Theory and Methodology

Ship scheduling with soft time windows: An optimisation based approach

Kjetil Fagerholt *

Department of Marine Systems Design, Norwegian University of Science and Technology, Otto Nielsens vei 10, 7034 Trondheim, Norway
Received 14 January 1999; accepted 6 March 2000

Abstract

This paper considers a real ship scheduling problem that can be considered as a multi-ship pickup and delivery problem with soft time windows (m-PDPSTW). The motivation for introducing soft time windows instead of hard is that by allowing controlled time window violations for some customers, it may be possible to obtain better schedules and significant reductions in the transportation costs. To control the time window violations, inconvenience costs for servicing customers outside their time windows are imposed. An optimisation based approach based on a set partitioning formulation is proposed to solve the problem. First, all (or a number of promising) feasible routes are enumerated. Second, the various possible schedules of each route are computed as well as the corresponding operating and inconvenience costs. Finally, the schedules are given as input to a set partitioning problem. The solution method also determines the optimal speeds for the ships on the various sailing legs. The computational results show that the proposed approach works on the real ship scheduling problem. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Scheduling; Transportation; Ships; Soft time windows

1. Introduction

Ocean shipping is the major transportation mode of international trade. Each year, more than 4 billion tons of cargo are lifted by more than 80,000 ships. A ship involves a major capital investment (usually millions of US dollars), and the daily operating costs of a ship can be thousands of dollars. Significant improvements of the economic performance of a fleet of ships may therefore be expected from proper routing and scheduling.

While there is an abundance of references in the OR/MS literature in vehicle routing and scheduling, see the survey by Laporte and Osman (1995), the number of references in routing and scheduling of ships is relatively small. A discussion of the reasons for the low attention on ship routing and scheduling is given in Ronen (1983), while a review summarising trends and published research in ship scheduling and related areas for the period 1983–1993 is presented in Ronen (1993).

*Tel.: +47-73-59-56-80; fax: +47-73-59-55-75. 
E-mail address: kjetil.fagerholt@marintek.sintef.no (K. Fagerholt).
In scheduling problems with hard time windows, in addition to a fixed demand, each customer has a time window within which service must begin. By transforming the hard time windows into soft time windows, the service is allowed to begin also outside the time windows at an appropriate penalty. This penalty can represent the cost of lost sales, goodwill, etc. due to the customer inconvenience for not meeting the time window, and is therefore often called an *inconvenience cost*.

This paper deals with a ship scheduling problem that can be considered as a multi-ship pickup and delivery problem with soft time windows (m-PDPSTW). The motivation for scheduling with soft time windows is that by allowing time window violations for some customers, it may be possible to obtain better schedules and significant reductions in the transportation costs. Soft time windows also reflect situations found in practice better than hard time windows and can be used to find a good trade-off between transportation costs and service quality to the customers, see Fagerholt (2000). It should also be emphasised that a scheduling model that incorporates soft time windows is more general and includes the hard time window case as well. If 100% on-time service is required, raising appropriately the inconvenience cost should encourage a solution with no time window violations.

To author's knowledge, there exists no work on ship scheduling with soft time windows in the literature. This paper presents a solution approach for this problem based on a set partitioning formulation. All (or a number of promising) candidate schedules are generated a priori together with their operating and inconvenience costs. Then, the scheduling problem is formulated and solved as a set partitioning problem, where the columns represent the candidate schedules generated a priori. The set partitioning approach has been widely used to optimally solve various ship scheduling problems with hard time windows, see for instance Brown et al. (1987), Fisher and Rosenwein (1989), Bausch et al. (1998) and Fagerholt and Christiansen (2000). The new contribution in this paper is that soft time windows replace the hard. To deal with this in the set partitioning approach, an additional schedule optimisation algorithm is added to the generation of candidate schedules. The proposed solution approach also brings the possibility to determine the optimal speeds on the various sailing legs in a ship’s schedule. This may be important since the fuel cost for a given sailing leg is approximately related to the second power of the ship’s speed.

When it comes to vehicle scheduling with soft time windows, there exist a few contributions in the literature. Among the earliest reported research, Sexton and Choi (1986) considered the single-vehicle problem with soft time windows involving both pickups and deliveries. The inconvenience cost of each load is assumed to be a weighted sum of the pickup and delivery time window deviations. The authors describe a Bender’s decomposition procedure and show that the scheduling problem is the dual of a network flow problem. Ferland and Fortin (1989) present a heuristic approach that attempts to adjust time windows of pairs of customers in order to reduce overall costs. Koskosidis et al. (1992) use an optimisation based heuristic approach to solve a vehicle routing problem with soft time windows. The customers are assigned to vehicles using the generalised assignment procedure, see Fisher and Jaikumar (1981). Then a travelling salesman problem with soft time windows is solved separately for each vehicle to determine its route.

Balakrishnan (1993) describe three simple heuristics for the vehicle scheduling problem with soft time windows. The algorithms are based on the nearest neighbour, Clarke–Wright savings, and space–time rules, respectively. The inconvenience cost for violating a time window is assumed to be a linear function of the amount of the violation. At each customer in a route, the heuristics simultaneously determine the time to begin service, and the next customer to serve. More recently, Taillard et al. (1997) described a tabu search heuristic for the vehicle routing problem with soft time windows. In their problem, only lateness at a customer is penalised. Ioachim et al. (1998) present an optimal dynamic programming algorithm to solve the shortest path problem with time windows and additional linear costs on the node service start times. In Desaulniers et al. (1998), a unified for-
mulation for vehicle routing problems with time windows is presented together with a branch-and-bound framework to solve the multi-commodity network flow models in this class. This general formulation also encompasses the problem studied here.

The purpose of this paper is to describe a new optimisation based approach for the m-PDPSTW. A detailed problem description is given in the next section. In Sections 3 and 4, the proposed solution approach is described. The main solution algorithm is given in Section 3, while schedule optimisation for fixed routes, which occurs as a subproblem, is studied in Section 4. A computational study based on data from a real ship scheduling problem is presented in Section 5, while concluding remarks follow in Section 6.

2. Description of the problem

The background of this work is a real problem, involving efficient scheduling of a heterogeneous fleet of ships engaged in pickup and delivery of bulk cargoes. Each cargo consists of a given quantity to be picked up at a given loading port and delivered at a corresponding discharging port. Time windows are imposed for both the pickup and delivery of the cargoes. Each ship in the fleet has a specific capacity, service speed and cost structure. Due to seasonal variations in demand, the fleet is not designed to handle all cargoes. Some of the cargoes may be serviced by spot carriers at a given cost, set by the market. These features give a problem similar to the m-PDPTW, which is treated in Dumas et al. (1991) and Desrosiers et al. (1995).

The cargo quantities and ship capacities are such that the ships may carry several cargoes simultaneously. The loading and discharging on a ship schedule may be interwoven, where a ship discharges only some of the cargoes onboard before loading a new cargo. This makes the problem more complex than many of the ship scheduling problems reported in the literature.

Let us denote the time window for the pickup node of cargo $i$ as $[A_i, B_i]$ and call this an inner time window. The inner time window for the corresponding delivery node is $[A_{n+i}, B_{n+i}]$, where $n$ is the number of cargoes to be serviced during the planning period. We define $D_i^{MAX}$ as the maximum violation of the inner time window at a given node. A larger violation of the time window is assumed to give an inconvenience to the customer that is not acceptable. Now, the outer time window for node $i$ is defined by $[E_i, L_i]$, where $E_i = A_i - D_i^{MAX}$ and $L_i = B_i + D_i^{MAX}$.

The time for start of service at a given node must be within the outer time window and preferably within the inner time window. We define the inconvenience cost function $P_i(t_i)$ on the outer time window $[E_i, L_i]$, where $t_i$ denotes the time at which service begins at node $i$. Let $P_i^{MAX}$ be the maximum cost for violating the time window at node $i$. The value of $P_i^{MAX}$ occurs for $t_i$ equal to $E_i$ and/or $L_i$. Various inconvenience cost functions can be considered, see for instance Fig. 1.

Functions (a) and (c) indicate an inconvenience cost increasing with the amount of violation. Function (b) indicates a non-convex inconvenience cost function, where the cost is not dependent on the amount of the time window violation, only if there is a violation or not. The latter will typically be the case when the customer is given a discount if the delivery is not on time.

![Fig. 1. Alternative inconvenience cost functions.](image-url)
By choosing an inconvenience cost function and a maximum value $P^\text{MAX}_i$ for node $i$, we also influence the policy regarding trade-off between transportation costs and customer service. For example, if a customer is important, $P^i(i)$ is set such that it is ‘expensive’ to violate the inner time window.

By introducing the soft time windows, the ship scheduling problem will become a m-PDPSTW. The objective of the m-PDPSTW is to determine the schedules that each ship in the fleet must perform and the cargoes to be serviced by spot carriers, such that the sum of transportation costs and inconvenience costs is minimised. A schedule is here defined as the visiting sequence as well as the time for start of service at each node. The transportation costs consist of operating costs for the fleet (mainly fuel and port costs) and the cost for the cargoes serviced by a spot carrier. Cargoes lifted by spot carriers are assumed to be serviced within the inner time windows, i.e., at no inconvenience cost.

3. An optimisation based algorithm

The proposed solution algorithm is based on a set partitioning formulation of the m-PDPSTW. First, all (or a number of promising) feasible candidate ship routes are enumerated. Second, the various possible schedules of each route are computed as well as the corresponding operating and inconvenience costs, according to the possible violation of the time windows. It may be noted that the operating cost is not completely determined by the route, since the fuel consumption depends on the ship’s speed and hence the timing of the stops on the route. Finally, the problem is solved as a set partitioning problem, where the columns represent the candidate schedules that are generated.

Section 3.1 presents the formulation of the m-PDPSTW as a set partitioning problem, while the generation of the candidate schedules is discussed in Section 3.2.

3.1. Formulation of the ship scheduling problem as a set partitioning problem

The m-PDPSTW can be reformulated as a set partitioning problem. Let $K$ be the set of ships, indexed by $k$. $N$ is the set of cargoes, indexed by $i$. $R^k$ is the set of candidate schedules for ship $k$, indexed by $r$. At this time, it is assumed that each schedule is optimal (i.e., has a visiting sequence and time for start of service at each node that minimises the sum of the operating costs and the inconvenience costs). $C^k_r$ is the operating cost for sailing schedule $r$ by ship $k$, while $C^k_r$ is the inconvenience cost. $C_{\text{SPON}}$ is the cost for cargo $i$ to be serviced by a spot carrier. $A^k_r$ is a constant that is equal to one if schedule $r$ for ship $k$ services cargo $i$ and zero otherwise. Let $x^k_r$ be a binary variable that is equal to one if ship $k$ sails schedule $r$ and zero otherwise. $s_i$ is a binary variable that is equal to one if cargo $i$ is serviced by a spot carrier and zero otherwise.

The m-PDPSTW can now be formulated as

\[
\begin{align*}
\min & \sum_{k \in K} \sum_{r \in R^k} (C^k_r + C^k_{\text{SPON}}) x^k_r + \sum_{i \in N} C_{\text{SPON}} s_i, \\
\sum_{k \in K} \sum_{r \in R^k} A^k_r x^k_r + s_i & = 1 \quad \forall i \in N, \\
\sum_{r \in R^k} x^k_r & \leq 1 \quad \forall k \in K, \\
x^k_r & \in \{0, 1\} \quad \forall k \in K, \ r \in R^k.
\end{align*}
\]

The objective function (1) minimises the transportation costs plus the inconvenience costs. Constraints (2) ensure that all cargoes are serviced, either by a ship in the fleet or by a spot carrier. (3) ensure that each ship in the fleet sails at most one of its candidate schedules, while (4) impose binary requirements on the variables. $s_i$ variables need not be defined as binary according to (2), since $A^k_r$ are binary constants and $x^k_r$ are binary variables.

3.2. Generation of the candidate schedules

The column vector $[A^k_{1r}, A^k_{2r}, \ldots, A^k_{nr}, 1]^T$ in the formulation (1)–(4) corresponds to one feasible candidate schedule belonging to the set $R^k$, where $n$ is the number of cargoes in the set $N$. The number of feasible schedules for a ship $k$ is too big to allow exhaustive enumeration for problems of general size. Therefore, the possibility of generating only
the most promising schedules is introduced. For this purpose, heuristic rules regarding the capacity utilisation of the ships are introduced. Only schedules that satisfy these heuristic rules are considered as candidate schedules for the set partitioning problem, while schedules with too low capacity utilisation are rejected. Two measures for the capacity utilisation are used:
- waiting time/unused time,
- utilisation of the capacity of the ship’s cargo hold.

Let us define $R_{\text{MAX}}$ as a factor to control the allowance level for both these two measures. Maximum allowed waiting time in a schedule will be an increasing function of $R_{\text{MAX}}$, while minimum utilisation of the cargo hold that is accepted is a decreasing function of $R_{\text{MAX}}$. By selecting the value of $R_{\text{MAX}}$, we control the number of schedules that are generated, and also the response time. For low $R_{\text{MAX}}$-values, only few candidate schedules are generated, while for high $R_{\text{MAX}}$-values, all schedules may be generated.

The generation of the schedules is performed a priori and in a systematic way by extending an existing schedule by adding one more cargo at a time. Since the service of a cargo consists of both pickup and delivery, we say that each cargo consists of two nodes. Suppose an existing schedule servicing cargoes 2 and 3, has the optimal visiting sequence 0–3–2–(n + 2)–(n + 3). Node 0 is the initial point for the given ship, while the nodes 2, 3 and (n + 2) and (n + 3) are the pickup and delivery nodes for cargoes 2 and 3, respectively. This schedule is to be extended by cargo $i$, with the corresponding nodes $i$ and (n + i). It is generally not optimal to add the nodes for cargo $i$ to the end of the existing schedule. It may also give an infeasible solution due to the time windows. To find the optimal visiting sequence and the start of service at each node for the new extended schedule, we have to solve a Travelling Salesman Problem with Capacity, Soft Time Windows and Precedence Constraints (TSP-CSTWPC) for the nodes 2, 3, $i$, (n + 2), (n + 3) and (n + i), with node 0 as the starting node. The capacity constraints ensure that the total cargo onboard the ship never exceeds the capacity. The precedence constraints ensure that the pickup node is visited before the corresponding delivery node. The soft time windows ensure that the new schedule is optimal with respect to the sum of the inconvenience cost and the operating cost.

Thus, in the generation of each new candidate schedule, we have to solve a TSP-CSTWPC for a given set of nodes. During the implementation of the soft time windows and the inconvenience cost for missing the inner time window, the ship scheduling problem was solved with the outer time windows as hard time windows, i.e., as an m-PDPTW. Here, a Travelling Salesman Problem with Capacity, Hard Time Windows and Precedence Constraints (TSP-CHTWPC) had to be solved in the generation of each schedule. By examining these TSP-CHTWPCs, it was found that the number of feasible solutions or visiting sequences for a given subset of nodes was on an average relatively small, due to the time windows. This motivated the idea of introducing the inconvenience cost and perform a schedule optimisation for each of the feasible visiting sequences to the TSP-CHTWPC. By doing this, the optimal schedule (for the given subset of nodes) with respect to the sum of the inconvenience and the operating cost is determined, and the TSP-CHTWPC (the soft version) is actually solved. By adding the optimal schedules to the set partitioning problem, we can solve the soft time window version of the ship scheduling problem, i.e., the m-PDSTW.

Suppose we want to determine the optimal schedule for the set of nodes $V = \{0, 1, 2, \ldots, m\}$ for ship $k$, where node 0 is the starting node. Suppose further that two feasible visiting sequences or routes are found when the TSP-CHTWPC is solved with the time windows $[E_i, L_i] \forall i \in V$. It may be noted that the ship’s maximum speed (no slow steaming) is used in the solution of the TSP-CHTWPC. Once the routes are determined, what is left open is the exact timing of the stops on the routes. The timing affects both the inconvenience cost and the operating cost (remember that the fuel consumption on a specific sailing leg is related to the second power of the ship’s speed). By optimising the schedules for both alternative routes with respect to the sum of the operating cost and the inconvenience cost, the schedule with the minimum sum of these two cost
components can be selected as a candidate schedule to the set partitioning problem. If there is only one feasible solution to the TSP-CHTWPC, the visiting sequence is given. Then, we optimise the schedule for this visiting sequence with respect to the sum of the operating and inconvenience cost to find the optimal times for start of service at each node. The schedule optimisation algorithm is described in Section 4.

Due to the outer time windows, the TSP-CHTWPC can often be solved on a subset of \( V \), thus operating on partial schedules only and reducing the problem size. In the example above, the outer time windows for the nodes may be such that it is not necessary to consider the possibility to insert node \( i \) before node \( (n+2) \) of the existing schedule, because this would involve a waiting time that is higher than allowed by the heuristic rules. In this case, we can operate with three nodes only \( ((n+3), i, (n+i)) \), with \( (n+2) \) as a starting node. To find the new extended visiting sequence, the visiting sequence for the subsets of the nodes can be merged together with the existing schedule up to the starting node \( (n+2) \).

It is not within the scope of this paper to describe the solution method to the TSP-CHTWPC, but it can due to the problem size easily be solved by a dynamic programming algorithm, see for instance Desrosiers et al. (1986) and Dumas et al. (1995).

When all candidate schedules satisfying the heuristic rules on capacity utilisation and waiting time are generated, the m-PDPSTW can be formulated as (1)–(4) and solved by commercial mixed integer programming software.

4. Schedule optimisation for a fixed route

The problem of optimising the schedule for a fixed visiting sequence in the case of inconvenience costs is to some extent studied in the literature. Sexton and Bodin (1985) consider a maximum time for start of service, \( L_i \), rather than a time window. The inconvenience cost is modelled by a linear function. Dumas et al. (1990) consider the case where the start time of service is constrained by hard time windows, but where some time for start of service is preferred. The inconvenience cost can be modelled by a linear, a quadratic, or a convex function. They propose a dual solution approach that consists, starting from a relaxed version of the problem, in reintroducing one by one of the violated constraints until all constraints are satisfied. Ioachim et al. (1998) propose another approach in the case of linear node costs, based on a definition and analysis of the node cost function at the last node of the path. It is shown that the node cost function at a given node in the path is piecewise linear.

All inconvenience cost functions considered in the literature are convex. The solution approach that is proposed here can handle non-convex functions as well. Section 4.1 presents a mathematical formulation for the problem of optimising the schedule for a given visiting sequence. The schedule optimisation problem is solved as a shortest path problem on a network where each node of the given visiting sequence is duplicated into a number of possible times for start of service. The description of the shortest path network and the solution algorithm is given in Sections 4.2 and 4.3, respectively.

4.1. Mathematical formulation

Let \( V = \{0, 1, 2, \ldots, m\} \) be the set of nodes in the visiting sequence \( 0 \rightarrow 1 \rightarrow 2 \rightarrow \cdots \rightarrow m \). Define \( T_{i,i+1} \) to be the sailing time between the nodes \( i \) and \( i+1 \) in the visiting sequence, \( i \in V \setminus \{m\} \). Let \( S_i \) be the time used for service at node \( i \in V \). For each node \( i \in V \), the variable \( t_i \) represents the time when service begins. The inconvenience cost function \( P_i(t_i) \) is defined on the outer time window \([E_i, L_i]\), for instance as in Fig. 1. The fuel consumption and hence the operating cost for a schedule also depend on the time for start of service. Changing the time for start of service at a given port may bring the possibility for slow steaming with reduced fuel cost. Define the operating cost on the sailing leg between node \( i \) and \( (i+1) \) as \( C_{i,i+1}(t_i, t_{i+1}) \), which is a non-linear function of the used sailing time between the two nodes (and hence the speed).

The problem of optimising the schedule for a given visiting sequence can now be formulated as
\min \sum_{i \in V} P_i(t_i) + \sum_{i \in V \setminus \{m\}} C_{i,i+1}(t_i, t_{i+1}), \quad (5)

t_i + S_i + T_{i,i+1} \leq t_{i+1} \quad \forall i \in V \setminus \{m\}, \quad (6)

E_i \leq t_i \leq L_i \quad \forall i \in V. \quad (7)

The objective function (5) minimises the sum of the inconvenience and the operating cost for the given route. It may be noted that the port costs for a route do not depend on the timing of the stops, and can thus be omitted in the objective function (5). Constraints (6) are linking constraints describing the compatibility of the schedule with the visiting sequence, while (7) ensures that the service at each node starts within the outer time windows.

4.2. Generation of shortest path network

To solve the schedule optimisation problem defined by (5)–(7), each node in the given route is duplicated into all possible times for start of service at that node. By discretising the time into steps of a given time unit, the number of possible arrival times can be reduced. Each of the new duplicated nodes will represent a state \((i, t_i)\) consisting of the physical node \(i\) and the start time \(t_i\) of service at the node. A value for the inconvenience cost, \(P_i(t_i)\), will be attached to each node in the new network, while a value for the operating cost, \(C_{ij}(t_i, t_j)\), will be attached to the arc between the two states \((i, t_i)\) and \((j, t_j)\). It may be noted that \(C_{ij}(t_1, t_2) > C_{ij}(t'_1, t'_2)\) if \(t_1 > t'_2\), since the latter allows for slower sailing speed with less fuel cost.

Example. Suppose the visiting sequence for the nodes \(V = \{0, 1, 2, 3\}\) is found by solving the TSP-CHTWP. The earliest possible time for start of service at each node is simultaneously determined. Fig. 2 indicates the visiting sequence and the earliest possible time for start of service together with the inner and outer time windows at each node.

Suppose that the possible times for start of service, \(t_i\), are discretised into time units of 6. The earliest possible start times are rounded to the nearest multiple of this time unit. By duplicating the nodes of the visiting sequence in Fig. 2 into all possible discretised times for start of service, we get the network illustrated in Fig. 3. The inconvenience cost is indicated only for some of the nodes in the network, while the operating cost is only indicated between the physical nodes 0 and 1.

We note that at node 2, there are more times for start of service within the outer time window than illustrated in Fig. 3, but these turn out to be infeasible, as they result in that node 3 cannot be reached within its outer time window.

4.3. Schedule optimisation by solving a shortest path problem

By discretising the possible times for start of service and duplicating the nodes in the visiting sequence as in Example 1, we get a nice structured acyclic network. The schedule optimisation can now easily be solved as a shortest path problem by...
a dynamic programming algorithm on this network. The start node will be the node corresponding to physical node 0, and the ending node will be any of the nodes corresponding to the physical node \( m \) (node 3 in Example 1).

Since \( C_{ij}(t_1^i, t_2^i) < C_{ij}(t_2^i, t_j) \) whenever \( t_1^i < t_2^i \), the state space when solving the shortest path problem can therefore be reduced according to the following proposition:

**Proposition.** Let \( f_i(t_i) \) be the least accumulated cost (inconvenience and operation) for reaching state \((i, t_i)\), where \( i \) is the physical node and \( t_i \) is the time for start of service at that node. For any inconvenience cost function \( P_j(t_j) \) at node \( j \) and an operating cost function \( C_{ij}(t_i, t_j) \) from \( i \) to \( j \) such that

\[
f_i(t_1^i) + P_j(t_j) + C_{ij}(t_1^i, t_j)
\]

is less than or equal to

\[
f_i(t_2^i) + P_j(t_j) + C_{ij}(t_2^i, t_j)
\]

since \( C_{ij}(t_1^i, t_j) \leq C_{ij}(t_2^i, t_j) \). Hence label \( f_i(t_1^i) \) can be eliminated from the optimisation process.

**5. Computational study**

**5.1. Description of the test problems**

The proposed solution algorithm for the m-PDPSTW is tested on data from a real problem considering sea transportation of bulk cargoes in Northern Europe. Four different cases are studied. Case 1 consists of 18 cargoes (36 nodes) distributed on a two week planning period. Case 2 consists of 25 cargoes (50 nodes) also for a period of two weeks. Case 3 has 26 cargoes (52 nodes), while case 4 has 35 cargoes (70 nodes), both cases corresponding to a 3-week planning period. The fleet available to perform the transportation tasks is assumed to consist of five ships, varying in their capacities and initial positions. The ships are assumed to be empty in the beginning of the planning period.

Five instances with various inconvenience cost functions are studied for each case, see Table 1. Each test problem is also solved a number of times, varying with the allowance factor \( R_{MAX} \). In this way, we can study how the heuristic rule for the schedule generation affects the solutions. The inconvenience cost functions are assumed to be equal for all nodes in each test problem, i.e., \( P_i(t_i) = P(t_i) \) and \( P_i^{MAX} = P^{MAX} \) for all nodes.

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Inconvenience cost functions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Instance</strong></td>
<td><strong>Inconvenience cost function</strong></td>
</tr>
<tr>
<td>1</td>
<td>Linear</td>
</tr>
<tr>
<td>2</td>
<td>Quadratic</td>
</tr>
<tr>
<td>3</td>
<td>‘Constant-cost’</td>
</tr>
<tr>
<td>4</td>
<td>‘Zero-cost’</td>
</tr>
<tr>
<td>5</td>
<td>‘Infinity-cost’</td>
</tr>
</tbody>
</table>
All nodes in the test problems have an inner time window width of 24 hours. The outer time window width is 72 hours, with a maximum allowable deviation from the inner time windows of 24 hours on each side. The coefficients \( a, \) \( b \) and \( P^{\text{MAX}} \) for instances 1, 2 and 3 in Table 1 are adjusted such that the maximum deviation from the inner time window of 24 hours has the same model cost as daytime charter for the largest ship in the fleet.

Instances 4 and 5 actually represent hard time window problems. For instance 4, the hard time windows correspond to the outer time windows, while the inner time windows represent the hard time windows for instance 5.

The test problems were solved with a time discretisation of 4 or 6 hours in the schedule optimisation. This is considered to be accurate enough for a ship scheduling problem with a planning period of 2–3 weeks.

5.2. Numerical results

The algorithm for generation of the candidate schedules and schedule optimisation is written and compiled in Borland Pascal 7.0. The set partitioning problem is implemented and solved by GAMS/CPLEX version 5.0. The problems are run on a PC with a Pentium 166 MHz processor having 64 MB of RAM.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Case 1</th>
<th></th>
<th>Case 2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning period [weeks]</td>
<td>2</td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td># nodes</td>
<td>36</td>
<td></td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>Time discretisation [hours]</td>
<td>4</td>
<td></td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>( R^{\text{MAX}} )</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>( \infty )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Schedule generation:</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td># candidate schedules</td>
<td>964</td>
<td>1171</td>
<td>1368</td>
<td>1493</td>
</tr>
<tr>
<td>Avg. # of TSP-CHTWPC solutions</td>
<td>1.07</td>
<td>1.08</td>
<td>1.08</td>
<td>1.09</td>
</tr>
<tr>
<td>CPU [seconds]</td>
<td>9.9</td>
<td>12.2</td>
<td>14.8</td>
<td>16.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Set partitioning:</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Total transportation cost [%]</td>
<td>62.4</td>
<td>64.4</td>
<td>64.4</td>
<td>64.4</td>
</tr>
<tr>
<td>Slow steaming [%]</td>
<td>2.8</td>
<td>3.2</td>
<td>3.2</td>
<td>3.2</td>
</tr>
<tr>
<td>Total violation of TW [hours]</td>
<td>144</td>
<td>132</td>
<td>132</td>
<td>132</td>
</tr>
<tr>
<td>CPU [seconds]</td>
<td>1.1</td>
<td>0.9</td>
<td>0.9</td>
<td>1.0</td>
</tr>
</tbody>
</table>
Table 3
Solutions with varying $R_{\text{MAX}}$, cases 3 and 4

<table>
<thead>
<tr>
<th>Problem</th>
<th>Case 3</th>
<th>Case 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning period [weeks]</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td># nodes</td>
<td>52</td>
<td>70</td>
</tr>
<tr>
<td>Time discretisation [hours]</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>$R_{\text{MAX}}$</td>
<td>1 2 3 $\infty$</td>
<td>1 2 3 $\infty$</td>
</tr>
<tr>
<td>Schedule generation:</td>
<td></td>
<td></td>
</tr>
<tr>
<td># candidate schedules</td>
<td>2988</td>
<td>10259</td>
</tr>
<tr>
<td>Avg. # of TSP-CHTWP solutions</td>
<td>1.32</td>
<td>1.05</td>
</tr>
<tr>
<td>CPU [seconds]</td>
<td>42.8</td>
<td>102.4</td>
</tr>
<tr>
<td>Set partitioning:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total transportation cost [%]</td>
<td>67.4</td>
<td>84.3</td>
</tr>
<tr>
<td>Slow steaming [%]</td>
<td>2.2</td>
<td>2.0</td>
</tr>
<tr>
<td>Total violation of TW [hours]</td>
<td>216</td>
<td>144</td>
</tr>
<tr>
<td>CPU [seconds]</td>
<td>2.3</td>
<td>18.0</td>
</tr>
</tbody>
</table>

solution is reached at $R_{\text{MAX}} = 2$, while for case 2 the optimal solution appears at $R_{\text{MAX}} = 3$. For case 3, the optimal solution is not reached until $R_{\text{MAX}} = \infty$. For case 1, it may be noted that the transportation cost is less for $R_{\text{MAX}} = 1$ than for $R_{\text{MAX}} > 1$. This reduction in transportation cost comes however at the sacrifice of more time window violations. What solution that is best in practice of the one obtained for $R_{\text{MAX}} = 1$ and the optimal one obtained for $R_{\text{MAX}} = \infty$ may be hard to tell.

The results show, as expected, that it is possible to achieve significant improvements in the total transportation costs by allowing controlled violations of some time windows. As indicated in Tables 2 and 3, the savings compared to the hard time window cases of instance 5 vary from approximately 20–40% between the different cases. The main reason for these considerable savings, is that for instance 5, a large amount of spot shipments that are costly compared to the operating costs are needed to manage servicing all customers within their inner time window. For instances 1–4, the need for spot shipments is much less. The savings due to the allowance of slow steaming are also significant, with approximately 2–3% of the total transportation cost of instance 5.

The generation of the candidate schedules requires more CPU-time than solution of the set partitioning problem. The time consumption for solving the set partitioning problems was low because there were no gaps between the values of the LP-solutions and the IP-solutions for any of the test problems solved. The computational experiences also indicate that CPU-time required to generate the candidate schedules heavily depends on the number of nodes to be serviced. This is because the number of candidate schedules typically increases exponentially with the number of nodes.

Table 4 shows the computational results for the various inconvenience cost functions. For cases 1, 2 and 3, $R_{\text{MAX}} = \infty$ is used, while for case 4, $R_{\text{MAX}} = 3$ is used. As expected, instance 4, where time window violations are not penalised at all, gives the lowest transportation costs. However, by penalising time window violations in the various ways of instances 1–3, we may achieve solutions that involve only slight higher costs compared with instance 4, but with significant reductions in the time window violations.

The total deviation of the inner time windows is higher for the quadratic inconvenience cost function (instance 2) than for the linear (instance 1) and the ‘constant’ function (instance 3). This is because all deviations are less penalised with this function than for the other two (except for the maximum deviation of 24 hours that is equally penalised). The results also show that since the quadratic function imposes very small costs for small deviations, the average time window violation is least for instance 2. On the contrary, we
might expect that instance 3 gives the highest average time window violation since all violations are penalised equally. This is true except for case 1, where instance 3 only gave an average time window violation of 9 hours.

Fig. 4 illustrates the CPU-time of the generation of the candidate schedules as a function of the time discretisation in the schedule optimisation. Time discretisations of 3, 4, 6, 8 and 12 hours are tested. The CPU-time for a time discretisation of 4 hours is set as 100%. The CPU-time is relatively sensitive for variations of the time discretisation around 4 hours. By decreasing the time discretisation unit to 3 hours, the CPU-time increases.

![Fig. 4. CPU-time as a function of time discretisation.](image-url)
6. Concluding remarks

This paper describes a new optimisation based approach for an m-PDPSTW. The motivation behind scheduling with soft time windows is that by allowing controlled time window violations for some customers, it may be possible to obtain better schedules and significant reductions in the transportation costs. The computational results obtained on data from a real ship scheduling problem also verify this. The effect of customer requirements (time windows) on transportation costs may be important to bear in mind when defining alliances between shippers, transport providers and customers in a supply chain.

Soft time windows also reflect situations found in practice better than hard time windows. To set an appropriate model cost for a given amount of violation of a time window is however difficult. This cost can for instance represent the cost of lost goodwill, lost sales or that the customer must be given a discount. However, by considering various inconvenience cost functions, a good trade-off between transportation costs and service quality to the customers can be found, see the discussion in Fagerholt (2000).

The main idea of the proposed solution method is to a priori generate a large number of promising (or all feasible) candidate schedules. Heuristic rules may be applied to reduce the number of candidate schedules that are generated. The scheduling problem can then be solved as a set partitioning problem, in which the columns represent the candidate schedules. To incorporate the soft time windows, an additional algorithm for schedule optimisation is added to the generation of the candidate schedules. The proposed solution method works for both convex and non-convex inconvenience cost functions and solutions on real sized ship scheduling problems are achieved in reasonable time. The solutions of three of the four cases are optimally solved, while the solution of the fourth case is not proven optimal. It should however be emphasised that the planning period for cases 3 and 4 is three weeks, while for the real problem a typical planning period for practical purposes is usually less than this, due to uncertainties in the future cargo movements.

The major drawback of the proposed solution method is that it cannot generate optimal solutions to large unconstrained problems where the number of feasible schedules is too high to enumerate them all. However, at least in the shipping business, the number of feasible schedules is often quite limited, as the problems are often not very large and also often well constrained due to ship capacities, port depths and time windows.

Acknowledgements

The author is grateful to the referees for their insightful and valuable comments, which improved the presentation of the paper.

References


