Solving the Vessel Routing and Scheduling Problem at a Canadian Maritime Transportation Company

Louis-Pierre Trottier[1] and Jean-François Cordeau[2]
HEC Montréal and CIRRELT, 3000 chemin de la Côte-Sainte-Catherine, Montréal, H3T 2A7 Canada

Abstract

This paper describes the development of a scheduling tool to help a Canadian shipping company optimize the routing and scheduling of their ships and allocate the right assets to the right cargo. The problem can be formulated as a variant of the vehicle routing problem with pickup and delivery and it is solved by adapting a tabu search heuristic originally developed for the latter problem. The decision support tool is currently implemented within the company and it is used by the scheduling team to validate assumptions and schedule their fleet over short to long term planning horizons. A user interface lets the users visualize results and carry out manual changes to the schedule. The decision support tool yields important savings compared to manual planning practices that were previously in place at the company.

1. Introduction

This study is motivated by the needs of a multi-national short-sea shipping company that operates mainly in North America, South America, Europe and Oceania. They specialize in short-sea shipping and transhipment operations with a fleet of specialized self-unloading vessels, cement vessels and bulk carriers. They own and operate more than 60 vessels worldwide. Short-sea shipping is characterized by short distances between cargo loading and discharging points. Instead of the usual long legs in shipping, frequent small legs between multiple ports are carried out, generally in coastal waters or inland water ways. Short sea shipping is in direct competition with rail and truck transportation, and is generally used for bulk or large cargos.

The company wanted to acquire an optimization-based decision support tool to improve their routing and scheduling decision process, and to help their tactical and strategic planning. The organisation is doing frequent small voyages and wants to know if they can reduce their fleet and repositioning costs while respecting their business constraints. To obtain a realistic model, we looked at existing optimization tools in different industries and adapted one to the short sea and dry bulk shipping context. An integrated solution approach capable of handling large instances and a friendly user interface were developed to facilitate the use of the algorithm and to help users run multiple scenarios.

The basic operation consists of cargo owners (customers) contacting ship owners (transportation companies) to move a quantity of dry bulk cargo from one port to another. Cargo owners sometimes request their cargo to be loaded or discharged at multiple locations. They pay a price per ton to the transportation company, and the latter is responsible to move the cargo within given laycans (time windows). This type of operation is known as tramp shipping due to the fact that ships do not follow fixed routes but instead travel according to customer needs. The quantity to move can be smaller, equal or greater than the size of the typical vessel of the shipping company. In the first two cases, only one ship will be required to move the cargo. The shipping company will thus assign one ship from the fleet to move the cargo within the given time windows. In the third case, the cargo will have to be divided into parcels and moved with more than one ship or in more than one voyage. In the simplest example, to minimize transportation costs the shipping company

[1] louis-pierre.trottier@hec.ca
[2] jean-francois.cordeau@hec.ca

will try to maximize the parcel size to minimize ship movements. In this article, we consider only the first two cases as the company contractually determines the size of each parcel.

When a shipping company agrees to move the cargo of a customer, a target quantity (parcel size) is decided as well as a range. Generally, the exact quantity that must be moved is not known beforehand. Instead, a minimum and a maximum are fixed and the shipping company or the customer can decide to load any quantity within these bounds. Under the minimum, the customer pays deadfreight and over the maximum, depending on the contract, additional tonnes might be charged at a different freight rate. Additional components of the freight rate are generally added or subtracted to the bill at the end, including demurrage, fuel surcharges, etc. The shipping company schedules its fleet to visit each series of ports to load and discharge the contracted cargo. The fleet can be homogenous (ships with the same costs, speed, constraints, etc.) or heterogeneous. In our case, the fleet is heterogeneous as the ships were mostly built at different times and have different cost components. A voyage consists of a positioning leg (ballast leg), the loading operation, the laden leg and the discharging operation. In the case of a dry bulk carrier, cargo is loaded and discharged with shore equipment while for self-unloaders cargo is discharged (and sometimes loaded) with equipment aboard the ship.

Large customers have generally well established supply chains with regular operations. They can manage their transportation network or subcontract it to external organizations. In the case of maritime transportation, long-term contracts are generally established between transporters and customers, and these contracts may range from a few months to several years. Contracts generally fix a number of tonnes per year, freight rates, parcel sizes and other clauses. With contracts, transporters have a rough idea of the required tonnage from their pool of customers for the next months and years. Generally, customers will confirm their shipping requirements a few weeks in advance and the shipping company will create a schedule based on those confirmations. When the schedule is approved internally, ships are nominated for the shipping tasks and confirmed to the customers.

The scheduling and routing process is generally done manually by a scheduler on a continuous basis. In our case, 60-day schedules are created on a rolling basis and sent to the customers. Sailing, loading and discharging times are estimated by the scheduler based on the historical performance of the nominated vessel.

To create a schedule from scratch, the scheduler uses a manual decision process. It generally starts with the most constrained voyages and the vessels that can supply those requests. The scheduler will insert those voyages in the routes of these vessels and, if capacity is left, standard contracts will be inserted in the schedule. Each voyage is inserted right after another to minimize slack between voyages and to reduce positioning voyages. After the constrained voyages are inserted, the remaining voyages are scheduled with the remaining fleet. When the first draft of the schedule is created, the scheduler checks voyage constraints (time windows, compatibility, etc.). If they are not fully respected, the scheduler will try a second iteration by exchanging voyages between ships and by checking if constraints are violated or not. A schedule is deemed satisfactory when all future voyages are scheduled and no major constraints are violated. If vessels remain empty, they are temporarily laid up or put in repair. This scheduling process is very time-consuming and is mostly based on the experience of the scheduler.

Each vessel costs tens of millions of dollars and even hundreds of millions of dollars for highly specialized ships. In addition to the fixed costs, operating a ship can cost up to thousands of dollars in maintenance, manning, fuel and insurance per day. Laying up or idling ships is generally cost prohibitive for ship-owners, and they usually try to minimize their fleet size for their given pool of customers (or maximize their pool of customers for a given fleet). Having planning tools to make better fleet sizing decisions is crucial to shipping companies. This article further explores the use of an optimization tool used primarily for operational and tactical planning for sizing decisions.

The aim of this paper is to present the development and the application of an algorithm to optimize the routing and scheduling of a fleet of vessels. Scheduling is a work intensive process that relies heavily on the experience of few individuals. Furthermore, manual schedules are hard to evaluate in terms of quality. To address these issues, we create a model based on existing operations and propose a solution method that can provide solutions in short computing times. We then assess the savings by comparing the current schedules of the company to the schedules produced by the algorithm.

The rest of this paper is organised as follows. In Section 2, we present a brief literature review of ship routing and scheduling problems. The problem is defined and formulated in Section 3, and we present our solution approach in Section 4. We then analyse the performance of the model on the company's operations and analyse the major differences with the current scheduling process in Section 5. Finally, we offer concluding remarks in Section 6.

2. Literature Review

The case studied in this article is obviously related to the well-known Vehicle Routing Problem (VRP). The classical VRP can be used in any mode of transport, but it often needs to be adapted to the specific context being considered. This is especially true for ship routing problems. Lawrence (1972) distinguishes three modes of operations: charter, tramp and liner. Ronen (1983) further classifies ship routing problem into various categories based on constraints, decision variables and other parameters. Shipping operations are varied and we will focus on a small portion here: tramping. For a complete literature review of ship routing problems, see Ronen (1983), Ronen (1993), Christiansen et al. (2004) and Christiansen et al. (2013).

Appelgren (1969) is one of the first researchers to describe and study routing and scheduling problems with a focus on tramp shipping. To solve the Vessel Routing and Scheduling Problem, he uses Dantzig-Wolfe decomposition but achieves non-integer solutions. Appelgren (1971) simplifies his first formulation and uses a branch-and-bound algorithm to find integral solutions to slightly larger shipping problems.

Ronen (1986) describes a short-term scheduling problem where a ship is loaded at a depot and discharged at one or various locations, then comes back to the depot. He also introduces a heterogeneous fleet with different cargo capacity, port-ship compatibility constraints and operating costs. He compares two solution approaches: a single-step cost minimization heuristic and a biased random insertion heuristic on small real-life instances.

Brown et al. (1987) study a vessel routing problem with time windows and speed as a decision variable. Ship speed intervals can be selected when the ship is in its repositioning leg with different fuel costs for different speeds. They use a set partionning approach to solve the problem. An algorithm generates feasible schedules a priori and then an optimization algorithm is used to solve the set-partitioning problem. Gatica and Miranda (2010) and Fagerholt et al. (2010) further explore the vessel routing and scheduling problem with speed as a decision variable. They discretize the arrival times and solve the problem as a shortest path problem. Norstad et al. (2011) and Wen et al. (2016) find that using variable speed in a ship routing and scheduling problem can have significant impacts on the operations costs of a fleet.

Instances with up to a hundred cargos can be solved in a reasonable time by the previous methods. However, instances are sometimes bigger and necessitate tailored solutions approaches to reach a good solution in reasonable time. Heuristics seem to be used often to solve the vessel routing and scheduling problem, with various mechanisms. Boffrey et al. (1979) describe a simple heuristic where ports are inserted subsequently in routes to maximize the objective function and respect the time windows. Korsvik et al. (2010) use a tabu search heuristic adapted from Cordeau et al. (2001) to solve a vessel routing and scheduling problem and add an intensification phase similar to the one developed by Brønmo et al. (2007). Korsvik et al. (2011)

define a large neighbourhood search algorithm that randomly constructs the initial solution. Then, a local search is performed and when a local optimum is reached an extended search is carried out. Malliappi et al. (2011) create a variable neighbourhood search algorithm that couples an insertion algorithm with the search algorithm.

In tramping operations, vessels generally load one cargo and travel directly to the discharge ports. This mode of transport is similar to a full truckload operation. Often, the parcel size is much smaller than the capacity of the ship, and loading multiple small parcels at one or many ports can be cost effective. Fagerholt and Christiansen (2000) explore a problem where a ship has holds of variable size and where small parcels can be picked up by the ship, and solve the problem with a set partitioning method. Jetlund and Karimi (2004) formulate a problem where a tanker has up to 20 holds, but the sequence and combination of cargo is constrained by the chemical reactivity of the transported product. They use a decomposition approach to solve the problem.

In the classical formulation, parcel sizes are fixed and known prior to the scheduling process. In real-life operations, parcel sizes are often bigger than the capacity of the biggest ship in a fleet, and must be separated in smaller cargos. Brønmo et al. (2007) create a formulation where the size of the parcel can be fractioned in smaller cargos, and transported alongside other parcels in a vessel. They use a local neighbourhood search algorithm where a large number of random-biased solutions are generated and a local search algorithm then tries to improve the solution. Andersson et al. (2011) propose a similar formulation with fractioned parcel size. Korsvik and Fagerholt (2010) couple the tabu search heuristic to solve the routing and scheduling problem to a quick search algorithm to find out the optimal parcel size to load on a ship.

In industrial operations, products are generally produced and stocked, and then transported to consumption points (other factories, warehouses, customers, etc.). Inventory Routing Problems (IRP) constitute a variant of the VRP where inventory must be managed as well as the routing and scheduling of vehicles. Christiansen and Nygreen (1998) formulate a ship routing and scheduling problem with inventory management. Instead of using regular time windows as the standard ship routing and scheduling problem, stock-level related time windows are used for each production point and each consumption point. Henning et al. (2011) formulate a problem where the transporter can choose between multiple production points to fulfill the demand.

Christiansen and Fagerholt (2002) introduce an interesting formulation of the ship routing and scheduling problem with a time window related penalty. If a ship is close to the end of a time window, a penalty is imposed to the solution, forcing good solutions to have cargos well within their time windows. In real-life applications, with highly variable sailing times, this approach should help find more robust solutions.

Finally, instead of using optimisation-based tools, Boykin and Levary (1985) develop a simulation-based approach to evaluate and analyse routing and scheduling decisions for an industrial fleet. Fagerholt et al. (2010) propose a hybrid system that combines optimisation and simulation to overcome the stochastic nature of the shipping. Miller (1987) develops a scheduling system with an algorithm that evaluates feasible moves and proposes them to a scheduler. The scheduler can see the real impact of each schedule change in terms of cost and constraints violation. In that same line of thought, Bausch et al. (1998) argue that scheduling systems must not replace schedulers but support them instead. Business constraints and parameters are very complex and user experience is necessary to make good decisions. Good scheduling systems are user-friendly and fast, and enable the user to interact with the data.

3. Problem Definition and Formulation

This section first describes the problem faced by the company in the Great Lakes and Saint Lawrence Seaway area. It then presents a mathematical formulation of the problem, followed by an explanation of how it can be adapted to other regions.

3.1. Characteristics of the Problem

As explained earlier, we are faced with a routing and scheduling problem for a fleet of dry bulk vessels in tramping operations. This problem is similar to the vehicle routing problem with pickup and delivery and the dial-a-ride problem. However, additional constraints and parameters must be considered to reflect the specific business conditions faced by the company.

The company's operations are concentrated in the Canadian Great Lakes and the Saint Lawrence Seaway System. The Great Lakes are an inland waterway linked to the Atlantic Ocean by the Saint Lawrence River and a series of locks and channels. Navigation is restricted to vessels under 225.5 meters in length, 24.4 meters in width and with a maximum draft of 8.08 meters. Due to ice, the waterway is closed to navigation after the city of Montreal from the end of December to the end of March.

The problem can be defined on a network composed of nodes and arcs. Nodes represent the actual loading and discharging ports, as well as the initial and ending locations of each vessel. Arcs are the segments between nodes of the network, characterized by a fixed time, a distance and a cost.

We gathered historical, actual and potential ports to have a list of all possible nodes. Each node is associated to many parameters and constraints, including the cost of loading and discharging, the loading and discharging rate for each type of vessel and for each type of cargo, as well as compatibility with each of the fleet's vessels. Each node is also classified within a region to help create the distance table and the weather matrix. The figure below illustrates the Great Lakes regions with each dot being the entry or exit point of a given region.

Figure 1 - Map of the regions in the problem

We gathered more than 170 potential nodes in the network, and we created a distance table between each pair of nodes. The distance table has more than 28,900 entries. Traditional distance and mapping software could not be used as they can generally only be applied for land distance calculation.

Since all nodes belong to one region, and each region is almost perfectly bounded by one or a few entry points, an approximation method is employed. By calculating the distance between each entry point in a region and each port, and by calculating the distance between the entry points of all regions, it is possible to estimate the distance for each pair of ports that are in different regions. For example, port A is in Lake Superior and has a distance of 400 nm with the sole entry point of Lake Superior. Port B is in Lake Erie and has a distance of 200 nm with the entry point between Lake Erie and Lake Huron. The distance between the two entry points is 500 nm. The total distance is the addition of each segment, thus 400 + 200 + 500 = 1100 nm.

To calculate intra-region distances (i.e. distances between two ports within the same region), an approximation method is again used. The geographical coordinates of each port are determined and the following equation is used to calculate the distance, where (x_1, y_1) and (x_2, y_2) are the port coordinates and the parameter l is calibrated for each part of the region with historical data and manually verified with a maritime distance calculator:

$$d_l = \sqrt[l]{(|y_2 - y_1|)^l - (|x_2 - x_1|)^l} \,. \tag{1}$$

Sailing times between each port are calculated by using the distance divided by the average speed of the ship in open waters. However, in inland waterways like the Great Lakes, numerous channels, locks and rivers reduce the average speed. In order to have an accurate estimate of the sailing time, a fixed time and distance are calculated for each entry point and applied to each port combination. For example, sailing through the Welland Canal at 12 knots would take around two hours if the pure distance was kept, but in reality the transit takes more than 10 hours. We thus use the following formula, where B_{ij} is the sailing time between points i and j, d_{ij} is the base distance, δ_{ij} is the entry point distance, v_{nm} is the average sailing speed in open waters for the ship depending on whether it is in ballast or laden, and τ_{ij} is the choke point fixed sailing time:

$$B_{ij} = \frac{d_{ij} - \delta_{ij}}{v_{nm}} + \tau_{ij} \,. \tag{2}$$

All sailing times between any two points depend on the selected ship for the route and the state of the ship (in ballast or laden). Contrary to the classical VRP, the vessels of the fleet are not based at a depot. The initial location of each vessel corresponds to its last port of visit, and the ending location can be chosen by the user or remain open. To keep the ending location open, a fictional port with sailing time and cost equal to 0 is created.

The fleet assigned to the Great Lakes is composed of about twenty vessels, each with different costs and attributes. Sister ships in the fleet have generally the same attributes but can have different operating costs based on their maintenance schedule and other factors. Variable operating costs include crewing cost, maintenance and lubrication, insurance, etc., and are calculated on a daily basis. If a ship operates for several days, the daily variable operating cost is simply multiplied by the number of operating days. Minimizing the utilization of the fleet will thus reduce the total costs. Fuel costs are estimated using a daily fuel consumption rate multiplied by the time spent in a specific state. Each ship has eight different fuel consumption figures for five states: sailing in ballast, sailing laden, loading, discharging and idling. Ships consume two types of fuel, a heavy fuel blend (IFO) and marine diesel (MDO). IFO can be consumed during the five states and MDO can be consumed only in the latter three.

Aside from the costs, the vessels have physical parameters and constraints. The capacity of the ship is measured by volume and weight. The volume corresponds to the usable space of the cargo holds while the weight is derived by the maximal lift of the ship at a specific draft with their deadweight scale. Ships also have compatibility constraints with ports and regions: some ships are not designed to go in coastal waters and some are too large to enter a port. Some vessels are self-unloaders, namely a ship equipped with a gravity-fed or mechanical discharging system that requires minimal or no shore-based equipment to discharge.

Each pickup and delivery request consists of one or several loading ports and one or several discharging ports. Each request has only one type of cargo with one or many grades and a target quantity is specified (parcel size) with lift range. The vessel must lift the minimum parcel size and not exceed the maximum quantity. In addition to these parameters, a compatibility indicatior with each of the vessels in the fleet is specified. Additional costs can be added to penalize or reward a ship-request combination. Compatibility between requests can also be imposed by putting a large fictional distance between the two pairs of nodes.

Each request is also associated with a time window that restricts the moment when the cargo can be picked up or delivered. The time window can be applied on the pickup, the delivery or both, and is imposed on the beginning of the service at the node. When a ship arrives at a node, it can load, discharge or wait depending on the type of node. Waiting time is incurred before the time-constrained node (i.e. the ship waits before the beginning of the service at the node on which the time window is applied). Service time can be fixed by the user or calculated with a loading or discharging rate.

The company operates ships that can lift a multitude of dry bulk cargo. Each request is specified by a cargo type with an average stowage factor in tonnes per cubic meters. Since vessels have two types of capacity, volume and weight, stowage factors determine which one of the two will be reached first. The lift is the weight of the cargo at the vessel’s maximum allowed draft or the weight of the maximum volume of cargo within the holds.

In order to maximize the lift of the fleet (assign large assets to the largest parcel size, and small assets to the smallest parcel size), a lift-related penalty was created. The penalty is calculated by taking the lift at the restricting draft (the minimum between the draft at the loading berth, during transit or at the discharging berth) and by comparing it to the maximum target tonnage. The difference is then multiplied by the freight rate. For instance, if the lift of vessel A is 22,000 tonnes and vessel’s B lift is 25,000 tonnes, and the target tonnage is 23,000 +/- 10%, a penalty of 3300 tonnes would be imposed to vessel A and one of 300 tonnes would be imposed to vessel B. If vessel C, holding 26,000 tonnes is added to the mix, a penalty of 700 tonnes would be imposed to reflect the opportunity cost of using large vessels for smaller cargos. The penalty can be adjusted with a parameter discussed in the next section.

Cleaning time is incurred before each loading operation. The length of the cleaning varies depending on the previous cargo and the next. For example, cleaning time to go from coal to grain will be greater than cleaning time from grain to coal. The cleaning operations can be done during the repositioning leg, but if the length of the leg is shorter, the vessel will have to wait until the operation is completed to enter the next node and load the cargo.

Pickup and delivery requests that comprise multiple loading or multiple discharge ports are handled by two different methods. If there is only one loading port but multiple discharging destinations, one request for each discharging port is created and the origin of each request is the original loading node. Parcels are adjusted to reflect the cargo split for each port. When there are multiple loading ports and one discharge port, the same principle is applied. In the case of multiple loading and discharging ports, the shortest distance path is created between each node and requests corresponding to each of the legs are created. This subproblem is similar to the classic traveling salesman problem and is solved with an enumeration method (since the list of candidate nodes is generally very small and constrained).

When a ship must stop for a dry-dock or a planned repair, a fictional request is created with the same loading and delivery port but with a service time equal to the length of the repair. The fictional request is set to be non-compatible with any of the other ships in the fleet.

Meteorological events and delays are also taken into account in the model. For each region and for each month, an average general delay was calculated using historical data from the company. The theoretical travel times calculated with the distance table were compared to the actual travel times in the company's records, and a delay was averaged per region and per month. The delay is then applied to the expected travel time based on the timing of the request. A weighted average of the delay from the previous month and the current month (if the request is made in the first half of the month) or from the current and the next (if the request is made in the second half of the month) is calculated and applied to the expected sailing time.

3.2. Mathematical Formulation

The problem to be solved can be summarized as follows:

1. Objective – minimize all transportation-related costs
   - Operating cost of the ship
   - Fuel cost (IFO and MDO)
   - Tolls and pilotage costs
   - Additional variable costs
   - Capacity penalty

2. Type of operation
   - Tramping
   - Dry bulk cargo
   - Pickup and delivery
   - Multiple loading and discharging ports
   - No depot or final destination
   - Sailing time depends of weather delays

3. Fleet
   - Fixed fleet of self-unloaders and bulk carriers
   - Different operating costs (including fuel consumption)
   - Different speeds
   - Different capacities (weight and volume)
   - Port-vessel and customer-vessel compatibility constraints

4. Cargos
   - Time windows applicable on loading and discharging
   - Cargo-vessel compatibility constraints
   - Cleaning time for each successive cargo
   - Each cargo has its own density
   - Each cargo must be serviced by the fleet

To formalize the problem, let N_l be a set of loading nodes {1,…, n} and N_d a set of discharging nodes {n + 1,…, 2n}. The set N is thus the union of N_l and N_d. Furthermore, V is the set of ships and N_v is the network associated to the ship v, which includes all feasible nodes for ship v, the initial location $o(v)$ and final destination $d(v)$. In addition, A_v is a set of feasible arcs for the ship, a combination of all feasible pairs of nodes in N_v. Finally, N_{lv} is set as $N_l \cap N_v$ and N_{dv} is set as $N_d \cap N_v$.

Following Christiansen et al. (2007), this problem can be formulated with three types of decision variables:

x_{ijv} = binary variable taking value 1 if node i is visited before node j by vessel v, and 0 otherwise;
t_{iv} = time when vessel v begins service at node i;
l_{iv} = total load after service at node i by vessel v.

The following parameters are used in the formulation:

C_{ijv} = total cost of servicing node i before node j by vessel v;
S_{ijv} = total time required to service node i before node j by vessel v;
Ts_{ijv} = total service time from node i to node j by vessel v;
$Tmin_{iv}$ = earliest time when node i can be serviced by vessel v;
$Tmax_{iv}$ = latest time when node i can be serviced by vessel v;
Q_i = fixed cargo quantity at node i;
V_{CAPv} = capacity of the ship v.

Our formulation is based on that of Christiansen et al. (2007):

$$min \sum_{v \in V} \sum_{(i,j) \in A_v} C_{ijv} x_{ijv} \tag{3}$$

subject to,

$$\sum_{v \in V} \sum_{j \in N_v} x_{ijv} = 1, \quad \forall\, i \in N_l\,, \tag{4}$$

$$\sum_{j \in N_{lv} \cup \{d(v)\}} x_{o(v)jv} = 1, \quad \forall\, v \in V, \tag{5}$$

$$\sum_{i \in N_v} x_{ijv} - \sum_{i \in N_v} x_{jiv} = 0, \quad \forall\, v \in V, j \in N_v \backslash \{o(v), d(v)\} \tag{6}$$

$$\sum_{j \in N_{dv} \cup \{o(v)\}} x_{id(v)v} = 1, \quad \forall\, v \in V, \tag{7}$$

$$x_{ijv}\left(t_{iv} + Ts_{ijv} - t_{jv}\right) \le 0 \qquad \forall\, v \in V, (i,j) \in\; A_v \tag{8}$$

$$Tmin_{iv} \le t_{iv} \le Tmax_{iv}, \quad \forall\, v \in V, i \in N_v \tag{9}$$

$$x_{ijv}\left(l_{iv} + Q_j + l_{jv}\right) = 0 \qquad \forall\, v \in V, (i,j) \in A_v | j \in N_{lv} \tag{10}$$

$$x_{i,n+j,v} + \left(l_{iv} - Q_j - l_{n+j,v}\right) = 0 \quad \forall v \in V, (i, n+j) \in A_v | j \in N_{lv} \tag{11}$$

$$l_{o(v)v} = 0 \quad \forall v \in V \tag{12}$$

$$\sum_{j \in N_v} Q_i x_{ijv} \le l_{iv} \le \sum_{j \in N_v} V_{CAPv} x_{ijv} \qquad \forall v \in V, i \in N_{lv} \tag{13}$$

$$0 \leq l_{n+i,v} \leq \sum_{j \in N_v} (V_{CAPv} - Q_i) x_{n+i,j,v} \quad \forall v \in V, i \in N_{lv} \tag{14}$$

$$t_{iv} + Ts_{i,n+i,v} - t_{n+i,v} \leq 0 \quad \forall v \in V, i \in N_{lv} \tag{15}$$

$$\sum_{j \in N_v} x_{ijv} - \sum_{j \in N_v} x_{j,n+i,v} = 0 \quad \forall v \in V, i \in N_{lv} \tag{16}$$

$$x_{ijv} \in \{0,1\}, \quad \forall\, v \in V, (i,j) \in A_v. \tag{17}$$

In (3), the objective of the problem is to minimize the sum of all transportation-related costs multiplied by the binary variable x_{ijv} indicating whether node *j* is serviced after node *i* by vessel *v*. Constraints (4) ensure that all nodes are serviced once. Constraints (5) and (7) ensure that the ships will depart from the origin *o(v)* and finish at destination *d(v)* while (6) are flow conservation constraints. Constraints (8) make sure that the beginning of the service at *j* is later than or equal to the beginning of service at *i* plus the service time between *i* and *j*. Constraints (9) ensure the service at node *i* by vessel *v* is within the time window $Tmin_{iv}$ and $Tmax_{iv}$. Constraints (10) and (11) link the binary flow variables with the load quantity on the ship. Constraints (12) give the initial load quantity at the origin node. Constraints (13) and (14) restrict the load quantity at the loading and discharge nodes to be positive and under the maximum capacity of the vessel. Constraints (15) impose that node *i* must be loaded before *n+i*, and finally constraints (16) make sure that the loading and discharging nodes are coupled together in the same vessel.

In our model, the cost component C_{ijv} in (3) is the sum of the voyage cost between nodes *i* and *j* and service costs at node *i*. These costs include the fuel consumed at sea and at port multiplied by the price of each type of fuel, the operational costs of the vessel, the ports dues, tolls and canal charges and the lift-derived penalty if node *i* is a loading node. The parameter Ts_{ijv} in (7) and (14) is composed of the time required to sail from *i* to *j* on vessel *v* as well as the service time at *i*. If the repositioning time between *i* and *j* is shorter than the cleaning time, the sailing time is replaced by the total time required to prepare the cargo hold between node *i* and *j* on vessel *v*. Our problem incorporates route duration constraints that restrict the length of a route for any vessel. If the vessel arrives at a node before the time windows allow it, a waiting time is incurred. In our model, the waiting time is part of the decision variables as it has a direct effect on the route duration and time windows constraints.

3.3. Adaptation to Other Regions

The model was initially developed for operations in the North American Great Lakes, but was subsequently expanded for short-sea shipping operations in other parts of the world. The short sea operations around the world remain similar from one region to another, but some variations have to be considered.

In different parts of the world, some vessels are larger and thus can transport multiple customers and shipping requests at once. In the Great Lakes, most of the requests are full ship load requests and the fleet has similar capacity. To incorporate the change and take into account the hold configuration of the vessels, we find the greatest common factor among each of the vessels' capacity in each fleet and the shipment sizes, and we divide each of the capacity and shipment sizes by the common factor. For instance, if a fleet has a 15,000 tonnes ship and a 20,000 tonnes ship, and parcel sizes of 10,000, the common factor is 5000. We then model the first ship with a capacity of 3 and the second one with a capacity of 4, and the parcel as a load of 2. We can then load two parcels in the second ship and only one in the first one.

To restrict shipment of incompatible cargos (e.g. sugar and coal) that would fit under the capacity constraints, we add multi-load restrictions to the model. The distance between two incompatible cargos loading nodes is set to a very large number, thus restricting the loading of two cargos side-by-side on the same vessel.

ECAs (Emission Control Areas) are zones around the coast where limits on sulfur emissions are imposed. In these areas, more expensive heavy fuel blends containing less sulfur or marine diesel can only be burned, thus increasing the fuel costs. To model it, we include an approximation of the distance within the ECA zone between each pair of nodes, and use the distance to derive an amount of time where the ship would burn high price fuel. To simplify the model, we assume that the ship burns diesel oil in ECAs at the same rate as it would burn heavy fuel oil.

4. Solution Approach

The formulation described in the previous section is useful to formalize the problem but it is not tractable for the instance size considered by the company. Instead, the problem is solved with a tabu search heuristic that is inspired from that of Cordeau and Laporte (2003) for the dial-a-ride problem. The dial-a-ride problem is a variant of the VRP with pickup and delivery that arises in the context of passenger transportation. In the DARP, users formulate requests for transportation from a specific origin to a specific destination. These requests are assigned to vehicles that can transport several users at the same time. However, ride time constraints are imposed to limit the time spent between the pickup and the drop off of a passenger. In addition, users must be picked up and dropped off within specified time windows. The problem addressed here can be seen as a special case of the DARP in which each request occupies the full capacity of the vehicle. Hence, the vehicle assigned to a request will travel directly from the origin to the corresponding destination without stopping at any intermediate location.

The heuristic of Cordeau and Laporte was modified in several ways to address the problem faced here. Heterogeneous vehicles were introduced, as well as initial and ending nodes for each vehicle. Compatibility matrices between vehicles and requests were created and incorporated in the heuristic to restrict the solution space. The objective function was also changed to reflect the problem's structure. We now describe the main components of this heuristic. We refer to Cordeau and Laporte (2003) for additional details regarding the original DARP heuristic.

4.1. Initial Solution

The tabu search heuristic starts either with a random solution or with a solution provided by the user. In the latter case, the algorithm inserts the requests in the routes according to a pre-existing schedule. In the former case, requests consisting of an origin node and a destination node are inserted randomly one after the other in each vessel until there is no request left. The compatibility between the vessel and the node is checked before any insertion, but the initial solution might violate other constraints (in Section 4.2 we explain how constraint violations are handled). A schedule is created by calculating the sailing time between each node, the service time for each node and applicable slack.

Slack time is created in the solution as the requests are inserted to reduce constraint violations. At first, we set the beginning of the service time at the node at its earliest possible time to minimize time window violations. Schedule times are calculated by inserting the minimum waiting time at each node. Then, to minimize route duration without increasing time window violations, the forward time slack at the node is computed. The forward time slack is the cumulative waiting time up to the node added to the difference between the end of the applicable time window and the start of service at the node. To make sure the maximum ride time constraint is not violated at a destination node, the forward time slack is instead equal to the minimum between the ride time constraint minus the ride time at the destination node and the difference between the end of the applicable time window and the start of service at the node plus the cumulative waiting time up to the node. The departure time is then updated with the forward time slack, and schedule times are recomputed. Finally, the beginning of service at each origin node is delayed to reduce ride time without affecting route duration and time window constraints.

4.2. Constraint Relaxation Mechanism

To facilitate the exploration of the solution space, we allow solutions to violate some constraints. The constraints that can be violated are the time window constraints, route duration constraints, and capacity constraints. Violations are expressed as the sum of each individual node violation in the solution, and are incorporated in the cost function to facilitate the search of good candidate solutions. We let α, β, and γ be the parameters associated with the violation of the capacity constraints, route duration constraints and time window constraints respectively. The cost function is defined as follows, where *c(s)* is the real cost of the solution, *q(s)* is the total capacity violation, *d(s)* is the total route duration violation, and *w(s)* is the total time window violation:

$$f(s) = c(s) + \alpha q(s) + \beta d(s) + \gamma w(s).$$

Each iteration, $(1+\delta)$ multiplies or divides α, β and γ in the cost function. For example, if the solution violates the capacity constraints, the value of α will be multiplied by $1+\delta$ whereas it will be divided by $1+\delta$ if the solution satisfies these constraints. This parameter is used during the search to adjust the weights of the constraint violations in the cost function. The value of δ is randomly selected at each iteration, and it is bounded by 0 and δ_{max}, a parameter the user can modify. The weight of the constraints in the objective function will increase over time if no feasible solutions are found until a point where it is reset to its original value.

4.3. Neighbourhood Definition and Evaluation

Starting from the initial solution, the search moves at each iteration from the current solution to its best neighbour. The direct neighbourhood of the current solution is evaluated in terms of solution cost and constraint violation, and the best solution that is not tabu is then selected. Solutions are defined by requests associated to routes. A move is defined by the repositioning of a request within a route or its insertion into a different one. During intra-route exchanges, a request is selected randomly and reinserted at the best position within the same route, and then the process is repeated for all requests. During inter-route exchanges, we consider moving each request to the route of a different vessel and the move that minimizes *f(s)* and that is non-tabu is performed. Each request consists of a pair of nodes with a time window applicable either on the delivery, the pickup or both. Each node is inserted within the route so as to minimize total time window violations. Moves are made according to a fixed proportion of 10-to-1 inter-route to intra-route exchanges. A binary compatibility matrix forbids the insertion of requests into non-compatible routes, thus reducing the size of the solution space and the move evaluation done at each iteration.

4.4. Tabu Tenures and Diversification

When a move is performed, the algorithm cannot perform the same move again for a given number of iterations. This is called the tabu tenure. The tabu tenure is randomly set within fixed bounds. When the tabu expires, the movement can be performed again by the algorithm. The tabu is also removed when a new best solution is found by the algorithm. The tabu tenure θ is randomly set each iteration between 0 and θ_{max}.

A diversification parameter proposed by Taillard (1993) and used by Cordeau and Laporte (2003) is also included in the algorithm to help explore alternative regions of the solution space. The parameter penalizes frequent moves, and over time forces the algorithm to consider less attractive ones. The following diversification penalty *p(s)* is added to the objective function:

$$p(s) = \zeta c(s)\sqrt{n * m}\rho_{ik} \ ,$$

where ζ is a parameter to control the intensity of the penalty, *c(s)* is the solution cost, *n* and *m* represent the number of requests and number of ships, and ρ_{ik} is the frequency of the moves that consists in inserting request *i* in the route of vehicle *k*. The parameter ζ is set randomly at each iteration between 0 and ζ_{max}.

To further explore the solutions space, solutions are destroyed and recreated if no solution is found for χ iterations. Requests are removed from each route and they are reinserted randomly. Diversification and constraint violation parameters are reset, as well as the tabu list. The previous best solution found is however kept, and used as the best solution. Only better solutions found after the destruction process will be considered as best solutions.

4.5. Termination Criterion

The algorithm stops when η iterations have been performed or if no solution is found within the first ψ iterations. The latter case indicates that the instance is too constrained or that the data are inconsistent.

4.6. Phantom Vessel

In order to relax highly constrained instances and still be able to obtain solutions, a phantom vessel was introduced in the fleet. This phantom vessel is able to service every request, but doing so yields a large increase in the solution cost. Only solutions where all cargo requests are inserted in routes are considered. Having a phantom vessel enables the algorithm to identify requests that could not be inserted in the regular fleet, and reflects cargos that might be outsourced to other carriers or chartered. The phantom vessel also helps the users identify parameter errors (time windows and capacity errors) or infeasible cargos. They can then run relaxed instances and see if cargos are still inserted in the phantom vessel. The users can run instances with or without the phantom vessel.

5. Results

The heuristic was coded in C and run on a personal computer running Windows 7 and Ubuntu 14.15 LTS, as well as on an enterprise server running Windows Server 2012. An excel-based user-interface was created to facilitate data management and instance creation. Users can manipulate constraints and parameters more rapidly and visualize results with the user interface. The decision support tool was deployed within the company and key performance indicators (KPIs) were incorporated into the tool to facilitate performance measurements.

To evaluate the impact of the decision support tool on the quality of the solutions, we gathered forward schedules and ran the algorithm with identical constraints and parameters. Forward schedules have generally a length of about 60 days, but once a year a budget schedule is created for 10 months. The heuristic was tested on the two types of schedules. Annual schedules have generally up to 500 pickup and delivery requests, while 2-month schedules have up to 150 pickups and delivery requests.

To evaluate schedules on the same cost basis, we used the reference schedule as an initial solution for the algorithm, and then ran the heuristic to create a new schedule. Costs and KPIs are expressed as a ratio of the heuristic-generated schedule over the reference schedule. To reflect a normal use of the tool, we ran the heuristic for 50,000 iterations once per reference schedule. Increasing the number of iterations and running multiple instances in parallel can potentially yield better solutions.

5.1. Parametrization

To increase the speed of the algorithm and to explore the solution space more efficiently, we carried out a series of tests to determine the best search parameters. A set of real-life instances were created with different characteristics (number of cargos, number of vessels, planning horizon and constraints) and different values of each parameter were subsequently tested. Each test ran for 100,000 iterations and was repeated between 20 and 40 times to get an average result.

Among the parameters discussed earlier, two seem to improve slightly the quality of the solution: the diversification penalty ζ and the constraint violation multiplier δ. Higher diversification penalties seem to improve the quality of the solution by penalizing more frequent moves, thus forcing the heuristic to explore alternative solutions. Increasing the maximum value of the parameter from a base value of 0.010 to 0.200 yielded improvements in solution quality by around 0.5% and up to 1.5%. On the contrary, a higher constraint violation multiplier reduced the quality of the solutions by penalizing infeasible solutions during the search, thus limiting the exploration of the solution space. Reducing this parameter from a base value of 0.50 to 0.10 yielded improvements in solution quality by 1.0% up to 2.8%. Changing the tabu tenure and changing the ratio of intra-route to inter-route exchange yielded no significant changes in our test instance. Likewise, the use of an additional destruction-reconstruction move did not yield improvement in the quality of the solution.

We also tested a different relaxation mechanism for the time window constraints. Under this mechanism, the weight of the constraint violation would still be included in the objective function but solutions violating the constraints up to a certain amount would be considered as feasible. A small relaxation of the time windows constraints (between 300 minutes and 5000 minutes) improved the solution but the improvement effect stabilized afterwards. In the instance with the smallest width of the time windows, increasing the constraint relaxation to 5000 minutes improved the solution by 2.21% while in most case, the improvement reached a plateau between 0.30% and 1.08% at 300 minutes. At high values, the parameter has little effect on the quality of the solution.

Finally, we varied the total number of iterations performed by the algorithm to better understand when solutions are found. The time required per iteration is roughly the same during the execution; a 50% decrease in the number of iterations will thus reduce the computing time by half. We ran the algorithm from 100,000 iterations to 500,000 iterations and saw that that the increase in the quality of the solution was very small (under 0.40%). We mapped the objective function versus the iteration number to find out when the algorithm should be stopped. At 50,000 iterations, the quality of the solution decreases by 0.35% for half of the base case's running time. This decrease in quality is very small. However, the high costs implicated can justify letting the algorithm run for a longer period of time.

The random parameters used in our solution approach yield different results for the same instance. In Figure 2, the objective functions over 100,000 iterations of ten identical instances are mapped. In some cases, early good candidate solutions are reached, yielding few or no improvement for the rest of the execution. The major improvements are reached within the first 5000 iterations and the objective function seems to stabilize between 40,000 and 50,000 iterations.

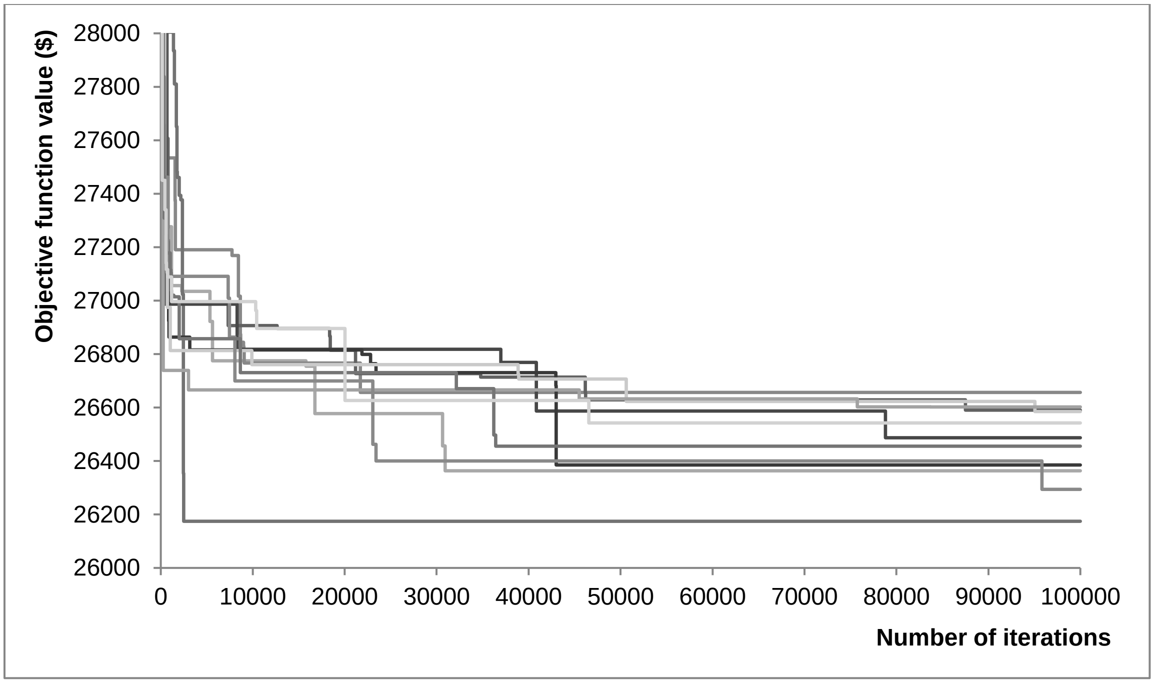


Figure 2 - Objective function per number of iteration in 10 runs of a real-life instance

After the initial experiments, we decided to set the parameter values as follow for all remaining experiments: δ_{max} = 0.2, ζ_{max} = 0.030, θ_{max} = 15, η = 100,000 and ψ = 1,000.

5.2. Assessing the Performance of the Algorithm

When tested on a one-year budget schedule, the scheduling tool improved total revenue by 0.41%, and decreased cost by 1.21% (yielding a gross benefit 3.58% higher). Operating costs were 99.13% of the original schedule, fuel costs were 99.38% in laden segments and 95.78% in ballast segments, MDO was at 99.18% and tolls were at 99.73% of the original costs. Operating costs and MDO are directly linked to the utilization of the fleet; the algorithm thus seems to have a smaller utilization than the reference schedule. Laden fuel costs are marginally the same because the same requests were serviced, but the small difference might be explained by an increased utilization of fuel efficient vessels. Thereby, tolls were almost at the same level as the original schedule. The largest improvement is the reduction of fuel cost for ballast segments. By pairing cargos differently and by trying to reduce unnecessary repositioning movements, the algorithm was able to reduce the ballast segments length by 3.08% (and fuel costs by 4.22%). The yearly schedule gives a good estimate of potential gains of the algorithm on the planning process. Gains may seem to be relatively low, but they are in fact significant for operations costing tens of millions of dollars.

We also tested the performance of the algorithm on a 2-month rolling schedule during a demand-intensive period. During those periods, almost the entirety of the fleet is scheduled. A larger fleet to schedule increases the complexity of the scheduling process, and additional gains can be achieved. The overall costs were 2.29% lower than the reference schedule, and the gross benefit was 3.85% higher. Operating costs were at 98.41%, laden fuel costs at 99.29%, ballast fuel costs at 92.12%, MDO fuel costs at 99.70% and tolls at 97.08%. Operating costs, MDO and laden fuel costs seem to follow the same pattern as in the case of the yearly schedule. They are slightly lower because of an overall lower fleet utilization or by using a different mix of vessel on different cargos. Tolls are decreased by 2.92%, because unnecessary repositioning segments going through tolls were decreased in the new schedule. Again, the biggest difference is the ballast fuel cost. The use of the algorithm seemed to reduce the overall length of the repositioning segment by 7.94% (saving 7.88% in fuel costs). Loading and discharging time decreased by 3.00% in the optimized schedule, mainly due to the fact that more self-unloading vessels were used instead of regular bulkers. Self-unloading vessels have a higher discharging rate than regular bulkers. Gains obtained on the shorter but demand-intensive scheduling horizon seem to yield better results than the overall year. The increased number of request and vessels has a non-negligible effect on the complexity of the schedule, and using the scheduling tool looks promising. Table 1 summarizes the gains achieved on four real-life instances.

Instance	Fleet costs	Fleet revenues	Gross profit
One year schedule	98.79%	100.41%	103.58%
2-month rolling horizon fall	97.71%	100.81%	103.85%

Table 1 – Gains reached with the decision support tool on two planning horizons

In a very constrained operation like this one, improvement opportunities are limited. The few avenues of improvement include allocating shipping requests more efficiently to reduce repositioning legs. The above results show that the optimization tool mainly improves the results through a reduction in the length of the repositioning legs. Several observations can be made concerning the source of the savings.

a) Sometimes, vessels are scheduled to do back and forth operations or some customers are assigned to a specific vessel. These scheduling decisions are often justified by the fact that it may decrease the chance of potential congestion or increase customer loyalty by having fewer vessels serving the customer. The optimization tool did not reproduce these results as no tangible cost or risk was associated with these scheduling decisions. Instead, all compatible vessels would service the customers according to their time windows. To reduce the risk of congestion, no overlapping time windows were selected instead of larger ones. Having small and precise time windows reduced considerably the congestion at port in recurring trades.

b) Fuel consumption is not generally taken into account by the scheduler. However, the optimizer inserts smaller legs into less fuel efficient vessels, thus increasing the time at port and reducing the fuel consumption while sailing. Fuel efficient vessels seem to be scheduled on longer legs.

c) In many cases, the algorithm was inserting cargos on vessels that did not seem to make sense for the schedulers. The divergences were sometimes due to incorrect parameters skewing the results, and sometimes due to new alternative solutions found to serve a customer. In the latter case, it was common practice to exclusively assign a vessel to large a customer. In many instances, we found that more than one vessel was used to service the customer and the overall solutions seemed to be lower. By inserting a diverse mix of requests within different vessels of the fleet, efficiency gains were achieved.

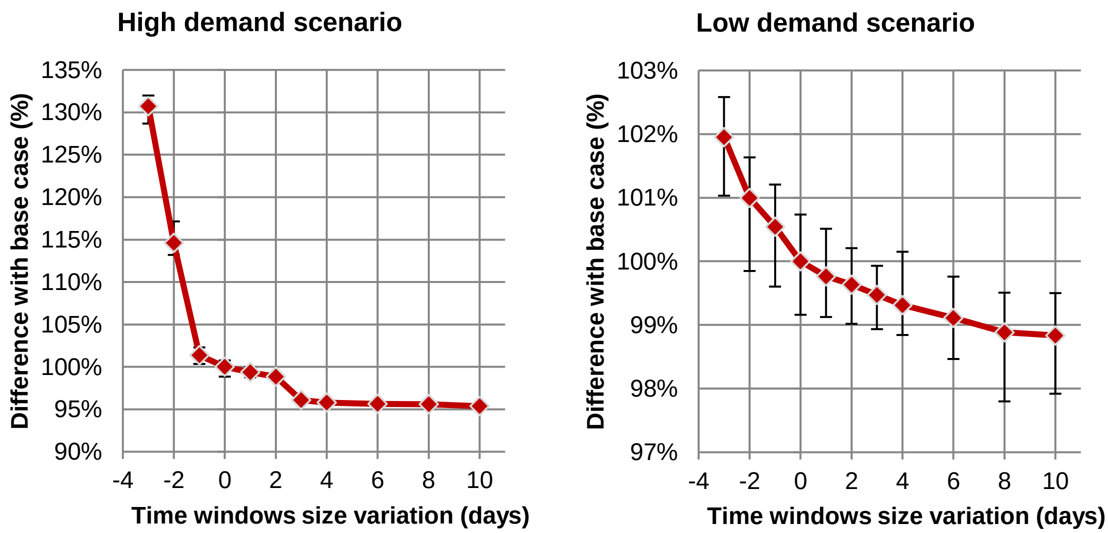
d) Often, ships navigate empty from one discharge port to the next load port. Schedulers try to minimize these repositioning segments by inserting cargos in-between, but trade imbalances between regions make it hard to find good repositioning cargos each time. Instead, the schedulers insert small requests within a long repositioning leg, even if the requests are not improving the overall repositioning position. This method was used primarily to ‘cut’ a ballast segment and spread the burden of the repositioning over multiple voyages even if the global length of the repositioning segment was still the same. Unlike the schedulers, the heuristic does not ‘cut’ long repositioning segments, but instead inserts the cargos as much as possible in the most cost-effective vessels.

e) The quality of the solutions obtained depends largely on the constraints and parameters the user inputs. By using the same parameters, costs and constraints used in the reference schedules, we were able to obtain small optimization gains for real-life instances. In some instances, by constraining too much the algorithm, we obtained very similar solutions to the ones created by the schedulers. Instead of telling exactly what the algorithm had to do, it was better to let it run with basic constraints (contractual obligations) and get alternative solutions that might yield better financial results in the end.

5.3. Sensitivity Analysis

We wanted to analyse the impact of certain parameters on the solution of the problem. After consulting the scheduling team, we found two main parameters that could greatly affect the solution: sailing speed and width of the time windows. The two parameters can be controlled to some extent by the company. Ships can increase or reduce their speed, affecting their fuel consumption but increasing the time needed to service a customer request. Likewise, contracts can be drafted to include wider or tighter time windows and can be used as a token of negotiation. Customers will prefer tighter time windows because it decreases the variability of the timing of the ship. Also, when the transportation company does not meet the time windows, contractual penalties can apply. We tested the two parameters with two demand scenarios (one with a high demand and one with a lower demand). The high-demand scenario has a fleet utilization of over 90% while the low demand scenario has a fleet utilization under 90%.

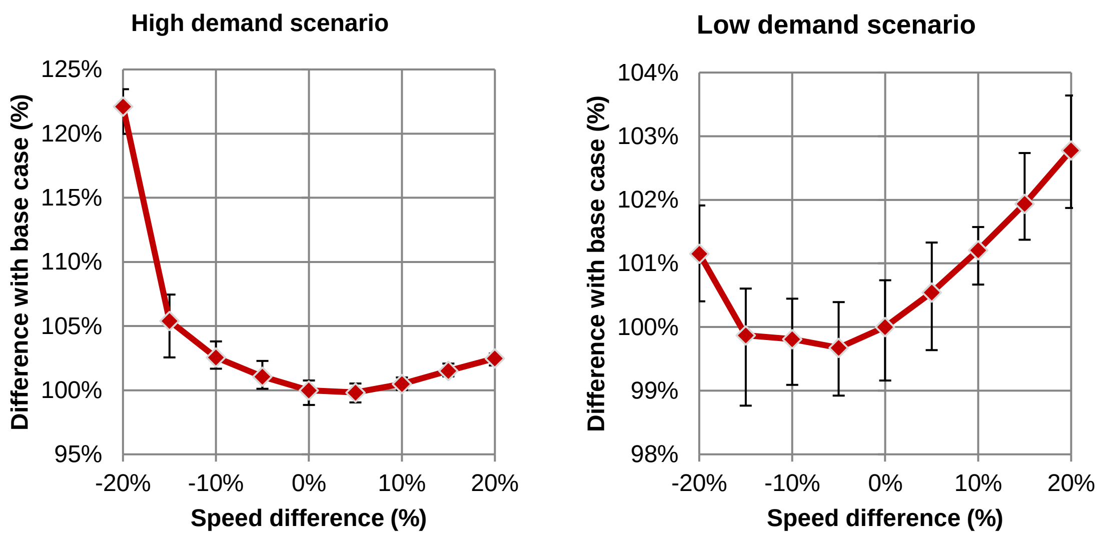
For the time windows analysis, we reduced the normal width by three days (except when the actual width was already under three days) and then increased it by 10 days. The average best solution of 20 instances was recorded at each time windows level. On the other hand, decreasing the time windows has a direct effect on the quality of the solution as the problem becomes more constrained. In the high demand scenario, the parameter's variation had a larger effect on the quality of the solution. The large increase in the value of the objective function is due to the fact that the problem becomes infeasible when the time windows are reduced by two days or more. The higher costs are due to the unscheduled requests. On the other hand, increasing the size of the time windows has a positive effect that tends to stabilize at three days. In the low demand scenario, decreasing the time windows does not yield infeasible solutions but increases the cost by up to 2% when reduced by three days.



For the speed analysis, we varied the speed of the vessel by 20% in both directions, a realistic speed range. We used the average historical speed and consumption as the baseline and increased and decreased the speed from this reference point. Since speed and fuel consumption do not have a linear relationship, and the fuel consumption patterns were not known for all the vessels in the fleet, we estimated them based on the methodology of Bialystocki and Konovessis (2016). We used the following equation to estimate the consumption for a given speed:

$$Consumption = 0.2525 * speed^2 - 1.6307 * speed \, .$$

Increasing speed has different effects on the solution depending on the scenario analyzed. In high demand scenario, slightly increasing the speed (+5%) has a small beneficial impact on the solution. Over 5%, the additional fuel consumption overcomes the added benefits of going faster (having more time to service the customers). Decreasing the speed has a negative effect on the cost solution and at 20%, the solution becomes unfeasible. On the other hand, in the low demand scenario, decreasing the speed reduces the costs. Since the instance is less constrained, the fuel savings are greater than the value of reduced time.



5.4. Impact on the Company's Operations

The manual scheduling process is an arduous task involving trial and error, and requires a lot of knowledge and experience. Schedulers need to know distances and sailing time, as well as every fleet constraint. Having a scheduling tool that holds data about ports, vessels and distance allows more individuals to get involved in the scheduling process within the company. The manual process is also time consuming for the scheduling team; schedules are created iteratively and if it is infeasible, the process must be restarted. Once a schedule is created, financial impacts of manual modifications are hard to track. The scheduling tool can be used to run schedules and track manual changes, as well as finding alternative schedules that improve financial results for the company. Adding KPIs and financial indicators to scheduling process helps schedulers understand the impact of minor changes on the financial performance of the company.

The scheduling tool was also used to find out what capacity was needed during the year. The technical department was able to have forward visibility on which vessel would be needed during the year and could therefore better plan the timing of their repairs. The annual budgeting process is long because schedules for the whole year must be created. Costs and revenues must be calculated afterwards on the expected schedules. Using the scheduling tool reduced the time needed to create schedules, and additional demand scenarios were computed to plan for variability. The additional scenarios were useful to map the fleet utilization and to help fill out gaps in the demand by contacting potential customers. During the budgeting process and throughout the year, the impact of new customers on the book of business and on schedules was evaluated with the scheduling tool. Fictional requests corresponding to the new customers were inserted in the schedule and the impact on the fleet, the costs and revenue could be assessed. Evaluating the capacity is critical in the company because each additional ship added to the fleet (from long term layup or from long term chartering) imply large fixed costs. If an additional ship is needed, its utilization must be justified by demand. The scheduling tool was thus able to show the utilization of additional asset based on different demand scenarios or new customers.

Furthermore, the scheduling tool was used to evaluate assumptions within the company. Ships of different characteristics were scheduled individually on a basket of trade, and cost and revenues could be noted. The performance of the ship could then be assessed on specific trades, and cost components could be better understood.

6. Conclusion

The scheduling tool developed (known as the ‘Optimizer’ within the company) has a direct impact on the processes of the company. It quickly creates good schedules that can be used in the daily operations of the company or for tactical planning, and enables the users to modify and create different scenarios to analyse. Besides, it allows the user to obtain realistic schedules, modify them and automatically assess the impact of a schedule change. It brought direct cost reduction versus the traditional manual process in the form of better trading patterns and optimized asset allocation. The tool was also used to analyse the fuel consumption and speed decisions within the company, and could be used in the future for fuel consumption management. In addition to that, the tool decreased the complexity of the scheduling process; additional scenarios could be analyzed by many people within the organization. No expertise is needed to schedule the fleet because most of the data and knowledge is put within the tool and is not restricted to a few individuals. The tool can also be used for pricing and customer request analysis. Traditionally, customer analysis and pricing was done in isolation to the scheduling process; the tool allows users to input fictional requests and assess the impact of the new customers on the existing trading patterns. Better pricing can thus be developed and can have favourable impacts on the business development of the organisation.

Acknowledgement

This research was supported by MITACS and by the Natural Sciences and Engineering Research Council of Canada. This support is gratefully acknowledged.

References

Andersson, H., Christiansen, M., Fagerholt, K. (2011). The Maritime Pickup and Delivery Problem with Time Windows and Split Loads. INFOR, 49, 79-91.

Appelgren, L. H. (1969). A Column Generation Algorithm for a Ship Scheduling Problem. Transportation Science, 3, 53-68.

Appelgren, L. H. (1971). Integer Programming Methods for a Vessel Scheduling Problem. Transportation Science, 5, 64-78.

Bausch, D., Brown, G., Ronen, D. (1998). Scheduling Short-term Marine Transport of Bulk Products. Maritime Policy and Management, 25, 335-348.

Bialystocki, N., Konovessis, D. (2016). On the Estimation of Ship's Fuel Consumption and Speed Curve: A Statistical Approach. Ocean Engineering and Science, 1, 157-166.

Boffey, T. B., Edmond, E. D., Hinxman, A. I., Pursglove, C. J. (1979). Two Approaches to Scheduling Container Ships with an Application to the North Atlantic Route. Journal of the Operational Research Society, 30, 413-425.

Boykin, R., Levary, R. (1985). An Interactive Decision Support System for Analyzing Ship Voyage Alternatives. Interfaces, 15, 81-84.

Brønmo, G., Christiansen, M., Nygreen, B. (2007). Ship Routing and Scheduling with Flexible Cargo Sizes. The Journal of the Operational Research Society, 58, 1167-1177.

Brønmo, G., Christiansen, M., Fagerholt, K., Nygreen, B. (2007). A Multi-start Local Search Heuristic for Ship Scheduling – A Computational Study. Computers and Operations Research, 34, 900-917.

Brown, G. G., Graves, G. W., Ronen, D. (1987). Scheduling Ocean Transportation of Crude Oil. Management Science, 33, 335-346.

Christiansen, M., Fagerholt, K. (2002). Robust Ship Scheduling with Multiple Time Windows. Naval Research Logistics, 49, 611-625.

Christiansen, M., Nygreen, B. (1998). A Method for Solving Ship Routing Problems with Inventory Constraints. Annals of Operations Research, 81, 357-378.

Christiansen, M., Fagerholt, K., Ronen, D. (2004). Ship Routing and Scheduling: Status and Perspectives. Transportation Science, 38, 1-18.

Christiansen, M., Fagerholt, K., Nygreen, B., Ronen, D. (2007). Maritime Transportation. In C. Barnhart, G. Laporte, Handbook in Operational Research and Management Science, 14, 189-282.

Christiansen, M., Fagerholt, K., Nygreen, B., Ronen, D. (2013). Ship Routing and Scheduling in the New Millennium. European Journal of Operational Research, 228, 467-483.

Cordeau, J.-F., Laporte, G. (2003). A Tabu Search Heuristic for the Static Multi-Vehicle Dial-a-Ride Problem. Transportation Research Part B, 37, 579-594.

Fagerholt, K., Christiansen, M. (2000). A Combined Ship Scheduling and Allocation Problem. The Journal of the Operational Research Society, 51, 834-842.

Fagerholt, K., Christiansen, M., Hvattum, L., Johnsen, T. A., Vabo, T. (2010). A Decision Support Methodology for Strategic Planning in Maritime Transportation. Omega, 38, 465-474.

Fagerholt, K., Laporte, G., Norstad, I. (2010). Reducing Fuel Emissions by Optimizing Speed on Shipping Routes. The Journal of the Operational Research Society, 61, 523-529.

Gatica, R. A., Miranda, P. A. (2010). Special Issue on Latin-American Research: A Time Based Discretization Approach for Ship Routing and Scheduling with Variable Speed. Networks and Spatial Economics, 11, 465-485.

Hennig, F., Nygreen, B., Furman, K. C., Song, J., Kocis, G. R. (2011). Crude Oil Tanker Routing and Scheduling. Information Systems and Operational Research, 49, 153-170.

Jetlund, A. S., Karimi, I. A. (2004). Improving the logistics of Multi-Compartment Chemical Tankers. Computers and Chemical Engineering, 28, 1267-1283.

Korsvik, J. E., Fagerholt, K. (2010). A Tabu Search Heuristic for Ship Routing and Scheduling with Flexible Cargo Quantities. Journal of Heuristics, 16, 117-137.

Korsvik, J. E., Fagerholt, K., Laporte, G. (2011). A Large Neighbourhood Search Heuristic for Ship Routing and Scheduling with Split Loads. Computers and Operations Research, 38, 474-483.

Korsvik, J., Fagerholt, K., Laporte, G. (2010). A Tabu Search Heuristic for Ship Routing and Scheduling. Journal of the Operational Research Society, 61, 594-603.

Lawrence, S. (1972). International Sea Transport: The Years Ahead. Lexington: Lexington Books.

Malliappi, F., Bennell, J. A., Potts, C. N. (2011). A Variable Neighborhood Search Heuristic for Tramp Ship Scheduling. In J. Böse, H. Hu, C. Jahn, X. Shi, R. Stahlbock, S. Voss, Computational Logistics, 273-285. Berlin: Springer-Verlag.

Miller, D. M. (1987). An Interactive, Computer-Aided Ship Scheduling System. European Journal of Operational Research, 32, 363-379.

Norstad, I., Fagerholt, K., Laporte, G. (2011). Tramp Ship Routing and Scheduling with Speed Optimization. Transportation Research Part B, 19, 853-865.

Ronen, D. (1983). Cargo Ships Routing and Scheduling: Surveys of models and problems. European Journal of Operational Research, 12, 119-126.

Ronen, D. (1986). Short-Term Scheduling of Vessels for Shipping Bulk or Semi-Bulk Commodities Originating in a Single Area. Operations Research, 34, 164-173.

Ronen, D. (1993). Ship Scheduling: The Last Decade. European Journal of Operational Research, 71, 325-333.

Taillard, É. (1993). Parallel Iterative Search Methods for Vehicle Routing Problems. Networks, 23, 661-673.

Wen, M., Ropke, S., Petersen, H. L., Larsen, R., Madsen, O. G. (2016). Full-Shipload Tramp Ship Routing and Scheduling with Variable Speeds. Computers and Operations Research, 70, 1-8.