Choi, Byung-Cheon; Briskorn, Dirk; Lee, Kangbok; Leung, Joseph; Pinedo, Michael

Working Paper

Allocating containers to ships with fixed departure times

Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 641

Provided in Cooperation with:
Christian-Albrechts-University of Kiel, Institute of Business Administration

Manuskripte
aus den
Instituten für Betriebswirtschaftslehre
der Universität Kiel

No. 641

Allocating Containers to Ships with Fixed Departure Times

Byung-Cheon Choi[1], Dirk Briskorn[1,2,*], Kangbok Lee[1], Joseph Leung[3], Michael Pinedo[1]

October 2008

[1]: Stern School of Business
New York University
44 West 4^{th} Street, New York, NY 10012, USA
http://www.stern.nyu.edu/∼mpinedo
bchoi@stern.nyu.edu, klee3@stern.nyu.edu, mpinedo@stern.nyu.edu

[2]: Christian-Albrechts-Universität zu Kiel
Institut für Betriebswirtschaftslehre
Olshausenstr. 40, 24098 Kiel, Germany
http://www.bwl.uni-kiel.de/bwlinstitute/Prod
briskorn@bwl.uni-kiel.de

[3]: Department of Computer Science
New Jersey Institute of Technology
University Heights, Newark, NJ 07102, USA
http://web.njit.edu/∼leung/
leung@cis.njit.edu

*: supported by a fellowship within the Postdoc-Program
of the German Academic Exchange Service (DAAD)

Abstract

We consider the problem of allocating containers to ships in which the size of container is 1 or 2, and each ship has its own capacity and fixed departure time. The fixed departure times implies the completion times of containers belonging to the same ship are identical. As objectives, L_{max}, $\sum w_j C_j$, $\sum w_j U_j$ and $\sum T_j$ are considered. We verify the problems are closely related with the scheduling problem with eligibility constraint or the generalized assignment problem. The polynomial-time algorithms are developed for each problem, and moreover the more efficient algorithms are presented for the cases with $\sum C_j$ and $\sum U_j$,

Keywords: Scheduling, Container Allocation, Fixed Departure Time

1 Introduction

In various regions of the world, double-digit growth rates in container handling have been common during the last years and, hence, a substantial number of container vessels is built each year. Additionally, capacity of new vessels is increasing constantly. Thus, the capacity of the worldwide container vessel fleet increases year by year. Obviously, this enlarges the pressure on container terminal operators to improve the performance, often measured as throughput per time-unit. Accordingly, new container terminals are built as well as existing container terminals are enlarged and better equipped.

To make efficient use of the container handling equipment there is an enormous need for methods to plan the operations taking place in the container terminal. Due to its practical relevance, container terminal logistics and its support by operational research methods have been prominent fields of research in recent years. Comprehensive literature surveys can be found for example in Stahlbock and Voß [12], Steenken et al. [14], and Vis and de Koster [15]. In optimization problems the handling of containers is considered, e.g. quai crane planning (see Goodchild and Daganzo [6] and Park and Kim [11]), and straddle carrier scheduling (see Kim and Kim [8] and Steenken et al. [13]). Furthermore, routing and scheduling of ships has been worked on intensively, see Christiansen et al. [3] for a survey. Some authors look at the assignment of cargo to ships. Fagerholt [4] and Fagerholt and Christiansen [5] consider a ship routing problem where allocating cargo is part of the decision. Chen et al. [2] propose a pretty simple model involving two ports only. Cargo shall be assigned to vessels of an infinite fleet such that the sum of ship cost (fixed per used ship) and inventory cost is minimized.

Surprisingly, in literature one central problem has only received minor attention. A major shipping company usually has a number of ships travelling on a fixed route along a certain set of ports. In each port there are containers to load onto one of those ships. The shipping company's planner has to decide which container to assign to which ship. Of course containers originating from different ports may compete for the same capacity. In this paper we consider due date and flow time related objectives to evaluate different assignments.

The paper is organized as follows. In Section 2 we propose a model concerning several ports and show how to reduce it to a problem where only one port is involved. Moreover, we give strong relations to other optimization problems. In Sections 3, 4, 5 and 6, we develop polynomial-time dynamic programming for problems considering well-known objective functions L_{max}, $\sum w_j C_j$, $\sum w_j U_j$, and $\sum T_j$. Additionally, we present more efficient algorithms for a special case. Finally, we complete the paper with a summary and concluding remarks.

2 Problem Definition

2.1 The Port Model

We consider a set of ports being visited by a set of feeder vessels. In comparison to mega container carriers developed recently feeder vessels are smaller. Mainly, they are employed to connect a major seaport container terminal to smaller ports that are either impossible to reach by mega container carriers or can only be reached uneconomically. Feeder vessels often go on a fixed route along a set of ports and either collect containers to be loaded onto mega carriers in a major seaport near by or distribute containers that have been arrived on a mega carrier. Arrival and departure time of each vessel i in each port are known in advance. Furthermore, we assume that ships travel with identical speed, i.e., departure times of two ships in two consecutive ports have the same time distance. Figure 1 illustrates the setting.

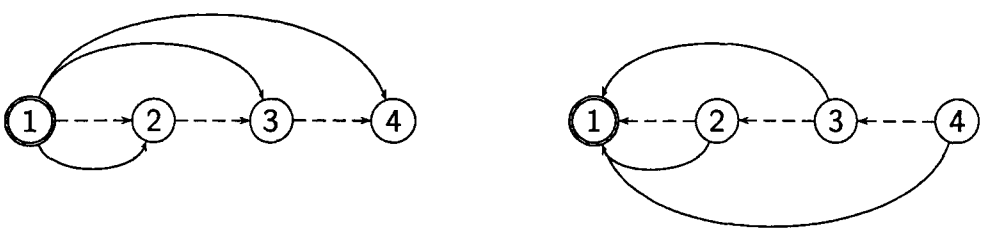

Figure 1: Port Structure

In general dashed lines and solid lines represent the route of vessels and container flows, respectively. On the left hand side feeder ships go from the major seaport 1 via smaller ports 2 and 3 to 4. Containers are transported from the major seaport to all smaller ports but no containers are loaded onto the ship in one of the smaller ports. On the right hand side feeder ships go from 4 via 3 and 2 to the major seaport 1. Containers are transported from each of the smaller ports to the major seaport but no containers are unloaded in one of the smaller ports.

We regard the point of time when container j reaches its destination as completion time C_j, and the point of time when container j should be reached the destination as due date d_j. Due-date related objectives number of tardy jobs, weighted number of tardy jobs, and total tardiness are considered. On the other hand total (weighted) completion time gives a objective measuring the total amount of time containers are in the system until they reach their destination. Obviously, all objectives have to be minimized.

Now, we show that in both cases it is enough to consider the major seaport only. This is obvious for the setting where containers are brought to the major seaport. Completion time C_j of container j is regarded as the arriving time of ship i at the major seaport t_i^a if container j is loaded to ship i. So, we can consider all ships arriving over time (at predefined points of time t_i^a for all ships i) and decide which container should arrive on which ship. For the setting where containers are picked up at the major seaport we can calculate completion times of jobs to the major seaport as follows. Completion time C_j of job j equals the sum of travelling time to its destination and departure time t_i^d if container j is loaded to ship i. Since travelling time is assumed to be constant for each container j we can consider $C_j = t_i^d$ if container j is loaded to ship i. Obviously, we have to project due date d_j to the major seaport as well. This can be done by reducing d_j by the travelling time of job j and, thus, d_j represents the point of time j should leave the major seaport. To unify the notation we consider a single time t_i representing t_i^d and t_i^a, respectively.

Naturally, each ship i has limited capacity c_i that must not be exceeded by the set of containers assigned to the ship. We consider two types of containers for the development of algorithms. The size of each container depends on its type. In the real world almost all containers have identical heigth and width and have length of either 1 TEU (twenty-foot equivalent unit) or 2 TEU. Consequently, sizes of containers are 1 or 2 depending on the type. However, our algorithms are generalizable to other sizes or more types, respectively, and we will mention details after the development of each algorithm.

2.2 Formal Optimization Problem

In the following, we model the problem of allocating single containers to ships, formally. Henceforth, we refer to a container and ship as a job and batch, respectively. The following notations are used throughout the paper.

Notations

n_s : number of jobs of size s, $s = 1, 2$;

n : number of jobs of all sizes, i.e., $n_1 + n_2$;

r : number of batch ;

J_j^s : job j of size s, $s = 1, 2$;

t_i : departure time of batch i, $i = 1, 2, ..., r$;

c_i : capacity of batch i, $i = 1, 2, ..., r$;

w_j^i : weight of job j of size i ;

d_j^s : due date of job j of size s ;

r_j^s : release date of job j of size s ;

C_j^s : completion time of job j of size s ;

s_j : size of job j ;

By using the above notations, our problem is defined as follows.

Problem Definition: There are r batches such that batch i having capacity c_i is suppose to depart at t_i, and there are n_1 jobs of size 1 and n_2 jobs of size 2. Job j of size s_j has its own due date d_j^i and weight w_j^s, $j = 1, 2, ..., n_s$, $s = 1, 2$. The objectives are to find the allocation of jobs to ship to minimize L_{max}, $\sum w_j C_j$, $\sum w_j U_j$ and $\sum T_j$.

The completion time of each job is equal to the departure time of the ship in which the job is scheduled, and the capacity of each batch is different. Let these properties be described as $batch(i)$. Following the classification scheme of Graham et al. [7], we can refer to the problems under consideration as follows.

Problem I $1|batch(i),\ s_j \in \{1,2\}|L_{max}$;

Problem II $1|batch(i),\ s_j \in \{1,2\}|\sum w_j C_j$;

Problem III $1|batch(i),\ s_j \in \{1,2\}|\sum w_j U_j$;

Problem IV $1|batch(i),\ s_j \in \{1,2\}|\sum T_j$.

2.3 Related Work

In this section, we derive the strong relationships between our problems and other combinatorial problems. First of all, we show there exists a relationship between Problem I and the scheduling problem with eligibility constraints, e.g., *Interval* and *Great of Service(GoS)* which are defined as follows: (*Interval*) If job j can be processed in machines α and β, then job j can also be processed in machines $\alpha+1$, $\alpha+2$,.., and $\beta-1$, and (*GoS*). If job j can be processed in machines α, then job j can also be processed in machines 1,2,..., and $\alpha-1$.
Let **Scheduling Problem with Eligibility Constraints (SPEC)** be defined as follow: Given n jobs, processing times $p_j \in \{1,2\}$, interval eligibility constraint M_j and release times r_j, find an optimal schedule to minimize the makespan on m parallel machines. Following the classification scheme of Graham et al. [7], the above problems with *Interval* or *GoS* eligibility constraints are denoted as $P|M_j(Interval), p_j \in \{1,2\}|C_{max}$ and $P|M_j(GoS), p_j \in \{1,2\}|C_{max}$, respectively.

Theorem 1. *Problem I with release times is equivalent to* $P|M_j(Interval), p_j \in \{1,2\}|C_{max}$.

Proof First of all, consider that the decision versions of Problem I and SPEC are as follows: Given thresholds T_1 and T_2, *Decision version of Problem I with release times:* "Is there a schedule σ such that $L_{max}(\sigma) \leq T_1$ in Problem I with release times?", and *Decision version of SPEC:* "Is there a schedule σ such that $C_{max}(\sigma) \leq T_2$ in SPEC?", where T_1 and T_2 are given thresholds.
We can verify that Problem I is the special case of SPIEC by the following reduction: Let $c_{max} = \{c_i | i = 1,2,...,r\}$, and $T_2 = c_{max}$. For job j with size of 1, 1et $p_j = 1$ and

$$M_j = \{i \mid r_j^1 \leq t_i \leq d_j^1 + T_1,\ i = 1,2,...,r\},\ j = 1,2,...,n_1,$$

and for job j with size of 2, let $p_j = 2$ and

$$M_j = \{i \mid r_j^2 \leq t_i \leq d_j^2 + T_1,\ i = 1,2,...,r\},\ j = 1,2,...,n_1.$$

Finally, construct $(c_{max} - c_i)$ jobs with unit processing times and $\{i\}$ as eligibility constraint, $i = 1,2,...,r$. Note that eligibility constraints are *interval*. In other way, it is shown that SPEC is the special case of Problem I by the following reduction: Let $T_1 = 0$. If $s_j = p_j = 1$, $d_j^1 = max\{i|i \in M_j\}$ and $r_j^1 = min\{i|i \in M_j\}$, while if $s_j = p_j = 2$, $d_j^2 = max\{i|i \in M_j\}$ and $r_j^2 = min\{i|i \in M_j\}$, $j = 1,2,...,n$. Let $c_i = T_2$ and $t_i = i$, $i = 1,2,...,m$. This completes proof. ■
It is unknown whether the time complexity of $P|M_j(Interval), p_j \in \{1,2\}|C_{max}$ is NP-hard or not. This implies that Problem I with release times is a challenging problem.
Corollary 1 Problem I is equivalent to $P|M_j(GoS), p_j \in \{1,2\}|C_{max}$.
Proof It immediately holds from the proof of Theorem 1. ■
Leung [9] and Leung [10] considered the parallel machine problem with k type of processing times, in which the objective is to minimize the makespan. In Leung [9], the problem was shown to be solve in $O(n^{2(k-1)} \log p \log m)$, where let $p = max\{p_i | i = 1,2,...,k\}$. Also, Leung [10] showed that the problem with *GoS* eligibility constraint is polynomially solvable in $O(mn^{2(k-1)})$. By Corollary, it implies that there exists a $O(rn^2)$ algorithm to solve Problem I. In section 3, we will suggest the better algorithm.
Problems II, III and IV can be formulated as generalized assignment problem, which is defined as follows.
Generalized Assignment Problem (GAP): Given positive integer h_{ij}, a_j and b_i, find an optimal solution $x = (x_1, x_2, ..., x_n)$ such that

$$\begin{aligned}
&\text{Minimize(Maximize)} && \sum_{i=1}^{m}\sum_{j=1}^{n} h_{ij}x_{ij} \\
&\text{Subject to} && \sum_{j=1}^{m} a_{ij}x_{ij} \le b_i \quad \text{for } i = 1,2,...,m \\
& && \sum_{i=1}^{m} x_{ij} \le 1 \quad \text{for } j = 1,2,...,n \\
& && x_{ij} \in \{0,1\} \quad \text{for } i = 1,2,...,m,\ j = 1,2,...,n.
\end{aligned}$$

If $m = r$, $n = n_1 + n_2$, $a_{ij} = s_j \in \{1,2\}$ and $b_i = c_i$, then the set of feasible schedule is equivalent to the feasible set of Problems II, III and IV, e.g., $x_{ij} = 1$ means job j is loaded to batch i. According to the objective function, h_{ij} can be as follows:

Problem II $h_{ij} = w_j t_i$;

Problem III $h_{ij} = w_j$ if $t_i > d_j$, while $h_{ij} = 0$, otherwise ;

Problem IV $h_{ij} = max\{0, t_i - d_j\}$.

3 Problem I: $1|batch(i),\ s_j \in \{1,2\}|L_{max}$

Suppose that

$$d_1^s \le d_2^s \le ... \le d_{n_s}^s, \quad s = 1,2.$$

Let $\Delta = \{t_i - d_j^s \mid i = 1,\ldots,r, s = 1,2, j = 1,\ldots,n_s\}$, and T^* be the optimal value. Clearly, $T^* \in \Delta$. If due dates are changed as $d_j^s = d_j^s + T^*$, $j = 1,2,...,n_s$, $s = 1,2$, then there exists a schedule with $L_{max} = 0$ in the modified problem. We can determine whether a given value T^* is an optimal value or not through the following algorithm.

Algorithm for Feasibility (Algorithm F)

Step 1 Set $i = 1$, $j_1 = 1$ and $j_2 = 1$.

Step 2 Find the index $i^* = \min\{i|c_i \ge 2\}$, and $i = i^*$.

Step 3 If $j_2 = n_2 + 1$, go to Step 6. Otherwise, put job j_2 of size 2 in batch i.

Step 4 If the job is not tardy, then $c_i = c_i - 2$. Otherwise, STOP (INFEASIBILITY).

Step 5 If $c_i \ge 2$, then $j_2 = j_2 + 1$ and go to Step 3. Otherwise, go to Step 2.

Step 6 Find the index $i^* = \min\{i|c_i \ge 1\}$. Let $i = i^*$.

Step 7 If $j_1 = n_1 + 1$, STOP (FEASIBILITY). Otherwise, put job j_1 of size 1 in batch i.

Step 8 If the job is not tardy, then $c_i = c_i - 1$. Otherwise, STOP (INFEASIBILITY).

Step 9 If $c_i \ge 1$, then $j_2 = j_2 + 1$ and go to Step 7. Otherwise, go to Step 6.

The time complexity of the above algorithm is $O(n)$.

Algorithm for Problem I (Algorithm I)

Step 1 If $|\Delta| = 1$, then STOP. Select the median in Δ. Let δ be the value of the median, and let Δ_R and Δ_L be the the higher and lower half of Δ, respectively.

Step 2 Modify the problem by letting $d_j^s = d_j^s + \delta$, $j = 1, 2, ..., n_s$, $s = 1, 2$.

Step 3 Check the feasibility for δ through Algorithm F. If δ is feasible, then $\Delta = \Delta_L$ and go to Step 1. Otherwise, $\Delta = \Delta_R$ and go to Step 1.

Since the total number of iterations of Step 1 is $O(\log|\Delta|)$ time, and Steps 2 and 3 requires $O(n)$ time, the time complexity of Algorithm I is $O(n \log|\Delta|)$. Since $\log|\Delta| = \log rn = \log n + \log r$, however, the time complexity reduces to $O(n \log n)$.

Theorem 2. *Problem I can be solved in $O(rn + n \log n)$ time.*

Proof Before Algorithm I is applied, all values in $|\Delta|$ should be calculated, which requires $O(rn)$ times. This completes the proof. ■

4 Problem II: $1|batch(i),\ s_j \in \{1,2\}|\sum w_j C_j$

In this section, we show that Problem II can be solved in polynomial time by reducing it to the shortest path problem, and develop an efficient algorithm for the case with identical weights.

4.1 Reduction to Shortest Path Problem

Suppose that for $i = 1, 2$,

$$w_1^i \geq w_2^i \geq \cdots \geq w_{n_i}^i.$$

We can formulate Problem II as the shortest path problem of finding the shortest path from a starting vertex s to a destination vertex h in a directed graph $G(N, A)$ as follow: Let $N(k, a_1, a_2)$ be a node meaning the state that a_1 jobs of size 1 and a_2 jobs of size 2 are scheduled in batches $1, 2, ...,$ and k, $a_1 = 1, 2, ..., n_1$, $a_2 = 1, 2, ..., n_2$. Let the starting node s and the destination node h be defined as $N(0, 0, 0)$ and $N(r, n_1, n_2)$, respectively. The edge connecting node $N(k, a_1, a_2)$ with node $N(k+1, b_1, b_2)$ means that the $b_1 - a_1$ jobs of size 1 and $b_2 - a_2$ jobs of size 2 are scheduled to batch $k+1$. Since the total sizes of the jobs scheduled to batch $k+1$ should be less than c_{k+1}, one of two conditions should be satisfied.

i) $(b_1 - a_1) + 2(b_2 - a_2) < c_{k+1}$ and $b_1 = n_1$

ii) $(b_1 - a_1) + 2(b_2 - a_2) = c_{k+1}$

Given k and a_2, $N(k, a_1, a_2)$, a_1 can be calculated as follows.

$$a_1 = \min\{\sum_{i=1}^{k} c_i - 2a_2, n_1\}. \tag{1}$$

Let $L_{k+1}(N(a_1, a_2), N(b_1, b_2))$ be the length of the arc between $N(k, a_1, a_2)$ and $N(k+1, b_1, b_2)$. The length can be calculated as follows.

$$L_{k+1}\big(N(a_1, a_2), N(b_1, b_2)\big) = \Big(\sum_{j=a_1+1}^{b_1} w_j^1 + \sum_{j=a_2+1}^{b_2} w_j^2\Big) t_{k+1}.$$

The shortest path can be interpreted in the context of Problem I as follow. If $(N_k(a_1, a_2), N_{k+1}(b_1, b_2))$ belongs to the shortest path, then it means that the job set

$$\{J_{a_1+1}, J_{a_1+2}, ..., J_{b_1}, J_{a_2+1}, J_{b_1+2}, ..., J_{b_2}\}$$

is scheduled to batch $k+1$.

Lemma 1 The total number of arcs is at most $O(rn^2)$.

Proof By Equation (1), a_1 can be determined by k and a_2. Thus, the number of nodes is bounded by k and a_2, implying that the total number of nodes is at most $O(rn)$. Also, since the number of outgoing arcs from each node is less than or equal to n_2, the total number of arcs is at most $O(rn^2)$. ■

Theorem 3. *Problem II can be solved in $O(rn^2)$ time.*

Proof Since the constructed graph is acyclic, the algorithm of Ahuja et al. [1] can be solve the above shortest path problem in $O(|A|)$, where $|A|$ is the number of arcs. Since the number of arcs in the above graph is at most $O(rn^2)$ by Lemma 1, Problem II is polynomially solvable in $O(rn^2)$. ■

4.2 Identical weights

Let k be the index of the batch in which the last job of size 1, i.e., $J^1_{n_1}$ is loaded in an optimal schedule. Let T be the set of batches with odd capacity which are located before batch k.

Lemma 2 There should exist at least one job of size 1 in each batch belonging to T.

Proof Suppose that there exists no job of size 1 in batch l such that $l \in T$. Since the capacity of batch l is odd, there should be at least one empty slot in batch l. Thus, the objective value can be decreased by moving the job of size 1 located in batch k to batch l. This is contradiction. ■

From Lemma 2, we can reduce the original problem into Problem II-1, which is defined as follows: There are $(n_1 - |T|)$ jobs of size 1 and n_2 jobs of size 2. For $i \leq k$, set $\bar{c}_i = c_i - 1$ if c_i is odd, while $\bar{c}_i = c_i$ otherwise. Other features are the same with Problem II.

Lemma 3 In Problem II-1, there exists an optimal schedule such that jobs of size 1 are scheduled before jobs of size 2. This implies that the index of the batch to which the last job of size 1 is located is less than or equal to the index of the batch to which the first job of size 2 is located.

Proof Suppose that job α of size 2 is scheduled in batch i and job β of size 1 is scheduled in batch $i+j$ in an optimal schedule. Since $i+j \leq k$ by assumption, there exists at least one empty slot or two jobs of size 1 in batch $i+j$. If there exists one empty slot, then we can exchange the locations of jobs α and β without inceasing the objective value. If there exists other job of size 1 except job β in batch $i+j$, then we can decrease the objective value by exchanging the locations of two jobs of size 1 and job β. We can apply this argument until Lemma 3 obtained. This completes the proof. ■

Based on Lemma 3, we develop the algorithm to solve Problem II-1.

Algorithm for Problem II-1 (Algorithm II-1)

Step 1 Schedule jobs of size 1 from batch 1 until all jobs of size 1 have been scheduled. Let batch h be the last batch which the job of size 1 is located.

Step 2 If $h > k$, Stop, implying there exists no optimal schedule such that the last job of size 1 is located in batch k in the original problem.

Step 3 Schedule jobs of size 2 from batch h until all jobs of size 2 have been scheduled.

The time complexity of Algorithm II-1 is $O(n)$

Theorem 4. *Problem II with identical weights can be solved in $O(n \log n)$ time.*

Proof The original problem can be solved thorough the $O(\log n)$ iterations of Algorithm II-1. This completes the proof. ■

5 Problem III: $1|batch(i),\ s_j \in \{1,2\}|\sum w_j U_j$

In this section, we show that Problem III can be solved in polynomial time by reducing it to the shortest path problem and develop the efficient algorithm for the case with identical weights.

5.1 Reduction to Shortest Path Problem

We assume that the jobs of each size are ordered in early due date (EDD) order, i.e., for $i = 1, 2$,

$$d_1^i \leq d_2^i \leq \cdots \leq d_{n_i}^i.$$

We can formulate Problem III as the shortest path problem of finding the shortest path from a starting vertex s to a destination vertex h in a directed graph $G(N, A)$ as follow: Let $N(k, a_1, a_2, b_1, b_2)$ be a node meaning the state that (1) $\{J_1^1, J_2^1, ..., J_{a_1}^1\}$ and $\{J_1^2, J_2^2, ..., J_{a_2}^2\}$ have been considered until now, and (2) b_1 jobs of size 1 and b_2 jobs of size 2 have been scheduled in batch k. Let starting node s and destination node h be defined as $N(1, 0, 0, 0, 0)$ and $N(r+1, n_1, n_2, 0, 0)$, respectively. $N(k, a_1, a_2, b_1, b_2)$ can be connected to at most five nodes, depending on the capacity condition of batch k as follows.

(1) $N(k, a_1 + 1, a_2, b_1, b_2)$;

(2) $N(k, a_1 + 1, a_2, b_1 + 1, b_2)$ if $(b_1 + 1) + 2b_2 \leq c_k$ and $t_k \leq d_{b_1+1}^1$;

(3) $N(k, a_1, a_2 + 1, b_1, b_2)$;

(4) $N(k, a_1, a_2 + 1, b_1, b_2 + 1)$ if $b_1 + 2(b_2 + 1) \leq c_k$ and $t_k \leq d_{b_2+1}^2$;

(5) $N(k + 1, a_1, a_2, 0, 0)$.

Note that the conditions of $(b_1 + 1) + 2b_2 \leq c_k$ and $b_1 + 2(b_2 + 1) \leq c_k$ are on the capacity of batch k, and the conditions of $t_k \leq d_{b_1+1}^1$ and $t_k \leq d_{b_2+1}^2$are on the due date of jobs. Connection (1) means that job $(a_1 + 1)$ of size 1 is not scheduled within its own due date, while connection (2) means that job $(a_1 + 1)$ of size 1 is scheduled within its own due date in batch k. Connection (3) means that job $(a_2 + 1)$ of size 2 is not scheduled within its own due date, while connection (4) means that job $(a_2 + 1)$ of size 2 is scheduled within its own due date in batch k. Finally, connection (5) means that henceforth, we will not schedule jobs in batch k any more.
Let $L^{(i)}(N(k, a_1, a_2, b_1, b_2))$ be the length of the arc corresponding to connection (i) of $N(k, a_1, a_2, b_1, b_2)$, $i = 1, 2, ..., 5$. Note that there may not exist arcs of connections (2) and (4), depending on whether it is satisfied with the conditions or not. The length can be calculated as follows:

(1) $L^{(1)}(N(k, a_1, a_2, b_1, b_2)) = w_{a_1+1}^1$;

(2) $L^{(2)}(N(k, a_1, a_2, b_1, b_2)) = 0$ if the connection exists ;

(3) $L^{(3)}(N(k, a_1, a_2, b_1, b_2)) = w_{a_2+1}^2$;

(4) $L^{(4)}(N(k,a_1,a_2,b_1,b_2)) = 0$ if the connection exits ;

(5) $L^{(5)}(N(k,a_1,a_2,b_1,b_2)) = 0$.

Note that $N(r+1,n_1,n_2,0,0)$ have no outgoing arc. Note that the shortest path can be interpreted in the context of Problem III as follow. For example, if the arc corresponding to connection (4) of $N(k,a_1,a_2,b_1,b_2)$ belongs to the shortest path, then it means that job (a_2+1) is scheduled to batch k.

Lemma 4 The total number of arcs is at most $O(rn^4)$.

Proof Since the number of nodes is at most $O(rn^4)$ and the number of outgoing arcs from each node is less than or equal to 5, the total number of arcs is at most $O(rn^4)$. ■

Theorem 5. *Problem III can be solved in $O(rn^4)$ time.*

Proof Since the constructed graph is acyclic, the algorithm of Ahuja et al. [1] can be solve the above shortest path problem in $O(|A|)$, where $|A|$ is the number of arcs. Since the number of arcs in the above graph is at most $O(rn^4)$ by Lemma 4, Problem III is polynomially solvable in $O(rn^4)$. ■

5.2 Identical weights

Let $\{\alpha(1),\alpha(2),...,\alpha(l)\}$ be the set of batches with odd capacity such that $t_{\alpha(1)} < t_{\alpha(2)} < \cdots < t_{\alpha(l)}$. Let $J_{\beta(j)}$ be the job of size 1 such that

$$d_{\beta(1)} = \min\{d_j^1 | d_j^1 \geq t_{\alpha(1)}\},$$

and for $k = 2,3,...,l$,

$$d_{\beta(k)} = \min\{d_j^1 | d_j^1 \geq t_{\alpha(k)} \ \ and \ \ d_j^1 > d_{\beta(k-1)}\}.$$

Lemma 5 There exists an optimal schedule such that $J_{\beta(j)}$ is scheduled in batch $\alpha(j)$, $j = 1,2,...,l$.

Proof Suppose that there exists an optimal schedule such that $\{J_{\beta(1)},...,J_{\beta(k-1)}\}$ are scheduled in batches $\alpha(1)$,...,$\alpha(k-1)$ and $J_{\beta(k)}$ is not scheduled in batch $\alpha(k)$. Without loss of generality, suppose that there exists a $J_{\bar{\beta}}$ of size 1 in batch $\alpha(k)$. By the definition of $J_{\beta(k)}$,

$$d_{\beta(k)} \leq d_{\bar{\beta}}.$$

It implies that the locations of $J_{\beta(k)}$ and $J_{\bar{\beta}}$ can be exchanged without the increase of the objective value. We can apply this argument until Lemma 5 is obtained. This completes the proof. ■

From Lemma 5, we can fix $J_{\beta(j)}$ to batch $\alpha(j)$ in advance, $j = 1,2,...,l$. Henceforth, without out loss of generality, we can assume that all batches have even capacities. Let I_i be the set of job j such that

$$I_i = \{J_j | t_i \leq d_j < t_{i+1}\}, \quad i = 1,2,...,r.$$

Without lose of generality, assume that

$$\sum_{i=k}^{r} c_r \leq \sum_{j \in I_k \cup I_{k+1} \cup \cdots \cup I_r} s_j, \quad k = 1,2,...,r.$$

Note that if job can be put into batch i, then it can also be put into batch $(i-1)$, $i = 2,3,...,r$.

Consider Problem III-1 which is defined as follows. For each job j of size 2, divide the job into two jobs denoted as jobs $j[1]$ and $j[2]$ such that

$$s'_{j[1]} = s'_{j[2]} = 1, \quad d'_{j[1]} = d'_{j[2]} = d_j \quad and \quad w'_{j[1]} = w'_{j[2]} = \frac{1}{2},$$

while for each job j of size 1,

$$s'_j = 1, \quad d'_j = d_j \quad and \quad w'_j = 1.$$

The objective is to minimize the weighted number of tardy jobs.
We can construct an algorithm for Problem III-1. Let jobs be stocked in the decreasing order of w'_j, implying that the job located in the lowermost position has the largest weight in each batch.

Algorithm for Problem III-1

Step 1. Set $i = r$ and $T = \emptyset$.

Step 2. Put job j to batch i in the decreasing order of w'_j, $j \in I_i \cup I_{i+1} \cup \cdots \cup I_r \setminus T$ until the capacity of batch i is fully filled. If the tie happens, then schedule the job with the largest due dates prior to other jobs. Let T_i be the set of the jobs allocated to batch i. $T = T \cup T_i$ and $i = i - 1$ (Without loss of generality, suppose that job $j[1]$ is selected prior to job $j[2]$).

Step 3. If $i > 0$, go to Step 2.

Step 4. Reorder the scheduled jobs according to the Earliest Due Date (EDD) rule.

The time complexity of Greedy Algorithm is $O(rn + n \log n)$. Let $\bar{S}$ and $v(\bar{S})$ be the optimal schedule and the optimal value of Problem III-1, respectively. Note that $\bar{S}$ may not be feasible in Problem III. In $\bar{S}$, one of the following cases occurs.

Case (i) Jobs $j[1]$ and $j[2]$ are scheduled in the same batch ;

Case (ii) Job $j[1]$ is scheduled uppermost in the batch and job $j[2]$ is scheduled in immediately previous batch, or

Case (iii) Both of jobs $j[1]$ and $j[2]$ are not allocated to $\bar{S}$;

Case (iv) Jobs $j[1]$ is scheduled uppermost in batch 1.

If case (iv) happens, then it implies that all jobs of size 1 have been scheduled and the number of these jobs are odd. In this case, the upper bound of the optimal value of the original problem is $v(\bar{S}) - \frac{1}{2}$. This implies that there exists an optimal schedule in an original problem with at least one empty slot in batch 1. If case (iv) does not happen, then $v(\bar{S})$ is the upper bound of the optimal value.

Lemma 6 We can transform the schedule $\bar{S}$ into an optimal schedule S^* of the original problem.

Proof If case (iv) happens, then we can delete the job located in the uppermost position of batch 1 because the upper bound is $v(\bar{S}) - \frac{1}{2}$. Henceforth, without loss of generality, we assume that case (iv) does not occur. The schedule $\bar{S}$ can be decomposed into k clusters which are the set of batches as follows: Let cluster i, $\{\alpha_i, \alpha_i + 1, ..., \alpha_{i+1} - 1\}$, $i = 1, 2, ..., k$, be defined as the set of batches such that the job in the lowermost position of batch $\alpha_i + j$

and the job in the uppermost position of batch $\alpha_i + j + 1$ come from the same job of size 2 in the original problem. Note that $\alpha_1 = 1$ and $\alpha_{k+1} - 1 = r$.
We can make the schedule $\bar{S}$ feasible to the original problem without the decrease of the objective value as follows: Consider cluster l, $l = 1, 2, ..., k$. Without loss of generality, the number of batches in cluster l is larger than 1. Since the capacity of batch is even, at least one of jobs allocated to batch $\alpha_{l+1} - 1$ has the original size 1. Let this job be job q, and the set of jobs scheduled earlier than job q be R in cluster l. We can move job q into the uppermost position of batch α_l, and push all jobs of R down by one unit while the jobs of R allocated to the lowermost position of each batch should be moved into the uppermost location of the next batch. Since the job in the lowermost position of one batch and the job in the uppermost location of the next batch come from the same job in the original problem, this transformation can make the cluster part l of $\bar{S}$ feasible in the original problem without the decrease of the objective value. This completes the proof. ■

Theorem 6. *Problem III with identical weights can be solved in $O(rn + n \log n)$ time.*

Proof It immediately hold from Lemma 6. ■

6 Problem IV: $1|batch(i),\ s_j \in \{1, 2\}| \sum T_j$

In Problem IV, there exists an optimal schedule such that jobs of each size are scheduled in EDD order, respectively. Thus, we can apply the reduction of Problem II straightforwardly.

Theorem 7. *Problem IV can be solved in $O(rn^2)$.*

For the weighted case, however, we cannot determine the sequence of jobs of each size in advance. Thus, the approach of Problem II does not seem to be promising.

7 Concluding Remarks and Future Works

We consider the problem of scheduling containers with sizes 1 or 2 to ships whose capacity and the departure time are fixed. The objectives are to minimize L_{max}, $\sum w_j C_j$, $\sum w_j U_j$, and $\sum T_j$. We verify that the problem with L_{max} is equivalent to the scheduling problem with GoS eligibility constraint, and the problems with $\sum w_j C_j$, $\sum w_j U_j$, and $\sum T_j$ are the special case of generalized assignment problem. The results for four problems are follows.

	L_{max}	$\sum w_j C_j$	$\sum w_j U_j$	$\sum w_j T_j$
general	$O(rn + n \log n)$	$O(rn^2)$	$O(rn^4)$	Open
$w_j = 1$		$O(n \log n)$	$O(rn + n \log n)$	$O(rn^2)$

As seen above, however, it is unknown whether there exists an algorithm to solve the problem with $\sum w_j T_j$ or not.

References

[1] R. Ahuja, K. Mehlhorn, J. Orlin, and R. Tarjan. Faster Algorithms for the Shortest Path Problem. *Journal of the Assocation for Computing Machinery*, 37:213–223, 1990.

[2] Z.-L. Chen, L. Lei, and H. Zhong. Container vessel scheduling with bi-directional flows. *Operations Research Letters*, 35:186–194, 2007.

[3] M. Christiansen, K. Fagerholt, and D. Ronen. Ship Routing and Scheduling: Status and Perspectives. *Transportation Science*, 38(1):1–18, 2004.

[4] K. Fagerholt. Ship scheduling with soft time windows: An optimisation based approach. *European Journal of Operational Research*, 131:559–571, 2001.

[5] K. Fagerholt and M. Christiansen. A Combined Ship Scheduling and Allocation Problem. *Journal of the Operational Research Society*, 51(7):834–842, 2000.

[6] A. V. Goodchild and C. F. Daganzo. Double-Cycling Strategies for Container Ships and Their Effect on Ship Loading and Unloading Operations. *Transportation Science*, 40(4): 473–483, 2006.

[7] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimisation and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 5:236–287, 1979.

[8] K. Y. Kim and K. H. Kim. A routing algorithm for a single straddle carrier to load export containers onto a container ship. *International Journal of Production Economics*, 59: 425–433, 1999.

[9] J. Y.-T. Leung. On scheduling independent tasks with restricted execution times. *Operations Research*, 30(1):163 –171, 1982.

[10] J. Y.-T. Leung. Personal communication. 2008.

[11] Y.-M. Park and K. H. Kim. A scheduling method for berth and quai cranes. *OR Spectrum*, 25:1–23, 2003.

[12] R. Stahlbock and S. Voß. Operations research at container terminals: a literature update. *OR Spectrum*, 30:1–52, 2008.

[13] D. Steenken, A. Henning, S. Freigang, and S. Voß. Routing of straddle carriers at a container terminal with the special aspect of internal moves. *OR Spectrum*, 15:167–172, 1993.

[14] D. Steenken, S. Voß, and R. Stahlbock. Container terminal operations and operations research – a classification and literature review. *OR Spectrum*, 26:3–49, 2004.

[15] I. F. A. Vis and R. de Koster. Transshipment of containers at a container terminal: An overview. *European Journal of Operational Research*, 147:1–16, 2003.