Container Scheduling: Complexity and Algorithms

Byung-Cheon Choi

Department of Business Administration,
Chungnam National University,
79 Daehangno, Yuseong-gu, Daejeon 305-704, Korea

Kangbok Lee

Department of Supply Chain Management & Marketing Sciences,
Rutgers Business School,
1 Washington Park, Newark, NJ 07102, USA

Joseph Y-T. Leung

Department of Computer Science,
New Jersey Institute of Technology,
Newark, NJ 07102, USA

Michael L. Pinedo

Department of Information, Operations & Management Sciences,
Stern School of Business, New York University,
44 West 4th Street, New York, NY 10012-1126, USA

Dirk Briskorn

Universität zu Köln,
Wirtschafts- und Sozialwissenschaftliche Fakultät,
Albertus-Magnus-Platz, 50923 Köln, Germany

Abstract

We consider the transport of containers through a fleet of ships. Each ship has a capacity constraint limiting the total number of containers it can carry and each
ship visits a given set of ports following a predetermined route. Each container has a release date at its origination port, and a due date at its destination port. A container has a size 1 or size 2; size 1 represents a 1 TEU (Twenty-foot Equivalent Unit) and size 2 represents 2 TEUs. The delivery time of a container is defined as the time when the ship that carries the container arrives at its destination port. We consider the problem of minimizing the maximum tardiness over all containers. We consider three scenarios with regard to the routes of the ships, namely, the ships having (i) identical, (ii) nested, and (iii) arbitrary routes. For each scenario, we consider different settings for origination ports, release dates, sizes of containers, and number of ports; we determine the computational complexity of various cases. We also provide a simple heuristic for some certain cases, with its worst case analysis. Finally, we discuss the relationship of our problems with other scheduling problems that are known to be open.

Keywords: liner shipping; container allocation; on-time delivery, scheduling rules, computational complexity

Original submission: May 2009
1st revised version submission: Feb 2010
2nd revised version submission: Sep 2010
Acceptance: Oct 2010

1 Introduction

We consider the transport of containers through a fleet of ships. Each ship has a capacity constraint limiting the total number of containers it can carry and each ship visits a given set of ports following a predetermined route and at predetermined times. Because of the capacity constraints of the ships, the company in charge of the shipments often may have to decide which containers to ship immediately and which containers to ship later. In this paper we analyze various scheduling problems that arise in this context.

* Corresponding author.

Email addresses: polytime@cnu.ac.kr (Byung-Cheon Choi), kangblee@business.rutgers.edu (Kangbok Lee), leung@cis.njit.edu (Joseph Y-T. Leung), mpinedo@stern.nyu.edu (Michael L. Pinedo), briskorn@wiso.uni-koeln.de (Dirk Briskorn).

1 Work supported by the Korea Research Foundation Grant KRF-2008-357-D00289.
2 Work supported in part by the NSF Grant CMMI-0969830.
3 Work supported in part by the NSF Grant CMMI-0969755.
4 Work supported by a fellowship within the Postdoc-Program of the German Academic Exchange Service (DAAD).
As of 2006, maritime trade accounted for 89.6% of all global trade in terms of volume and 70.1% in terms of value (Rodrigue (2009)). Containerization has played for many years an important role in international trade. The container delivery market, as reported in the Drewry Annual Container Market Review and Forecast 2006/2007 by Dekker (2006) has grown steadily at a rate of over 9.0%. Container transport has given rise to a host of interesting combinatorial problems that have received a lot of research attention in the past.

Several comprehensive surveys have appeared in the literature; see, for example, Stahlbock and Voß (2008), Steenken et al. (2004) and Vis and Koster (2003). Container handling operations at terminals have been formulated as optimization problems, e.g., quay crane planning (see Goodchild and Daganzo (2006) and Park and Kim (2003)), container storage and retrieval (see Vis and Roodbergen (2009)) and straddle carrier scheduling (see Kim and Steenken et al. (1993)). Furthermore, routing and scheduling of ships have also received a considerable amount of attention, see Christiansen et al. (2004) for a survey. Fagerholt (2001) and Fagerholt and Christiansen (2000) consider ship routing problems where the allocation of containers is part of the decision making process. Chen et al. (2007) propose a simple model with two ports. There are several researches on shipments by a fleet of trucks at an air cargo terminal, for example, Ou et al. (2010). However, problems concerning the assignment of containers to the various ships of a given shipping operator have, to our knowledge, not received any attention so far.

In this paper we analyze various scheduling problems that arise in our container shipping model. The ports, ships, routes, and containers can be characterized as follows.

- **Ports:** A port is a location where containers depart from or arrive at. The set of all ports is denoted by \( \Pi = \{1, \ldots , N_p\} \), where \( N_p \) denotes the total number of ports.
- **Ships:** Ship \( i \) has capacity \( c_i \) and operates on a given route \( \rho^i \), calling on the set of ports \( \pi^i \), a subset of \( \Pi \), in a predetermined order. The set of ships is denoted by \( \{1, \ldots , m\} \), where \( m \) is the number of ships.
- **Routes:** The route of ship \( i \) is denoted by \( \rho^i \), and the \( l \)th port in \( \rho^i \) is denoted by \( \rho^i(l) \), i.e., \( \rho^i : \rho^i(1) \to \rho^i(2) \to \cdots \to \rho^i(|\pi^i|) \), where \(|\pi^i|\) denotes the cardinality of set \( \pi^i \). Let \( t_{i,l} \) denote the date ship \( i \) arrives at port \( l \) and if \( l \notin \pi^i \), then, for convenience, we set \( t_{i,l} \) equal to \( \infty \). Thus, \( t_{i,\rho^i(k+1)} - t_{i,\rho^i(k)} \) is the transportation time from port \( \rho^i(k) \) to port \( \rho^i(k+1) \); loading and unloading times are included in the transportation time.
- **Containers:** Container \( j \) has an origination port \( u_j \) and a destination port \( v_j \), where \( u_j, v_j \in \Pi \) and \( u_j < v_j \). In the real world, most containers have the same height and width but differ in their length. The length is either 1 TEU (twenty-foot equivalent unit) or 2 TEUs. Thus, we may assume that the size \( s_j \) of container \( j \) is either 1 or 2 and the total
size of containers being loaded on a ship cannot exceed the capacity of the ship. Let \( r_j \)
denote the release date of container \( j \) at its origination port and let \( d_j \) denote the due date
at its destination port. If container \( j \) is unloaded from ship \( i \) at port \( v_j \), then the delivery
date is \( D_j = t_{i,v_j} \). Let \( J^1 \) and \( J^2 \) denote the sets of containers of size 1 and 2, respectively.
Assume \( |J^1| = n_1 \) and \( |J^2| = n_2 \). Let \( J = \{1, \ldots, n\} \) denote the set of all containers, i.e.,
\( J = J^1 \cup J^2 \) and \( n = n_1 + n_2 \).

Throughout the paper, we make the following assumptions.

(i) **Unidirectional Routes** The \( N_p \) ports are geographically located in such a way, that
ships always go from a port with a lower index to a port with a higher index. This
implies that all ships consistently travel in one direction. The problem with containers
having the reverse direction can be solved separately.

(ii) **Non-overtaking Routes** A ship cannot pass another ship while travelling from one
port to the next, i.e., \( t_{h,l} < t_{i,l} \) for \( 1 \leq h < i \leq m \) and \( l \in \pi_h \cap \pi_i \).

(iii) **No-transfer Shipping** A container can be transported by only one ship. In the real
world, it may be the case that a ship brings a container to a transfer port and another
ship takes it from the transfer port to its final destination. However, in this paper, we
restrict ourselves to the no-transfer shipping case.

The first two assumptions are easily justified in the case of liner shipping or in the case of
shipping along a major river. Liner shipping implies that the maritime transport of goods
follows a regular service; liner shipping companies usually publicize their schedules in ad-
ance. Liner container ships operate in the international maritime trade with cargo that
is consolidated from a large number of consignments from different shippers. The shipping
networks are maintained with large vessels that follow either pendulum routes or circular
routes and are supported by transshipment services. The route involves a regular itinerary
between a number of ports, often served because of geographical proximity. In practice, often
a cluster of ports along one seaboard is serviced and then an ocean is crossed. This process
is repeated on a regular basis.

For example, Hapag-Lloyd maintains various global routes with its container vessels. Its NCE
(North/Central China Eastcoast Express) service maintains services between Asia and North
America (NCE website (2009)) following a circular route that consists of the ports Busan,
Qingdao, Ningbo, Yangshan, New York, Norfolk, Savannah and Busan. The interdeparture
times at consecutive ports are two, two, one, twenty two, two, two and twenty three days.
(Fig. 1) Every Monday a vessel departs from Busan and the vessels may have different
capacities. In such a case, we consider an extended linear route that visits ports from Busan
to Savannah, then again Busan, Qingdao, Ningbo, and finally arriving into Yangshan. Since
in the planning horizon for a schedule, the same ship typically does not appear twice, we
can deal with containers from Asia to North America and containers from North America
to Asia at the same time. Thus, we can assume that all ships travel in one direction.

As a second example, consider the Crowley Maritime Corporation which provides liner shipping service to and from Puerto Rico, the Bahamas, the Caribbean and Central America. The Bahamas Service is a service between the U.S. and Nassau and its route has a pendulum structure (Bahamas service website (2009)). A vessel departs from Port Everglades on Tuesday, visits Nassau on Wednesday and arrives at Jacksonville on Friday. In the reverse direction, it starts from Jacksonville on Friday, visits Nassau on Monday and arrives at Port Everglades on Tuesday. Clearly, the containers to be delivered from Port Everglades to Nassau, from Port Everglades to Jacksonville, and from Nassau to Jacksonville have to use south bound ships. On the other hand, containers with opposite originations and destinations have to use north bound ships. So we can easily split the routes into a north bound segment and a south bound segment.

The constraints under consideration are straightforward. Because of the release date of a container, a container can only be loaded onto a ship at its origination port after its release date; container \( j \) can be loaded onto ship \( i \) only if \( r_j \leq t_{i,u_j} \). Because of the No-Transfer assumption, a container can only be loaded onto a ship that visits both its origination port and its destination port. Each ship has a capacity constraint limiting the total number of containers that it can carry at any point in time. The objective under consideration is to minimize the maximum tardiness of the containers, i.e., \( T_{\text{max}} = \max_{j=1,2,...,n} \{0, D_j - d_j\} \), where \( D_j = t_{i,v_j} \) assuming container \( j \) is delivered at port \( v_j \) by ship \( i \).
In this paper we consider three scenarios with regard to the routes of the ships, namely:

(a) The routes of all ships are identical, implying that all ships visit all ports, i.e., $\pi^i = \Pi$ for all $i = 1, \ldots, m$.

(b) The routes of all ships are nested, implying that, for any two ships $h$ and $i$, $\pi^i = \pi^h$, or $\pi^i \subset \pi^h$, or $\pi^i \supset \pi^h$. (Some ships may be bigger and can visit only the larger ports, while other ships are smaller and can visit all ports.)

(c) The routes are arbitrary, implying that $\pi^i$ is an arbitrary subset of $\Pi$ with at least two distinct ports.

Scenario (b) is quite natural, since each ship may be of a certain size and with a certain capacity. A ship of a certain size may only be able to serve those ports that can handle ships of its size. The smaller a ship, the larger the set of ports that it can serve. The important levels of the ports may cause the various sets of ports to be nested.

We introduce the following classification scheme for our container scheduling problems: a problem can be described via a three field notation $\alpha \mid \beta \mid \gamma$. The $\alpha$ field contains two entries specifying details regarding the routing structure $R$ and the number of ports $N_p$; the $\beta$ field contains three entries specifying the release dates $r_j$ of the containers, whether or not they all originate from the same port $u_j$, and the sizes $s_j$ of the containers; the $\gamma$ field entry refers to the objective function $T_{\text{max}}$.

The routing structure $R$ may be $I$ (identical), $N$ (nested), or $A$ (arbitrary). If the number of ports $N_p$ is an arbitrary fixed number, then the symbol $N_p$ appears in the first field; otherwise, the number of ports is variable, and an $X$ appears. If $r_j = 0$, then all containers are released at the same time; if the entry is $r_j$, then the containers are released at arbitrary points in time. If $u_j = 1$, then all containers originate from the same port, i.e., port 1; if the entry only says $u_j$, then the containers are released at different ports. In case of $u_j = 1$, it may imply that port 1 is a major port and therefore a distribution center (say Hong Kong). For the case $u_j = 1$, the maximum load of a ship always occurs at the common origination port, so we explicitly assume that $\sum_{i=1}^{m} c_i \geq n_1 + 2n_2$ to avoid infeasibility. If $s_j = 1$, then all containers are of the same size; if $s_j \in \{1, 2\}$, then the containers may be either of size 1 or of size 2. In this paper the only objective function to be considered is $T_{\text{max}}$. To deal with the maximum tardiness $T_{\text{max}}$ as an objective seems to be quite reasonable. From a theoretical point of view, minimizing $T_{\text{max}}$ should be the first problem to be considered among the ones dealing with due date related objectives. From a practical point of view, checking whether there exists a schedule such that all jobs are scheduled on time (that is, $T_{\text{max}} = 0$), is critical since it is a question about whether we can actually make a commitment with regard to the current due dates. Note that in each parameter setting, the former is a special case of the
latter, implying that a problem with the former setting is not more difficult than a problem with the latter in a complexity sense.

We present an illustrative example of $A, 5 \mid r_j, u_j, s_j \in \{1, 2\} \mid T_{\text{max}}$. Three ships ($m = 3$) visit five ports ($N_p = 5$) with the following capacities and routes.

<table>
<thead>
<tr>
<th>visiting times</th>
<th>Port 1</th>
<th>Port 2</th>
<th>Port 3</th>
<th>Port 4</th>
<th>Port 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>ship 1 ($c_1 = 5$)</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>ship 2 ($c_2 = 4$)</td>
<td>-</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>8</td>
</tr>
<tr>
<td>ship 3 ($c_3 = 4$)</td>
<td>-</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>-</td>
</tr>
</tbody>
</table>

The container data are the following.

<table>
<thead>
<tr>
<th>container $j$</th>
<th>$s_j$</th>
<th>$u_j$</th>
<th>$v_j$</th>
<th>$r_j$</th>
<th>$d_j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$j = 1$</td>
<td>$s_1 = 2$</td>
<td>$u_1 = 1$</td>
<td>$v_1 = 5$</td>
<td>$r_1 = 0$</td>
<td>$d_1 = 7$</td>
</tr>
<tr>
<td>$j = 2$</td>
<td>$s_2 = 2$</td>
<td>$u_2 = 1$</td>
<td>$v_2 = 5$</td>
<td>$r_2 = 0$</td>
<td>$d_2 = 7$</td>
</tr>
<tr>
<td>$j = 3$</td>
<td>$s_3 = 1$</td>
<td>$u_3 = 1$</td>
<td>$v_3 = 4$</td>
<td>$r_3 = 0$</td>
<td>$d_3 = 5$</td>
</tr>
<tr>
<td>$j = 4$</td>
<td>$s_4 = 1$</td>
<td>$u_4 = 4$</td>
<td>$v_4 = 5$</td>
<td>$r_4 = 5$</td>
<td>$d_4 = 7$</td>
</tr>
<tr>
<td>$j = 5$</td>
<td>$s_5 = 2$</td>
<td>$u_5 = 1$</td>
<td>$v_5 = 5$</td>
<td>$r_5 = 0$</td>
<td>$d_5 = 5$</td>
</tr>
<tr>
<td>$j = 6$</td>
<td>$s_6 = 2$</td>
<td>$u_6 = 1$</td>
<td>$v_6 = 5$</td>
<td>$r_6 = 0$</td>
<td>$d_6 = 5$</td>
</tr>
<tr>
<td>$j = 7$</td>
<td>$s_7 = 2$</td>
<td>$u_7 = 2$</td>
<td>$v_7 = 4$</td>
<td>$r_7 = 3$</td>
<td>$d_7 = 6$</td>
</tr>
<tr>
<td>$j = 8$</td>
<td>$s_8 = 2$</td>
<td>$u_8 = 2$</td>
<td>$v_8 = 3$</td>
<td>$r_8 = 3$</td>
<td>$d_8 = 4$</td>
</tr>
<tr>
<td>$j = 9$</td>
<td>$s_9 = 1$</td>
<td>$u_9 = 3$</td>
<td>$v_9 = 4$</td>
<td>$r_9 = 4$</td>
<td>$d_9 = 6$</td>
</tr>
</tbody>
</table>

In a feasible schedule (See Fig. 2), containers 3, 4, 5 and 6 are delivered by ship 1, containers 1 and 2 are delivered by ship 2, and containers 7, 8 and 9 are delivered by ship 3. The maximum tardiness is determined by containers 5 and 6 and its value is two.

This paper is organized as follows. In Sections 2, 3 and 4, we study the case of identical routes, nested routes, and arbitrary routes, respectively. We determine the computational complexity of most cases in our framework. We also provide in Section 2 a simple heuristic for some cases, the Minimum Slack first (MS) rule, with a worst case analysis. In Section 5, we discuss the relationships between our container scheduling problems and other machine scheduling problems. Finally, we conclude the paper in Section 6 with some extensions of the results and a discussion of future research directions.

The results obtained are summarized in the table below.
2 Identical Routes

In this section, we show that \( I, X | r_j = 0, u_j, s_j = 1 | T_{\text{max}} \), with the number of ports being a variable, is strongly NP-hard. We subsequently consider the case where all containers have the same port of origination but are released at different points in time and we show that both \( I, N_p | r_j, u_j = 1, s_j = \{1, 2\} | T_{\text{max}} \) and \( I, X | r_j, u_j = 1, s_j = \{1, 2\} | T_{\text{max}} \) can be solved in polynomial time. We conclude this section with an analysis of a heuristic that is based on the Minimum Slack first rule and provide a worst case analysis.
2.1 Strong NP-hardness Result

In order to prove a strong NP-hardness result, we introduce the following variant of the Interval Scheduling Problem.

Hierarchical Interval Scheduling with $T$ Machine Types (HIS($T$)): Given $m_i$ machines and $n_i$ jobs of type $i$, $i = 1, \ldots, T$, such that

- $m = \sum_{i=1}^{T} m_i$ and $n = \sum_{i=1}^{T} n_i$,
- a job of type $i$ can only be processed on a machine of type $k$, when $1 \leq k \leq i$;
- each job $j$ should be processed within the interval $[b_j, e_j]$, where $b_j$ and $e_j$ are integers and $1 \leq b_j < e_j \leq 2n$, $j = 1, 2, \ldots, n$,

is there an assignment of the $n$ jobs to the $m$ machines such that no two jobs processed on the same machine overlap in time?

Kolen et al. (2007) have shown that HIS($T$) is strongly NP-hard even for $T = 3$.

**Theorem 1** Even when all containers are of the same size, the problem $I, X \mid r_j = 0, u_j, s_j = 1 \mid T_{\text{max}}$ is strongly NP-hard.

**Proof** The proof of this theorem is relegated to the Appendix.

2.2 Polynomially Solvable Cases

In this subsection, we show that $I, X \mid r_j, u_j = 1, s_j \in \{1, 2\} \mid T_{\text{max}}$ can be solved in polynomial time if either one or both of the following two conditions hold:

(i) The origination ports of all containers are the same, i.e., $u_j = 1$, $j = 1, 2, \ldots, n$.
(ii) The destination ports of all containers are the same, i.e., $v_j = N_p$, $j = 1, 2, \ldots, n$.

To solve $I, X \mid r_j, u_j = 1, s_j \in \{1, 2\} \mid T_{\text{max}}$, we can conduct a binary search in a set of possible values for the optimal $T_{\text{max}}$, just like in the proof of Theorem 1. For each value $\lambda$ obtained in the binary search, we want to determine whether there is a schedule $\sigma$ such that $T_{\text{max}}(\sigma) \leq \lambda$. We now consider the following decision problem.

The cases satisfying either (i) or (ii) can be solved in polynomial time by the same algorithm that solves the case for $N_p = 2$, i.e., $I, 2 \mid r_j, u_j = 1, s_j \in \{1, 2\} \mid T_{\text{max}}$. (We will verify the above assertion later.) For the time being, we focus on the decision version of $I, 2 \mid r_j, u_j = 1, s_j \in \{1, 2\} \mid T_{\text{max}}$. 
For a given instance of the decision version of \( I, 2 \mid r_j, u_j = 1, s_j \in \{1, 2\} \mid T_{\text{max}} \), let \( \mathcal{I}(\lambda) \) denote an instance obtained from \( I, 2 \mid r_j, u_j = 1, s_j \in \{1, 2\} \mid T_{\text{max}} \) by modifying \( d_j \) to \( d_j + \lambda, j = 1, 2, \ldots, n \). It is clear that there is a schedule \( \sigma \) with \( T_{\text{max}}(\sigma) \leq \lambda \) in \( I, 2 \mid r_j, u_j = 1, s_j \in \{1, 2\} \mid T_{\text{max}} \) if and only if there is a schedule \( \tilde{\sigma} \) with \( T_{\text{max}}(\tilde{\sigma}) = 0 \) in \( \mathcal{I}(\lambda) \). From now on, we will be considering the decision version of \( \mathcal{I}(\lambda) \).

Let the containers be classified as follows: let \( J^1_{h,i} \) and \( J^2_{h,i} \) denote the sets of containers of sizes 1 and 2, respectively, that are to be assigned to ships \( h, h+1, \ldots, i \). In Fig. 3, the square nodes indicate ships and the round nodes indicate \( J^1_{h,i} \). The existence of a path between \( J^1_{h,i} \) and square node \( k \) means that the containers in \( J^1_{h,i} \) can be loaded onto ship \( k, k = h, h+1, \ldots, i \). The structure of the set of containers of size 2 can be illustrated similarly as in Fig. 3. Without loss of generality, we may assume that \( \mathcal{I}(\lambda) \) satisfies \( \sum_{i=1}^{m} c_i = n_1 + 2n_2 \); otherwise, we can add \( (n_1 + 2n_2 - \sum_{i=1}^{m} c_i) \) dummy containers of size 1 to \( J^1_{1:m} \).

![Fig. 3. The structure of the set of containers of size 1](image)

We will be constructing an undirected graph \( G = (N, E) \) with \( N \) being the set of nodes and \( E \) being the set of edges. The nodes are of the form \( N(i, a_i, b_i) \) with \( 0 \leq i \leq m \) and \( a_i \) and \( b_i \)
being integer. $N(i, a_i, b_i)$ indicates that $a_i$ containers of size 1 and $b_i$ containers of size 2 have been assigned to ships $1, 2, …, i$; see Fig. 4. Let $a_0 = b_0 = 0$. Let $N(0, 0, 0)$ and $N(m, n_1, n_2)$ be the only nodes for the case of $i = 0$ and $i = m$, respectively. $N(k, a_k, b_k)$ can be defined only if the following conditions (node-feasibility conditions) are satisfied:

- $|\bigcup_{1 \leq h \leq i \leq k} J_{h,i}^1| \leq a_k \leq n_1 - |\bigcup_{k+1 \leq h \leq m} J_{h,i}^1|$.
- $|\bigcup_{1 \leq h \leq i \leq k} J_{h,i}^2| \leq b_k \leq n_2 - |\bigcup_{k+1 \leq h \leq m} J_{h,i}^2|$.
- $a_k + 2b_k = \sum_{i=1}^{k} c_i$.

Let $N(k, a_k, b_k)$ be connected to $N(l, a_l, b_l)$, $k < l$, if the following two conditions (arc-feasibility conditions) are satisfied:

- $a_l - a_k \geq |\bigcup_{k+1 \leq h \leq l} J_{h,i}^1|$ and $b_l - b_k \geq |\bigcup_{k+1 \leq h \leq l} J_{h,i}^2|$.

![Fig. 4. The reduced clique problem](image)

Note that the resulting graph has $(m + 1)$ layers; nodes in the same layer are not connected. The graph can be viewed as an $(m + 1)$-partite graph. Our goal is to find a clique $N'$ of size $m + 1$ in the $(m + 1)$-partite graph. Note that $N'$ must contain exactly one node from each layer.
Lemma 1 There is a clique $N'$ of size $m + 1$ in the $(m+1)$-partite graph if and only if there is a feasible schedule in $\mathcal{I}(\lambda)$.

Proof The proof of this lemma is relegated to the Appendix.

Lemma 1 shows that finding a feasible schedule for $\mathcal{I}(\lambda)$ reduces to the problem of finding a clique of size $m + 1$ in the $(m+1)$-partite graph. Unfortunately, the problem of finding a clique of size $k$ in an arbitrary $k$-partite graph has been proven to be strongly NP-hard by Stoller and Schneider (1995). Fortunately, the clique problem with our $(m+1)$-partite graph has some nice properties that will enable us to develop a polynomial-time algorithm. For our $(m+1)$-partite graph, we assume that nodes at the same level are in increasing order of the $a_k$ values. That is, if $\bar{a}_k < \hat{a}_k$, then $N(k, \bar{a}_k, \bar{b}_k)$ appears to the left of $N(k, \hat{a}_k, \hat{b}_k)$; see Fig. 4.

Note that there is no tie of $a_k$ values in any layer since $\sum a_k + 2 \sum b_k = \sum_{i=1}^{k} c_i$.

Lemma 2 In the $(m+1)$-partite graph, let $\bar{a}_k < \hat{a}_k$ and $\bar{a}_l < \hat{a}_l$, where $k < l$. Suppose that $N(k, \bar{a}_k, \bar{b}_k)$ and $N(k, \hat{a}_k, \hat{b}_k)$ are connected to $N(l, \hat{a}_l, \hat{b}_l)$ and $N(l, \bar{a}_l, \bar{b}_l)$, respectively. Then $N(k, \bar{a}_k, \bar{b}_k)$ is also connected to $N(l, \bar{a}_l, \bar{b}_l)$.

Proof We will show that $N(k, \bar{a}_k, \bar{b}_k)$ is connected to $N(l, \hat{a}_l - 2, \hat{b}_l + 1)$. Observe that $\bar{a}_l \leq \hat{a}_l - 2$ and $\bar{b}_l \geq \hat{b}_l + 1$. Then,

$$\hat{a}_l - 2 - \bar{a}_k \geq \hat{a}_l - \bar{a}_k > \hat{a}_l - \bar{a}_k \geq \bigcup_{k+1 \leq b \leq i} J^{1}_{h, i},$$

$$\hat{b}_l + 1 - \bar{b}_k \geq \hat{b}_l - \bar{b}_k \geq \bigcup_{k+1 \leq b \leq i} J^{2}_{h, i},$$

and

$$(\hat{a}_l - 2 - \bar{a}_k) + 2(\hat{b}_l + 1 - \bar{b}_l) = (\hat{a}_l - \bar{a}_k) + 2(\hat{b}_l - \bar{b}_l) = \sum_{i=k+1}^{l} c_i.$$

Thus, $N(k, \bar{a}_k, \bar{b}_k)$ is connected to $N(l, \hat{a}_l - 2, \hat{b}_l + 1)$. The lemma is shown by applying the above argument repeatedly.

Based on Lemma 2, we can construct an algorithm to find a clique of size $m + 1$ in the $(m+1)$-partite graph if possible. For $i = 1, 2, ..., m$, let $L_i$ be the set of nodes in layer $i$ that are connected to $N(0, 0, 0)$.

Algorithm CLIQUE

Step 1 Initialization

- Construct the $(m+1)$-partite graph.
Let \( L = \{N(0, a_0, b_0), N(1, a_1, b_1), \ldots, N(m, a_m, b_m)\} \) be the set of the left most nodes from each layer, where \( N(i, a_i, b_i) \) is the left most node in \( L_i \) for \( i = 0, \ldots, m \).

**Step 2** Evaluating and Updating \( L \)

**Step 2.a** If \( L \) is a clique, then STOP. Otherwise, we can find two layers \( k \) and \( l \) such that \( N(k, a_k, b_k) \) and \( N(l, a_l, b_l) \) are not connected to each other.

**Step 2.b** If \( N(k, a_k, b_k) \) is not linked to any node after \( N(l, a_l, b_l) \) in \( L_l \), then
- If \( \{N(k, a_k + 2, b_k - 1)\} \) is the element of \( L_k \), then \( L = L \cup \{N(k, a_k + 2, b_k - 1)\} \setminus \{N(k, a_k, b_k)\} \).
- Otherwise, INFEASIBLE (there is no clique of size \( m + 1 \)).

**Step 2.c** If \( N(l, a_l, b_l) \) is not linked to any node after \( N(k, a_k, b_k) \) in \( L_k \), then
- If \( \{N(l, a_l + 2, b_l - 1)\} \) is the element of \( L_l \), then \( L = L \cup \{N(l, a_l + 2, b_l - 1)\} \setminus \{N(l, a_l, b_l)\} \).
- Otherwise, INFEASIBLE (there is no clique of size \( m + 1 \)).

**Step 3** Go to Step 2.

In Steps 2.b and 2.c, Algorithm CLIQUE always deletes either \( N(k, a_k, b_k) \) or \( N(l, a_l, b_l) \) or both. Suppose Algorithm CLIQUE does not delete \( N(k, a_k, b_k) \) nor \( N(l, a_l, b_l) \), then, by Lemma 2, \( N(k, a_k, b_k) \) and \( N(l, a_l, b_l) \) should also be connected to each other, which is a contradiction.

In Step 1, to construct an initial graph requires \( O(mn^2) \) time. Note that the number of nodes in each layer is bounded by \( n \) and the total number of nodes and edges are bounded by \( O(mn) \) and \( O(mn^2) \), respectively. In Step 2.a, checking the existence of a clique of size \( m + 1 \) requires \( O(m^2) \) time and if a clique is not constructed the algorithm identifies two layers such that their left most nodes are not connected to each other. Through Steps 2.b and 2.c, we update \( L \) if possible and it takes constant time. Step 2 is repeated at most \( O(mn) \) times. Therefore, Algorithm CLIQUE will be terminated in \( O(mn^2 + m^3n) \) time.

**Theorem 2** If the number of ports is 2, then the problem \( I, 2 \mid r_j, u_j = 1, s_j \in \{1, 2\} \mid T_{\text{max}} \) can be solved in polynomial time.

**Proof** We can solve \( I(\lambda) \) in \( O(mn^2 + m^3n) \) time by Algorithm CLIQUE. To obtain an optimal value of \( T_{\text{max}} \), we need to conduct a binary search with \( O(\log mn) \) iterations. By Lemma 1, it requires \( O(n + m^2) \) time to construct a schedule from the obtained clique. Thus, \( I, 2 \mid r_j, u_j = 1, s_j \in \{1, 2\} \mid T_{\text{max}} \) can be solved in \( O((mn^2 + m^3n) \log mn) \) time. \( \blacksquare \)

**Theorem 3** The problem \( I, X \mid r_j, u_j = 1, s_j \in \{1, 2\} \mid T_{\text{max}} \) can be solved in polynomial time. If the condition \( u_j = 1 \) is replaced by the condition \( v_j = N_p \), the problem can be solved in polynomial time as well.
Proof We can classify the containers into $J_{hi}^1$ and $J_{hi}^2$ for $1 \leq h \leq i \leq n$ in $I, X \mid r_j, u_j = 1, s_j \in \{1, 2\} \mid T_{\text{max}}$ just like in the two-port case. The number of containers assigned to each ship is monotonically increasing or decreasing while each ship departs from node 1 or arrives at node $N_p$. This implies that the capacity constraint of each ship can be considered just like in the two-port case. Thus, we can solve $I, X \mid r_j, u_j = 1, s_j \in \{1, 2\} \mid T_{\text{max}}$ in polynomial time as well as the problem with condition $u_j = 1$ replaced by condition $v_j = N_p$. ■

The result of Theorem 2 can be extended to a more general setting. Assume again identical routes. Each container has an origination port in one continent (for example, Asia), but the origination ports of the containers are now not necessarily the same. The destination port of each container is located in another continent (for example, North America). Each ship visits port $u$ as its last port in the first continent and port $v$ as the first port in the second continent. More formally, there exist a $u$ and $v$ such that $u_j \leq u < v \leq v_j$ for all $j$. In this case, it is enough to check the capacity constraint on the path from the last port in the first continent to the first port in the second continent. Therefore, if we set a target objective function value, the available ships for the assignment of each container always form an interval of ships dependent upon the release date and due date of the container. This implies that all complexity results for problems with identical routes and a common origination port are exactly the same as those for problems with identical routes and a conjoint path.

2.3 A Heuristic Based on the Minimum Slack first Rule

The polynomial-time algorithms described in Sections 2.2 and 4.2 have a very high running time. Therefore, they are more of theoretical interest than of practical interest. In this section, we consider a heuristic that is very fast, although it may not produce an optimal schedule. We will show that the heuristic may produce poor results for $I, X \mid r_j = 0, u_j, s_j = 1 \mid T_{\text{max}}$, when the containers have different ports of origination and different ports of destination. However, if either $u_j = 1$ for $j = 1, 2, ..., n$ or $v_j = N_p$ for $j = 1, 2, ..., n$, then there is a worst-case absolute bound for the heuristic, relative to the optimal solution. The heuristic can be described as follows.

Let $(t_{i,v_j} - d_j)$ denote the slack of container $j$ if it is assigned to ship $i$. The ships are ranked in increasing order of their departure time from port 1. In order to load ship $i$, rank all the available containers in increasing order of their slack times. Load ship $i$ by applying the Minimum Slack first (MS) rule among the available containers. (Note that this rule does not take the sizes of the containers into account.) Let $T_{\text{max}}(OPT)$ denote the value of the objective function under the optimal assignment and let $T_{\text{max}}(MS)$ denote the value of the objective function under the MS rule.
In order to guarantee a feasible schedule by the MS rule and to obtain some insight into the rule, we make the following assumptions.

- \( m \) is sufficiently large to load all containers.
- All ships have the same capacity \( c \), i.e. \( c_i = c \) for \( i = 1, \ldots, m \).
- The inter-arrival times between consecutive ships at port \( l \) are all equal to \( \Delta \), i.e., \( t_{i,l} = t_{1,l} + (i-1)\Delta \) for \( i = 2, \ldots, m \) and \( l = 1, \ldots, N_p \).

We will show that the MS rule performs poorly for \( I, X | r_j = 0, u_j, s_j = 1 | T_{\text{max}} \), when the containers have different ports of origination and destination. Consider the following example. There are \( 2^k \) ports, \( k \) being a positive integer. Ship \( i \) visits port \( l \) at time \( t_{i,l} = (i + l)\Delta \), \( i = 1, 2, \ldots, m \) and \( l = 1, 2, 3, \ldots, 2^k \). The containers all have release dates equal to 0. Let \((g, h, j)\) denote container \( j \) which belongs to group \((g, h)\), where \( 0 \leq g \leq k - 1, 1 \leq h \leq 2^{k-1} \) and \( 1 \leq j \leq 2^{k-2}c \).

For \( g = 0, h = 1, 2, 3, \ldots, 2^{k-1} \) and \( j = 1, 2, \ldots, c \), we have

\[
u_{0,h,j} = 2h - 1, \quad v_{0,h,j} = 2h, \quad \text{and} \quad d_{0,h,j} = v_{0,h,j}\Delta - (2^{k-1} - 1)\epsilon.
\]

For \( g = 1, 2, \ldots, k - 1, h = 1, 2, 3, \ldots, 2^{k-g} \) and \( j = 1, 2, 3, \ldots, 2^{g-1}c \), we have

\[
u_{g,h,j} = \begin{cases} 2^g(h-1) + 1 & \text{if } h \text{ is odd} \\ 2^g(h-1) & \text{otherwise,} \end{cases} \quad v_{g,h,j} = \begin{cases} 2^g(h-1) + 1 + 2^g & \text{if } h \text{ is odd} \\ 2^g(h-1) + 2^g & \text{otherwise,} \end{cases}
\]

\[
\text{and } d_{g,h,j} = v_{g,h,j}\Delta - \left(2^{k-1} - (v_{g,h,j} - u_{g,h,j})\right)\epsilon,
\]

where \( \epsilon \) is a sufficiently small positive value. Note that the containers in group \((g, h)\) are assigned in increasing order of \( g \), because for container \( j \) belonging to group \((g, h)\),

\[
t_{i,j} - d_j = i\Delta + (2^{k-1} - 2^g)\epsilon.
\]

As an example, for \( k = 4 \), we have the following instance.
The optimal algorithm assigns all containers to $2^{k-1}$ ships with $T_{\text{max}}(OPT) = 0$, while the MS rule assigns all containers to $(2^k - 1)$ ships with $T_{\text{max}}(MS) = (2^k - 1)\Delta$; see Fig. 6 for the two schedules where $c = 1$. Thus, $T_{\text{max}}(MS) - T_{\text{max}}(OPT) \geq (2^k - 1)\Delta$.

From the above, we observe that the MS rule requires almost twice as many ships as the optimal rule and the performance of the MS rule gets worse when $k$ and $\Delta$ increase.

We now analyze the performance of the MS rule for $I, X | r_j, s_j \in \{1, 2\} | T_{\text{max}}$, when either $u_j = 1$, $j = 1, 2, ..., n$ or $v_j = N_p, j = 1, 2, ..., n$. For convenience, we refer to this problem as Problem A. Let $I, 2 | r_j, u_j = 1, s_j \in \{1, 2\} | T_{\text{max}}$ be denoted as Problem B.

**Lemma 3** Let $\mathcal{I}$ be an instance of problem A. If the absolute bound of the MS rule for $\mathcal{I}$ is $\tau^*$, then there is an instance $\mathcal{I}'$ of problem B for which the absolute bound of the MS rule for $\mathcal{I}'$ is also $\tau^*$.

**Proof** Let $T_j(MS)$ denote the tardiness of container $j$ for $\mathcal{I}$ under the MS rule. There are two cases to consider.

**Case 1:** $u_j = 1$, $j = 1, 2, ..., n$.

We construct an instance $\mathcal{I}'$ of Problem B as follows.

$u'_j = 1, v'_j = 2, r'_j = r_j, \text{ and } d'_j = d_j - (t_{1,v_j} - t_{1,2}) \text{ for } j = 1, 2, ..., n.$

The priorities of the containers in $\mathcal{I}$, under the MS rule, are in decreasing order of $t_{i,v_j} - d_j$. 

<table>
<thead>
<tr>
<th>$g$</th>
<th>$h$</th>
<th>No. of containers</th>
<th>$u_{g,h,j}$</th>
<th>$v_{g,h,j}$</th>
<th>$d_{g,h,j}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>$c$</td>
<td>1</td>
<td>2</td>
<td>$2\Delta - 7\epsilon$</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>$c$</td>
<td>3</td>
<td>4</td>
<td>$4\Delta - 7\epsilon$</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>0</td>
<td>8</td>
<td>$c$</td>
<td>15</td>
<td>16</td>
<td>$16\Delta - 7\epsilon$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$c$</td>
<td>1</td>
<td>3</td>
<td>$3\Delta - 6\epsilon$</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>$c$</td>
<td>2</td>
<td>4</td>
<td>$4\Delta - 6\epsilon$</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>$c$</td>
<td>14</td>
<td>16</td>
<td>$16\Delta - 6\epsilon$</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>$2c$</td>
<td>1</td>
<td>5</td>
<td>$5\Delta - 4\epsilon$</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>$2c$</td>
<td>4</td>
<td>9</td>
<td>$9\Delta - 4\epsilon$</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>$2c$</td>
<td>9</td>
<td>13</td>
<td>$13\Delta - 4\epsilon$</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>$2c$</td>
<td>12</td>
<td>16</td>
<td>$16\Delta - 4\epsilon$</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>$4c$</td>
<td>1</td>
<td>9</td>
<td>$9\Delta$</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>$4c$</td>
<td>8</td>
<td>16</td>
<td>$16\Delta$</td>
</tr>
</tbody>
</table>
Fig. 5. An illustrative example of the MS rule for $2^4$ ports case.

The priorities of the containers in $I'$, under the MS rule, are in decreasing order of

$$t_{i,v'_j} - d'_j = t_{i,2} - (d_j - (t_{1,v_j} - t_{1,2})) = t_{1,v_j} + (t_{i,2} - t_{1,2}) - d_j = t_{i,v_j} - d_j.$$

Therefore, the priorities of the containers are in both cases the same. Since the release time of the containers is the same in both cases, they are assigned to the same ship in both cases. Let $\sigma'(MS)$ denote the schedule obtained with the MS rule for $I'$. Then, we have

$$T_j(MS) = \max\{0, t_{1,v_j} + \Delta(i - 1) - d_j\} \text{ and}$$

$$T_j(\sigma'(MS)) = \max\{0, t_{1,v'_j} + \Delta(i - 1) - d'_j\} = \max\{0, t_{1,v_j} + \Delta(i - 1) - d_j\}.$$
Thus, $T_j(\sigma) = T_j(\sigma')$, $j = 1, 2, \ldots, n$.

**Case 2:** $v_j = N_p$, $j = 1, 2, \ldots, n$.

We construct an instance $I'$ of Problem B as follows.

$$u'_j = N_p - 1, v'_j = N_p, r'_j = r_j + (t_{1,N_p-1} - t_{1,u_j}) \quad \text{and} \quad d'_j = d_j, j = 1, 2, \ldots, n.$$  

Note that the source port of all containers is $N_p - 1$ and the destination port of all containers is $N_p$. Therefore, this is a two-port instance. In $I$, $r_j \leq t_{i,u_j} + \Delta$ if and only if $r'_j = r_j + (t_{1,N_p-1} - t_{1,u_j}) \leq t_{i,N_p-1} + \Delta$ in $I'$. In other words, container $j$ can be loaded onto ship $i$ in $I$ if and only if container $j$ can be loaded onto ship $i$ in $I'$. In addition, $t_{i,v_j} - d_j = t_{i,N_p} - d_j = t_{i,v'_j} - d'_j$, $j = 1, 2, \ldots, n$. This implies that $\sigma = \sigma'$ and $T_j(\sigma) = T_j(\sigma')$, $j = 1, 2, \ldots, n$.

From Lemma 3, we see that a worst-case absolute bound for problem B also serves as a worst-case absolute bound for problem A. In the remaining part of this section we focus on $I, 2 | r_j, u_j = 1, s_j \in \{1, 2\} | T_{\text{max}}$ only.

**Theorem 4** For $I, 2 | r_j, u_j = 1, s_j \in \{1, 2\} | T_{\text{max}}$, we have

$$T_{\text{max}}(MS) \leq T_{\text{max}}(OPT) + \left\lceil \frac{m' - 1}{c - 1} \right\rceil \Delta,$$

where $m' = \left\lceil \frac{n_1 + 2n_2}{c} \right\rceil$ and this absolute error bound is tight.

**Proof** We will prove the theorem by contradiction. Let $I'$ denote the smallest counterexample that violates the bound; i.e., $I'$ has the smallest number of containers. Let container $n$ be the one with the largest due date, i.e., $d_n = \max\{d_j | 1 \leq j \leq n\}$ and ship $h$ be the one that delivers container $n$. Let $\bar{c}_i$ be the remaining capacity of ship $i$ under the MS rule. Since $I'$ is the smallest counterexample, we may assume that $I'$ satisfies the following conditions.

(i) $T_{\text{max}}(OPT) = 0$.
(ii) $t_{h,2} - d_n = T_{\text{max}}(MS)$.
(iii) $\bar{c}_i \leq 1$, $i = 1, 2, \ldots, h - 1$.

If (i) fails to hold, we can consider another instance with due date defined by $d_j + T_{\text{max}}(OPT)$. This will reduce $T_{\text{max}}(OPT)$ and $T_{\text{max}}(MS)$ by the same amount, namely $T_{\text{max}}(OPT)$. If (ii) fails, then we can delete container $n$ to obtain a smaller counterexample. If (iii) fails, then there is a ship $h'$, $h' < h$, such that $\bar{c}_{h'} \geq 2$. This means that the containers loaded onto ship
i, i = h' + 1, h' + 2, ..., h have release dates greater than \( t_{h',1} \). Thus, we can construct a smaller counterexample by deleting the containers loaded onto ship \( i, i = 1, 2, ..., h' \). Henceforth, we will assume that \( I' \) satisfies the above three conditions. Note that by (ii), ship \( h \) is the last ship used by the MS rule. Otherwise, we can delete all containers that are assigned to ships \( h + 1, h + 2, \ldots, m \) and construct a smaller counterexample.

Note that the optimal algorithm uses at least \( m' \) ships. If \( d_n < t_{m',2} \), then all containers have due dates less than \( t_{m',2} \). It implies that in the optimal schedule the containers assigned to ship \( m' \) or later has tardiness greater than 0, which contradicts (i). Thus, \( d_n \geq t_{m',2} \).

Since \( t_{h,2} - d_n > \left\lceil \frac{m' - 1}{c - 1} \right\rceil \Delta \) and \( d_n \geq t_{m',2} \), we have

\[
t_{h,2} > d_n + \left\lceil \frac{m' - 1}{c - 1} \right\rceil \Delta \geq t_{m',2} + \left\lceil \frac{m' - 1}{c - 1} \right\rceil \Delta.
\]

This implies that container \( n \) cannot be assigned to the first \( m' + \left\lceil \frac{m' - 1}{c - 1} \right\rceil \) ship in the schedule generated by the MS rule.

We now consider two cases, depending on the size of container \( n \).

**Case 1:** \( s_n = 2 \).

Since container \( n \) is loaded after ship \( m' + \left\lceil \frac{m' - 1}{c - 1} \right\rceil \), we have

\[
n_1 + 2n_2 \geq \left( m' + \left\lceil \frac{m' - 1}{c - 1} \right\rceil \right)(c - 1) + 2 \geq m'c + 1,
\]

which contradicts the definition of \( m' \).

**Case 2:** \( s_n = 1 \).

In this case, we consider two cases concerning the remaining capacity of ship \( m' + \left\lceil \frac{m' - 1}{c - 1} \right\rceil \). If the remaining capacity is zero, then we obtain the following inequalities:

\[
n_1 + 2n_2 \geq \left( m' + \left\lceil \frac{m' - 1}{c - 1} \right\rceil - 1 \right)(c - 1) + c + 1 \geq m'c + 1,
\]

which contradicts the definition of \( m' \). On the other hand, if the remaining capacity of ship \( m' + \left\lceil \frac{m' - 1}{c - 1} \right\rceil \) is one, then the release date of container \( n \) should be greater than the departure time of ship \( m' + \left\lceil \frac{m' - 1}{c - 1} \right\rceil \). Because the total size of all containers assigned to ships
1, \ldots, m' + \left\lceil \frac{m'-1}{c-1} \right\rceil \text{ is at least }

\left( m' + \left\lceil \frac{m'-1}{c-1} \right\rceil \right) (c-1) \geq m'c - 1,

all containers, except container \( n \), are assigned to ships 1 to \( m' + \left\lceil \frac{m'-1}{c-1} \right\rceil \). Thus, container \( n \) must be assigned to the first available ship after ship \( m' + \left\lceil \frac{m'-1}{c-1} \right\rceil \). But the optimal algorithm must also assign container \( n \) to the first available ship after ship \( m' + \left\lceil \frac{m'-1}{c-1} \right\rceil \), contradicting (i).

So in all cases, we have a contradiction. Therefore, \( T' \) does not exist and we have \( T_{\text{max}}(MS) \leq T_{\text{max}}(OPT) + \left\lceil \frac{m'-1}{c-1} \right\rceil \Delta \).

Fig. 6. The worst case example for the MS rule in case of a common destination

To show that the bound is tight, consider the following example. Suppose \( c \) and \( m' \) are odd integers. For convenience we call a container of size 1 a small container and a container of size 2 a big container. In the example, the number of small containers, \( n_1 \) is \( m' - 1 \) and the
number of big containers, \( n_2 \) is \( m'\left(\frac{c-1}{2}\right) \).

There are \( \frac{m'+1}{2} \) groups of containers. In group \( g \) for \( g = 1, \ldots, \frac{m'-1}{2} \), there are two small containers and \( \left(\frac{c-3}{2}\right) \) big containers with common due date \((\frac{m'-1}{2} + g)\Delta\). The small containers have identical release date \( g\Delta - 1 \) and the big containers have release date 0. In group \( \frac{m'+1}{2} \), there are \( m'\left(\frac{c-1}{2}\right) - \left(\frac{c-3}{2}\right)\left(\frac{m'-1}{2}\right) \) big containers with release date 0 and due date \( m\Delta \).

In the optimal schedule, ship \( i \) for \( i = 1, \ldots, m' - 1 \) has one small container and \( \frac{c-1}{2} \) big containers. Ship \( m' \) has \( \frac{c-1}{2} \) big containers and \( T_{\text{max}}(\sigma^*) = 0 \). Under the MS rule, all containers in group \( g \), for \( g = 1, \ldots, \frac{m'-1}{2} \), are assigned to ship \( g \). The remaining containers are assigned to ships \( \frac{m'+1}{2} \) to \( m' + \left[\frac{m'-1}{c-1}\right] \), resulting in \( T_{\text{max}}(\sigma^G) = \left[\frac{m'-1}{c-1}\right] \Delta \).

The structure of the worst case bound in Theorem 4 is actually quite intuitive. As the absolute error is a multiple of \( \Delta \), the coefficient of \( \Delta \) shows how poor the heuristic can be. Since \( n_1 + 2n_2 \) represents the total size of containers and their sizes and \( c \) is the common capacity of the ships, \( m' \) approximately represents the number of ships used, implying the problem size. Therefore, we can observe a tendency that when the problem size (the number of ships being used) increases, the absolute error increases almost proportionally, whereas when the capacity of the ships increases the absolute error decreases at a rate \( \frac{1}{c} \). This is in agreement with our intuition that says that a problem with a small number of large capacity ships is easier to analyze than a problem with a larger number of small capacity ships, since in the case of smaller ships the sizes of the containers may have a greater impact on the quality of the solution. In practice, it also may imply that the solution by the MS rule is close enough to the optimal solution as modern container ships can carry up to 15,000 TEU and even a feeder ship has an average capacity of 300 to 500 TEU.

3 Nested Routes

In this section, we consider the case of nested routes, which can be defined as follows: There are \( k \) route types \( \{\tilde{\rho}^1, \ldots, \tilde{\rho}^k\} \) with \( \tilde{\pi}^k \subset \tilde{\pi}^{k-1} \subset \cdots \subset \tilde{\pi}^1 \), where \( \tilde{\pi}^l \) represents the set of ports that are part of route \( \tilde{\rho}^l \), \( l = 1, 2, \ldots, k \). Without loss of generality, \( \tilde{\pi}^1 = \{1, 2, \ldots, m\} \). Note that \( \tilde{\rho}^l \) is different from \( \rho^l \). This case can be described as \( N, X \mid r_j = 0, u_j, s_j \in \{1, 2\} \) | \( T_{\text{max}} \).

Since identical routes are a special case of nested routes, this case is also strongly NP-hard. Thus, we show that if all containers have either the same origination port or the same destination port, then the problem is polynomially solvable when all containers are released at the same time. Since the same argument can be applied to the case of an identical origination port and to the case of an identical destination port, we restrict ourselves to the case of an
identical origination port. The next lemma is instrumental in showing this result.

**Lemma 4** For \( N, X \mid r_j = 0, u_j = 1, s_j \in \{1, 2\} \mid T_{\text{max}} \), with a common origination port, the problem of deciding whether there exists a schedule \( \sigma \) with \( T_{\text{max}} \leq \lambda \) is solvable in polynomial time when the number of different route types \( (k) \) is fixed.

**Proof** Let \( J_t^l \) be the set of containers of size \( s \) whose destination ports belong to \( \pi^l \setminus \pi^{l+1} \), \( l = 1, 2, \ldots, k \), where \( \pi^{k+1} = \emptyset \). Then, it is observed that containers in \( J_t^l \cup J_t^2 \) can be delivered by ships with routes in \( \{\rho^1, \ldots, \rho^s\} \), \( l = 1, 2, \ldots, k \). Let

\[
J^s_{l,i} = \{ j \in J^s_l | t_{i,v_j} - d_j \leq \lambda \}, \quad \text{where} \quad t_{i,v_j} = \infty \text{ if } v_j \notin \pi^i.
\]

Note that \( J^s_{l,i} = \emptyset \) if \( j \in J^s_l \) and \( v_j \notin \pi^i \). Then, \( J^s_l = \bigcup_{i=1}^m J^s_{l,i}, s = 1, 2 \). By our assumptions of unidirectional routes and non-overtaking routes, there is an optimal schedule such that for each ship a container with an earlier due date is assigned before a container with a later due date.

Let \( N(i, (a_1, b_1), \ldots, (a_k, b_k)) \) denote a node that represents the following: \( a_l \) containers in \( J^l_t \) and \( b_l \) containers in \( J^2_t \) have been assigned to ships \( 1, 2, \ldots, l \), \( l = 1, 2, \ldots, k \). For convenience, let this node be abbreviated as \( N(i, [a, b]) \), i.e., \([a, b] \equiv (a_1, b_1), \ldots, (a_k, b_k)\). Let \( N(0, [0, 0]) \) and \( N(m, [J^1, J^2]) \) be the source and sink nodes, respectively. Consider that \( \rho^{l+1} = \rho^\infty \). Then, let \( N(i, [a, b]) \) be connected to \( N(i+1, [\hat{a}, \hat{b}]) \) such that

\[
\begin{align*}
\cdot & \hat{a}_l = a_l \text{ and } \hat{b}_l = b_l, \text{ for } l = 1, 2, \ldots, \alpha - 1, \\
\cdot & \sum_{h=1}^{i+1} |J^1_{l,h}| \leq \hat{a}_l, \sum_{h=1}^{i+1} |J^2_{l,h}| \leq \hat{b}_l, \text{ for } l = \alpha, \alpha + 1, \ldots, k, \\
\cdot & \sum_{l=\alpha}^{k} \left( (\hat{a}_l - a_l) + 2(\hat{b}_l - b_l) \right) \leq c_{i+1}.
\end{align*}
\]

Clearly, if there is a path between the source and sink nodes in the reduced graph, then there is a schedule \( \sigma \) such that \( T_{\text{max}}(\sigma) \leq \lambda \). The number of nodes in the reduced graph is at most \( O(mn^{2k}) \). Since the number of arcs emanating from each node is at most \( O(n^{2k}) \), the total number of arcs is at most \( O(mn^{4k}) \). We can now use the algorithm of Ahuja et al. (1990) to check the connectivity between the source and sink nodes. The running time of the algorithm becomes \( O(mn^{4k}) \).

**Theorem 5** If the number of different route types \( (k) \) is fixed and all containers have a common origination port, then \( N, X \mid r_j = 0, u_j = 1, s_j \in \{1, 2\} \mid T_{\text{max}} \) is solvable in polynomial time.

**Proof** We can conduct a binary search among all possible values of the optimal \( T_{\text{max}} \), denoted by \( T_{\text{max}}^* \). Note that \( T_{\text{max}}^* \) is in the set \( \{0\} \cup \{ t_{i,v_j} - d_j \mid 1 \leq j \leq n, 1 \leq i \leq m \} \). Thus, there
will be $O(\log mn)$ iterations in the binary search. For each value $\lambda$ in the binary search, we can use the algorithm described in Lemma 4 to decide whether there is a schedule $\sigma$ such that $T_{\text{max}}(\sigma) \leq \lambda$. Thus, the overall running time will be $O(mn^4k \log mn)$.

4 Arbitrary Routes

In this section we study the case of arbitrary routes. We first show that $A, X \mid r_j = 0, u_j, s_j \in \{1, 2\} \mid T_{\text{max}}$ is strongly NP-hard even when the destination port $v_j = N_p$ for all $j$, $j = 1, 2, ..., n$. We then show that $A, N_p \mid r_j = 0, u_j, s_j \in \{1, 2\} \mid T_{\text{max}}$ can be solved in polynomial time when the number of ports $N_p$ is fixed.

4.1 Strong NP-hardness Result

We will show that $A, X \mid r_j = 0, u_j, s_j \in \{1, 2\} \mid T_{\text{max}}$ is strongly NP-hard through a reduction from the following machine scheduling problem.

Parallel Machine Scheduling with Eligibility Constraints: Given $n$ jobs with processing times $p_j \in \{1, 2\}$ and eligibility constraint sets $M_j$, find a schedule that minimizes the makespan on $m$ parallel machines. Note that $M_j$ refers to the set of machines on which job $j$ may be processed.

The above scheduling problem has been shown to be strongly NP-hard by Glass and Kellerer (2007). Following the standard classification scheme in scheduling theory of Pinedo (2008), this problem can be denoted by $P \mid M_j, p_j \in \{1, 2\} \mid C_{\text{max}}$.

**Theorem 6** $A, X \mid r_j = 0, u_j, s_j \in \{1, 2\} \mid T_{\text{max}}$ is strongly NP-hard even if all containers have a common destination port.

**Proof** The proof of this theorem is relegated to the Appendix.

Note that, by symmetry, $A, X \mid r_j = 0, u_j = 1, s_j \in \{1, 2\} \mid T_{\text{max}}$ is strongly NP-hard even if all containers have a common origination port.

4.2 Polynomially Solvable Cases

In this subsection, we consider polynomially solvable cases with arbitrary routings. Note that the problem $A, X \mid r_j, u_j = 1, s_j = 1 \mid T_{\text{max}}$ can be solved in polynomial time, since its decision version problem can be modeled as an assignment problem.
We show that \( A, N_p \mid r_j = 0, u_j, s_j \in \{1, 2\} \mid T_{\text{max}} \) can be solved in polynomial time if the number of ports, \( N_p \), is fixed. The next lemma is instrumental in showing this result.

**Lemma 5** For \( A, N_p \mid r_j = 0, u_j, s_j \in \{1, 2\} \mid T_{\text{max}} \), the problem of deciding whether there is a schedule \( \sigma \) such that \( T_{\text{max}}(\sigma) \leq \lambda \) is solvable in polynomial time if \( N_p \) is fixed.

**Proof** Let \( J_{u,v}^s \) be the set of containers of size \( s \) with port \( u \) as origination and port \( v \) as destination, \( 1 \leq u < v \leq N_p \) and \( s = 1, 2 \). Because of our non-overtaking assumption, there is an optimal schedule such that for each ship, a container with an earlier due date is assigned before a container with a later due date. Let \( d_{u,v}^s(j) \) be the due date of the container with the \( j \)th smallest due date in \( J_{u,v}^s \). For convenience, let \( d_{u,v}^s(j) = \infty \) for \( j > |J_{u,v}^s| \), where \( |J_{u,v}^s| \) is the cardinality of \( J_{u,v}^s \).

Let \( N(i, (a_{1,2}, b_{1,2}), (a_{1,3}, b_{1,3}), \ldots, (a_{N_p-1,N_p}, b_{N_p-1,N_p})) \) denote a node that represents the following: \( a_{u,v} \) containers in \( J_{u,v}^1 \) and \( b_{u,v} \) containers in \( J_{u,v}^2 \) have been assigned to ships 1, 2, \ldots, \( i \), \( 1 \leq u < v \leq N_p \). Note that \( 0 \leq a_{u,v} \leq |J_{u,v}^1| \) and \( 0 \leq b_{u,v} \leq |J_{u,v}^2| \) for \( 1 \leq u < v \leq N_p \). For convenience, let the above node be abbreviated by \( N(i, [a_{u,v}, b_{u,v}]) \), i.e.,

\[
[a_{u,v}, b_{u,v}] = (a_{1,2}, (a_{1,3}, b_{1,3}), \ldots, (a_{N_p-1,N_p}, b_{N_p-1,N_p}).
\]

Let \( N(0, [0,0]) \) and \( N(m, |J_{u,v}^1|, |J_{u,v}^2|) \) be the source and destination nodes, respectively. Let \( N(i, [a_{u,v}, b_{u,v}]) \) be connected to \( N(i + 1, [\bar{a}_{u,v}, \bar{b}_{u,v}]) \) if the following three conditions are satisfied:

(i) \( \sum_{u=1}^{l} \sum_{v=l+1}^{N_p} \left( (\bar{a}_{u,v} - a_{u,v}) + 2(\bar{b}_{u,v} - b_{u,v}) \right) \leq c_{i+1} \) for \( l = 1, \ldots, N_p - 1 \).

(ii) In case ship \( i + 1 \) visits both ports \( u \) and \( v \), then

\begin{align*}
&\cdot \bar{a}_{u,v} - a_{u,v} \geq 0, \text{ and if } \bar{a}_{u,v} > a_{u,v}, \text{ then } t_{(i+1),v} - d_{u,v}^1(a_{u,v} + 1) \leq \lambda \text{ for } 1 \leq u < v \leq N_p. \\
&\cdot \bar{b}_{u,v} - b_{u,v} \geq 0, \text{ and if } \bar{b}_{u,v} > b_{u,v}, \text{ then } t_{(i+1),v} - d_{u,v}^2(b_{u,v} + 1) \leq \lambda \text{ for } 1 \leq u < v \leq N_p.
\end{align*}

(iii) In case ship \( i + 1 \) does not visit both ports \( u \) and \( v \), then

\begin{align*}
&\cdot \bar{a}_{u,v} = a_{u,v} \text{ and } \bar{b}_{u,v} = b_{u,v}.
\end{align*}

Condition (i) ensures that the total size of containers loaded onto ship \( i + 1 \) is not more than the capacity \( c_{i+1} \). Condition (ii) ensures that when at least one container in \( J_{u,v}^s \), \( s = 1 \) or 2, is delivered by ship \( i + 1 \), the tardiness of the container cannot exceed \( \lambda \). Condition (iii) ensures that if ship \( i + 1 \) does not visit both ports \( u \) and \( v \), no container in \( J_{u,v}^s \), \( s = 1 \) or 2, can be delivered by ship \( i + 1 \).

Clearly, if there is a path between the source and destination nodes in the reduced graph, then there is a schedule \( \sigma \) such that \( T_{\text{max}}(\sigma) \leq \lambda \). The number of nodes in the reduced
The number of arcs emanating from each node is at most \( O(mn^{N_p(N_p-1)}) \). Since the number of arcs is at most \( O(n^{N_p(N_p-1)}) \), the total number of arcs is at most \( O(mn^{2N_p(N_p-1)}) \). We can now use the algorithm of Ahuja et al. (1990) to check the connectivity between the source and destination nodes. The running time of the algorithm becomes \( O(mn^{2N_p(N_p-1)}) \).

**Theorem 7** If the number of ports \( N_p \) is fixed, then \( A, N_p \mid r_j = 0, u_j, s_j \in \{1, 2\} \mid T_{\text{max}} \) is polynomially solvable.

**Proof** We can conduct a binary search among all possible values of the optimal \( T_{\text{max}} \), denoted by \( T^*_\text{max} \). \( T^*_\text{max} \) is in the set \( \{0\} \cup \{t_{i,v_j} - d_j \mid 1 \leq j \leq n, 1 \leq i \leq m\} \). Thus, there will be \( O(\log mn) \) iterations in the binary search. For each value \( \lambda \) obtained in the binary search, we can use the algorithm described in Lemma 5 to decide whether there is a schedule \( \sigma \) such that \( T_{\text{max}}(\sigma) \leq \lambda \). Thus, the overall running time will be \( O(mn^{2N_p(N_p-1)} \log mn) \).

This result can be further generalized to the case without the unidirectional assumption. Then we define \( J^*_{u,v} \) for \( 1 \leq u \leq N_p, 1 \leq v \leq N_p \) and \( u \neq v \) and the solution structure \([a_{u,v}, b_{u,v}]\) can be defined for \( 1 \leq u \leq N_p, 1 \leq v \leq N_p \). The first connecting condition between \( N(i, [a_{u,v}, b_{u,v}]) \) and \( N(i+1, [a_{u,v}, b_{u,v}]) \) is changed as follows:

\[
\sum_{u \in \pi^{i+1}(l)} \sum_{v \in \pi^{i+1}(-l)} \left( (\bar{a}_{u,v} - a_{u,v}) + 2(\bar{b}_{u,v} - b_{u,v}) \right) \leq c_{i+1} \quad \text{for} \quad l = 1, \ldots, |\pi^{i+1}| - 1
\]

where \( \pi^{i+1}(l) = \bigcup_{k=1}^{l} \{\rho^{i+1}(k)\} \) and \( \pi^{i+1}(-l) = \pi^{i+1} \setminus \pi^{i+1}(l) \). In the second and third conditions, the case where ship \( i + 1 \) visits ports \( u \) and \( v \) and visits port \( u \) earlier than port \( v \) is under consideration. Then, the same almost argument is applied and the problem remains to be solved in polynomial time for fixed \( N_p \).

**5 Relationships with Other Scheduling Problems**

Interestingly, \( I, 2 \mid r_j, u_j = 1, s_j \in \{1, 2\} \mid T_{\text{max}} \) is closely related to various well-known scheduling problems, e.g., the multiprocessor task scheduling problem and the parallel machine scheduling problem with eligibility constraints. Brucker and Krämer (1996) introduced a scheduling problem in which job \( j \) requires simultaneously a number of machines for processing, the number of machines required being \( \text{size}_j \), and provided polynomial time algorithms for various objective functions. They also referred to several open problems, one of which being \( P \mid p_j = 1, r_j, \text{size}_j \in \{l_1, l_2, \ldots, l_k\} \mid L_{\text{max}} \) with \( k \) fixed.

We first analyze the relationship between \( I, 2 \mid r_j, u_j = 1, s_j \in \{1, 2\} \mid T_{\text{max}} \) and \( P \mid p_j = 1, r_j, \text{size}_j \in \{1, 2\} \mid L_{\text{max}} \). For any due date related objective function value, each container
(job) can go on any ship that belongs to a set of consecutive ships (time slots); this set of ships is determined by the release date of the container and by the objective function value. Also, each ship (time slot) has a capacity (the number of machines) that the total size of containers (jobs) allocated to the ship (time slots) cannot exceed. Intuitively, problem \( I, 2 | r_j, u_j = 1, s_j \in \{1, 2\} | T_{\text{max}} \) is equivalent to problem \( P | p_j = 1, r_j, \text{size}_j \in \{1, 2\} | L_{\text{max}} \). From Theorem 2 and this relationship, we can conclude that \( P | p_j = 1, r_j, \text{size}_j \in \{l_1, l_2, \ldots, l_k\} | L_{\text{max}} \) has been partially solved.

As a special case of \( P | p_j = 1, r_j, \text{size}_j \in \{l_1, l_2, \ldots, l_k\} | L_{\text{max}} \), the problem with two job sizes of 1 and \( m \), \( P | p_j = 1, r_j, \text{size}_j \in \{1, m\} | T_{\text{max}} \), referred to as scheduling Tall/Small Multiprocessor Tasks, has been studied in literature. Baptiste and Schieber (2003) proved that it can be solved in polynomial time by providing an algorithm with a complexity of \( O(n^4) \), and Durr and Hurand (2009) improved the time complexity to \( O(n^3) \) by using a linear programming model and finding a total unimodularity property. Even though we consider the case where containers have a size of either 1 or 2, it can be generalized into a more general case where containers have a size of either 1 or an arbitrary integer \( \mu \). Thus, scheduling Tall/Small Multiprocessor Tasks can be solved even faster.

Furthermore, \( I, 2 | r_j, u_j = 1, s_j \in \{1, 2\} | T_{\text{max}} \) is also related to the parallel machine scheduling problem with eligibility constraints. The eligibility constraints imply that job \( j \) can only be processed on a machine that belongs to set \( M_j \) which is a set of machines referred to as the eligible set for job \( j \). We consider two types of eligibility constraints, namely, \textit{Interval eligibility} (Lee et al. (2009)) and \textit{Grade of Service (GoS) eligibility} (Hwang et al. (2004)) which are defined as follows:

\begin{itemize}
    \item \textit{(Interval eligibility)}: If job \( j \) can be processed on machines \( \alpha \) and \( \beta \), then job \( j \) can also be processed on machines \( \alpha + 1, \alpha + 2, \ldots, \beta - 1 \);
    \item \textit{(GoS eligibility)}: If job \( j \) can be processed on machine \( \alpha \), then job \( j \) can also be processed on machines \( 1, 2, \ldots, \) and \( \alpha - 1 \).
\end{itemize}

If the \textit{Interval} or \textit{GoS} eligibility constraint is added, then the problem is referred to as \( P | M_j(\text{Interval}), p_j \in \{1, 2\} | C_{\text{max}} \) or \( P | M_j(\text{GoS}), p_j \in \{1, 2\} | C_{\text{max}} \), respectively.

**Theorem 8** \( I, 2 | r_j, u_j = 1, s_j \in \{1, 2\} | T_{\text{max}} \) is equivalent to \( P | M_j(\text{Interval}), p_j \in \{1, 2\} | C_{\text{max}} \).

**Proof** Consider the following decision versions of the two problems given thresholds \( T_1 \) and \( T_2 \).
Decision version of $I, 2 \mid r_j, u_j = 1, s_j \in \{1, 2\} \mid T_{\text{max}}$: 
"Is there a schedule $\sigma$ such that $T_{\text{max}}(\sigma) \leq T_1$ in $I, 2 \mid r_j, u_j = 1, s_j \in \{1, 2\} \mid T_{\text{max}}$?"

Decision version of $P \mid M_j(\text{Interval}), p_j \in \{1, 2\} \mid C_{\text{max}}$: 
"Is there a schedule $\sigma$ such that $C_{\text{max}}(\sigma) \leq T_2$ in $P \mid M_j(\text{Interval}), p_j \in \{1, 2\} \mid C_{\text{max}}$?"

We can first verify that $I, 2 \mid r_j, u_j = 1, s_j \in \{1, 2\} \mid T_{\text{max}}$ is a special case of $P \mid M_j(\text{Interval}), p_j \in \{1, 2\} \mid C_{\text{max}}$ by the following reduction: Let $c_{\text{max}} = \max\{c_i \mid i = 1, 2, ..., m\}$, and $T_2 = c_{\text{max}}$. For container $j$ with size $s$, let $p_j = s$, $s = 1, 2$, and for $j = 1, 2, ..., n$,

$$M_j = \{i \mid r_j \leq t_{i,s} \text{ and } t_{i,t} \leq d_j + T_1, i = 1, 2, ..., m\}.$$ 

Finally, we create $(c_{\text{max}} - c_i)$ jobs with unit processing times and $\{i\}$ as an eligible set for them. Note that the eligibility constraints are interval.

Conversely, we can show that $P \mid M_j(\text{Interval}), p_j \in \{1, 2\} \mid C_{\text{max}}$ is a special case of $I, 2 \mid r_j, u_j = 1, s_j \in \{1, 2\} \mid T_{\text{max}}$ through the following reduction: Let $T_1 = 0$. For $j = 1, 2, ..., n$, let $d_j = \max\{i \mid i \in M_j\}$ and $r_j = \min\{i \mid i \in M_j\}$. Let $c_i = T_2$ and $t_{i,s} = t_{i,t} = i$, $i = 1, 2, ..., m$. This completes the proof.

While $P \mid M_j, p_j \in \{1, 2\} \mid C_{\text{max}}$ has been shown to be strongly NP-hard by Glass and Kellerer (2007), it follows from Theorems 2 and 8 that $P \mid M_j(\text{Interval}), p_j \in \{1, 2\} \mid C_{\text{max}}$ can be solved in polynomial time.

**Corollary 1** $I, 2 \mid r_j = 0, u_j = 1, s_j \in \{1, 2\} \mid T_{\text{max}}$ can be solved in $O(mn^2)$ time.

**Proof** Ou et al. (2008) showed that $P \mid M_j(\text{GoS}), p_j \in \{l_1, l_2, ..., l_k\} \mid C_{\text{max}}$ can be solved in $O(mn^{2(k-1)})$, and GoS is a special property of Interval. Thus, it immediately follows from Theorem 8.

6 Conclusions

This paper focuses on problems related to the shipment of containers from their ports of origination to their ports of destination while minimizing the maximum tardiness. Three routing structures are considered, namely, identical, nested, and arbitrary routes. We determine the computational complexity of most cases and analyze a simple heuristic based on the Minimum Slack first rule with regard to its worst case analysis.
According to the computational complexity results we obtain, we can deal with more general real world problems heuristically. For some NP-hard cases, algorithms for polynomial solvable problems can be repeatedly applied. For example, in case of a fixed number of ports, all problems with \( r_j = 0 \) can be solved in polynomial time. If we only consider containers with zero release dates and create a schedule, we do not deal with containers that arrive later on, but only with containers at hand. As time goes by, new containers arrive and at a certain moment we iteratively can generate a new schedule with containers that already have arrived. This is a reasonable and practical heuristic. Furthermore, it is mentioned at the end of section 4 that the problem with arbitrary routes and a fixed number of ports can be solved in polynomial time when we drop the unidirectional routing assumption. However, since the analysis of other exact and approximate algorithms rely on the unidirectional routing assumption, this assumption is kept in place.

There are many other problems within this area of research that need to be investigated. Other objectives, besides \( T_{\text{max}} \), include the minimization of the sum of the delivery times of the containers (such an objective is akin to minimizing total inventory costs) and the minimization of the total number of ships needed to move all containers. More complicated models would include a transfer hub in which a container has to be unloaded from one ship (a feeder) and transferred to another ship. The following research directions may also be of interest.

- Given the demand for container transport for a particular planning horizon, the routes of the ships, a fleet of ships each with a capacity \( c \), and a threshold value of \( T_{\text{max}} \), what is the minimum number of ships needed and the assignment of ships to routes, including the start time of each ship, to achieve a threshold value for \( T_{\text{max}} \)?
- Consider a fleet of ships available with varying capacity \( c_i \) and cost \( z_i \). Given the demand for container transport for a particular planning horizon, the routes for the ships, and a threshold value for \( T_{\text{max}} \), what is the minimum cost assignment of ships to routes, including the start time of each ship, to achieve a threshold value for \( T_{\text{max}} \)?
- Consider a fleet of available ships with varying capacities \( c_i \) and costs \( z_i \). Given the demand for container transport for a particular planning horizon and a threshold value for \( T_{\text{max}} \), a more general problem is to determine the routes as well as the number of ships that must be assigned to each route, the start time of each ship, and the capacity of each ship for the minimum cost assignment of ships to routes to achieve a threshold value for \( T_{\text{max}} \).
References


APPENDIX

Proof of Theorem 1 The decision version of $I, X | r_j = 0, u_j, s_j = 1 | T_{\text{max}}$ can be stated as follows: Given a threshold $\lambda$, is there a schedule $\sigma$ such that $T_{\text{max}}(\sigma) \leq \lambda$?

Given an instance of HIS($T$), we can construct an instance of $I, X | r_j = 0, u_j, s_j = 1 | T_{\text{max}}$ as follows. There are $2n$ ports, $\{1, 2, \ldots, 2n\}$, $n$ containers, $\{1, \ldots, n\}$, and $T$ ships, $\{1, \ldots, T\}$. Job $j$ of type $i$ in HIS($T$) is reduced to container $j$ of size 1 with origination port $u_j = b_j$,
destination port $v_j = e_j$, release date $r_j = 0$ and due date $d_j = i + e_j - 2$, $j = 1, 2, \ldots, n$ and $i = 1, 2, \ldots, T$. Ship $i$ has capacity $c_i = m_i$ and visits all ports; it arrives at port $l$ at time $t_{i,l} = i + l - 2$, $i = 1, 2, \ldots, T$ and $l = 1, 2, \ldots, 2n$. Note that all ships visit all ports in the same order, i.e., for $i = 1, 2, \ldots, T$,

$$\rho^i : 1 \to 2 \to \cdots \to (2n - 1) \to 2n.$$ 

Finally, we choose $\lambda$ to be 0. The objective is to decide if there is a schedule with $T_{\text{max}} = 0$. Obviously, the transformation can be done in polynomial time.

Suppose there is a feasible schedule for HIS($T$) such that

(i) no job overlaps with any other job on the same machine,
(ii) job $j$ of type $i$ is scheduled on a machine of type $k$ during $[b_j, e_j]$, where $k \leq i$.

From (ii), we can construct a schedule for the reduced problem by loading container $j$ of type $i$ onto ship $k$. By (i), it is clear that the number of containers loaded onto ship $k$ is no more than $m_k$. It implies that the constructed schedule is feasible. Furthermore, since $d_j = i + e_j - 2$, $k \leq i$ and $t_{k,e_j} = k + e_j - 2$, container $j$ must reach its destination port by its due date. Therefore, $T_{\text{max}} = 0$.

Suppose there is a feasible schedule for the reduced problem such that

(i’) the number of containers loaded onto ship $i$ at any time is no more than its capacity $c_i = m_i$, $1 \leq i \leq T$,
(ii’) container $j$ is loaded onto ship $i$ at port $b_j$ and unloaded at port $e_j$ at time $i + e_j - 2$,
(iii’) $T_{\text{max}} = 0$.

From (ii’) and (iii’), we can construct a schedule for HIS($T$) by assigning job $j$ to a machine of type $i$ during the interval $[b_j, e_j]$. Clearly, (i’) implies (i). Thus, the schedule is feasible.

**Proof of Lemma 1**

Suppose there is a clique $N'$ of size $m + 1$ in the $(m + 1)$-partite graph. Let

$$N' = \{N(0, 0, 0), N(1, a_1, b_1), N(2, a_2, b_2), \cdots, N(m, n_1, n_2)\}.$$ 

Note that $a_k = \sum_{i=1}^{k} c_i^1$ and $b_k = \sum_{i=1}^{k} c_i^2$ for $k = 1, 2, \ldots, m$, where $c_i^1 = a_i - a_{i-1}$ and $c_i^2 = b_i - b_{i-1}$ for $i = 1, 2, \ldots, m$. (Note that $c_i^1$ and $c_i^2$ denote the numbers of containers of sizes 1 and 2, respectively, assigned to ship $i$, $i = 1, 2, \ldots, m$.) We can construct a feasible schedule for $I(\lambda)$ by the following procedure.
Procedure TRANSFORMATION

**Step 1** \( h = 1 \) and \( \hat{J}_{0:i}^1 = \hat{J}_{0:i}^2 = \emptyset, i = 1, 2, ..., m; \)

**Step 2** \( \hat{J}_{h:i}^1 = J_{h:i}^1 \cup \bar{J}_{h-1:i}^1 \) and \( \hat{J}_{h:i}^2 = J_{h:i}^2 \cup \bar{J}_{h-1:i}^2, i = h, h + 1, ..., m; \)

**Step 3** Assign, by the Earliest Due Date rule, \( (a_h - a_{h-1}) \) containers from \( \bigcup_{i=h}^{m} \hat{J}_{h:i}^1 \) and \( (b_h - b_{h-1}) \) containers from \( \bigcup_{i=h}^{m} \hat{J}_{h:i}^2 \) onto ship \( h; \)

**Step 4** Let \( \hat{J}_{h:i}^1 \) and \( \hat{J}_{h:i}^2 \) be the sets of unassigned containers in \( J_{h:i}^1 \) and \( J_{h:i}^2 \), respectively, \( i = h, h + 1, ..., m; \)

**Step 5** \( h = h + 1. \) If \( h = m + 1 \), STOP; otherwise, go to Step 2;

Clearly, Procedure TRANSFORMATION can be done in \( O(n + m^2) \) time. Suppose a feasible schedule cannot be constructed from \( N' \) by Procedure TRANSFORMATION. Then, there is a (smallest) index \( l \) such that either (i) \( |\bigcup_{i=l}^{m} \hat{J}_{i:l}^1| < c_l^s \), or (ii) \( |\hat{J}_{i:l}^1| > c_l^1 \), or (iii) \( |\bigcup_{i=l}^{m} \hat{J}_{i:l}^2| < c_l^s \), or (iv) \( |\hat{J}_{i:l}^2| > c_l^2 \) at the \( l \)th iteration of the procedure. That is, all containers belonging to \( \bigcup_{1 \leq h \leq l, 1 \leq i \leq m} J_{h:i}^s \) have been assigned to ships \( 1, 2, ..., l - 1 \). We have the following cases to consider.

**Case I:** \( |\bigcup_{i=l}^{m} \hat{J}_{i:l}^s| < c_l^s \) for \( s = 1 \) or \( 2 \). (Cases (i) and (iii)).

In this case, all containers belonging to \( \bigcup_{1 \leq h \leq l, 1 \leq i \leq m} J_{h:i}^s \) have been assigned to ships \( 1, 2, ..., l \). This implies that

\[
|\bigcup_{1 \leq h \leq l, 1 \leq i \leq m} J_{h:i}^s| < \sum_{i=1}^{l} c_i^s. \tag{1}
\]

From inequality (1) and \( |\bigcup_{1 \leq h \leq i \leq m} J_{h:i}^s| = \sum_{i=1}^{m} c_i^s \), we have the following inequality.

\[
|\bigcup_{l+1 \leq h \leq i \leq m} J_{h:i}^s| > \sum_{i=l+1}^{m} c_i^s = \begin{cases} a_m - a_l & \text{if } s = 1 \\ b_m - b_l & \text{if } s = 2. \end{cases}
\]

However, the above inequality means that the arc-feasibility condition between \( N(l, a_l, b_l) \) and \( N(m, a_m, b_m) \) is violated, and hence there is no edge between \( N(l, a_l, b_l) \) and \( N(m, a_m, b_m) \). This contradicts the fact that \( N' \) is a clique.

**Case II:** \( |\hat{J}_{i:l}^s| > c_l^s \) for \( s = 1 \) or \( 2 \) (Cases (ii) and (iv)).

Let \( l' \) be the largest index, \( l' < l \), such that the containers that do not belong to \( \bigcup_{1 \leq h \leq i \leq l} J_{h:i}^s \) have been assigned to ship \( l' \). Then, only containers belonging to \( \bigcup_{l'+1 \leq h \leq i \leq l} J_{h:i}^s \) will be
assigned to ships $l' + 1, \ldots, l$. Therefore, the following inequality can be obtained.

$$ | \bigcup_{l' + 1 \leq h \leq l} J_{h,i}^s | > \sum_{i=l'+1}^{l} c_i^s = \begin{cases} a_l - a_{l'} & \text{if } s = 1 \\ b_l - b_{l'} & \text{if } s = 2. \end{cases} $$

However, the above inequality violates the arc-feasibility condition between $N(l', a_{l'}, b_{l'})$ and $N(l, a_l, b_l)$. Hence, there is no edge between $N(l', a_{l'}, b_{l'})$ and $N(l, a_l, b_l)$, contradicting our assumption that $N'$ is a clique.

Now, suppose there is a feasible schedule for $I(\lambda)$, expressed as follows.

$$ N(0, 0, 0) \rightarrow N(1, a_1, b_1) \rightarrow N(2, a_2, b_2) \rightarrow \cdots \rightarrow N(m, n_1, n_2). $$

In the above expression, node-feasibility and arc-feasibility conditions are satisfied by the feasibility of the schedule. Therefore,

$$ N' = \{ N(0, 0, 0), N(1, a_1, b_1), N(2, a_2, b_2), \ldots, N(m, n_1, n_2) \} $$

is a clique with $m + 1$ nodes. \hfill \blacksquare

**Proof of Theorem 6** The decision version of $A, X | r_j = 0, u_j, s_j \in \{1, 2\} | T_{\max}$ can be stated as follows: Given a threshold $\lambda$, is there an assignment of containers onto ships such that $T_{\max} \leq \lambda$? Clearly, this problem is in NP.

The decision version of $P | M_j, p_j \in \{1, 2\} | C_{\max}$ is defined as follows: Given a threshold $\omega$, is there a schedule $\sigma$ such that $C_{\max}(\sigma) \leq \omega$, where $C_{\max}(\sigma)$ is the makespan of schedule $\sigma$.

With respect to problem $P | M_j, p_j \in \{1, 2\} | C_{\max}$, let $K_i$ denote the set of all jobs that are eligible for processing on machine $i$, $i = 1, 2, \ldots, m$; i.e., $K_i = \{ j | i \in M_j \}$. Let $k_i(l)$ denote the $l$th smallest element of $K_i$, $1 \leq l \leq |K_i|$. Then $K_i = \{ k_i(1), \ldots, k_i(|K_i|) \}$, where $k_i(1) < k_i(2) < \ldots < k_i(|K_i|)$.

Given an instance of $P | M_j, p_j \in \{1, 2\} | C_{\max}$, we construct an instance of $A, X | r_j = 0, u_j, s_j \in \{1, 2\} | T_{\max}$ as follows. There are $n$ containers, $m$ ships and $n + 2$ ports, i.e., $N_p = n + 2$. For container $j$, let $u_j = j + 1$, $v_j = N_p$, $s_j = p_j$, $r_j = 0$, and $d_j = m + n$, $j = 1, 2, \ldots, n$. All ships have the same capacity $\omega$, i.e., $c_i = \omega$ for $i = 1, 2, \ldots, m$. Ship $i$ has a route $\rho^i$ defined as follows.

$$ \rho^i : 1 \rightarrow k_i(1) + 1 \rightarrow k_i(2) + 1 \rightarrow \cdots \rightarrow k_i(|K_i|) + 1 \rightarrow N_p, \quad i = 1, 2, \ldots, m. $$

Note that the ship routes all have a common source port (namely port 1) and a common
destination port (namely port $N_p$). Ports $2, 3, ..., n + 1$ are the source ports of jobs $1, 2, ..., n$. All containers have a common destination port, $N_p$. Ship $i$ visits all the ports that correspond to jobs that can be executed on machine $i$. For ship $i$, $i = 1, 2, ..., m$, $t_{i,1} = i - 1$, $t_{i,k_i(l)} = i - 1 + k_i(l)$ and $t_{i,N_p} = i + n$, where $1 \leq l \leq |K_i|$. The capacity of ship $i$ is $\omega$, the makespan threshold of $P | M_j, p_j \in \{1, 2\} | C_{\text{max}}$. Finally, let $\lambda = 0$. The objective is to determine if there is an allocation of containers to ships such that $T_{\text{max}} = 0$. We observe that

- There exists one container in port $l$, $l = 2, 3, ..., n + 1$, and no container in ports 1 and $N_p$;
- The reduced problem satisfies the non-overtaking assumption;
- Machine $i$ in $P | M_j, p_j \in \{1, 2\} | C_{\text{max}}$ corresponds to ship $i$ in $A, X | r_j = 0, u_j, s_j \in \{1, 2\} | T_{\text{max}}$.
- The capacity of every ship is exactly the makespan threshold in the problem $P | M_j, p_j \in \{1, 2\} | C_{\text{max}}$.

The transformation can be done in polynomial time. Clearly, there is a schedule $\sigma$ with $C_{\text{max}}(\sigma) \leq \omega$ in the decision version of $P | M_j, p_j \in \{1, 2\} | C_{\text{max}}$ if and only if there is an allocation of containers to ships with $T_{\text{max}} = 0$ in the reduced problem. ■