Schedule Template Design and Storage Allocation for Cyclically Visiting Feeders in Container Transshipment Hubs

Der-Horng Lee
(Corresponding author)
Department of Civil & Environmental Engineering
National University of Singapore
Block E1A, #07-16, 1 Engineering Drive 2
Singapore 117576
Phone: (+65)65162131
Email: dhl@nus.edu.sg

Jian Gang Jin
Department of Civil & Environmental Engineering
National University of Singapore
Block E1, #08-20, 1 Engineering Drive 2
Singapore 117576
Email: jin_jiangang09@nus.edu.sg

Jiang Hang Chen
Department of Civil & Environmental Engineering
National University of Singapore
Block E1, #08-20, 1 Engineering Drive 2
Singapore 117576
Email: chen_jianghang07@nus.edu.sg

Submission Date: Oct 28, 2011
Word Count: 5347 words + 3 figuires + 6 tables = 7597

ABSTRACT

This paper studies the management of transshipment flows in a container terminal which consists of designing a visiting schedule template for feeder vessels and determining storage locations for transshipment containers. A mixed integer programming formulation is developed with an objective of minimizing the total distance travelled by transshipment flows between quayside and storage yard as well as the workload imbalance in time. To solve the problem we devise an algorithm based on Lagrangian relaxation to find near-optimal solutions within short computational times. Numerical experiments are conducted to assess the effectiveness and efficiency of the algorithm and the benefit from adjusting feeder visiting schedule.

Keywords: Schedule template, Storage allocation, Container terminal, Transshipment flow

1. INTRODUCTION

Maritime freight transport is an important part of the global logistic system. With over 80 percent of world merchandize trade by volume being carried by sea, maritime transport remains the backbone supporting international trade and globalization (*1*). Containerization is an evolutionary change for maritime freight transport, and has increased the efficiency to a large extent. Benefiting from the economics of scale, the maritime shipping network has employed larger container vessels and introduced the hub-and-spoke system in which large container vessels (mother vessels) visit large transshipment terminals (hubs) while small vessels (feeders) connect the hubs with other ports (spokes). The need for an efficient management of logistic activities at modern container terminals, and especially at the major hubs, is well recognized (*2*). The operational efficiency of ports not only affects the efficiency of the whole container shipping network, but also determines the port operational costs.

As a key node of the maritime shipping network, a container transshipment hub provides container loading and discharging services to both mother vessels and feeders, and offers yard space for container temporary storage. Yard storage allocation is one of the typical decision problems faced by port operators, especially for large transshipment hubs with scarce-land issues. In a transshipment terminal, container batches are exchanged between mother vessels and feeders. Such transshipment flows are firstly discharged from their inbound vessels, placed in certain blocks of the storage yard temporarily and finally loaded to their corresponding outbound vessels. The container movements between the quayside and the storage yard are conducted by yard trucks. Longer distance travelled by yard trucks for moving a transshipment flow not only results in larger travel cost but also causes an unfavorable circumstance in terms of fast loading and discharging operations at the quayside. Thus, it is essential for port operators to develop a storage allocation plan in a broad view with consideration of all the transshipment flows.

Feeder scheduling is the specific motivation of this study: to support decisions at the tactical level for port operators of negotiating with shipping liners on their vessel arrival times. Vacca et al. (*3*) mentioned that the efficiency of yard operations in a transshipment hub can be improved by taking into account the peculiarities of transshipment flows when the arrival times of feeders are not known in advance but can be decided by the terminal. This idea adjusts the calling schedule of feeders in such a way that the quayside workload (loading and discharging) varies as little as possible over the time. The current convention is that shipping liners have dominance of the arrival times of container vessels (e.g., Monday morning shift), and container ports satisfies the requests in order to gain attractiveness. However, this practice often results in a concentration of the quayside workload over the time and it is unfavorable for quay crane operations. That's why port operators intend to gain benefits from adjusting feeder calling times.

This paper deals with modeling the integration of the feeder scheduling problem and the storage allocation problem for the transshipment flows between mother vessels and feeders. The feeder scheduling decision determines the arrival and departure times of transshipment flows while the storage allocation decides where to put the flows in the yard. In other words, the two types of decisions affect the transshipment flows temporarily and spatially. Thus, we intend to organize the transshipment flows in an optimal manner by incorporating both the temporal and spatial considerations. We remark that the contribution of this study lies on the following two aspects:

- The concept of feeder scheduling is studied in detail and modeled as a mixed

integer program.

- The storage allocation problem is integrated with feeder scheduling problem to improve the overall operational efficiency for transshipment flows.

This paper is organized as follows. Section 2 reviews relevant papers in the literature and Section 3 presents the detailed problem description as well as a mixed integer program. A Lagrangian relaxation algorithm is developed in Section 4 followed by computational experiments in Section 5. Finally, Section 6 draws the conclusion.

2. LITERATURE REVIEW

In this section we review some recent research outcomes which are most relevant to the storage allocation problem and the feeder scheduling problem. For a comprehensive review on container port operations, readers may refer to Steenken et al. (*4*) and Stahlbock and Voss (*5*). Yard storage space allocation is one of the typical decision problems faced by port operators. It is a common practice that the storage policies are container type dependent and the yard is partitioned into different areas according to their operational needs (*6*). Kim and Kim (*7*) considered how to allocate storage space for import containers under the segregation strategy. Relationship between stack height and number of re-handles was analyzed in order to minimize the expected total number of re-handles for outside trucks to pick up containers in the yard. Three arrival patterns were considered: constant, cyclic and dynamic arrival rate. Ng et al. (*6*) studied the export yard template designing problem for vessel services with a cyclical calling pattern. Given the daily arrival information of the export containers associated with each vessel service, the authors tried to determine the storage locations for the export container clusters with an objective of balancing the workload over the yard. The yard template designing problem was modeled as an integer program. Zhang et al. (*8*) applied a hierarchical approach to the storage allocation problem where import, export and transshipment containers are mixed together. The higher level balances the workload among all the blocks such that the berthing time can be minimized. The lower level minimizes the total distance between storage blocks and vessel berthing locations by allocating containers corresponding to each vessel to the storage space determined in the first level. Chen et al. (*9*) modeled the storage allocation as a special 2-dimensional packing problem and applied several heuristic approaches. Moccia and Astorino (*10*) proposed a *Group Allocation Problem* which is to study the transshipment container flows through the storage yard. A mathematical model was formulated considering all the costs of handling work occurring at discharging, loading and reallocation of container groups. Moccia et al. (*11*) developed column generation algorithms for the group allocation problem. From the previous literature, we can see that most works on storage space allocation focus on the storage yard side and assume the quayside operations (e.g., berth allocation, quay crane scheduling/assignment and feeder scheduling) have been already determined. However, to achieve better efficiency it is necessary to integrate related operations, like storage allocation and feeder scheduling, instead of solving them hierarchically.

Regarding the feeder scheduling problem, this concept was first introduced by Vacca et al. (*3*). However, the authors only proposed the idea and did not provide any mathematical formulation. In real-world maritime shipping, it is common that container vessels maintain cyclical arrival patterns, mostly weekly pattern. The feeder scheduling problem is a new tactical planning problem that allows port operators to control the quayside congestion and traffic issues by designing a good schedule template for feeders. With a determined schedule template, feeders visit the terminal accordingly on a weekly basis. In the literature, Moorthy and Teo (*12*) first studied the berth template problem

which consists of assigning a home berth to cyclically visiting vessels. It is also a tactical planning problem and its output is subsequently used as a key input for operational planning problems. Vacca et al. (*3*) stated that feeder scheduling can reduce the congestion (loading and discharging operations) at the quayside by adjusting the feeder arrival times. Such a schedule template problem resembles the berth template problem as both of them manage the feeder operations. However, the former focuses on the temporal control while the latter deals with spatial management.

3. MODEL FORMULATION

In this section we first provide a detailed description of the problem of schedule template design and storage allocation for cyclically visiting feeders. Then, a mixed integer programming formulation is developed.

3.1. Problem Description

The studied problem is to simultaneously consider two decisions from the viewpoint of port operators: one is to allocate storage space to transshipment flows and the other is to determine the calling schedule for feeders. Transshipment container flows between mother vessels and feeders need yard space resource for temporary storage and handling equipment (e.g., quay cranes, yard cranes and yard trucks) for container loading and discharging. The two decisions manage the transshipment flows and determine the operational cost directly. In this paper, we intend to tackle this problem from two aspects:

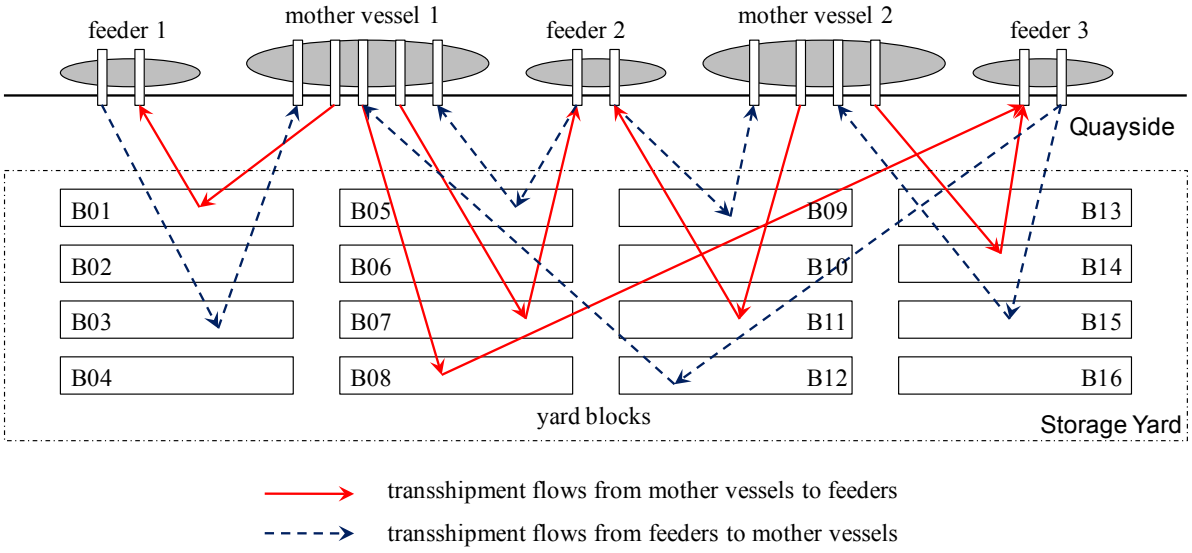

FIGURE 1: Transshipment flows between mother vessels and feeders.

a) Spatial planning:

Spatial planning aims at reducing the total distance of the transshipment flows from the quayside to the storage yard and vice versa. Determined by the assigned storage space for transshipment flows, the transportation distance has an impact on the operational efficiency as large distance is not in favor of fast loading and discharging. Figure 1 presents the flows in a geographical manner with 2 mother vessels, 3 feeders and 16 yard blocks. A transshipment flow is firstly discharged from its inbound vessel and then moved to a yard block for temporary storage. Finally, when the corresponding outbound vessel arrives, it is transported back to the quayside by yard trucks for loading. The storage positions of transshipment flows should be assigned in such a way that the total distance of container movements could be minimized. Note that in this study we focus on deciding the block locations for container storage, and simplify the detailed

considerations of container stacking and re-handles within a block. Besides, we consider the movements of container batches instead of individual containers. This is because the integrated feeder scheduling and storage allocation problem is at the tactical planning level, and the detailed stacking and re-handling decisions for individual containers should be determined at the operational level.

b) Temporal planning:

Temporal planning is to assign a service time slot (working shift) to container vessels in a proactive way from the terminal's perspective, unlike the convention that shipping liners establish the port staying time window and port operators have to provide service reactively according to the time window. In this paper, we assume that the service requests from mother vessels are always satisfied while the port operator has the authority on deciding the service times of feeders. We remark that this assumption is reasonable when there is an alliance between the port and shipping liners. With assigned service times by port operators, feeders follow the schedule and maintain a weekly arrival pattern. Such a proactive operational strategy provides port operators an opportunity of reducing the workload (container loading and discharging) congestion by adjusting feeder calling schedules.

The calling schedules of container vessels determine the arrival and departure times of transshipment flows, as well as the workload distribution in time for handling equipment. Figure 2 shows an illustrative example of the workload distribution over two planning horizons each of which has seven time periods. There are three transshipment flows each of which has three statuses: unloading, in storage and loading. During loading and unloading time periods port operators have to allocate handling equipment to conduct the operations, while allocated blocks are occupied when flows are in storage and during loading and unloading. In this example, Flow 1 arrives at the first time period and is loaded onto its outbound vessels at time period 5. Regarding Flow 2, as its outbound vessel arrives before its inbound vessel in the current planning horizon, the containers will stay in the terminal until the arrival of its outbound vessel in the next planning horizon. As can be seen from Figure 2, the three transshipment flows lead to an imbalanced workload distribution in time. From the operational point of view, an evenly distributed workload circumstance is preferred in that large workload imbalance would make handling equipment sometimes busy and sometimes idle. As the schedule of unloading and loading operations is determined by the vessel calling times, the workload imbalance can be reduced by adjusting the calling schedule of vessels.

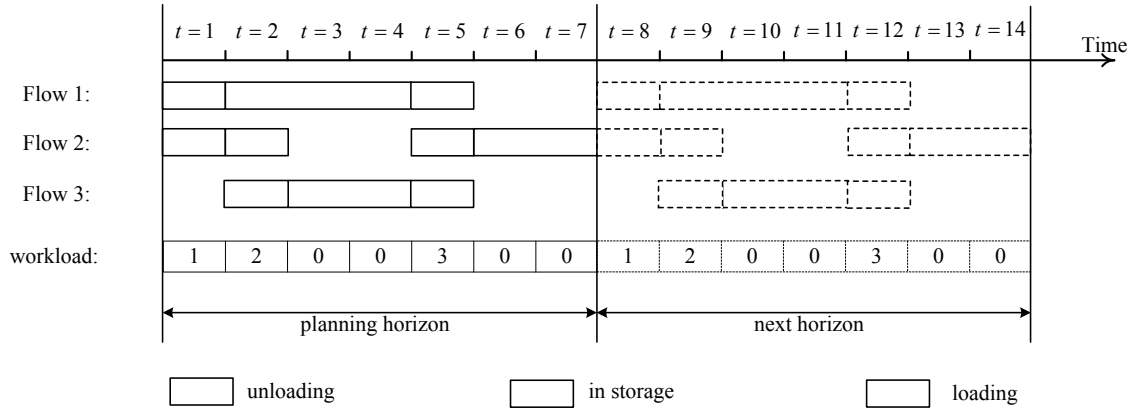

FIGURE 2: An illustrative example of workload distribution.

3.2. Assumptions

To simplify the problem, the following assumptions are made.

(1) The berth template for mother vessels and feeders is known.

(2) The container loading and unloading of a feeder can be done within one working shift.

The berth template design (assigning the home berth position for each vessel) is another important decision problem which also has an impact on the total travel distance of transshipment flows within the terminal. In order to achieve more advantageous flow movements, the schedule template design problem studied in this paper and the berth template design problem need to be solved iteratively. However, in this paper we only tackle the feeder schedule template design problem with a given berth template. Assumption (2) is reasonable as it is in line with the practice.

3.3. Formulation

Given the length of the cyclical planning horizon, e.g., a week, it is discretized into a series of working shifts. We consider two types of transshipment flows: flows from mother vessels to feeders and flows from feeders to mother vessels. For each flow, the service time related with its mother vessel is given while the service time of the corresponding feeder should be determined. Given a yard layout and a set of yard blocks, storage space should be allocated to the transshipment flows. Note that the assigned service times for feeders have to respect the allowable time windows, and storage capacity constraint of each block must be hold. The notation of the mathematical formulation is defined in Table 1.

Table 1: Formulation Notation

Sets:

- M : set of mother vessels
- N : set of feeders
- K : set of yard blocks
- T : set of working shifts, $T = \{1,2,\cdots,\bar{t}\}$
- I_1 : set of transshipment flows from mother vessels to feeders
- I_2 : set of transshipment flows from feeders to mother vessels
- I : set of all transshipment flows, $I = I_1 \cup I_2$
- S : set of quay positions

Parameters:

- θ_i^N : the corresponding feeder of flow $i \in I$
- o_i : $\in S$, the arrival quay position of flow $i \in I$
- d_i : $\in S$, the departure quay position of flow $i \in I$
- q_i : the amount of containers in flow $i \in I$, measured in Twenty-foot Equivalent Units (TEUs)
- t_i^a : the arrival working shift of flow $i \in I_1$
- $\left[\underline{T}_i^d, \overline{T}_i^d\right]$: [earliest, latest] feasible departure time of flow $i \in I_1$
- $\left[\underline{T}_i^a, \overline{T}_i^a\right]$: [earliest, latest] feasible arrival time of flow $i \in I_2$
- t_i^d : the departure working shift of flow $i \in I_2$
- l_{ks} : the travel distance between block $k \in K$ and quay position $s \in S$
- Q_k : the storage capacity of block $k \in K$

Decision variables:

- x_{ik} : binary, equal to 1 if flow $i \in I$ is put in block $k \in K$ for temporary storage, 0 otherwise

y_{it}^a	: binary, equal to 1 if flow $i \in I_2$ arrives at working shift $t \in T$, 0 otherwise
y_{it}^d	: binary, equal to 1 if flow $i \in I_1$ departs at working shift $t \in T$, 0 otherwise
z_i	: binary, equal to 1 if the departure working shift of flow i is later than its arrival working shift in one planning horizon, 0 otherwise
u_{it}	: binary, equal to 1 if flow $i \in I$ is within the terminal (unloading, in storage and loading statuses), 0 otherwise
w_h	: the highest workload per working shift
w_l	: the lowest workload per working shift

Note that the arrival and departure quay positions of transshipment flows o_i and d_i can be obtained from the berth template for container vessels which is assumed to be known. t_i^a and t_i^d are related to the service times of mother vessels. Regarding the feasible arrival/departure time windows, $\overline{T}_i^d$ ($\overline{T}_i^a$) are allowed to be smaller than $\underline{T}_i^d$ ($\underline{T}_i^a$) since the vessels visit the port cyclically. With the notation defined in Table 1, the problem is formulated as follows:

Objective function:

$$\text{Min} \quad \lambda \sum_{i \in I} \sum_{k \in K} \left(l_{ko_i} + l_{kd_i}\right) q_i x_{ik} + (1-\lambda)(w_h - w_l) \tag{1}$$

The objective function is a convex combination of two parts: spatial objective of minimizing the total travel distance of all flows between the quayside and storage yard, and temporal objective of minimizing the gap between the highest and lowest workload per working shift. The spatial objective reflects the transportation cost of yard trucks for carrying out all the container movements during the loading and unloading operations while the temporal objective indicates the quay crane workload imbalance in time. The two objectives are weighted by the parameter $\lambda \in [0,1]$. Note that λ is predetermined by port operators which takes into account the relative preferences and their monetary costs.

Constraints:

$$\sum_{k \in K} x_{ik} = 1 \quad \forall\, i \in I \tag{2}$$

$$\sum_{t \in T} y_{it}^d = 1 \quad \forall i \in I_1 \tag{3}$$

$$\sum_{t \in T | \underline{T}_i^d \le t \le \overline{T}_i^d} y_{it}^d = 1 \quad \forall i \in I_1 | \underline{T}_i^d < \overline{T}_i^d \tag{4}$$

$$\sum_{t \in T | t \le \overline{T}_i^d} y_{it}^d + \sum_{t \in T | t \ge \underline{T}_i^d} y_{it}^d = 1 \quad \forall i \in I_1 | \underline{T}_i^d > \overline{T}_i^d \tag{5}$$

$$\sum_{t \in T} y_{it}^a = 1 \quad \forall i \in I_2 \tag{6}$$

$$\sum_{t \in T | \underline{T}_i^a \le t \le \overline{T}_i^a} y_{it}^a = 1 \quad \forall i \in I_2 | \underline{T}_i^a < \overline{T}_i^a \tag{7}$$

$$\sum_{t \in T | t \le \overline{T}_i^a} y_{it}^a + \sum_{t \in T | t \ge \underline{T}_i^a} y_{it}^a = 1 \quad \forall i \in I_2 | \underline{T}_i^a > \overline{T}_i^a \tag{8}$$

$$0 \le z_i + \left(t_i^a - \sum_{t \in T} t y_{it}^d \right) / \overline{t} < 1 \quad \forall\, i \in I_1 \tag{9}$$

$$1 - t_i^a \le t(u_{it} + z_i - 1) \le \sum_{t' \in T} t' y_{it'}^d, \quad \forall i \in I_1, \forall t \in T \tag{10}$$

$$\sum_{t' \in T} t' y_{it'}^d - \bar{t} \le \left(\bar{t} + 1 - t\right)(u_{it} + z_i - 1) \le \bar{t} + 1 - t_i^a \quad \forall i \in I_1, \forall t \in T \tag{11}$$

$$\sum_{t \in T} (u_{it} + z_i - 1) = \sum_{t \in T} t y_{it}^d - t_i^a + 1 \quad \forall i \in I_1 \tag{12}$$

$$0 \le z_i + \left(\sum_{t \in T} t y_{it}^a - t_i^d\right) / \bar{t} < 1 \quad \forall i \in I_2 \tag{13}$$

$$1 - \sum_{t' \in T} t' y_{it'}^a \le t(u_{it} + z_i - 1) \le t_i^d \quad \forall i \in I_2, \forall t \in T \tag{14}$$

$$t_i^d - \bar{t} \le \left(\bar{t} + 1 - t\right)(u_{it} + z_i - 1) \le \bar{t} + 1 - \sum_{t' \in T} t' y_{it'}^a \quad \forall i \in I_2, \forall t \in T \tag{15}$$

$$\sum_{t \in T} (u_{it} + z_i - 1) = t_i^d - \sum_{t \in T} t y_{it}^a + 1 \quad \forall i \in I_2 \tag{16}$$

$$w_h \ge \sum_{i \in I_1 | t_i^a = t} q_i + \sum_{i \in I_1} q_i y_{it}^d + \sum_{i \in I_2} q_i y_{it}^a + \sum_{i \in I_2 | t_i^d = t} q_i \quad \forall t \in T \tag{17}$$

$$w_l \le \sum_{i \in I_1 | t_i^a = t} q_i + \sum_{i \in I_1} q_i y_{it}^d + \sum_{i \in I_2} q_i y_{it}^a + \sum_{i \in I_2 | t_i^d = t} q_i \quad \forall t \in T \tag{18}$$

$$\sum_{i \in I} q_i x_{ik} u_{it} \le Q_k \quad \forall k \in K, \forall t \in T \tag{19}$$

$$y_{it}^a = y_{jt}^a \quad \forall i, j \in I_2, \forall t \in T | \theta_i^N = \theta_j^N \tag{20}$$

$$y_{it}^d = y_{jt}^d \quad \forall i, j \in I_1, \forall t \in T | \theta_i^N = \theta_j^N \tag{21}$$

$$y_{it}^a = y_{jt}^d \quad \forall i \in I_2, \forall j \in I_1, \forall t \in T | \theta_i^N = \theta_j^N \tag{22}$$

$$x_{ik} \in \{0,1\} \quad \forall i \in I, \forall k \in K \tag{23}$$

$$y_{it}^a \in \{0,1\} \quad \forall i \in I_2 \tag{24}$$

$$y_{it}^b \in \{0,1\} \quad \forall i \in I_1 \tag{25}$$

$$z_i \in \{0,1\} \quad \forall i \in I \tag{26}$$

$$u_{it} \in \{0,1\} \quad \forall i \in I, \forall t \in T \tag{27}$$

$$w_h, w_l \ge 0 \tag{28}$$

Constraints (2) assign each transshipment flow to a certain yard block for temporary storage. Constraints (3) assure that one and only one working shift during the planning horizon is selected as the departure time for each transshipment flow $i \in I_1$. Constraint (4) and (5) guarantee that the assigned times of transshipment flows I_1 fall within their corresponding feasible time windows. Similarly, the arrival times of transshipment flows I_2 are specified by Constraints (6)-(8). Imposed by Constraints (9), the binary variable z_i is set to 1 if and only if the departure working shift $\sum_{t \in T} t y_{it}^d$ of transshipment flow $i \in I_1$ is later than its arrival working shift t_i^a. Similarly, Constraints (13) assign values to z_i for transshipment flows I_2. Constraints (10)-(12) and (14)-(16) define the variable u_{it} by enforcing its relationship with variable z_i, y_{it}^a and y_{it}^d. Figure 3

shows two scenarios of different arrival and departure schedules of transshipment flow $i \in I_1$ and the values of the corresponding variables. Take Figure 3(a) as an example, Constraints (10) set u_{i6} and u_{i7} to 0. Similarly, u_{i1} takes 1 as ensured by Constraints (11). Constraints (12), along with (10) and (11), assign 1 to the rest of u_{it}. Constraints (17)-(18) assign the highest and lowest workload per working shift to variables w_h and w_l, respectively. The total amount of containers within a block should not exceed the storage capacity at each working shift, as guaranteed by Constraints (19). Ensured by Constraints (20), any two transshipment flows in I_2 with the same inbound feeder should arrive at the same working shift. Constraints (21) and (22) impose similar restrictions. Finally, the domain of decision variables is defined by Constraints (23)-(28).

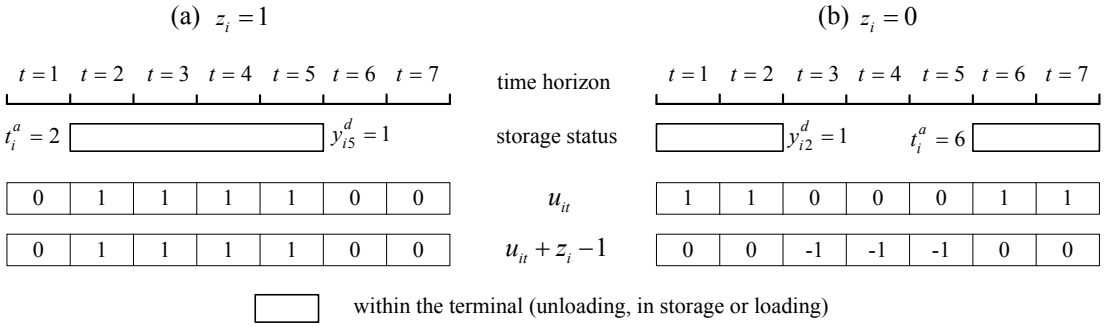

FIGURE 3: Two scenarios with different arrival and departure schedules for transshipment flow $i \in I_1$

Note that Constraints (19) related with block storage capacity is quadratic. Thus, we linearize it by defining an additional decision variable $\delta_{ikt} \in \{0,1\}$, which is set to 1 if $x_{ik} = u_{it} = 1$ and 0 otherwise. The related additional constraints are defined as follows:

$$\delta_{ikt} \geq x_{ik} + u_{it} - 1 \quad \forall i \in I, \forall k \in K, \forall t \in T \tag{29}$$

$$\delta_{ikt} \leq x_{ik} \quad \forall i \in I, \forall k \in K, \forall t \in T \tag{30}$$

$$\delta_{ikt} \leq u_{it} \quad \forall i \in I, \forall k \in K, \forall t \in T \tag{31}$$

$$\delta_{ikt} \in \{0,1\} \quad \forall k \in K, \forall t \in T \tag{32}$$

$$\sum_{i \in I} q_i \delta_{ikt} \leq Q_k \quad \forall k \in K, \forall t \in T \tag{33}$$

Therefore, the problem of schedule template design and storage allocation for cyclically visiting feeders can be formulated as a mixed integer linear program as follows:

$$\textbf{[P]} \quad \text{Min} \quad \lambda \sum_{i \in I} \sum_{k \in K} \left(l_{ko_i} + l_{kd_i}\right) q_i x_{ik} + (1-\lambda)(w_h - w_l)$$
$$s.t. \ (2) - (18) \text{ and } (20) - (33)$$

4. LAGRANGIAN RELAXATION ALGORITHM

In this section, we propose a Lagrangian relaxation solution method for the proposed mixed integer program **[P]**. Since the introduction of Lagrangian approach by Held and Karp (*13*, *14*), it has gained significant applications in many research fields. In the literature on container port operations, Zhang et al. (*15*) employed Lagrangian relaxation methods to solve the dynamic crane deployment problem in container storage yards, and Monaco and Sammarra (*16*) developed a Lagrangian algorithm for a strong formulation of the berth allocation problem. The merit of the Lagrangian relaxation method lies in the relaxation of complicated constraints by penalizing the violated constraints in the objective function. As the complicated constraints have been relaxed, the original

problem becomes easier to solve.

In our problem, based on the observation that the block capacity constraints (33) are the only constraints linking storage allocation decision and feeder scheduling decision, we apply the Lagrangian relaxation method to the original formulation **[P]** by relaxing Constraints (33). For given Lagrangian multipliers $\boldsymbol{\mu} = \{\mu_{kt}\} \geq \mathbf{0}$ associated with Constraints (33), the Lagrangian relaxation of the original problem **[P]** is defined as follows:

$$[\textbf{P-LR}(\boldsymbol{\mu})] \quad \text{LR}(\boldsymbol{\mu}) = \min \quad \lambda \sum_{i\in I}\sum_{k\in K}\left(l_{ko_i} + l_{kd_i}\right)q_i x_{ik} + (1-\lambda)(w_h - w_l) + \sum_{k\in K}\sum_{t\in T}\mu_{kt}\left(\sum_{i\in I} q_i \delta_{ikt} - Q_k\right)$$

$$s.t. \quad (2)-(18) \text{ and } (20)\text{-}(32)$$

Therefore, the Lagrangian dual problem is:

$$\textbf{[P-Dual]} \quad Z_D = \max_{\boldsymbol{\mu}\geq \mathbf{0}} \text{LR}(\boldsymbol{\mu}) \tag{34}$$

As the Lagrangian relaxation problems **[P-LR($\boldsymbol{\mu}$)]** are relatively easier to solve compared with the original problem **[P]**, we take advantage of MIP solvers to solve them to optimum or to find out near-optimal solutions if the computational time exceeds a pre-determined time limit. Lower bounds to the original problem **[P]** can be obtained from the Lagrangian relaxation problems. The subgradient method is employed to update the Lagrangian multipliers and the Lagrangian problems are iteratively solved with the updated multipliers.

In order to get primal bounds of the original problem **[P]**, we develop a heuristic based on the solutions to the Lagrangian relaxation problems. Given a solution to **[P-LR($\boldsymbol{\mu}$)]**, the decisions related with feeder scheduling $(\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{u}, \boldsymbol{w}) = \{y^a_{it}, y^d_{it}, z_i, u_{it}, w_h, w_l\}$ are selected. With the known feeder scheduling decisions, the remaining problem is to solve the storage allocation problem. The formulation of the resulting storage allocation problem is as follows:

$$\textbf{[P-Sub]} \quad C_{sub} = \min \quad \sum_{i\in I}\sum_{k\in K}\left(l_{ko_i} + l_{kd_i}\right)q_i x_{ik}$$

$$s.t. \quad \sum_{k\in K} x_{ik} = 1 \quad \forall i \in I$$

$$\sum_{i\in I} q_i \delta_{ikt} \leq Q_k \quad \forall k \in K, \forall t \in T$$

$$\delta_{ikt} \geq x_{ik} + u_{it} - 1 \quad \forall i \in I, \forall k \in K, \forall t \in T$$

$$\delta_{ikt} \leq x_{ik} \quad \forall i \in I, \forall k \in K, \forall t \in T$$

$$\delta_{ikt} \leq u_{it} \quad \forall i \in I, \forall k \in K, \forall t \in T$$

$$x_{ik} \in \{0,1\} \quad \forall i \in I, \forall k \in K$$

$$\delta_{ikt} \in \{0,1\} \quad \forall k \in K, \forall t \in T$$

Note that in **[P-Sub]**, only x_{ik} and δ_{ikt} are the decision variables while variables u_{it}, w_h and w_l are known. With both storage allocation and feeder scheduling decisions determined, a primal feasible solution can be obtained.

The subgradient procedure of the Lagrangian relaxation method is summarized in Table 1. The Lagrangian multipliers are initially set to 0. We solve the linear relaxation of the original problem **[P]** to get an initial lower bound *LB* while the initial upper bound *UB* is set to infinity. During each iteration of the subgradient procedure, firstly problem **[P-LR($\boldsymbol{\mu}$)]** is solved and *LB* is updated accordingly. After that, the feeder scheduling related

decisions are selected for the remaining problem **[P-Sub]**. UB and $Best$ can be updated once **[P-Sub]** is solved. Then, the standard subgradient method is applied to get $\boldsymbol{\mu}$ updated as equation (35). In the updating scheme, the step size t_n is computed by equation (36) where the parameter τ_n is initialized to 2 and halved once LB is not improved for three consecutive iterations.

$$\mu_{kt}^{n+1} = \mu_{kt}^{n} + t_n \left(\sum_{i \in I} q_i \delta_{ikt} - Q_k \right) \quad \forall k \in K, \forall t \in T \tag{35}$$

$$t_n = \frac{\tau_n (UB - \mathrm{LR}(\boldsymbol{\mu}^n))}{\sum_{k \in K} \sum_{t \in T} (\sum_{i \in I} q_i \delta_{ikt} - Q_k)^2} \tag{36}$$

At the end of each iteration, three stopping criteria are checked to determine whether or not the subgradient procedure should be terminated: the maximum number of iterations N_1 is reached; UB does not get improved for N_2 consecutive iterations; the gap between UB and LB is less than $UbGap$. In order to accelerate the solving process of the subproblems **[P-LR($\boldsymbol{\mu}$)]** and **[P-Sub]** by MIP solvers, instead of solving them to optimum we truncated the solving process according to the following two conditions: a time limit $TiLim$ and an optimality gap $EpGap$.

Table 1: Subgradient Procedure of the Lagrangian Relaxation Algorithm

Input: an instance
Output: a lower bound LB, an upper bound UB and the best feasible solution $Best$

Initialize multipliers $\boldsymbol{\mu} \leftarrow \mathbf{0}$, best lower bound $LB \leftarrow 0$, best upper bound $UB \leftarrow +\infty$;
Solve the linear relaxation of **[P]**, and update LB;
Repeat
    Solve the Lagrangian relaxation formulation **[P-LR($\boldsymbol{\mu}$)]**;
    if LR($\boldsymbol{\mu}$)>LB
        $LB \leftarrow$ LR($\boldsymbol{\mu}$);
    end if
    Obtain the feeder scheduling related decisions ($\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{u}, \boldsymbol{w}$);
    Solve the sub-problem **[P-Sub]** given ($\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{u}, \boldsymbol{w}$);
    if $\lambda C_{sub} + (1-\lambda)(w_h - w_l) < UB$
        $UB \leftarrow \lambda C_{sub} + (1-\lambda)(w_h - w_l)$
        Update $Best$;
    end if
    Update multipliers $\boldsymbol{\mu}$;
Until any one of the stopping criteria is met

5. COMPUTATIONAL EXPERIMENTS

In this section we first illustrate the generation of test instances and the settings of the Lagrangian relaxation algorithm. Then, we assess the efficiency and effectiveness of the developed algorithm by comparing its results with the optimal ones obtained by the MIP model **[P]**. Finally, the effectiveness of adjusting the feeder calling schedules is evaluated. The optimization model and the Lagrangian relaxation algorithm are coded in C++ and CPLEX 12.1 is used as the underlying MIP solver. All the numerical experiments are conducted on a PC with 3 GHz CPU and 4GB RAM.

5.1. Instance generation and algorithm settings

In order to validate our model and the Lagrangian algorithm, six sets of test instances are randomly generated each of which has 10 instances with the same parameters. The

amount of containers in each flow q_i is uniformly distributed within [500, 800] and the storage capacity of each block Q_k is set to 2,000. For each instance set, six parameters are pre-determined as listed in Table 2: number of mother vessels, number of feeders, number of transshipment flows, number of blocks, number of working shifts within one planning horizon and the length of the allowable time windows. For each instance, the corresponding mother vessel and feeder of each transshipment flow are randomly assigned. The service times of mother vessels and the allowable time windows of feeders are uniformly distributed within the horizon. The berth positions of all vessels are also randomly allocated at the quayside. Take Set 4 as an example, the terminal is visited by 10 mother vessels and 30 feeders each week (21 working shifts) and there are 60 transshipment flows. The length of the allowable time windows of feeders is 2 days (6 working shifts) and the storage yard has 24 yard blocks.

Table 2: Instance parameters

	$\|M\|$	$\|N\|$	$\|I\|$	$\|K\|$	$\|T\|$	L
Set 1	5	10	20	8	9	3
Set 2	6	15	30	12	12	6
Set 3	8	20	40	15	15	6
Set 4	10	30	60	24	21	6
Set 5	12	40	80	30	21	6
Set 6	15	50	100	36	21	6

L: the length of the allowable time windows

Several parameters of the Lagrangian algorithm are selected by trials and listed as follows:

- Maximum number of subgradient iterations: $N_1 = 50$.
- Maximum number of consecutive subgradient iterations with non-improved upper bound: $N_2 = \lceil \log|I| \cdot |K| \cdot |T| \rceil$.
- The gap between UB and LB: $UbGap$ =0.01.
- Time limit for solving the subproblems by CPLEX: $TiLim = \lceil \sqrt{|I| \cdot |K| \cdot |T|}/10 \rceil$.
- Optimality gap for solving the subproblems by CPLEX: $EpGap$ =0.003.

5.2. Results of Lagrangian relaxation algorithm

In order to assess the efficiency and effectiveness of the Lagrangian relaxation algorithm, the results are compared with the best solutions found by CPLEX. Table 3 lists the computational results of instance sets 1&2. A time limit of 43,200 seconds is set for CPLEX. In case the MIP model cannot be solved by the time limit, the best value of the objective function and the best lower bound are returned, as listed in the second and third columns. Four types of information of the Lagrangian algorithm are reported in columns (5)-(8): the number of subgradient iterations, the gap between lower and upper bounds, the best solution found by the algorithm and the computational time consumed. As can be seen, CPLEX is able to find optimal solutions in 18 out of 20 instances, and the computational time grows exponentially as the instance scale increases. The last two columns provide the gap between the solutions of the algorithm and the optimal ones and the lower bounds. The Lagrangian relaxation performs very well as the average of Gap^1 is less than 1% and Gap^2 is also small. In terms of efficiency, Lagrangian algorithm does not show much advantage over CPLEX for such relatively small-scale instances.

Table 4 compares the performance of the algorithm with CPLEX for instance sets 3&4. As can be seen, the instances become more computationally intractable as the instance scale increases. CPLEX can find optimal solutions in 7 out of 10 instances for Set 3 and none of the instances in Set 4 can be solved to optimum by CPLEX. The result

Table 3: Computational Results of Instance Sets 1&2

Instance	CPLEX			Lagrangian				Gap	
	Optimal	LB	Time(sec)	NoIter	Gap^{LR}(%)	Result	Time(sec)	Gap^{1}(%)	Gap^{2}(%)
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
Set1-I01	**9071.0**		1.9	10	7.17	9129.9	2.8	0.65	
Set1-I02	**8241.4**		0.3	12	7.51	8249.2	2.1	0.09	
Set1-I03	**7209.3**		0.7	10	7.74	7211.3	2.2	0.03	
Set1-I04	**7867.9**		0.9	10	7.52	7884.6	2.3	0.21	
Set1-I05	**7616.0**		0.7	14	5.66	7616.0	3.3	0.00	
Set1-I06	**8189.4**		0.9	12	11.83	8189.4	2.6	0.00	
Set1-I07	**9332.3**		1.5	12	4.00	9332.3	3.2	0.00	
Set1-I08	**6139.3**		0.3	10	7.32	6154.9	1.8	0.25	
Set1-I09	**7781.6**		1.3	14	6.11	7793.2	4.0	0.15	
Set1-I10	**5556.9**		0.5	11	6.70	5559.9	2.2	0.05	
Set2-I01	**8834.7**		701.0	11	13.07	8913.3	75.9	0.89	
Set2-I02	**11852.6**		240.0	12	10.18	11861.9	59.7	0.08	
Set2-I03	**10534.8**		4710.8	20	7.31	10534.8	147.0	0.00	
Set2-I04	**9810.3**		272.0	16	6.89	9877.5	59.8	0.68	
Set2-I05	**10795.3**		14.6	14	6.41	10838.5	89.6	0.40	
Set2-I06	**9360.1**		3236.0	11	13.94	9441.3	76.7	0.87	
Set2-I07	14795.0	14422.5	>43200	31	6.98	14796.0	244.9	0.01	2.59
Set2-I08	10275.6	10023.1	>43200	18	12.40	10424.2	160.2	1.45	4.00
Set2-I09	**11358.2**		242.9	19	8.88	11358.2	123.5	0.00	
Set2-I10	**9649.8**		164.5	13	9.40	9649.8	49.4	0.00	
Average					8.38			0.29	3.30

Gap^{1} = [(7)-(2)]/(2)×100%
Gap^{2} = [(7)-(3)]/(3)×100%

Table 4: Computational Results of Instance Sets 3&4

Instance	CPLEX			Lagrangian				Gap	
	Optimal	LB	Time(sec)	NoIter	Gap^{LR}(%)	Result	Time(sec)	Gap^1(%)	Gap^2(%)
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
Set3-I01	14345.9	14222.6	>43200	34	7.24	14345.9	471.2	0.00	0.87
Set3-I02	12505.1	12388.2	>43200	13	6.58	12631.4	132.6	1.01	1.96
Set3-I03	19227.7	18577.8	>43200	14	14.53	19226.7	252.6	-0.01	3.49
Set3-I04	**16568.0**		512.3	14	8.41	16605.6	141.4	0.23	
Set3-I05	**14241.2**		8882.0	25	7.51	14472.2	336.8	1.62	
Set3-I06	**19147.0**		29437.5	12	8.22	19191.7	155.8	0.23	
Set3-I07	**15517.0**		8602.6	18	7.23	15600.6	191.6	0.54	
Set3-I08	**12681.0**		1435.4	18	5.54	12681.0	185.7	0.00	
Set3-I09	**15963.0**		285.5	19	6.49	16111.1	111.3	0.93	
Set3-I10	**13902.8**		10289.2	17	8.67	13970.1	165.9	0.48	
Set4-I01	21577.7	21440.2	>43200	16	9.98	21676.2	338.8	0.46	1.10
Set4-I02	27680.9	26062.9	>43200	32	6.63	27462.7	1157.7	-0.79	5.37
Set4-I03	26637.0	25643.5	>43200	18	9.89	26554.3	627.8	-0.31	3.55
Set4-I04	26220.9	24126.2	>43200	16	12.64	25785.5	581.0	-1.66	6.88
Set4-I05	26301.8	26035.9	>43200	26	8.67	26465.7	942.3	0.62	1.65
Set4-I06	22491.8	21380.2	>43200	26	9.23	22324.2	896.0	-0.75	4.42
Set4-I07	23882.6	23268.9	>43200	22	7.99	23953.5	588.6	0.30	2.94
Set4-I08	27476.1	27067.6	>43200	15	9.76	27548.6	298.9	0.26	1.78
Set4-I09	25444.4	25264.0	>43200	37	6.74	25604.7	746.3	0.63	1.35
Set4-I10	22662.3	21467.1	>43200	18	11.47	22867.3	652.0	0.90	6.52
Average					8.67			0.24	3.22

$Gap^1 = [(7)-(2)]/(2)\times 100\%$
$Gap^2 = [(7)-(3)]/(3)\times 100\%$

Table 5: Computational Results of Instance Sets 5&6

Instance	CPLEX			Lagrangian				Gap	
	Optimal	LB	Time(sec)	NoIter	Gap^{LR}(%)	Result	Time(sec)	Gap^1(%)	Gap^2(%)
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
Set5-I01	37163.4	33617.6	>43200	14	16.61	37050.9	661.3	-0.30	10.21
Set5-I02	40879.2	36254.0	>43200	13	19.87	40462.2	601.6	-1.02	11.61
Set5-I03	35759.7	32891.5	>43200	16	14.72	35308.9	754.7	-1.26	7.35
Set5-I04	44015.3	42023.0	>43200	18	12.26	43595.3	847.3	-0.95	3.74
Set5-I05	36403.6	33682.7	>43200	13	14.18	35674.5	619.0	-2.00	5.91
Set5-I06	31257.6	30050.2	>43200	19	8.79	31211.1	897.7	-0.15	3.86
Set5-I07	35925.8	32968.3	>43200	15	9.61	35588.0	709.2	-0.94	7.95
Set5-I08	35832.0	32509.7	>43200	14	8.26	34431.4	660.4	-3.91	5.91
Set5-I09	42922.7	40047.8	>43200	23	9.08	41842.5	1081.4	-2.52	4.48
Set5-I10	38916.4	34573.1	>43200	22	9.31	37800.5	1032.5	-2.87	9.34
Set6-I01	51196.9	44639.0	>43200	17	13.28	48908.8	986.6	-4.47	9.57
Set6-I02	56330.8	46521.7	>43200	16	10.58	53418.9	930.3	-5.17	14.83
Set6-I03	54713.6	45723.2	>43200	25	9.58	51348.4	1439.3	-6.15	12.30
Set6-I04	47195.3	43072.8	>43200	14	16.78	46917.1	819.6	-0.59	8.93
Set6-I05	53031.8	43810.8	>43200	18	16.53	50171.2	1044.3	-5.39	14.52
Set6-I06	56208.7	49381.8	>43200	16	10.35	54750.8	922.6	-2.59	10.87
Set6-I07	50945.1	45434.6	>43200	19	11.74	50482.3	1103.7	-0.91	11.11
Set6-I08	44993.5	41136.0	>43200	19	9.48	44657.5	1103.9	-0.75	8.56
Set6-I09	50716.1	46085.6	>43200	15	13.67	50172.5	865.6	-1.07	8.87
Set6-I10	58358.8	51530.5	>43200	15	17.85	56598.5	874.2	-3.02	9.83
Average					12.63			-2.30	8.99

$Gap^1 = [(7)-(2)]/(2)\times 100\%$
$Gap^2 = [(7)-(3)]/(3)\times 100\%$

in column Gap^1 indicate that the algorithm can find very good near-optimal solutions and can even find better solutions than CPLEX in 5 out of 20 instances. For these medium-scale instances, the computational times demonstrate the algorithm's advantage in efficiency. The results of the instance sets 5&6 are reported in Table 5. As can be seen, for all of the twenty instances the Lagrangian relaxation algorithm outperforms CPLEX both in efficiency and solution quality. However, as shown in column Gap^2 the gap between the solution found by the algorithm and the best lower bound *LB* is large. We remark that this is probably because CPLEX fails in providing tight lower bounds.

From the above numerical experiments we observe that *UB* converges towards the optimal objective value faster than *LB* during the subgradient procedure of the Lagrangian relaxation algorithm. That's why the procedure is terminated after a small number of iterations and Gap^{LR} is large. With more iterations, better *LB* from the Lagrangian relaxation algorithm is expected to be found and thus Gap^{LR} can be improved. Another observation is that as the instance scale increases the solution quality of the Lagrangian relaxation algorithm becomes better compared with CPLEX. Besides, the Lagrangian relaxation algorithm is also relatively more efficient in obtaining near-optimal solutions for larger-scale instances than CPLEX.

5.3. Improvement from feeder scheduling

Computational experiments are also conducted to see how much benefit can be gained by adjusting the calling schedule of feeders. Two scenarios are compared: one allowing adjusting feeder schedule and another with fixed calling schedule. For the latter scenario, ten schedule templates are generated by randomly assigning feeders' service times within their allowable time windows. With determined calling schedule, the workload imbalance can be easily calculated and the minimum of total distance of container movements can be obtained by solving the resulting storage allocation problem. The mean objective value of the latter scenario serves as a benchmark and the improvement from feeder scheduling can be obtained as the gap to the benchmark. Table 6 shows the improvement in percentage for all the instances. As can be seen, there is a significant improvement if the feeder calling schedule is allowed to be adjusted.

Table 6: Improvement from Feeder Scheduling (%)

	I01	I02	I03	I04	I05	I06	I07	I08	I09	I10	Average
Set1	20.5	23.9	29.1	26.6	32.4	25.5	23.9	27.2	28.5	36.9	27.4
Set2	36.7	34.0	32.1	35.7	35.4	34.0	19.2	33.2	30.9	37.8	32.9
Set3	30.0	29.0	23.2	29.3	27.1	19.7	27.2	34.5	25.6	30.7	27.6
Set4	25.8	21.7	21.1	18.7	19.3	26.9	24.4	24.1	19.1	24.9	22.6
Set5	21.3	18.1	20.6	21.3	21.6	28.0	21.0	21.5	13.8	21.8	20.9
Set6	19.5	21.1	20.3	23.1	22.8	17.5	22.7	23.3	21.4	17.2	20.9
Average											25.4

6. CONCLUSION

In this paper we have studied the management of transshipment flows in a container terminal which consists of designing a visiting schedule template for feeder vessels and determining storage locations for transshipment containers. Unlike the previous literature, we adjust the calling schedule of feeders from the container ports' perspective so as to reduce the workload imbalance in time. This feeder scheduling problem is integrated with storage allocation problem in order to organize the transshipment flows in an optimal manner with both temporal and spatial considerations. A mixed integer programming

formulation is presented and a Lagrangian relaxation algorithm is developed. Computational experiments have shown that the Lagrangian algorithm can find near-optimal solutions with very small gaps within short computational times, and significant improvement can be gained by adjusting the calling schedule of feeders.

As a next step, berth template for feeders can be included in the developed model as it is assumed to be an input in this paper. We expect a lower spatial cost with the inclusion of the berth template.

REFERENCES

1. UNCTAD, 2008. Review of Maritime Transport. United Nations, New York and Geneva.
2. Giallombardo, G., Moccia, L., Salani, M., Vacca, I., 2010. Modeling and solving the Tactical Berth Allocation Problem. *Transportation Research Part B: Methodological* 44(2), 232-245.
3. Vacca, I., Bierlaire, M., and Salani, M., 2007. Optimization at Container Terminals: Status, Trends and Perspectives. *Technical report TRANSP-OR 071204*. Transport and Mobility Laboratory, ENAC, EPFL.
4. Steenken, D., Voss, S., Stahlbock, R., 2004. Container terminal operation and operations research - a classification and literature review. *OR Spectrum* 26(1), 3-49.
5. Stahlbock, R., Voss, S., 2008. Operations research at container terminals: a literature update. *OR Spectrum* 30(1), 1-52.
6. Ng, W.C., Mak, K.L., Li, M.K., 2010. Yard planning for vessel services with a cyclical calling pattern. *Engineering Optimization* 42(11), 1039-1054.
7. Kim, K.H., Kim, H.B., 1999. Segregating space allocation models for container inventories in port container terminals. *International Journal of Production Economics* 59(1-3), 415-423.
8. Zhang, C., Liu, J., Wan, Y., Murty, K.G., Linn, R.J., 2003. Storage space allocation in container terminals. *Transportation Research Part B: Methodological* 37(10), 883-903.
9. Chen, P., Fu, Z.H., Lim, A., Rodrigues, B., 2004. Port yard storage optimization. *IEEE Transactions on Automation Science and Engineering* 1(1), 26-37.
10. Moccia, L., Astorino, A., 2007. The group allocation problem in a transshipment container terminal, *Proceedings of World Conference on Transport Research*, University of California at Berkeley.
11. Moccia, L., Cordeau, J.F., Monaco, M.F., Sammarra, M., 2009. A column generation heuristic for a dynamic generalized assignment problem. *Computers & Operations Research* 36(9), 2670-2681.
12. Moorthy, R., Teo, C.P., 2006. Berth management in container terminal: the template design problem. *OR Spectrum* 28(4), 495-518.
13. Held, M., Karp, R.M., 1970. The traveling-salesman problem and minimum spanning trees. *Operations Research* 18(6), 1938-1962.
14. Held, M., Karp, R.M., 1971. The traveling-salesman problem and minimum spanning trees: Part II. *Mathematical Programming* 1(1), 6-25.
15. Zhang, C., Wan, Y.W., Liu, J.Y., Linn, R.J., 2002. Dynamic crane deployment in container storage yards. *Transportation Research Part B: Methodological* 36(6), 537-555.
16. Monaco, M.F., Sammarra, M., 2007. The berth allocation problem: A strong formulation solved by a Lagrangean approach. *Transportation Science* 41(2), 265-280.