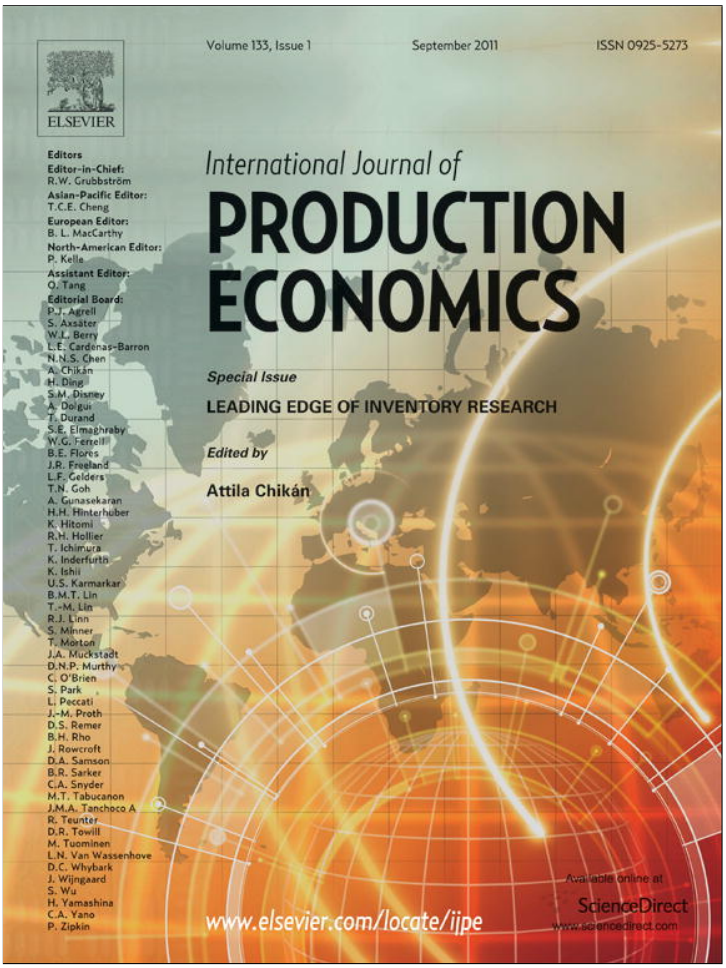

Int. J. Production Economics 133 (2011) 351–359

Contents lists available at ScienceDirect

Int. J. Production Economics

journal homepage: www.elsevier.com/locate/ijpe

Modeling and optimization of a container drayage problem with resource constraints

Ruiyou Zhang [a], Won Young Yun [b], Il Kyeong Moon [b,*]

[a] Institute of Systems Engineering, College of Information Science and Engineering, Northeastern University, Shenyang 110004, China
[b] Department of Industrial Engineering, Pusan National University, Busan 609-735, Korea

ARTICLE INFO

Article history:
Received 8 September 2008
Accepted 9 February 2010
Available online 13 February 2010

Keywords:
Container transportation
Drayage
Time windows
Traveling salesman problem (TSP)
Reactive tabu search (RTS)

ABSTRACT

This paper investigates the problem faced by firms that transport containers by truck in an environment with resource constraints. The considered area is export-dominant. As a result, there are three types of container movements as inbound full, outbound full, and inbound empty movements. Both the time windows at the terminal and at the customers' places and the operation times are considered. Empty containers are also regarded as separate transportation resources besides trucks. The total operating time including waiting time of all the trucks in operation is minimized. The problem is first formulated as a directed graph and then mathematically modeled based on the graph. It falls into a multiple traveling salesman problem with time windows (m-TSPTW) with resource constraints. An algorithm based on reactive tabu search (RTS) is developed to solve the problem. A number of randomly generated examples indicate that the algorithm can be applied to the real world.

1. Introduction

The transportation of containers usually involves a variety of transportation modes such as truck, train, vessel, and so on. Among all the transportation modes, the transportation by truck is a necessary mode since the other modes cannot provide a door-to-door service. The short-haulage container transportation by truck between a terminal and a shipper/receiver is usually called drayage operation (Sinclair and Van Dyk, 1987; Macharis and Bontekoning, 2004). Although the transport distance of drayage operation is relatively short, the transportation cost per TEU (20 ft equivalent unit) per kilometer is relatively high. Furthermore, this part of transportation is usually the sources of road congestion and shipment delay (Cheung et al., 2008). However, despite the importance of drayage operation in container transportation, research on the truck scheduling in container drayage operation has been scant (Vis and de Koster, 2003).

Wang and Regan (2002) formulated a container drayage problem as a multiple traveling salesman problem with time windows (m-TSPTW) and developed a solution method based on window partition. Jula et al. (2005) modeled a container movement problem by truck as an m-TSPTW with social constraints and developed three solution methods to solve it. Coslovich et al. (2006) built and solved an integer programming model that was based on Lagrangian relaxation from the perspective of the container transportation company. Chung et al. (2007) studied several transportation models about the drayage operations of containers, one with time windows, and one without time windows, for single or multi-commodity and for different vehicle types. Imai et al. (2007) addressed a problem of vehicle routing that arises in picking up and delivering full container load from/to a container terminal. Namboothiri and Erera (2008) considered the container truck transportation problem arising from access appointment systems for truck that some intermodal ports have developed.

Driver and truck are two types of necessary resources in all vehicle routing problems. As a result, they are seldom mentioned as transportation resources. In container drayage operation, there are two types of additional resources as chassis and container. On one hand, a chassis and a tractor can be separated or connected. On the other hand, a chassis and a container can be put at a customer's place when the tractor departs from there. In addition, an empty container may be transported between different areas. The container drayage problem becomes extremely complicated if driver, tractor, chassis, and container are all regarded as separated resources.

There is very little literature addressing vehicle routing problems with resource constraints. Assuming that each driver is assigned to a single vehicle route, Desaulniers et al. (1999) studied three driver-scheduling aspects. Fischer et al. (1999) applied multi-agent system (MAS) in the scheduling and planning of the transportation resources. Smilowitz (2006) introduced an application of a multi-resource routing problem in drayage operations. However, tractors and trailers (but not containers),

* Corresponding author. Tel.: +82 51 510 2451; fax: +82 51 512 7603.
E-mail address: ikmoon@pusan.ac.kr (I.K. Moon).

0925-5273/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.ijpe.2010.02.005

were considered. Cheung and Hang (2003) and Cheung et al. (2008) applied a multi-attribute model to research the land transportation of air-cargo freight and a cross-border container drayage problem, respectively.

In this paper, we study a container drayage operation problem with empty container constraints. An export-dominant area such as China or Korea is considered. Therefore, there are three types of container movements as inbound full, outbound full, and inbound empty movements. Each full container (or a container of freight) has both an origin time window and a destination time window. For a sub-fleet of trucks, the truck dispatcher is the decision maker. His objective is to minimize the total operating time, including traveling and waiting time, of all the trucks in operation. Empty containers are considered as a type of separate resources besides trucks. The problem is first formulated as a directed graph. Then, the problem falls into a type of m-TSPTW with resource constraints. The container drayage problem is mathematically modeled and is solved by means of the reactive tabu search (RTS). A lower bound that is obtained from the commercial tool CPLEX is used to test the performance of the RTS algorithm. A number of randomly generated examples indicate that the developed RTS algorithm can be applied to real world situations.

The remaining part of this paper is organized as follows. A container drayage problem is given in Section 2. The problem is first formulated as a graph model in Section 3, and then mathematically modeled based on the graph in Section 4. A RTS based algorithm is developed to solve the problem in Section 5, and tested by using a number of randomly generated examples in Section 6. Finally, Section 7 concludes this paper.

2. Container drayage problem

2.1. Problem setting

A transportation company owns a number of containers and trucks. The containers are used to contain freight to be transported. The trucks, with the containers, are used to transport freight for customers. All the trucks are divided into several sub-fleets, each of which is handled by a single load manager (dispatcher). The load manager of a sub-fleet is the decision maker of the considered problem. In addition, the company owns several depots in a local area. The depots can be used to store containers and park trucks which belong to the company.

A sub-fleet has several container transportation orders between customers and terminals in the local area. In this paper, a *terminal* means a sea-side port, a train hub station, an air port, or an off-dock container yard. At the terminal, the mode of transportation changes to "by truck" or "not by truck", as the case may be. In this paper, a *customer* means the original shipper's location, where freight is packed inside containers and transported to a given terminal, or the final receiver's location, where containers are delivered and unpacked and freight is removed. Both the customers and the terminals operate according to certain time windows, during which the corresponding activities of the transportation orders must be started.

If a container from one terminal is delivered to another terminal and needs to be removed by truck, it is called an *inbound* container, even after it is has been loaded onto the truck. If a container of freight is to be transported to a terminal by truck and then to other terminals, it is called an *outbound* container even before it is packed. We can consider an area such as China or South Korea to be export-dominant in containerized trade. That is, in a given time period, the number of containers exported is much larger than the number imported. Because of this disparity, such an area will, in irregular cycles, experience a shortage of empty containers. The inverse of this can be seen in an import-dominant area such as the US where there will at times be a surplus of empty containers. A truck transportation company in an export-dominant area, therefore, often needs to import a number of empty containers. Such a company thus has three distinct types of orders (movements, or containers): inbound full containers, outbound full containers, and inbound empty containers.

Fig. 1 shows the representative journey of one truck in a small system, which in this example, consists of one terminal, one depot, and two customers. The cycle of this truck involves an inbound full container of Customer 1, an outbound full container for/from Customer 2, and an inbound empty container. The truck is used to transport these containers that fulfill the three different types of orders. The number in the bracket beside the lines in Fig. 1 indicates the sequence of the entire journey. First, the truck leaves the depot without a container. The inbound full container is picked up after the truck arrives at the terminal, is delivered to Customer 1, and is unpacked. The recently emptied container is delivered to the depot. The inbound empty container is picked up after the truck arrives at the terminal for the second time, and is delivered to Customer 2. The freight is packed within the container and delivered to the terminal. Finally, the truck returns to the depot.

It can be observed that a truck can return to the depot to pick up or deliver empty containers whenever necessary. The truck does not, however, need to return to the depot after the transportation of each container. After the transportation of one container, the truck transfers another container. The transfer between different containers may involve different activities. Furthermore, if, and only if a truck can begin its activities within its assigned time window, it can go to the customer (or the terminal).

The aim of our investigation is to determine which containers should be transported by a single truck and in which order. The typical planning horizon is one day. The objective is to minimize the total operating time of all the trucks in operation. The total operating time consists of the combined waiting and traveling time of all trucks. The trucks' journeys can involve the transportation of full containers, as represented by a solid line in Fig. 1, or the transportation of empty containers and the empty travel without containers (Zhang and Yun, 2008), as represented by a dashed line in Fig. 1.

2.2. Assumptions

The following assumptions are introduced to formulate the problem:

(i) There is only one terminal in the system.
(ii) There is only one depot in the system. The depot can be visited at any time. The depot has enough room for empty

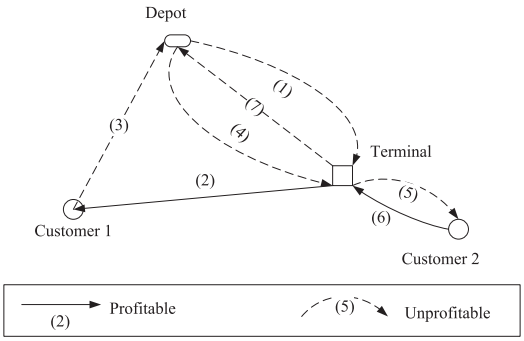

Fig. 1. Representative journey of a truck in a small system.

containers to be stored. The initial number of empty containers at the depot is given.

(iii) All of the containers are of the same type, which are 40 ft dry containers.

(iv) All the trucks are identical. Each truck can carry only one container at any time. All the trucks are initially located at the depot and should be finally returned to the depot. The travel time is the same for loaded trips and empty trips, is determinate, and depends only on the traveling distances.

(v) The time for pick-up or drop-off a container is the same. However, the packing time or unpacking time differs from container to container.

(vi) Each inbound or outbound full container has an origin time window and a destination time window. Each inbound empty container has only an origin time window. Inbound empty containers have no determinate destinations, and hence no destination time windows. All the time windows are hard, and hence cannot be violated.

(vii) All the containers should be delivered directly to their destinations without being stored at the depot.

(viii) During the time period of packing and unpacking, the truck must stay at the customer's location.

(ix) The freight of each customer that is to be exported can be held in integer times of one container. Each inbound full container has a single customer (receiver).

2.3. Parameters

The following given parameters are introduced to describe the problem:

n_I	number of inbound full containers
n_O	number of outbound full containers
n_E	number of inbound empty containers
n	number of trucks
m	initial number of empty containers at the depot. Where, $m+n_I+n_E \geq n_O$, which is necessary. Otherwise, the problem has no feasible solution
$[\tau_{OLi}, \tau_{OUi}]$	origin time window of container i $(1 \leq i \leq n_I+n_O+n_E)$, where $\tau_{OLi} \leq \tau_{OUi}$. The picking up of the inbound (full or empty) container and the packing of the outbound full container should be started during the corresponding origin time windows. τ_{OLi} and τ_{OUi} are the lower bound and upper bound of the window, respectively
$[\tau_{DLi}, \tau_{DUi}]$	destination time window of the inbound or outbound full container i $(1 \leq i \leq n_I+n_O)$, where $\tau_{DLi} \leq \tau_{DUi}$. The unpacking of the inbound full container and the dropping off of the outbound full container should be started during the corresponding destination time window
t	amount of time for pick-up or drop-off a container
t_i	packing/unpacking time of inbound/outbound full container i $(1 \leq i \leq n_I+n_O)$, where $t_i \geq 0$
t_{ij}	travel time between locations l_i and l_j. Where, l_0 means the location of the depot, l_i $(1 \leq i \leq n_I)$ means the receiver's location of the inbound full container i, l_i $(n_I+1 \leq i \leq n_I+n_O)$ means the shipper's location of the outbound full container i, and $l_{n_I+n_O+1}$ means the location of the terminal $t_{ij} \geq 0,\ t_{ij} = t_{ji}$

3. Graphical formulation of the problem

The considered container drayage problem is complicated as: a container to be transported involves multiple activities; and the transfer between different types of containers involves different activities. A graphical formulation of the problem is introduced in this section so that we can easily construct a mathematical model and solve the problem.

3.1. Graph definition

A graph $G=(N,A)$ is introduced to formulate the problem, where, $N=\{0\} \cup C$ and $A=\{(i,j)|i \in N, j \in N, i \neq j\}$ are the node set and arc set, respectively.

Definition 1. A start/return node is defined as the initial start from and final return to the depot.

Node 0 is the single start/return node in the graph. It involves no actual activities. It is introduced only for the completeness of the graph.

Definition 2. A container node is defined as the continuous determinate activities involved in the transportation of the corresponding container.

A node $i \in C$ is a container node. Specifically, we introduce three subsets of C to describe the three types of containers to be transported. Inbound full container node $i \in C_I$ involves the activities of pick-up the container i $(1 \leq i \leq n_I)$ at the terminal, delivering it to its receiver, dropping off and unpacking it. Outbound full container node $i \in C_O$ involves the activities of packing the container i $(n_I+1 \leq i \leq n_I+n_O)$ at its shipper, picking it up, delivering it to the terminal, and dropping it off. Inbound empty container node $i \in C_E$ involves the activity of pick-up the container i $(n_I+n_O+1 \leq i \leq n_I+n_O+n_E)$ at the terminal.

Definition 3. An arc $(i,j) \in A$ is defined as the transfer of a truck from node i to node j.

A transfer from node i to node j means that a truck will transport container j after the transportation of container i. Here, container nodes (not the start/return node) are taken as an example. The activities on arcs are to be determined.

Definition 4. A graph is named as a determinate-activities-on-node (DAON) graph if all the determinate activities are denoted by nodes and the indeterminate activities are denoted by arcs.

The graph G proposed here is a DAON graph.

3.2. Attributes of the start/return node

Attribute 1. The truck-number attribute of the start/return node is the number of trucks, n.

Attribute 2. The container-number attribute of the start/return node is the initial number of empty containers at the depot, m.

3.3. Attributes of container nodes

We propose a combination template of time windows, based on which the attributes of container nodes can be calculated.

It is assumed that two activities should be executed serially. In other words, Activity B can be started only after Activity A is finished. The amounts of time consumed by the Activities A and B are a and b, respectively. The two activities should be started during the time windows $[L_A, U_A](L_A \leq U_A)$ and $[L_B, U_B](L_B \leq U_B)$, respectively.

The time windows are reasonable and hence both of the two activities can be executed if and only if the following condition

is satisfied:

$$L_A + a \le U_B \tag{1}$$

If Activity A is started before the time $L_B - a$, Activity A will be finished before the time L_B. Hence, Activity B cannot be started immediately after Activity A is finished. Therefore, it is unnecessary for Activity A to be started before the time $L_B - a$. Furthermore, Activity A must be started before the time $U_B - a$. Otherwise, Activity A will be finished after the time U_B. Considering the own time window $[L_A, U_A]$ of Activity A, we can obtain the following time window $[L,U]$ during which the combined activity should be started.

$$L = \min(\max(L_A, L_B - a), U_A) \tag{2}$$

$$U = \min(U_A, U_B - a) \tag{3}$$

According to the above analysis, the amount of time consumed before Activity B is started can be formulated as $\max(L_B - U_A, a)$. As a result, we can get the total consumed time, including the unavoidable waiting time, of the combined activity as follows.

$$T = \max(L_B - U_A, a) + b \tag{4}$$

Attribute 1. The service-time attribute of a container node $i \in C$ is the amount of time consumed by the activities. It is denoted by T_i.

Attribute 2. The time-window attribute of a container node $i \in C$ is the time period during which the activities should be started. It is denoted by $[T_{Li}, T_{Ui}](T_{Li} \le T_{Ui})$.

The two attributes of a container node $i \in C_I \cup C_O$ can be calculated based on the combination template. For instance, for an inbound full container node $i \in C_I$, Activity A means picking up container i and delivering it to its receiver; and Activity B means dropping off and unpacking the container. Therefore, $a = t + t_{n_I+n_O+1,i}$, $b = t + t_i$, $[L_A, U_A] = [\tau_{OLi}, \tau_{OUi}]$, $[L_B, U_B] = [\tau_{DLi}, \tau_{DUi}]$. Only one activity is associated with each inbound empty container node $i \in C_E$. Therefore, the two attributes of node $i \in C_E$ can be directly obtained from the given parameter.

The service-time attribute of node $i \in C$ is

$$T_i = \begin{cases} \max(\tau_{DLi} - \tau_{OUi}, t + t_{n_I+n_O+1,i}) + t + t_i, & i \in C_I \\ \max(\tau_{DLi} - \tau_{OUi}, t_i + t + t_{i,n_I+n_O+1}) + t, & i \in C_O \\ t, & i \in C_E \end{cases} \tag{5}$$

The time-window attribute of node $i \in C$ is

$$T_{Li} = \begin{cases} \min(\max(\tau_{OLi}, \tau_{DLi} - t - t_{n_I+n_O+1,i}), \tau_{OUi}), & i \in C_I \\ \min(\max(\tau_{OLi}, \tau_{DLi} - t_i - t - t_{i,n_I+n_O+1}), \tau_{OUi}), & i \in C_O \\ \tau_{OLi}, & i \in C_E \end{cases} \tag{6}$$

$$T_{Ui} = \begin{cases} \min(\tau_{OUi}, \tau_{DUi} - t - t_{n_I+n_O+1,i}), & i \in C_I \\ \min(\tau_{OUi}, \tau_{DUi} - t_i - t - t_{i,n_I+n_O+1}), & i \in C_O \\ \tau_{OUi}, & i \in C_E \end{cases} \tag{7}$$

3.4. Attributes of arcs

Attribute 1. The transfer-time attribute of an arc $(i,j) \in A$ is the amount of time consumed by the corresponding activities. It is denoted by T_{ij}.

The activities of arcs are presented in Table 1. The transfer-time attribute can be calculated based on the activities and are shown at the end of each cell in Table 1. For example, if $i=0$, $j \in C_I$ or $j \in C_E$, the truck should start from the depot to transport an inbound container. Therefore, the activity is traveling to the terminal; and hence $T_{ij} = t_{0,n_I+n_O+1}$. However, if $i=0$ and $j \in C_O$, an empty container should be picked up at the depot, delivered to the shipper and dropped off; and hence $T_{ij} = t_{0j} + 2t$.

Attribute 2. The change-number attribute of an arc $(i,j) \in A$ is the change of the number of empty containers at the depot that is caused by the arc (i,j). It is denoted by α_{ij}.

Attribute 3. The change-time attribute of an arc $(i,j) \in A$ is the elapsed time when the number of empty containers at the depot is changed by the arc after the arc is started. It is denoted by β_{ij}.

Change-number and change-time are two important attributes of arcs in the graph. They can be calculated based on the activities of arcs presented in Table 1 and formulated as follows:

$$\alpha_{ij} = \begin{cases} -1 & \text{if } i = 0,\ j \in C_O \\ 1 & \text{if } i \in C_I,\ j \in \{0\} \cup C_I \cup C_E \\ -1 & \text{if } i \in C_O,\ j \in C_O \\ 1 & \text{if } i \in C_E,\ j \in \{0\} \cup C_I \cup C_E \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

Table 1
Activities and transfer-time attribute of arcs.

From node	Activities and transfer-time attribute of the arc (i,j)			
	To node $j=0$	To node $j \in C_I$	To node $j \in C_O$	To node $j \in C_E$
$i=0$	–	Traveling to the terminal $T_{ij} = t_{0,n_I+n_O+1}$	Picking up an empty container, delivering it to the shipper of container j, and dropping it off $T_{ij} = t_{0j} + 2t$	Traveling to the terminal $T_{ij} = t_{0,n_I+n_O+1}$
$i \in C_I$	Picking up the recently emptied container, delivering it to the depot, and dropping it off $T_{ij} = t_{i0} + 2t$	Picking up the recently emptied container, delivering it to the depot, dropping it off, and traveling to the terminal $T_{ij} = t_{i0} + t_{0,n_I+n_O+1} + 2t$	Picking up the recently emptied container, delivering it to the shipper of container j, and dropping it off if the receiver of container i is not the shipper of container j; no activity, otherwise $T_{ij} = \begin{cases} t_{ij} + 2t & \text{if } t_{ij} \neq 0 \\ 0 & \text{if } t_{ij} = 0 \end{cases}$	Picking up the recently emptied container, delivering it to the depot, dropping it off, and traveling to the terminal $T_{ij} = t_{i0} + t_{0,n_I+n_O+1} + 2t$
$i \in C_O$	Traveling to the depot $T_{ij} = t_{n_I+n_O+1,0}$	No activity $T_{ij} = 0$	Traveling to the depot, picking up an empty container, delivering it to the shipper of container j, and dropping it off $T_{ij} = t_{n_I+n_O+1,0} + t_{0j} + 2t$	No activity $T_{ij} = 0$
$i \in C_E$	Delivering container i to the depot, and dropping it off $T_{ij} = t_{n_I+n_O+1,0} + t$	Delivering container i to the depot, dropping it off, and traveling to the terminal $T_{ij} = 2t_{n_I+n_O+1,0} + t$	Delivering container i to the shipper of the container j, and dropping it off $T_{ij} = t_{n_I+n_O+1,j} + t$	Delivering container i to the depot, dropping it off, and traveling to the terminal $T_{ij} = 2t_{n_I+n_O+1,0} + t$

$$\beta_{ij}=\begin{cases}0 & \text{if } i=0,\ j\in C_O\\ t_{i0}+2t & \text{if } i\in C_I,\ j\in\{0\}\cup C_I\cup C_E\\ t_{n_I+n_O+1,0} & \text{if } i\in C_O,\ j\in C_O\\ t_{n_I+n_O+1,0}+t & \text{if } i\in C_E,\ j\in\{0\}\cup C_I\cup C_E\\ 0\ \text{(meaningless)} & \text{otherwise}\end{cases} \tag{9}$$

In summary, the problem is formulated as a DAON graph. The truck-number and container-number of the single start/return node is n and m, respectively. Each container node $i\in C$ has a service-time T_i and a time-window $[T_{Li}, T_{Ui}]$. The transfer-time of arc $(i,j)\in A$ is T_{ij}. The number of empty containers at the depot will be changed by the value α_{ij} at time β_{ij} after the arc is started. The problem is to find a number of circuits. Each container node should be visited exactly by one circuit. The constraints are the time window of container nodes, the number of trucks and the accumulative number of empty containers at the depot at any time.

4. Mathematical model

The following decision variables are introduced:

$$x_{ij}=\begin{cases}1 & \text{if arc } (i,j)\in A \text{ is included in the solution}\\ 0 & \text{otherwise}\end{cases}$$

y_i time when node $i\in C$ is started
z_{ij} time when arc $(i,j)\in A$ is started

The following mathematical model is able to formulate the problem by means of the proposed DAON graph.

4.1. Objective function

If a node $j\in C$ is the first node that is visited by a truck, this truck initially starts from the depot at time z_{0j}. Similarly, if a node $i\in C$ is the last node that is visited by this truck, this truck finally returns to the depot at time $z_{i0}+T_{i0}x_{i0}$. Therefore, the amount of operating time of this truck is $z_{i0}+T_{i0}x_{i0}-z_{0j}$. Furthermore, if a node $j\in C$ is not the first node visited by a truck, $z_{0j}=0$ (see constraint (17)). Similarly, if a node $i\in C$ is not the last node visited by a truck, $z_{i0}=0$ (see constraint (18)) and hence $z_{i0}+T_{i0}x_{i0}=0$. Consequently, $\sum_{i\in C}(z_{i0}+T_{i0}x_{i0})-\sum_{j\in C}z_{0j}$ denotes the total operating time of all the trucks in operation. The following objective is obtained.

$$\min\sum_{i\in C}(z_{i0}+T_{i0}x_{i0})-\sum_{j\in C}z_{0j} \tag{10}$$

4.2. Basic constraints

The following basic constraints that can be found in most VRPs should be satisfied:

$$\sum_{j\in N}x_{ji}=\sum_{j\in N}x_{ij}=1,\quad \forall i\in C \tag{11}$$

$$\sum_{j\in N}x_{0j}\le n \tag{12}$$

$$\sum_{i\in S}\sum_{j\in S}x_{ij}\le |S|-1,\quad \forall S\subseteq C, S\neq\Phi \tag{13}$$

Constraint (11) states that any container node should be visited (entered and left) exactly once. Constraint (12) means that the number of trucks which leave the start/return node should not exceed the number of trucks available. It should be noticed that it is unnecessary to introduce the constraint $\sum_{i\in N}x_{i0}\le n$ since $\sum_{j\in N}x_{0j}=\sum_{i\in N}x_{i0}$ can be satisfied automatically. Constraint (13) removes sub-tours among container nodes. Where, S is a non-empty sub set of container node set, C, and Φ means an empty set.

4.3. Time window constraints

The activities of a container node $i\in C$ should be started during the corresponding time window. Specifically

$$T_{Li}\le y_i\le T_{Ui},\quad \forall i\in C \tag{14}$$

If an arc (i,j) is included in the solution, this arc should be started after the node i is finished; and node j should be started after this arc is finished. That is

$$y_i+T_i-z_{ij}\le(1-x_{ij})M,\quad \forall i\in C,\ j\in N \tag{15}$$

$$z_{ij}+T_{ij}-y_j\le(1-x_{ij})M,\quad \forall i\in N,\ j\in C \tag{16}$$

where M is a big enough constant.

Furthermore, the following constraint is introduced:

$$z_{0j}\le x_{0j}M,\quad \forall j\in C \tag{17}$$

If $x_{0j}=0$, constraint (17) becomes $z_{0j}\le 0$. The negative coefficient in the objective function forces z_{0j} to be zero. Similarly, the following constraint is introduced to impose z_{i0} to be zero if $x_{i0}=0$:

$$-z_{i0}\le x_{i0}M,\quad \forall i\in C \tag{18}$$

4.4. Empty container constraint

As it is defined, the activities of an arc (i,j) affect the number of empty containers at the depot at time $z_{ij}+\beta_{ij}$. Therefore, the set of arcs which affect the number of empty containers before this arc is $\{(p,q)|z_{pq}+\beta_{pq}\le z_{ij}+\beta_{ij}\}$. The accumulative total number of empty containers at the depot should be larger than or equal to zero at any time. Specifically

$$\sum_{(p,q)\in\{(p,q)|z_{pq}+\beta_{pq}\le z_{ij}+\beta_{ij}\}}\alpha_{pq}+m\ge 0,\quad \forall i\in N,\ j\in N \tag{19}$$

4.5. Variable type constraints

$$x_{ij}\in\{0,1\},\quad \forall i\in N,\ j\in N \tag{20}$$

$$y_i:\ \text{real variable},\quad \forall i\in C \tag{21}$$

$$z_{ij}:\ \text{real variable},\quad \forall i\in N, j\in N \tag{22}$$

5. Solution method to solve the model

If the empty container constraint (19) and all the related constraints are relaxed, the problem falls into the traditional m-TSPTW. The mathematical model is NP-hard since the m-TSPTW is NP-hard.

5.1. Reactive tabu search algorithm

Tabu search (TS) (Glover, 1989) is a famous meta-heuristic algorithm that has been successfully applied in various optimization fields. Intensification and diversification are two important mechanisms in TS and a number of other similar meta-heuristic algorithms. Reactive search was introduced into TS in order to automatically balance the two mechanisms as intensification and diversification (Battiti and Tecchiolli, 1994). This type of improved

version of TS is called reactive tabu search (RTS) (Blochliger and Zufferey, 2008; Braysy and Gendreau, 2005).

5.1.1. Basic settings of RTS

A RTS algorithm is developed to solve the model in this section. A solution of the algorithm is denoted by $n_I+n_O+n_E$ natural numbers and n zeros. Each natural number means a container node. A segment of solution separated by two zeros means the container nodes that are visited by one truck in their orders. If two randomly selected elements of the current solution are exchanged, a neighborhood solution is obtained. If a neighborhood solution is better than the best solution found in history, it is accepted regardless of its tabu status. Otherwise, we continue to generate neighborhood solutions until a certain number of feasible solutions are generated. The best none-tabu neighborhood solution is chosen as the current solution.

A pair of two exchanged numbers forms a cell of the tabu list. The length of the tabu list is modified automatically according to the iteration procedure. If a certain number of solutions are repeatedly found for a certain times, an escape mechanism is triggered; see Wang et al. (2007) for detailed description of the RTS algorithm. All the accepted solutions are stored in a binary tree. If the maximum number of iterations is reached, the solution ends.

5.1.2. Initial solution

First of all, the graph model is compacted. An arc $(i,j)\in A$ and a node $j\in N$ are combined using the combination template described in Section 3.3.

Actually, if $i\in C$, $j\in C$, the time and time window of an arc $(i,j)\in A$ can be considered as T_{ij}, and $[T_{Li}+T_i, T_{Ui}+T_i]$, respectively. Of course, the time and time window of a node $j\in C$ are T_j, and $[T_{Lj}, T_{Uj}]$, respectively. Therefore, arc $(i,j)\in A$ and node $j\in$C can be combined according to the combination template. If i=0, or j=0, there is no time window imposed to the arc or the node, then it is relatively easy to combine the arc and the node. The original service-time and time-window attributes of nodes, as well as the transfer-time attribute of arcs are discarded after the combination. The service time and time window of the combined arc $(i,j)\in A$ are denoted by T'_{ij} and $[T_{Lij}, T_{Uij}]$, respectively.

An initial solution can be obtained based on the compacted graph as follows:

Step 1: Find a container node according to a number of criteria. A new route that will be visited by one truck is constructed.

Step 2: If any un-routed container node can be appended to the current route, find one among them according to several other criteria until no un-routed container node can be appended to the current route.

Step 3: If there exist several un-routed container nodes, go to Step 1 to construct a new route; otherwise, end.

There are two levels of criteria that are used to construct a new route in Step 1. The first level can be formulated as follows:

$$P\{j|\alpha_{0j}=0\}>P\{j|\alpha_{0j}\neq 0\} \tag{23}$$

where $P\{\cdot\}$ denotes the priority of nodes. Criterion (23) means that the first container node of a route is preferred to be an inbound (full or empty) container node rather than an outbound container node. The second level of the criteria is the lower bound of the time when the arc is finished, i.e., $T_{L0j}+T'_{0j}$. If an arc from the start/return node can be finished earlier, it has a higher priority.

If a container node is appended to a route, it is combined with the route according to the combination template. Therefore, it can be easily checked whether a node can be appended to a route. There are also two similar criteria that are used in Step 2. The first level is shown in

$$P\{j|\alpha_{ij}=0\}>P\{j|\alpha_{ij}\neq 0\} \tag{24}$$

where i in (24) denotes the last node of the current route since the tail node of the arc in Step 2 is not the start/return node. The second level of the criteria is the upper bound of the time when the arc is finished. If an arc from the last node (node i) can be finished earlier, it has a higher priority.

5.1.3. Feasibility of solutions

Because the solutions are initialized route by route, it is convenient to check whether the trucks are enough. When a neighborhood solution is generated, it is unnecessary to check the truck number constraint since it is satisfied automatically.

The time window constraint should be checked when a neighborhood solution is obtained. For a segment of the solution separated by two zeros, the arcs are tried to be combined one by one. If the segment, which is actually a route, can be successfully combined, it is feasible from the viewpoint of time window. If the empty container constraint of a given solution is checked exactly according to constraint (19), it is very difficult. To exactly check this constraint means to solve a difficult problem since the start times of arcs, z_{ij}, are not given in the solution. Therefore, the following checking method is proposed. For a given route that is followed by a truck, the accumulative maximum number of empty containers needed is checked. For example, if the change-numbers of arcs of the route are supposed as follows:

$$0,-1,0,1,-1,-1,1$$

The accumulative numbers after each arc is finished can be obtained as

$$0,-1,-1,0,-1,-2,-1$$

Therefore, the accumulative maximum number is -2. If the summation of this accumulative maximum numbers of all routes plus m (the given number of empty containers) is larger than or equal to zero, the solution is feasible.

Such type of checking method of the empty container constraint is a little stronger than the original definition. However, it is very convenient and practical.

5.2. Lower bound

The mathematical model is not the same as the typical m-TSPTW even if constraint (19) is relaxed. The main difference lies in the objective function and the related constraints (15)–(18). The relaxed model is also NP-hard. However, it is a linear mixed integer programming. It can be solved by using some commercial tools such as CPLEX if its size is relatively small. The result of the relaxed model is a lower bound of the original model. It can be used to prove the performance of the RTS algorithm. If constraint (19) is relaxed, most elements of the decision variables $z_{ij}(i\in C, j\in C)$ are removed with only $z_{0j}(j\in C)$ and $z_{i0}(i\in C)$ remained.

Constraints (15) and (16) can be replaced by constraints (25)–(27) as follows:

$$y_i+T_i+T_{ij}-y_j\leq(1-x_{ij})M,\quad \forall i\in C,\ j\in C \tag{25}$$

$$z_{0j}+T_{0j}-y_j\leq(1-x_{0j})M,\quad \forall j\in C \tag{26}$$

$$y_i+T_i-z_{i0}\leq(1-x_{i0})M,\quad \forall i\in C \tag{27}$$

Constraints (21) and (22) can be rewritten as constraints shown below

$$y_i, z_{i0}, z_{0j}:\ \text{real variable},\quad \forall i\in C,\ j\in C \tag{28}$$

Table 2
Information and results of the first seven examples.

Example	No. of containers	No. of trucks	Initial no. of empty containers	Lower bound from CPLEX		Result from RTS algorithm		Gap (%)
				Lower bound	CPU time (s)	OBJ value	CPU time (s)	
1	5 (2, 2, 1)	4	2	771	2.15	771	0.34	0
2	10 (4, 4, 2)	6	3	1524	3.25	1529	9.25	0.3
3	15 (8, 7, 0)	12	6	3225	83.29	3225	622.27	0
4	16 (8, 8, 0)	12	7	2912	636.57	2912	546.77	0
5	17 (8, 8, 1)	14	7	4120	1521.90	4120	725.02	0
6	18 (8, 8, 2)	15	7	3718	1217.14	3718	695.88	0
7	19 (8, 8, 3)	16	7	NA	NA	4926	589.34	NA

6. Numerical experiments

6.1. General setting of experiments

A number of examples are generated randomly by using a paragraph of Matlab function. All the locations as the terminal, the depot, the shippers of outbound full containers, and the receivers of inbound full containers are randomly generated on the Euclidean plane with a length and width equal to a 3-h's distance by truck. The time for pick-up or drop-off a container is 5 min (Chung et al., 2007). The packing/unpacking time of full containers is uniformly distributed in the range from 5 to 60 min. The lengths of the origin and destination time windows are distributed in the range of 0–4 h, and the range of 0–5 h, respectively. In addition, the traveling time between locations that can be calculated based on the coordinates is considered when we generate the lower bound of the time windows.

The graph model and the RTS algorithm are coded in Matlab R2006b by Math Works Inc. All the examples are solved by using the RTS algorithm. Meanwhile, the relaxed form of those examples with small size is solved by using the commercial tool CPLEX 10.0 in ILOG OPL Development Studio 4.2. All the experiments are carried out on a personal computer with Intel® Pentium CPU 3.40, 3.39 GHz, and 0.99 GB memory.

The parameters of the RTS algorithm used in the experiments are set as follows. The initial length of the tabu list is 5. If a repeated solution is found during the iteration, the length of the tabu list is multiplied by 1.01. If no repeated solution is found during 20 iterations, the length of the tabu list is multiplied by 0.9. If several repeated solutions are found for 30 times, the escape mechanism is triggered. The maximum size of the neighborhood is 5 for Example 1, 20 for Example 2, and 100 for other examples. The maximum number of iterations is 20 for Example 1, 100 for Example 2, and 1000 for other examples.

6.2. Experiments on small-sized examples

First of all, seven examples with the number of containers increased from 5 to 19 are tested. Table 2 presents the information of the examples and the calculation results. The three numbers in the brackets in the second column of Table 2 indicate the numbers of inbound full, outbound full, and inbound empty containers, respectively. The lower bound from CPLEX and the objective value obtained from the RTS algorithm are in minutes. The gap is the difference between the lower bound and the objective value divided by the lower bound.

The lower bound of Example 7 cannot be obtained because this requires too much computation time. It seems that the CPU time that is consumed to obtain the lower bound from CPLEX is too long if the number of containers is larger than 18. Therefore, 10 more examples each of which has the similar information as Example 6 are generated and solved. The results are presented in Table 3.

Table 3
Results of 10 examples each of which has 18 containers.

Example	Lower bound from CPLEX		Result from RTS algorithm		Gap (%)
	Lower bound	CPU time (s)	OBJ value	CPU time (s)	
8	4037	1442.23	4037	684.70	0
9	3964	781.07	3965	746.38	0.03
10	5024	526.09	5077	1038.00	1.1
11	4488	2450.34	4488	765.36	0
12	3809	3593.04	3809	559.19	0
13	3230	5008.01	3375	422.97	4.5
14	NA	$>25\,200$	2373	453.53	NA
15	4775	1291.90	4775	821.20	0
16	4767	10 943.46	4767	675.33	0
17	4190	1482.03	4190	814.83	0

It can be seen that even if the number of containers is fixed at 18, the CPU time consumed by CPLEX is very long. For example, it takes more than 3 h to obtain the lower bound of Example 16. Furthermore, the lower bound of Example 14 cannot be obtained from CPLEX within 7 h. Both Tables 2 and 3 indicate that all the presented examples can be solved by using the proposed RTS algorithm in about 10 min. The optimum solutions of most of the examples have been obtained. The biggest gap between the objective value and the lower bound is only 4.5%. It is completely acceptable in industrial application.

6.3. Experiments on realistic-sized examples

According to Wang and Regan (2002), the maximum number of containers that a sub-fleet can handle in one day is about 75. Based on Wang and Regan's findings, four examples which involve about 75 containers were generated and tested in this subsection.

Example 18 has 70 containers; and each of the other three examples has 75 containers. All the four examples are independently solved using the RTS algorithm for three times. The objective values and the computation times are shown in Figs. 2 and 3, respectively. The lower bound of these four examples cannot be obtained from CPLEX because this requires too much computation time and too much memory.

Fig. 2 indicates that the performance of the RTS algorithm is stable. For each of the four examples, the objective value obtained from each repeat is quite close to that of the other repeats. Example 21 has the largest difference in objective values between repeats. However, this difference is only 1.58% (between the second and third repeats).

It can be observed from Fig. 3 that the computation time of the RTS algorithm is congenial to actual application. The addressed container drayage problem needs to be solved off-line. For example, the RTS algorithm can be executed during the night

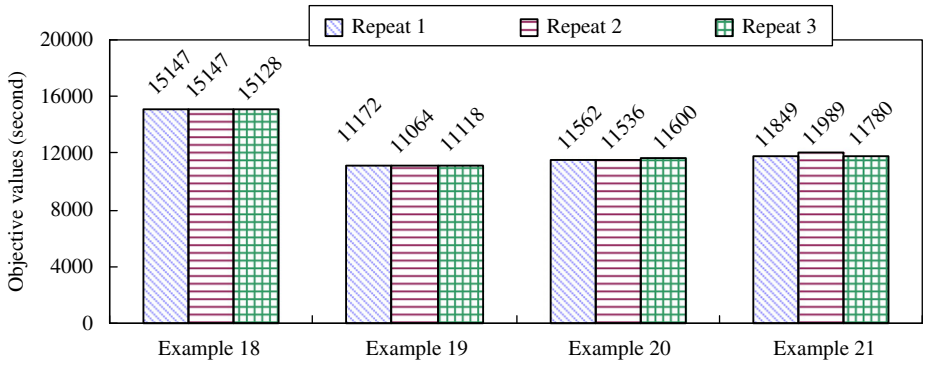

Fig. 2. Objective values of realistic-sized examples.

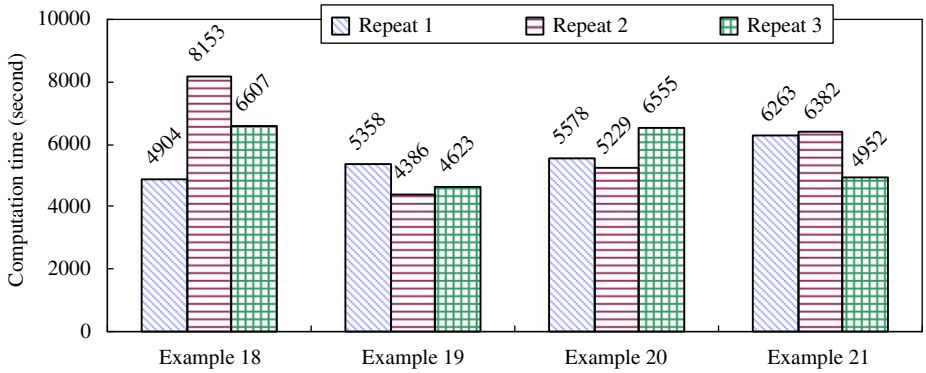

Fig. 3. Computation times of realistic-sized examples.

$\mathbf{0},{}^{32},{}_{66},{}^{57},{}_{67},{}^{64},\mathbf{0},{}^{50},{}_{21},{}^{61},{}_{6},{}^{41},\mathbf{0},{}^{39},{}_{69},\mathbf{0},{}_{29},{}^{45},{}_{27},\mathbf{0},{}_{70},{}^{58},{}_{15},{}^{48},{}_{4},\mathbf{0},{}_{1},{}^{33},{}_{68},\mathbf{0},{}_{72},$
$^{36},{}_{71},\mathbf{0},{}_{26},{}^{59},{}_{25},\mathbf{0},{}_{28},{}^{49},{}_{14},\mathbf{0},{}_{24},{}^{31},{}_{74},\mathbf{0},{}^{60},{}_{2},{}^{42},\mathbf{0},{}_{3},{}^{52},\mathbf{0},{}^{43},{}_{73},\mathbf{0},{}_{18},{}^{38},\mathbf{0},{}_{16},{}^{34},\mathbf{0},$
$_{11},{}^{56},{}_{12},\mathbf{0},{}_{23},\mathbf{0},{}_{22},{}^{40},{}_{75},\mathbf{0},{}_{17},{}^{44},\mathbf{0},{}_{9},{}^{65},\mathbf{0},{}^{63},{}_{20},{}^{54},\mathbf{0},{}^{35},{}_{8},\mathbf{0},{}^{37},{}_{7},{}^{62},\mathbf{0},{}^{51},{}_{13},{}^{53},{}_{30},\mathbf{0},$
$^{46},{}_{19},{}^{47},\mathbf{0},{}_{5},{}^{55},{}_{10},\mathbf{0}$

Fig. 4. A solution of Example 19.

before the containers are transported. The CPU time for 11 of the 12 repeats is <2 h, which is an acceptable time period.

A solution for Example 19 is shown in Fig. 4. In the example there are 30 inbound full containers numbered 1–30, 35 outbound full containers numbered 31–65, and 10 inbound empty containers numbered 66–75. In Fig. 4, the number **0** in bold type designates the start/return node. Each segment separated by two '**0**'s represents the containers transported by one truck. In Fig. 4, all inbound containers are written in subscript and all outbound containers are written in superscript. We can see that the journey of any one truck involves neither continuous inbound containers nor continuous outbound containers. The reason for this is that the transfer-time from inbound containers to outbound containers, or from outbound containers to inbound containers, is usually quite short. Most of them are zero, which can be found in Table 1.

Furthermore, the parameters of the RTS algorithm can be fixed, except that the maximum number of iterations and the maximum size of the neighborhood must be increased as the size and complexity of the situation increase.

In summary, the developed RTS algorithm is relatively stable and robust. The optimum solution for small-sized instances can be obtained in a short period of time, and larger realistic-sized instances can also be solved within an acceptable computation time.

7. Conclusions

In this paper, we investigated a container drayage problem with empty container constraints. An export-dominant area was considered, and there are three distinct types of container movements; inbound full, outbound full, and inbound empty movements. Each full container (inbound or outbound) has time windows at both its origin and its destination. Empty containers were also considered as separate transportation resources besides trucks. The initial number of empty containers at the depot was given. The total operating time including the waiting time of all the trucks in operation were minimized.

The problem was first formulated as a determinate-activities-on-node (DAON) graph and then mathematically modeled based on the graph. It can be considered as a multiple traveling salesman problem with time windows (m-TSPTW) with additional resource constraints. A meta-heuristics based on reactive tabu search (RTS) was developed to solve the problem. A lower bound with the empty container constraint relaxed was able to

prove the performance of the algorithm. A number of randomly generated examples indicate that the algorithm can be applied to the real world.

Acknowledgements

The authors are grateful to two reviewers for their constructive comments. This work was supported by the Grant of the Korean Ministry of Education, Science and Technology (The Regional Core Research Program/Institute of Logistics Information Technology).

References

Battiti, R., Tecchiolli, G., 1994. The reactive tabu search. ORSA Journal on Computing 6 (2), 126–140.

Blochliger, I., Zufferey, N., 2008. A graph coloring heuristic using partial solutions and a reactive tabu scheme. Computers & Operations Research 35 (3), 960–975.

Braysy, O., Gendreau, M., 2005. Vehicle routing problem with time windows, part II: metaheuristics. Transportation Science 39 (1), 119–139.

Cheung, R.K., Hang, D.D., 2003. Multi-attribute label matching algorithms for vehicle routing problems with time windows and backhauls. IIE Transactions 35 (3), 191–205.

Cheung, R.K., Shi, N., Powell, W.B., Simao, H.P., 2008. An attribute-decision model for cross-border drayage problem. Transportation Research Part E: Logistics and Transportation Review 44 (2), 217–234.

Chung, K.H., Ko, C.S., Shin, J.Y., Hwang, H., Kim, K.H., 2007. Development of mathematical models for the container road transportation in Korean trucking industries. Computers & Industrial Engineering 53 (2), 252–262.

Coslovich, L., Pesenti, R., Ukovich, W., 2006. Minimizing fleet operating costs for a container transportation company. European Journal of Operational Research 171 (3), 776–786.

Desaulniers, G., Desrosiers, J., Lasry, A., Solomon, M.M., 1999. Crew pairing for a regional carrier. Lecture Notes in Economics and Mathematical Systems 471, 19–41.

Fischer, K., Chaib-draa, B., Muller, J.P., Pischel, M., Gerber, C., 1999. A simulation approach based on negotiation and cooperation between agents: a case study. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 29 (4), 531–545.

Glover, F., 1989. Tabu search: part I. ORSA Journal on Computing 1 (3), 190–206.

Imai, A., Nishimura, E., Current, J., 2007. A Lagrangian relaxation-based heuristic for the vehicle routing with full container load. European Journal of Operational Research 176 (1), 87–105.

Jula, H., Dessouky, M., Ioannou, P., Chassiakos, A., 2005. Container movement by trucks in metropolitan networks: modeling and optimization. Transportation Research Part E: Logistics and Transportation Review 41 (3), 235–259.

Macharis, C., Bontekoning, Y.M., 2004. Opportunities for or in intermodal freight transport research: a review. European Journal of Operational Research 153 (2), 400–416.

Namboothiri, R., Erera, A.L., 2008. Planning local container drayage operations given a port access appointment system. Transportation Research Part E: Logistics and Transportation Review 44 (2), 185–202.

Sinclair, M., Van Dyk, E., 1987. Combined routing and scheduling for the transportation of containerized cargo. Journal of the Operational Research Society 38 (6), 487–498.

Smilowitz, K., 2006. Multi-resource routing with flexible tasks: an application in drayage operations. IIE Transactions 38 (7), 577–590.

Vis, I.F.A., de Koster, R., 2003. Transshipment of containers at a container terminal: an overview. European Journal of Operational Research 147 (1), 1–16.

Wang, D., Wang, J., Wang, H., Zhang, R., Guo, Z., 2007. Intelligent Optimization Methods. Higher Education Press, China, Beijing.

Wang, X., Regan, A.C., 2002. Local truckload pickup and delivery with hard time window constraints. Transportation Research Part B: Methodological 36 (2), 97–112.

Zhang, R., Yun, W.Y., 2008. An optimum route modeling for container truck transportation. In: Proceedings of Asia Conference on Intelligent Manufacturing & Logistics Systems, Kitakyushu, Japan, pp. 356–362.