Tramp Ship Scheduling with Speed Decisions: A Tabu Search Approach

Marc Kuo
Erasmus School of Economics, Erasmus University Rotterdam, Rotterdam, The Netherlands, kuomarc2@gmail.com

Slow steaming is anecdotally known to save costs, but for the first time, this paper quantifies this method explicitly with ship scheduling. This study analyzed the potential of incorporating speed decisions within the tramp ship scheduling problem. Two mathematical models were devised for a direct comparison. One conventional model uses a constant sailing speed, while a new model allows for variable sailing speeds. We developed a tabu search algorithm with an efficient embedded speed-optimization heuristic, which optimizes both the shipping schedules and the sailing speeds simultaneously. A computational study was performed with various simulated test cases. On 70 out of the 72 test cases, the Tramp Ship Scheduling model with Speed Decisions significantly outperformed the general Tramp Ship Scheduling model. The new model may enable a shipping company to save more than a million dollars a year on fuel consumption per ship and cut the associated carbon dioxide emissions.

Key words: ship scheduling, slow steaming, tabu search, tramp shipping, transportation

1. Introduction

Seaborne shipping is the most important mode of transport for international trade. More than 80 percent of world merchandise trade by volume is carried by sea. The total amount of seaborne trade has increased from 2.5 billion tonnes in 1970 to 6 billion tonnes in 2000, and up to 8.2 billion tonnes in 2008, as reported in UNCTAD (2010).

Carriers burn between twenty to fifty metric tonnes of bunker fuel a day, which not only lead to enormous operating costs, but also leaves a large carbon footprint. In 2007 over a billion tonnes of carbon dioxide emissions came from ships, which amounts for 3.3 percent of carbon dioxide emissions from global fuel combustion, as reported in Buhaug et al. (2009). Good planning or scheduling is not only of economical essence, but also an environmental necessity. Despite these compelling figures, the science of maritime transport is not nearly as explored in literature as compared to the other modes of transport, such as aircraft and truck (Christiansen et al. 2007).
1.1. Ship Scheduling with Speed Decisions

According to Manning (1956), a ship’s fuel consumption is directly related to the third power of the speed. This means that by slowing down 20%, a ship will reduce its fuel consumption by approximately 50%. A medium sized ship that burns 40 tonnes of bunker fuel a day may save 20 tonnes this way. With the Bunkkerworld Index fluctuating around $600 and $1,400 between 2007 and 2011, this may result in savings of up to $20,000 per day per ship. Fuel consumption is the main driver of operating costs of a ship.

Maritime transport planning problems can be classified in three overlapping levels, i.e. strategic, tactical, and operational. Among strategic problems one can think of network design problems or fleet size/mix problems. These problems have the highest impact and the longest planning horizon. Ship routing and scheduling is a tactical problem, whereas speed selection is an operational one.

We studied the problem of deciding on cruising speed on the operational level, while solving for the scheduling problem on the tactical level. To this end we propose a mathematical model for the ship scheduling problem in which sailing speed is explicitly taken into account.

Ronen (1982) has studied the effect of slow steaming, using the opportunity costs of chartering out ships as a trade-off. Brown, Graves, and Ronen (1987) addressed the problem of deciding on an optimal cruising speed inherently with creating the optimal ship schedule. However, only decisions regarding cruising speed is made on ballast (empty) legs.

1.2. Tramp Shipping

Three modes of operation in the shipping industry are distinguished in Lawrence (1972): liner, industrial, and tramp operations. Liners can be compared to bus lines, operating according to published schedules. Industrial shippers often own the cargo and the ships themselves and are vertically integrated in the company. Tramp ships are like taxis; their courses are determined by the demands of their clients. Contracts of affreightment are long-term contracts that specify the amount of cargo that need to be transported between specific ports within a specific time window. A fixed price per unit of cargo is agreed upon. Meanwhile a tramp ship tries to maximize its profits by transporting additional cargoes; filling orders that arise from the demands of the spot markets.

Looking at a literature review by Christiansen, Fagerholt, and Ronen (2004), which summarizes studies on ship routing and scheduling of the past decade, we see that little research has been done on tramp shipping. Many large manufacturing companies were vertically integrated and had their own fleets for transport. As tramp shipping companies grow, so do the size of their fleets, adding to the complexity of planning problems and the need for advanced decision support systems. This paper contributes to the research in the tramp shipping industry.

A typical tramp ship scheduling problem was described first by Appelgren (1969). Since it is unsure when new demand occurs in the spot markets, they prepared for these opportunities by
allocating scheduled cargoes to smaller ships, hence allowing for more flexibility. They introduced a so called ‘time value’ for each ship, which is a daily premium for idle time after completion of the last scheduled cargo.

The assumption of fixed cargo quantities is relaxed later by an extension in Brønmo, Christiansen, and Nygreen (2007b) and Brønmo, Nygreen, and Lysgaard (2010). They present a more realistic model using flexible cargo sizes; Cargo quantities are given in intervals. Advantages of utilizing this flexibility – mentioned in Korsvik and Fagerholt (2008) – include reducing cargo quantities in order to transport additional spot cargoes or increasing it when feasible for more revenue.

Spot charters are external ships chartered for a single voyage that can be deployed when the fleet is at capacity and cannot deliver all contracted cargoes. Tramp scheduling studies that incorporated spot charters in their models are: Fagerholt (2004), Brønmo, Christiansen, and Nygreen (2007b), Brønmo et al. (2007a), Brønmo, Nygreen, and Lysgaard (2010).

In this paper, we implemented two different models. One model solves the tramp shipping scheduling problem with cruising speed decisions and the other without. Both models were assessed using various simulated test cases and compared directly with each other on solution quality and computational effort. From the results, we concluded statistically that the new model, which incorporates cruising speed decisions, significantly outperforms the conventional model.

Parallels can be drawn between ship scheduling problems and the well-studied Pickup and Delivery Problem with Time Windows (PDPTW) (c.f. Berbeglia et al. 2007, Parragh, Doerner, and Hartl 2008)). The PDPTW is an extension of the Vehicle Routing Problem (VRP), which is known to be an NP-hard problem. This implies that finding the optimal solution becomes intractable for larger cases (Garey and Johnson 1979). Many metaheuristics have been applied to solve the VRP, including simulated annealing, deterministic annealing, genetic algorithms, ant systems, tabu search, and neural networks. Tabu search clearly stands out as one of the best algorithms for the VRP as shown in Cordeau and Laporte (2005) and is the preferred choice to solve our models.

Tabu search is very customizable; specific detail of the problem at hand can be incorporated into the design of the algorithm. Another reason why we opted for tabu search, is the speed of the algorithm. A quality solution can be obtained very quickly. At the same time, this solution can be improved upon for as long as one allows.

The rest of the paper is structured as follows. Section 2 describes the tramp ship scheduling problem and gives the mathematical formulations. We also propose a new model that accounts for cruising speed decisions. In Section 3 we describe the tabu search algorithm that solves these models. It also includes the description of the embedded speed optimization heuristic. The experimental setup and the results of the computational study are presented in Section 4. We conclude the paper in Section 5, with a few recommendations for future research.
2. The Tramp Ship Scheduling Problem

The objective of the Tramp Ship Scheduling (TSS) problem is to maximize the profits. Revenues are generated by transporting cargoes; costs are incurred by the fuel consumption of the ships and the hiring of spot charters. The cargoes are defined by a pickup and delivery point with hard time windows, a quantity and the corresponding revenue. They can be either mandatory (contracted) or optional (from spot markets). The shipping company owns a heterogeneous fleet with loading capacities and maximum sailing speeds. It is allowed to carry different cargoes on one ship. The ships may be located anywhere at the start of the planning horizon. Spot charters are allowed to be hired for a single voyage, and are assumed to be duly available.

First, we present a general TSS model derived from Brønmo, Nygreen, and Lysgaard (2010) in Section 2.1. Note however, that they modeled flexible cargo sizes. This extension is practical, but is out of the scope of this paper. Then we extended the general TSS model with speed decisions, to incorporate the model of Ronen (1982), which relates bunker fuel consumption with sailing speed. This Tramp Ship Scheduling with Speed Decisions (TSS-SD) model is described in Section 2.2. We used, where possible, the same notation as in Brønmo, Nygreen, and Lysgaard (2010) for consistency. The notation introduced below is summarized in Table 1.

We denote the fleet of available ships with \( V \), indexed by \( v \). The set of cargoes, and its associated loading ports, \( N_P \) is indexed by \( i \), which consist of the set of contracted cargoes \( N_C \) and the set of optional spot cargoes \( N_O \); \( N_P = N_C \cup N_O \). Associated with each cargo is a loading port and an unloading port, represented by a node \( i \) and \( N+i \) respectively, in which \( N \) is the total number of cargoes to be serviced during the planning horizon. Note that different nodes may represent the same port. The set of loading nodes is given by \( N_P = \{1, \ldots, N\} \) and the set of unloading nodes by \( N_D = \{N+1, \ldots, 2N\} \).

Two additional artificial nodes are used for each ship. Let \( o(v) \) denote the origin depot and \( d(v) \) the destination depot of ship \( v \). The origin depot can both be a port that is last visited or a position at sea at the beginning of the planning horizon. The artificial destination depot denotes the last planned unloading port it visits. So if ship \( v \) is unused, \( o(v) \) and \( d(v) \) represent the same location. The set of all nodes is then given by \( N = N_P \cup N_D \cup \bigcup_{v \in V} o(v) \cup \bigcup_{v \in V} d(v) \).

A ship may not be feasible to visit some ports or transport certain cargo. Let \( N_v \) represent the set of feasible ports to ship \( v \). Here, \( N_{P,v} \) denotes the set of feasible loading nodes and \( N_{D,v} \) the set of feasible unloading nodes for ship \( v \). Let a network associated with a ship \( v \) be given by \( (N_v, A_v) \) where \( A_v \) represents the set of feasible arcs, which is a subset of \( N_v \times N_v \). This set is calculated based on time and capacity constraints, and also precedence constraints of visiting a cargo’s loading port before visiting its unloading port.
servicing is given by a spot charter. Let \( \pi \) denote the profit of servicing a cargo \( i \) by a spot charter. Note that \( \pi \) may also be negative if the revenue created by servicing the cargo is lower than the cost of a spot charter.

We use the binary flow variable \( x_{ijv} \in \{0,1\}, v \in \mathcal{V}, (i,j) \in \mathcal{A}_v \) to represent the decision to let ship \( v \) sail directly from node \( i \) to node \( j \). Another decision variable that we introduce is \( s_i \in \{0,1\} \), which decides on deploying a spot charter for cargo \( i \in \mathcal{N}_P \).

The time variable \( t_{iv}, v \in \mathcal{V}, i \in \mathcal{N}_V \) denotes the time of start of service at node \( i \) with ship \( v \). This variable is derived from the present schedule of ship \( v \) and the sailing times between the

#### Table 1 Mathematical notation.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathcal{V} )</td>
<td>The fleet of available ships</td>
</tr>
<tr>
<td>( \mathcal{N}_C )</td>
<td>The set of contracted cargoes and its associated loading node</td>
</tr>
<tr>
<td>( \mathcal{N}_O )</td>
<td>The set of optional cargoes’ loading nodes</td>
</tr>
<tr>
<td>( \mathcal{N}_P )</td>
<td>The set of all loading nodes, ( \mathcal{P} = \mathcal{N}_C \cup \mathcal{N}_O )</td>
</tr>
<tr>
<td>( \mathcal{N}_D )</td>
<td>The set of all unloading nodes</td>
</tr>
<tr>
<td>( \mathcal{N} )</td>
<td>The set of all nodes, ( \mathcal{N} = \mathcal{N}_P \cup \mathcal{N}<em>D \cup \bigcup</em>{v \in \mathcal{V}} o(v) \cup u(v) d(v) )</td>
</tr>
<tr>
<td>( \mathcal{N}_v )</td>
<td>The set of feasible nodes for ship ( v \in \mathcal{V}, \mathcal{N}_v \subset \mathcal{N} )</td>
</tr>
<tr>
<td>( \mathcal{N}_P_v )</td>
<td>The set of feasible loading nodes for ship ( v \in \mathcal{V} )</td>
</tr>
<tr>
<td>( \mathcal{N}_D_v )</td>
<td>The set of feasible unloading nodes for ship ( v \in \mathcal{V} )</td>
</tr>
<tr>
<td>( \mathcal{A} )</td>
<td>The set of arcs, ( \mathcal{A} = \mathcal{N} \times \mathcal{N} )</td>
</tr>
<tr>
<td>( \mathcal{A}_v )</td>
<td>The set of feasible arcs for ship ( v \in \mathcal{V}, \mathcal{A}_v \subset \mathcal{A} )</td>
</tr>
</tbody>
</table>

The capacity of a ship \( v \) is given by \( V_{CAP_v} \). The sailing time of ship \( v \) departing from node \( i \) and arriving on node \( j \) at maximum speed is given by \( S_{ijv} \). The associated costs are denoted by \( C_{ijv} \). The time window of a cargo is given by an earliest possible time for start of service \( T_{MINi} \) and a latest possible time for start of service \( T_{MAXi} \) at node \( i \). Time required to load or unload cargo at node \( i \in \mathcal{N}_P \cup \mathcal{N}_D \) is given by \( T_i \). The quantity of cargo \( i \in \mathcal{N}_P \) is given by \( Q_i \) and its revenue for servicing is given by \( R_i \). Let \( \pi_i \) denote the profit of servicing a cargo \( i \in \mathcal{N}_P \) by a spot charter. Note that \( \pi_i \) may also be negative if the revenue created by servicing the cargo is lower than the cost of a spot charter.

We use the binary flow variable \( x_{ijv} \in \{0,1\}, v \in \mathcal{V}, (i,j) \in \mathcal{A}_v \) to represent the decision to let ship \( v \) sail directly from node \( i \) to node \( j \). Another decision variable that we introduce is \( s_i \in \{0,1\} \), which decides on deploying a spot charter for cargo \( i \in \mathcal{N}_P \).
scheduled nodes. For nodes \( i \) that are not visited by the respective ship \( v \), the value of \( t_{iv} \) is 0. Let \( l_{iv}, v \in V, i \in N_v \) represent the total load on board ship \( v \) right after departure from node \( i \).

For the TSS-SD we require some additional parameters and another decision variable. Let us introduce \( \nu_{ijv} \in \mathbb{R} \), which is a continuous decision variable for the sailing speed of ship \( v \in V \) on arc \((i,j) \in A_v \). We also need to split up the cost of sailing \( C_{ijv} \) in fixed and variable costs. For sailing each arc \( v \in V, (i,j) \in A_v \) a ship needs to pay fixed port costs, denoted by \( P_{ijv} \). Variable fuel costs are denoted by \( F_{ijv} \), which depend on the cruising speed. Here, the value of \( F_{ijv} \) is set to the fuel costs associated with a maximum cruising speed \( V_0 \) of ship \( v \in V \) sailing on arc \((i,j) \in A_v \).

### 2.1. General Tramp Ship Scheduling Model

The general TSS model can be formulated as follows:

\[
\max \left[ \sum_{i \in N_P} \sum_{v \in V} \sum_{j \in N_v} R_i x_{ijv} - \sum_{v \in V} \sum_{(i,j) \in A_v} C_{ijv} x_{ijv} + \sum_{i \in N_P} \pi_i s_i \right], \tag{1}
\]

subject to

1. \[
\sum_{v \in V} \sum_{j \in N_v} x_{ijv} + s_i = 1, \quad \forall i \in N_C, \tag{2}
\]
2. \[
\sum_{v \in V} \sum_{j \in N_v} x_{ijv} + s_i \leq 1, \quad \forall i \in N_O, \tag{3}
\]
3. \[
\sum_{j \in N_{P_v} \cup d(v)} x_{o(\nu)jv} = 1, \quad \forall v \in V, \tag{4}
\]
4. \[
\sum_{i \in N_v} x_{ijv} - \sum_{i \in N_v} x_{jiv} = 0, \quad \forall v \in V, j \in N_v \setminus \{o(v), d(v)\}, \tag{5}
\]
5. \[
\sum_{i \in N_v} x_{id(v)jv} = 1, \quad \forall v \in V, \tag{6}
\]
6. \[
x_{ijv}(t_{iv} + T_i + S_{ijv} - t_{jv}) \leq 0, \quad \forall v \in V, (i,j) \in A_v, \tag{7}
\]
7. \[
T_{MN_i} \leq t_{iv} \leq T_{MX_i}, \quad \forall v \in V, i \in N_v, \tag{8}
\]
8. \[
x_{ijv}(l_{iv} + Q_j - l_{jv}) = 0, \quad \forall v \in V, (i,j) \in A_v | j \in N_{P_v}, \tag{9}
\]
9. \[
x_{i,N+j,v}(l_{iv} - Q_j - l_{N+j,v}) = 0, \quad \forall v \in V, (i, N + j) \in A_v | j \in N_{P_v}, \tag{10}
\]
10. \[
0 \leq l_{iv} \leq V_{CAP_v}, \quad \forall v \in V, i \in N_{P_v}, \tag{11}
\]
11. \[
t_{iv} + T_i + T_{S_{N+i,v}} - t_{N+i,v} \leq 0, \quad \forall v \in V, i \in N_{P_v}, \tag{12}
\]
12. \[
\sum_{j \in N_v} x_{ijv} - \sum_{j \in N_v} x_{j,N+i,v} = 0, \quad \forall v \in V, i \in N_{P_v}, \tag{13}
\]
13. \[
x_{ijv} \in \{0, 1\}, \quad \forall v \in V, (i,j) \in A_v, \tag{14}
\]
14. \[
s_i \in \{0, 1\}, \quad \forall i \in N_C. \tag{15}
\]
In Equation (1), the first term accounts for the total revenue generated by transporting cargoes, be it contracted or optional. The second term calculates the variable operating costs, i.e. total sailing and port costs of the entire fleet. Term three calculates the profits (positive or negative) of using spot charters to transport cargoes. Note that we deliberately excluded fixed costs, such as capital and maintenance costs in this equation. These costs are hard to quantify, and incorporating them have no direct impact on the model’s solution.

Constraints (2) make sure that all contracted cargoes are served, either with an own ship or with a spot charter. Constraints (3) state that all optional cargoes are not served more than once. Constraints (4)-(6) formulate the sailing flow of a ship, and ensures that it departs from an artificial origin node and arrives at the artificial destination node.

Calculations regarding service times are done in constraints (7). Here, the time of starting a service at node $j$ cannot be earlier than the start of service at node $i$ plus the service duration and the sailing time between the nodes. Note that the inequality sign allows for waiting at node $j$, if the ship arrives before the earliest allowable time of service $T_{MNj}$. Constraints (8) make sure that (un)loading is done within the specified time window.

Load of the ships are calculated in constraints (9) and (10) after loading and unloading respectively. These loads are required to stay within a ship’s capacity at all times, which is stated by constraints (11). Ships that visit a loading node $i \in \mathcal{N}_p$ are also required to deliver the cargo at its unloading node $N+i \in \mathcal{N}_d$. This is captured by constraints (13). Furthermore, an unloading node cannot be visited before its loading node is visited, which is stated by constraints (12).

The decision variables $x_{ijv}$ and $s_i$ are binary, as captured in constraints (14)-(15).

### 2.2. Tramp Ship Scheduling Model with Speed Decisions

Following, we present an extension of the general TSS model by incorporating cruising speed decisions – the TSS-SD model. The bunker fuel consumption of the main engines of a motor ship is directly related to the third power of the speed. The relation is given by (16); even though this equation is an empirical one, it remains a useful approximation.

$$F = \left( \frac{V}{V_0} \right)^3 F_0, \quad (16)$$

where $V$ is the sailing speed, $V_0$ is the maximum speed, $F_0$ is the fuel consumption at maximum speed and $F$ is the actual fuel consumption. Note that a minimum speed of 50% is required for proper steerage.

We will make use of (16) to calculate operating costs in the objective function of our TSS-SD model:
max \left[ \sum_{i \in \mathcal{N}} \sum_{v \in \mathcal{V}} \sum_{j \in \mathcal{N}_v} R_{ij} x_{ijv} - \sum_{v \in \mathcal{V}} \sum_{(i,j) \in \mathcal{A}_v} x_{ijv} \left( \frac{\nu_{ijv}}{V_{0v}} \right)^3 F_{ijv} + P_{ijv} \right] + \sum_{i \in \mathcal{N}_p} \pi_i s_i . \quad (17)

In (17) we adjusted the second term of (1) by splitting the sailing cost parameter $C_{ijv}$ into fixed and variable costs. $P_{ijv}$ are fixed port costs that need to be payed when arc $(i, j) \in \mathcal{A}_v$ is sailed on by ship $v \in \mathcal{V}$. The variable bunker fuel costs are denoted by $F_{ijv}$ and is dependant on the sailing speed. This introduces a new decision variable for sailing speed $\nu_{ijv}, (i, j) \in \mathcal{A}_v, v \in \mathcal{V}$, which is continuous.

Sailing slower obviously results in a later arrival time at the next destination. When sailing twice as slow, the sailing time will be twice as long. We need to adjust constraints (7) and (12):

\begin{align*}
x_{ijv}(t_{iv} + T_i + S_{ijv} \left( \frac{V_{0v}}{\nu_{ijv}} \right) - t_{jv}) & \leq 0, \quad \forall v \in \mathcal{V}, (i, j) \in \mathcal{A}_v, \quad (18) \\
t_{iv} + T_i + T_{S_{i,N+i,v}} \left( \frac{V_{0v}}{\nu_{i,N+i,v}} \right) - t_{N+i,v} & \leq 0, \quad \forall v \in \mathcal{V}, i \in \mathcal{N}_{P_v}, \quad (19) \\
\nu_{ijv} & \in \mathbb{R}, \quad \forall v \in \mathcal{V}, (i, j) \in \mathcal{A}_v . \quad (20)
\end{align*}

We need to impose a speed range constraint for $\nu_{ijv}$, which also requires to be positive, but this is already implied by constraints (21).

\begin{equation}
\frac{1}{2} V_{0v} \leq \nu_{ijv} \leq V_{0v}, \quad \forall v \in \mathcal{V}, (i, j) \in \mathcal{A}_v . \quad (21)
\end{equation}

The TSS-SD can now be formulated by maximizing (17) subject to (2)-(6), (8)-(11), (13)-(15) and (18)-(21).

We compare the performance of the TSS and the TSS-SD models with respect to the profits achieved, and the computational effort. The TSS-SD has a larger solution space and an increased model complexity. The computational comparison is presented in Section 4.

3. Tramp Tabu Search

This section describes our implementation of tabu search, which solves the models formulated in Section 2. A good introduction to tabu search can be found in Gendreau (2003) and Glover and Laguna (1997). A treatise on tabu search heuristics applied to the vehicle routing problem is given by Cordeau and Laporte (2005). Tabu search is a term originally coined by Glover (1986). In short, tabu search can be described as a local search technique, that allows for non-improving moves to overcome local optima. Tabu search explores the solution space by moving from a solution $x_t$ at iteration $t$ to the best, or first improving, solution $x_{t+1}$ in a subset of the neighborhood $N(x_t)$, until
a certain stopping condition. Here, $x_{t+1}$ is not necessarily improving upon $x_t$, hence the search might go back to old solutions and keep cycling over a sequence of solutions. This is prevented using tabu lists; attributes of recent moves are recorded and potential moves that contain these attributes are not allowed for a number of iterations.

Many different tabu search implementations are possible, as tabu search allows the user to tailor the implementation to a specific problem. We experimented extensively with our algorithm on the versatile benchmark data of Li and Lim (2001). Though the benchmark data is for the PDPTW problem, it remains a useful testing ground for our algorithm, as no benchmark data are available for the TSS problem. When deciding on algorithmic designs, we not only considered solution quality, but also regarded computational effort to maintain practicality. Below, we describe the resulting tabu search algorithm tailored for the Tramp Ship Scheduling problem, which we will call Tramp Tabu Search (TrampTS).

3.1. Neighborhood Structure

We defined a simple neighborhood move called the ‘single paired insertion move’. It selects an order, be it an order that is already scheduled on a ship, an order scheduled for the spot charter, or a neglected optional order on the dummy ship, and inserts it into either one of the former. It examines all possible selections and insertions of orders, considering both the respective loading and unloading nodes at the same time. Incremental changes in the objective value are calculated exactly for each move. Note that it is also possible to insert the order back into the same ship, allowing the paired insertion move to improve within routes as well. During our search, we do not allow moves that violate capacity or time constraints. Nor do we allow to neglect contracted orders.

Additionally, we make use of elite candidate lists to speed up the search process. All moves that improve upon the current solution are recorded. The best move is applied and removed from the list. Remaining moves that are not applicable anymore on the new solution, will be removed as well. During the subsequent iterations, the moves on the candidate list are reevaluated, from which the best one is chosen, until no improving moves remain on the list. Experiments on benchmark data, showed that our algorithm ran twice as fast by deploying elite candidate lists, without a trade-off in solution quality.

3.2. Tabu List

The tabu list records every move that is performed for a number of iterations, depending on the tenure value. Two attributes are recorded, i.e. the ID of the order $i$ that is being moved, and the ship $v$ that it is inserted in. For subsequent iterations, the move of inserting order $i$ onto the route of ship $v$ is tabu, irrespective of its origin. A higher the tenure value causes the search to be more thorough, but more computational effort is required.
Whenever a new best solution is found through a move that is on the tabu list, we override the tabu list by means of the aspiration criteria. Without such an aspiration criteria, we risk that better solutions are not found.

We set the maximum iterations to 1000, but we also put a stopping condition in place. This stops the search when for a consecutive number of iterations no new best solution has been found. This number we define as double the tenure value, since the search starts cycling if it cannot seem to escape from a local optima using the present length of the tabu list. However, we inhibit a minimum of 10 iterations in case the tenure value drops below 5 and stops prematurely.

### 3.3. Initial Feasible Solution

We generate initial feasible solutions using a simple greedy heuristic, which inserts orders one by one. They are assigned to the best ship available, and the loading and unloading nodes scheduled in the best possible position. This is determined by minimizing additional costs. When we encounter contracted orders that cannot fit feasibly anymore in our available fleet, we assign the order to a spot charter. However, if the respective order concerns an optional cargo, we simply neglect it, thus putting it on a dummy ship. Since we assume that an external spot charter is always available, it implies that a feasible initial solution always exists.

The sequence of insertion is the main determinant for the resulting initial solution. In our case, we randomize the order of insertion, taking into account, that contracted orders will be scheduled prior to the optional ones. To account for the perils of random initialization, we run the TrampTS multiple times, from which we can select the best solution. This makes our TrampTS similar to a multi-start local search heuristic (c.f. Brønmo et al. 2007a), which in turn diversifies our search, as each starting point would represent a different region in the solution space.

When increasing the number of runs, i.e. the number of different randomly generated initial solutions from which each independent search proceeds, we increase our eventual solution quality. Obviously, the CPU time is linearly related to the number of runs performed. Benchmark experiments on 100-order problems from Li and Lim (2001) showed that by using only 5 runs we obtain a solution on average within 20 seconds, but the solution we get is on average more than 6% away from the best known optimal solution. When we set the number of runs to 10, we improve the solution quality with 1.5%, but we double the CPU time. The results, when ranging the number of runs from 5 to 200, are plotted in Figure 1. The solution quality gets better, though in a decreasing fashion. It is up to the user to decide for how long the algorithm needs to search.

In addition to random initialization, we also deploy a deterministic circled sweep heuristic. This heuristic starts by inserting the pickup order located closest to the depot, followed by the second closest, until all orders are inserted.
When we set the algorithm to run 5 times, this implies that one initial solution is generated using the circled sweep heuristic, and four others randomly. From every initial solution, the algorithm runs until a stopping condition is met. After the TrampTS is finished running, the best solution from all runs is chosen.

3.4. Speed Optimization

One way to optimize sailing speeds is to optimize it after solving the scheduling problem. This simple approach has a major drawback that you do not search the full search space. Since slow steaming directly affects the feasibility region of scheduling decisions, and vice versa, we need to optimize both decision variables simultaneously.

The consequences of arriving later at a certain node is two-fold. Not only may this thwart us from scheduling additional cargo, but it also results in less slow steaming opportunities on later legs. For example, consider a situation in which going at half speed to load the cargo at port A forces you to go at full speed to deliver it just in time at port B. Due to the nature of the relation given in Equation (16), more savings can be realized if we arrive at port A earlier, so we can exploit the leg to port B.

In the paper of Fagerholt (2001), which studies the ship scheduling problem with soft time-windows, time is made discrete. They made duplicate dummy nodes with each node representing a different arrival time. The arrival times on a route are then optimized, which in effect decides the cruising speed. Using this approach, the search space would explode when we increase the granularity of the speed variable, resulting in intractability for mid-sized problem. We present a fast heuristic that is embedded in the tabu search without exploding the search space.
3.4.1. Embedded Speed Optimization Heuristic. Here we describe the speed optimization heuristic that we devised and embedded in the TrampTS algorithm. The subproblem of optimizing sailing speeds of a single route is an LP-problem on itself. To address an LP-solver to solve this subproblem every time would be computationally intractable. The speed optimization heuristic must be fast, since every feasible move requires the heuristic to re-optimize up to two ship schedules. We devised a quick iterative heuristic, which lowers the speed on the most expensive leg of a specific ship schedule with $\delta_s$, and updates the cost of the leg. We select the most expensive leg, because slow steaming on this leg will result in the most significant savings.

First we set all speeds on the selected route to the maximum. When we select $q$, which is the leg with the highest fuel consumption, we check whether on this leg it is feasible to lower the speed with $\delta_s$. When lowering the speed on leg $q$ with $\delta_s$ results in arriving too late at any of the subsequent nodes, the speed on that leg is infeasible to lower. The next most expensive leg that is feasible to lower is selected instead. When no more slow steaming opportunities remain the heuristic stops. Note also that the sailing speed cannot go lower than half of the maximum speed.

The pseudo code for optimizing a single route with $N$ legs is given in the box below. Note that CPNM stands for the cost per nautical mile at maximum speed. This value is dependant on the fuel price, the maximum speed and the daily fuel consumption of the respective ship. Both the route from which the order is taken, and the route in which the order is inserted, are optimized before the assessment of the move.

```plaintext
for i = 1 to N do
    leg[i] ← maxSpeed
    fuelCons[i] ← distance[i] * CPNM
end for
feasibleToLower ← true
while feasibleToLower do
    q ← Select most expensive leg that is feasible to lower with $\delta_s$
    if q = null then
        feasibleToLower ← false
    else
        speed ← leg[q].speed
        leg[q].speed ← speed - $\delta_s$
        fuelCons[q] ← distance[q] * CPNM * speedFactor(speed) {Eq. 16}
    end if
end while
```
4. Quantitative effects of Speed Decisions

We compare the performance of the regular TSS model with the TSS-SD model on a few simulated cases described in Section 4.1. The two performance measures are solution quality and computation time. The quantitative effects of incorporating speed decisions are reported and discussed in Section 4.2. We used an Intel Core2 Duo, 2.1Ghz, 2GB Ram system to run the experiments.

4.1. Experimental setup

Below, we describe how we estimated the parameters of the model in an attempt to simulate reality. We opt to match realistic numbers, since the significance of slow steaming depends on the relative proportions of (sailing) costs and revenues. Furthermore, it would give us some practical insight in the range of the potential increase in profits.

4.1.1. Model parameters estimation. The dry-bulk trip charter rates of iron ore from Brazil to China in 2007 ranged from $35.50 per tonne at the beginning of the year to $86.35 per tonne at the end of the year. In May 2008 it was $101.80 per tonne and in June 2009 it was back down to $43.45 per tonne (UNCTAD 2010). For our experiments we simplify this to range from $50 per tonne to $100 per tonne, which represents the range of the revenue rates created from transporting cargo.

The global average vessel size of bulk carriers were about 80,000 tonnes at the end of 2006 and about 89,000 tonnes at the end of 2009. For our model we range the capacity of the ships between 25,000 and 150,000 tonnes. We randomize the weight of each order between 5,000 tonnes and 120,000 tonnes.

The time it requires to load or unload an order at a port is dependant on the speed of the crane and the quantity of the cargo. Dry bulk cranes’ speed ranges from 1,000 tonnes per hour to 10,000 tonnes per hour, which is the range in which we randomize. Additional to the actual loading and unloading, a ship that is visiting a port requires some fixed time to enter the port, setup the crane and to exit the port. We estimate the total fixed time per port visit to be 3 hours.

We assume a bunker fuel price of $1,000 per tonne. The fuel consumption increases with the size of the ship, ranging from 20 to 45 tonnes a day. The maximum sailing speed remains constant around 14 knots, which is about 26 km/h, with one knot equal to 1.852 km/h, i.e. one nautical mile per hour. Our fleet is composed of ships listed in Table 2.

A single voyage between Brazil and China takes on average about 37 days compared with a single voyage from Australia to China, which takes about 15 days; the trip from Beijing to Hamburg lasts for 30 days (UNCTAD 2010). The sailing distance between Brazil and China is approximately 10,000 nautical miles. These numbers give us an idea of the distances that the bulk carriers sail.
Table 2 Features of a few bulk carriers.

<table>
<thead>
<tr>
<th>Ship Nr</th>
<th>Capacity (D.W.T.)</th>
<th>Speed (knots)</th>
<th>Fuel consumption (Tonnes/day)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>25,000</td>
<td>13</td>
<td>21</td>
</tr>
<tr>
<td>2</td>
<td>30,000</td>
<td>13</td>
<td>23</td>
</tr>
<tr>
<td>3</td>
<td>54,500</td>
<td>14</td>
<td>30</td>
</tr>
<tr>
<td>4</td>
<td>70,000</td>
<td>14</td>
<td>33</td>
</tr>
<tr>
<td>5</td>
<td>150,000</td>
<td>14</td>
<td>45</td>
</tr>
</tbody>
</table>

Our model has a maximum size of 10,000 nautical miles, and distances between a pickup and a delivery node has a minimum of 3,000 nautical miles.

The sizes of the test case studies performed in Korsvik and Fagerholt (2008), Brønmo, Christiansen, and Nygreen (2007b), Brønmo, Nygreen, and Lysgaard (2010) range from 14 to 75 contracted orders and 4 to 20 optional orders. The fleet sizes range from 4 ships to 19, and the planning horizon ranged from 15 to 120 days. These test cases were derived from a real shipping companies, hence are practically sized. From all cargoes, on average, about 25% are spot cargoes. For our test cases, we used similar sizes and proportions. We randomized the size of the time windows, while taking into account the feasibility.

The small test cases have 4 ships and 20 orders. The medium sized test cases use a fleet of 8 ships and have 40 orders. The large test cases use a fleet of 15 ships with 100 orders. We consider a tight, normal, and wide planning horizon, i.e. 40, 100 and 200 days.

In total we have 9 different types of test cases, each of which we generate 8 instances, totalling 72 randomly generated test case using the parameters described above. We depict each data set using a prefix in their name, e.g. mw3 denotes the third out of the eight test cases that are generated with 40 orders and a planning horizon of 200 days.

4.1.2. The paired-observation t test. In order to compare the results of both models and draw conclusions that are statistically valid, we make use of the paired-observation t-test, since pairing gets at the difference between two populations more directly. If the data is paired in some way, the difference between the two conveys more information about the difference between the two models (Aczel 2002, Ch. 8.2). Our data is paired, because the models are tested on the same test cases using the same fleets and the same random seeds when generating initial solutions. This does not mean that the actual initial schedule is the same, since this is dependant on which model you use. Beside randomness in generating insertion sequences for the initial solution, the TrampTS algorithm remains deterministic.

The test statistic for the paired-observation t-test in (Aczel 2002, Ch. 8.2) is given by:

\[ t = \frac{\bar{D} - \mu_{D_0}}{s_D / \sqrt{n}} \] (22)
where \( \bar{D} \) is the average difference between the two populations, \( s_D \) is the standard deviation of the differences, \( n \) is the number of test observations, and \( \mu_D \) is the mean difference under the null hypothesis (which is 0 in our case).

### 4.2. Computational results

We ran both models on the 72 test cases, using 100 runs per experiment. For these experiments we used a fixed tenure of 30. For the TSS-SD model, we choose a \( \delta_s \) of 0.1, but we also performed the experiments using a \( \delta_s \) of 0.01. We report the best profits, the mean profits over 100 runs, the standard deviation and the total CPU time in seconds.

### 4.2.1. TSS-SD dominates TSS

The results of the small test cases are summarized in Table 3. At first glance, we see that the TSS-SD clearly dominates the TSS when it comes to profits. The TSS-SD model (\( \delta_s = 0.1 \)) outperforms the TSS model on all test cases with an extremely high significance at an alpha of 0.01% (with only one exception of test case \( st2 \) at an alpha of 1%, which is still highly significant). On these test cases, the TSS-SD (\( \delta_s = 0.1 \)) model achieves $7.86 million
Table 4  Comparative computational results for 24 medium sized test cases, using 8 ships and 40 orders.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Best Mean Stdv CPU</th>
<th>Best Mean Stdv CPU</th>
<th>Best Mean Stdv CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TSS model</td>
<td>TSS-SD model ($\delta_s = 0.1$)</td>
<td>TSS-SD model ($\delta_s = 0.01$)</td>
</tr>
<tr>
<td></td>
<td>100.34 91.21 5.2 131</td>
<td>111.05 99.55 4.51 329</td>
<td>111.09 99.36 4.26 2175</td>
</tr>
<tr>
<td></td>
<td>82.23 70.4 5.64 164</td>
<td>96.87 80.21$a$ 6.87 698</td>
<td>93.56 80.3 6.62 5216</td>
</tr>
<tr>
<td></td>
<td>83.5 66.27 7.15 160</td>
<td>88.12 67.78 8.29 574</td>
<td>83.49 67.43 7.7 4414</td>
</tr>
<tr>
<td></td>
<td>88.68 76.83 6.58 150</td>
<td>100.37 79.14$a$ 8.08 449</td>
<td>99.91 78.2 8.06 2918</td>
</tr>
<tr>
<td></td>
<td>92.38 70.73 7.22 134</td>
<td>102.96 79.31$a$ 7.8 404</td>
<td>97.92 80.18 8.06 2802</td>
</tr>
<tr>
<td></td>
<td>79.64 73.59 3.83 132</td>
<td>89.75 80.14$a$ 5.76 519</td>
<td>92.85 80.3 5.46 3992</td>
</tr>
<tr>
<td></td>
<td>69.77 44.37 9.54 114</td>
<td>69.75 46.40$c$ 9.12 430</td>
<td>67.07 46.99 9.56 3064</td>
</tr>
<tr>
<td></td>
<td>93.86 84.62 5.59 119</td>
<td>99.28 82.05$a$ 7.85 393</td>
<td>100.27 82.71 7.68 2575</td>
</tr>
<tr>
<td>Avg mt</td>
<td>86.30 72.25 6.34 138</td>
<td>94.66 76.82 7.29 475</td>
<td>93.27 76.93 7.07 3369</td>
</tr>
<tr>
<td></td>
<td>78.63 69.84 6.72 209</td>
<td>99.09 77.58$a$ 7.64 2006</td>
<td>99.09 78.37 8.33 16896</td>
</tr>
<tr>
<td></td>
<td>61.15 56.66 6.04 186</td>
<td>82.16 73.37$a$ 7.26 2178</td>
<td>82.17 73.36 7.27 18586</td>
</tr>
<tr>
<td></td>
<td>84.25 77.82 5.73 191</td>
<td>104.61 88.84$a$ 7.69 1574</td>
<td>103.71 88.77 7.07 13679</td>
</tr>
<tr>
<td></td>
<td>92.8 84.32 5.46 173</td>
<td>113.24 94.66$a$ 7.38 1192</td>
<td>111.31 96.07 8.04 11045</td>
</tr>
<tr>
<td></td>
<td>70.68 66.77 3.13 178</td>
<td>90.26 81.53$a$ 6.55 1433</td>
<td>89.79 81.82 6.59 13180</td>
</tr>
<tr>
<td></td>
<td>68.84 61.34 4.03 146</td>
<td>89.01 76.08$^a$ 5.49 1061</td>
<td>88.49 75.39 5.48 9223</td>
</tr>
<tr>
<td></td>
<td>87.14 81.95 4.06 201</td>
<td>106.86 92.68$a$ 7.05 1537</td>
<td>106.93 92.62 7.08 13889</td>
</tr>
<tr>
<td>Avg mn</td>
<td>75.88 68.34 5.46 178</td>
<td>95.56 80.59 7.14 1493</td>
<td>95.09 80.71 7.28 13063</td>
</tr>
<tr>
<td></td>
<td>91.49 83.44 6.39 253</td>
<td>114.3 94.14 $^a$ 10.44 3362</td>
<td>114.31 93.97 10.27 32770</td>
</tr>
<tr>
<td></td>
<td>86.37 83.25 3.47 172</td>
<td>109.82 103.73$^a$ 6.42 2459</td>
<td>109.76 103.37 6.45 22456</td>
</tr>
<tr>
<td></td>
<td>89.99 64.6 14.17 111</td>
<td>117.55 101.50$^a$ 10.1 1238</td>
<td>117.56 101.69 9.15 10915</td>
</tr>
<tr>
<td></td>
<td>82.39 72.9 6.48 153</td>
<td>102.63 88.87$^a$ 8.95 1379</td>
<td>107.69 89.61 8.91 12254</td>
</tr>
<tr>
<td></td>
<td>94.54 55.48 15.94 129</td>
<td>106.19 76.12$^a$ 13.15 1165</td>
<td>106.2 75.37 12.94 9926</td>
</tr>
<tr>
<td></td>
<td>82.32 79.3 2.49 235</td>
<td>104.18 98.02$^a$ 5.72 2651</td>
<td>104.16 97.91 5.96 23531</td>
</tr>
<tr>
<td></td>
<td>68.88 55.77 8.78 169</td>
<td>83.31 67.25$^a$ 7.57 1365</td>
<td>83.15 66.61 7.25 12227</td>
</tr>
<tr>
<td></td>
<td>88.49 69.37 9.34 136</td>
<td>99.63 75.02$^a$ 10.22 1104</td>
<td>99.64 75.07 9.89 9562</td>
</tr>
<tr>
<td>Avg mw</td>
<td>85.56 70.51 8.38 170</td>
<td>104.7 88.08 9.07 1840</td>
<td>105.31 87.95 8.85 16705</td>
</tr>
<tr>
<td>Total Avg</td>
<td>82.58 70.37 6.73 162</td>
<td>98.31 81.83 7.83 1269</td>
<td>97.89 81.86 7.73 11046</td>
</tr>
</tbody>
</table>

The values are obtained through 100 runs per test case. The TSS-SD ($\delta_s = 0.1$) model outperforms the TSS model significantly with only one exception, i.e. mt3 with a 10% alpha. For test case mt7 this is a 5% alpha significance and mt3 a 1% alpha significance. All other test cases are dominated by the TSS-SD with a 0.01% alpha. Profits are reported in millions of USD. CPU time is given in seconds for 100 runs.

$a$ Outperforms with an alpha of 0.01%.

$b$ Outperforms with an alpha of 1%.

$c$ Outperforms with an alpha of 5%.

of savings on average. However, it requires almost 7 times more computational effort. In absolute terms, this is only 1 second more on average per run, hence negligible given the tremendous savings potential.

The experimental results of the medium sized test cases are reported in Table 4. Likewise, the TSS-SD ($\delta_s = 0.1$) model dominates the TSS model, with the exception of test cases mt3 and mt8. The savings that are realized on medium sized test cases are higher, and amount on average $11.5 million. The computational efforts are also about 7 times higher, but in this case this is on average about 11 seconds per run.

In Table 5 we report the results for the large test cases. Again, on all these test cases, the TSS-SD model outperforms the TSS model with high significance. On average, the savings amount to $17.8 million. Considering best solutions found, the difference can climb up to $47 million, as was the case in lw2. It requires 3 times as much computational effort, which is 14 seconds more per run in absolute terms.
The analysis of the results are summarized in Table 6. The bigger the test case, the higher the potential savings from slow steaming. Additional computational effort increases too, but remains overshadowed by the potential savings. For instance, for large test cases, the TSS-SD model requires 21 seconds per run. In 10 minutes, the TSS-SD model is still able to create 30 solutions from which the best schedule may be implemented. Recall Figure 1 which shows that after 30 runs, the slope of improving solution quality decreases quickly. We can also note from Table 6 that wider time windows allow for more slow steaming opportunities.

### 4.2.2. A smaller step size is unnecessary.

The $\delta_s$ denotes the step size of the speed heuristic. Lowering the step size allows the model to search more granular. Not only may this result in more accurate optimal speeds on the legs, the resulting schedule might be different. On all test cases, we ran the model again using a $\delta_s$ of 0.01 instead of 0.1. The results are reported in the right columns of Tables 3, 4 and 5.

We see that the model with a $\delta_s$ of 0.01 does not perform significantly better than the model with a $\delta_s$ of 0.1. In fact, sometimes it returns a worse solution. The computational time also increases.
Table 6  Summary of experimental results – average differences.

<table>
<thead>
<tr>
<th>Test cases</th>
<th>Savings (factor)</th>
<th>CPU (factor)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>tight</td>
<td>4.58 (6.37)</td>
</tr>
<tr>
<td>small</td>
<td>normal</td>
<td>7.64 (1.66)</td>
</tr>
<tr>
<td></td>
<td>wide</td>
<td>11.35 (1.15)</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>7.86 (1.87)</td>
</tr>
<tr>
<td>medium</td>
<td>tight</td>
<td>4.57 (1.06)</td>
</tr>
<tr>
<td></td>
<td>normal</td>
<td>12.25 (1.18)</td>
</tr>
<tr>
<td></td>
<td>wide</td>
<td>17.57 (1.25)</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>11.46 (1.16)</td>
</tr>
<tr>
<td>large</td>
<td>tight</td>
<td>9.92 (n/a)</td>
</tr>
<tr>
<td></td>
<td>normal</td>
<td>17.83 (n/a)</td>
</tr>
<tr>
<td></td>
<td>wide</td>
<td>25.75 (3.58)</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>17.84 (n/a)</td>
</tr>
</tbody>
</table>

Results reported in this table report the average differences between the TSS-SD (δ_s = 0.1) model and the TSS model. Values are derived from Tables 3, 4 and 5. Savings are reported in millions USD and represent the average amount of savings that the TSS-SD model realizes as compared to the TSS model. CPU time is given in seconds for 100 runs.

* The TSS model makes a negative profit, whereas the TSS-SD model makes a positive one, hence no difference factor can be calculated.

dramatically. It is therefore not advised to opt for an even smaller step size. A δ_s of 0.1 is small enough, and remains very fast.

5. Conclusion

We developed two distinct models to quantify the consequences of incorporating cruising speed decisions in the tramp ship scheduling problem. One model does not optimize cruising speeds and optimizes the schedules using the conventional way of assigning a fixed speed. The other model simultaneously optimizes the sailing routes and the cruising speeds on each leg, with an accuracy to a tenth of a knot. The additional computation time that is required for the extended model is overshadowed by the potential savings. This is due to the efficient heuristic that we developed and embedded into the tabu search algorithm, making no compromises in the search space.

Our new model clearly dominates the conventional model; incorporating speed decisions may result in multi-million dollar savings in fuel costs per ship per year, without major additional computational effort. Carbon dioxide emissions are therefore substantially reduced.

These models apply customized tabu search algorithms for quick and good solutions, which makes the tool practical for the dynamic tramp industry. The tool enables the user to quickly generate new schedules whenever a new order occurs on the spot market. Our algorithm can deliver a solution within seconds, but also gives the user the flexibility to run the algorithm for as long as his situation permits, consequently maximizing his profits.

Since research in tramp shipping is still in its infancy, no benchmark test cases have yet been published in literature. We generated our own benchmark test cases for our research, in which we
attempt to simulate a wide range of real-life scenarios. The test cases we generated may deviate from real-life situations, resulting in a different proportion of savings when compared to those reported in this paper. Despite the strong case we make with our computational study, further research is required – utilizing cases with real shipping companies – in order to span the bridge between academics and practice.

Even though we studied the tramp ship scheduling problem, our model with embedded speed optimization heuristic is also applicable to other ship scheduling problems, such as industrial shipping and liner shipping.

Acknowledgments
We would like to express our thanks to Robin Schilham and Uzay Kaymak for their stimulating discussions and edits by Suzanne Ma for improving the readability of this paper.

References


