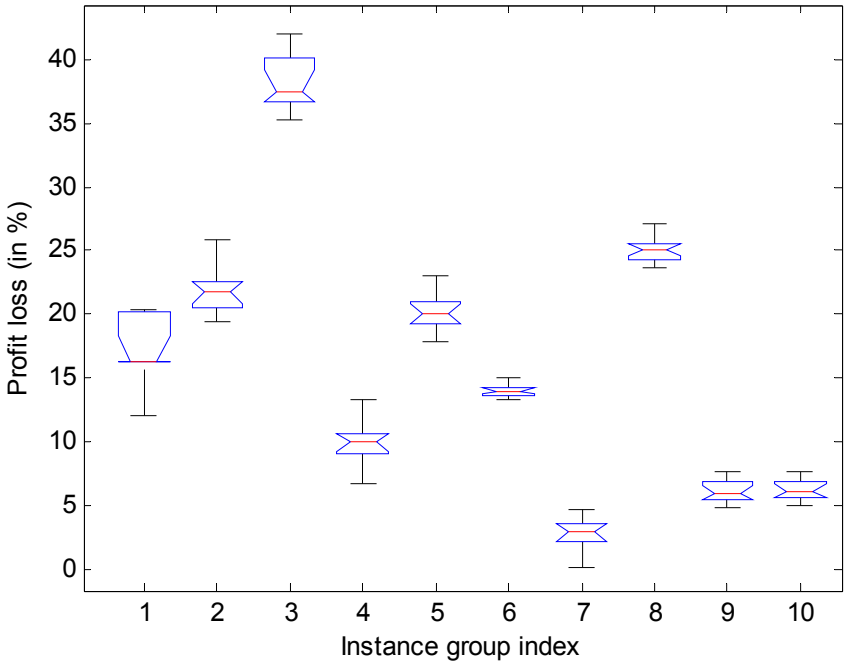

Figure 5: Box chart of the improvement of the solutions obtained at the end of the ICGR heuristic compared to its initial solution

Finally, we include Figure 5 to show the effect of Algorithm 3.6 – Algorithm 3.8. It gives a box chart showing the difference between the initial solution obtained at Step 7 in Algorithm 3.1 in the first iteration, and the best solution found after the complete search. The primary axis is divided in 10 and each group consists of 4 instances with the same number of cargoes and ships and different number of inventory pairs which varies from 1 to 4, as described in Table 3. The ICGR has been run 10 times for each instance and the average over 4 different number of inventory pairs in a group has been considered.

The chart indicates the 25th and 75th percentiles and median reduction in profit if the initial solution would be used, as well as a 95 % confidence interval for the reduction in profit within the instance group. The figure shows that the value of adding Algorithm 3.6 – Algorithm 3.8 is significant and large.

5 Concluding remarks

Tramp shipping companies traditionally use contracts of affreightment (COAs) to regulate the service provided to their customers. Replacing traditional COAs with vendor managed inventories (VMI) reduces the costs of providing an acceptable service to the customers of a tramp shipping company. Previous work (Stålhane et al., 2014) presented formulations and exact solution methods for the tactical planning problem of a shipping company offering both COAs and VMIs. The results indicated a very large potential benefit of being able to convert COAs into VMIs.

This work presented a novel heuristic solution method based on iteratively converting inventory pairs into cargoes and solving, as a subproblem, an associated cargo routing problem using an adaptive large neighborhood search heuristic. While previous work had been able to assess the value of introducing VMI only for small instances, the ICGR heuristic allowed an analysis based on realistically sized

instances. The results show that the value of introducing VMI is not as large as previously indicated, and that it depends in particular on the number of contracts that are converted, the number of vessels in the fleet, and the number of additional cargoes available for transportation.

Acknowledgements

This research was carried out with financial support from the Research Council of Norway through the DOMinant II project.

References

[1] Al-Khayyal F, Hwang SJ (2007) Inventory constrained maritime routing and scheduling for multi commodity liquid bulk, Part I: Applications and model. European Journal of Operational Research 176(1):106–130

[2] Brønmo G, Christiansen M, Fagerholt K, Nygreen B (2007a) A multi-start local search heuristic for ship scheduling – a computational study. Computers & Operations Research 34(3):900–917

[3] Brønmo G, Christiansen M, Nygreen B (2007b) Ship routing and scheduling with flexible cargo sizes. Journal of the Operational Research Society 58(9):1167–1177

[4] Christiansen M (1999) Decomposition of a combined inventory and time constrained ship routing problem. Transportation Science 33(1):3–16

[5] Christiansen M, Fagerholt K, Nygreen B, Ronen D, Barnhart C, Laporte G (2007) Maritime transportation. Transportation, Handbooks in Operations Research and Management Science 14(North-Holland, Amsterdam) 189–284

[6] Christiansen M, Fagerholt K, Flatberg T, Haugen Ø, Kloster O, Lund EH (2011) Maritime inventory routing with multiple products: A case study from the cement industry. European Journal of Operational Research 208(1):86–94

[7] Christiansen M, Fagerholt K, Nygreen B, Ronen D (2013) Ship routing and scheduling in the new millennium. European Journal of Operational Research 228(3):467–483

[8] Engineer FG, Furman KC, Nemhauser GL, Savelsbergh MWP, Song JH (2012) A branch-price-and-cut algorithm for single-product maritime inventory routing. Operations Research 60(1):106–122

[9] Fagerholt K, Lindstad H (2007) Turborouter: an interactive optimization-based decision support system for ship routing and scheduling. Maritime Economics and Logistics 9:214–233

[10] Grønhaug R, Christiansen M, Desaulniers G, Desrosiers J (2010) A branch-and-price method for a liquefied natural gas inventory routing problem. Transportaion Science 44(3):400–415

[11] Hemmati A, Hvattum LM, Norstand I, Fagerholt K (2014) Benchmark suite for industrial and tramp ship routing and scheduling problems. INFOR, 52(1):28–38

[12] Korsvik JE, Fagerholt K, Laporte G (2010) A tabu search heuristic for ship routing and scheduling. Journal of the Operational Research Society 61(4):594–603

[13] Lawrence SA (1972) International sea transport: The years ahead, Lexington Books, Lexington.

[14] MacQueen J (1967) Some methods for classification and analysis of multivariate observations. Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, California, USA, 281–297

[15] Malliappi F, Bennell JA, Potts CN (2011) A variable neighborhood search heuristic for tramp ship scheduling. Computational Logistics, Lecture Notes in Computer Science 6971:273–285

[16] Papageorgiou DJ, Nemhauser GL, Sokol J, Cheon MS, Keha AB (2014) MIRPLib – A library of maritime inventory routing problem instances: Survey, core model, and benchmark results. European Journal of Operational Research 235(2):350–366

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

[18] Ropke S, Pisinger D (2006) An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. Transportation Science 40(4):455–472

[19] Shaw P (1997) A new local search algorithm providing high quality solutions to vehicle routing problems. Technical report, University of Strathclyde, Glasgow

[20] Song JH, Furman KC (2013) A maritime inventory routing problem: Practical approach. Computers & Operations Research 40(3):657–665

[21] Stålhane M, Andersson H, Christiansen M, Fagerholt K (2014) Vendor managed inventory in tramp shipping. Omega 47:60–72

[22] UNCTAD (2013) Review of Maritime Transport. United Nations, New York and Geneva