Transportation Research Part E 45 (2009) 904–914

Contents lists available at ScienceDirect

Transportation Research Part E

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

A reactive tabu search algorithm for the multi-depot container truck transportation problem

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

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

ARTICLE INFO

Article history:
Received 2 July 2008
Received in revised form 30 January 2009
Accepted 21 April 2009

Keywords:
Traveling salesman problem
Time windows
Multiple depots
Container truck transportation
Reactive tabu search
Heuristic

ABSTRACT

A container truck transportation problem that involves multiple depots with time windows at both origins and destinations, including the reposition of empty containers, is formulated as a multi-traveling salesman problem with time windows (m-TSPTW) with multiple depots. Since the problem is NP-hard, a cluster method and a reactive tabu search (RTS) algorithm are developed to solve the problem. The two methods are compared with the mixed integer program which can be used to find optimum solutions for small size problems. The computational results show that the developed methods, particularly the RTS algorithm, can be efficiently used to solve the problem.

1. Introduction

Nowadays, container transportation plays a key role in international logistics. Furthermore, the importance of container transportation and the proportion that it holds in the transportation market have been increased significantly over the last two decades. Container transportation is a very complicated problem because of the numerous container types and container sizes and the different transportation modes, and due to the increasing number of customer requests. Numerous studies related to different container transportation problems have been published. For example, Yun and Choi (1999) proposed a simulation model of the transportation and operation of containers in yards using the simulation software SIMPLE ++. Shintani et al. (2007) examined a container shipping network design problem with empty container repositioning and developed a GA (genetic algorithm) – based heuristic algorithm to solve it. Hsu and Hsieh (2007) formulated a two – objective model in order to determine the optimal liner routing, ship size and sailing frequency for container carriers by minimizing the shipping cost and inventory cost. Furthermore, Vis and de Koster (2003), Macharis and Bontekoning (2004), and Steenken et al. (2004) classified the container operation problems in an inter-modal terminal and reviewed the related literature.

Containers are initially transported from their shippers to a terminal and finally transported from another terminal to their receivers. This process of container transportation is usually done by truck. Although the distance of this container truck transportation is relatively short compared to that of maritime transportation, the transportation cost per 20-foot equivalent unit (TEU) is relatively high. Therefore, the container truck transportation is a very important problem. A short review of the literature addressing container truck transportation is presented as follows. Coslovich et al. (2006) modeled a real-world container movement problem as an integer programming problem, which they solved by decomposing it into

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

1366-5545/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.tre.2009.04.012

three simpler sub problems. Imai et al. (2007) addressed the vehicle routing problem that arose during the picking up and delivering of a full container load from/to an inter-modal terminal, and developed a sub-gradient heuristic algorithm based on a Lagrangian relaxation in order to identify a near optimal solution. Chung et al. (2007) constructed a number of models for a container truck transportation problem. After a basic model was constructed, the problem was extended in order to include the cases that involve time windows and multiple commodities, and where different types of trucks are utilized. Since a number of ports have been equipped with an appointment-based access control system, Namboothiri and Erera (2008) studied a local container truck transportation problem for such a port. Cheung et al. (2008) constructed an attribute-decision model for a cross-border container transportation problem, using Hong Kong as the study example. Additionally, several container truck transportation problems were formulated as either a multi-traveling salesman problem with time windows (m-TSPTW) or an m-TSPTW with social constraints, and were solved based on either heuristics or meta-heuristics (Wang and Regan, 2002; Jula et al., 2005).

The transportation of containers either from the initial shipper or to the final receiver differs from other types of container transportation. The shipping company usually owns the containers that are being transported. The containers are a type of transportation resource used to contain freight. After a container of freight (i.e., an inbound container) is delivered to its receiver, the container should be emptied and the recently emptied container should be returned to a depot or to an alternative similar location. Conversely, when an amount of freight is to be exported, additional empty containers are needed. Therefore, the reposition of empty containers should be considered when we research the container truck transportation. However, to the authors' knowledge, such research has not been carried out in any of the literature reviewed above.

The main contributions made by this paper are as follows:

(1) A container truck transportation problem that involves multiple depots with time windows at the origin and destination, including the reposition of empty containers, is formulated as a multiple graph and modeled as an asymmetric m-TSPTW with multiple depots which is a mixed integer program.
(2) A cluster heuristic algorithm is developed to solve the m-TSPTW with multiple depots. The cluster algorithm consists of two main steps: construction of container vertex sequences, and assignment of start depot vertices to the sequences. The cluster algorithm has been tested by using a number of randomly generated examples and compared with the solutions obtained from the mixed integer program. The cluster algorithm can solve large size problems very quickly and the obtained results are acceptable.
(3) A reactive tabu search (RTS) algorithm is developed to solve the problem. In order to automatically balance the two optimization abilities as intensification and diversification, the length of the tabu list is adaptable and an escape mechanism is introduced. The RTS algorithm has been tested and compared with other methods. The results indicate that the RTS algorithm is able to find the optimal solutions for small-sized examples in a short period of time, and the RTS algorithm is sufficiently robust to solve large-sized instances.

The remaining content of this paper is organized as follows. The multi-depot container truck transportation problem is described in Section 2, formulated as a directed graph in Section 3, and mathematically modeled in Section 4. A cluster method and a reactive tabu search (RTS) algorithm are developed in Section 5, and are tested and analyzed using a number of randomly generated examples in Section 6. The paper is concluded in Section 7.

2. Multi-depot container truck transportation problem

There are three types of sites that need to be considered in the container transportation problem: the container terminal, depot, and the customer. A container terminal is a site where various containers are transshipped. It can be a sea port, or a railway hub station. A container depot is a warehouse where various empty containers are stacked and container trucks can be parked. A customer can be regarded as a plant that will send or receive freight by containers. It is assumed that there is a single container terminal and several container depots in the considered local area. A truck company with a number of container trucks serves the container movement in the area. Each truck can be parked at any depot at any time. Furthermore, the trucks can be used to pick up and drop off empty containers at any depot at any time.

There are several types of container movements as follows. The inbound full containers should be picked up at the terminal and delivered to their receivers. After an inbound full container is unpacked, the truck should be used to deliver the recently emptied container to a depot, or to a customer who will export freight at that time. Similarly, a certain number of empty containers should be delivered to the customers who will use them for export freight. After a container is packed, the truck should be used to deliver the container to the terminal for transshipment. In addition, a number of empty containers are sometimes required to be either imported or exported due to the occasional trade imbalance between areas. Therefore, a number of empty containers should be picked up from or delivered to the terminal. This is a special type of container movement.

It is assumed that the truck company personnel know all the transportation tasks to be carried out in advance of the transportation. All trucks are standardized and each truck can carry exact one container at any one time (e.g., a 40-foot container). All trucks are initially located at some depots. A truck can be sent to any depot after its tasks are finished. The pickups and deliveries of containers at the terminal and at the customer's location must begin during their corresponding time periods. All the containers should be delivered directly to their destinations without being stacked at the depots. The objective of this task is to minimize the total unprofitable traveling time.

Let D be a set of depots, i.e., $D = \{d_1, d_2, \ldots, d_m\}$, where m is the number of depots. There are n_i trucks originally located at the depot $d_i \in D$. Let C be a set of containers (cargo) to be transported during the considered time period, i.e., $C = C_{IF} \cup C_{OF} \cup C_{IE} \cup C_{OE}$, where C_{IF}, C_{OF}, C_{IE}, and C_{OE} are the sets of inbound full, outbound full, inbound empty, and outbound empty containers, respectively. The total number of inbound/outbound full containers is p; and the total number of inbound/outbound empty containers is q. Therefore, the container set can also be described as $C = \{c_{m+1}, c_{m+2}, \ldots, c_{m+p}, c_{m+p+1}, \ldots, c_{m+p+q}\}$. Note that the container set, C, is not exactly related to physical containers, but is related to the tasks that are to be finished. For example, the cargo to be exported has not been packed, but a container of cargo to be exported is still called an *outbound full container*. In other words, it is an element of set C_{OF}. Furthermore, the containers containing export cargo, including the cargo in them, are known as outbound full containers, regardless of where the containers originate from.

The origin (pick up) time window of container $c_i \in C$ is $[\tau_{Ai}, \tau_{Bi}]$ $(\tau_{Ai} \leqslant \tau_{Bi})$. The destination (delivery) time window of an inbound/outbound full container $c_i \in C_{IF} \cup C_{OF}$ is $[\tau_{Ci}, \tau_{Di}]$ $(\tau_{Ci} \leqslant \tau_{Di})$. Here, $[\tau_{Ai}, \tau_{Bi}]$ and $[\tau_{Ci}, \tau_{Di}]$ are hard time windows associated with customer locations and the terminal. If a truck arrives at a location before the lower bound of the time window, i.e., τ_{Ai} or τ_{Ci}, it must wait until the lower bound. A truck cannot serve if it arrives after the upper bound of the time window.

Let $t(t \geqslant 0)$ be the loading/unloading time of a container; and $t_i(t_i \geqslant 0)$ be the packing/unpacking time of an inbound/outbound full container $c_i \in C_{IF} \cup C_{OF}$ at its customer's location.

Each inbound full container $c_i \in C_{IF}$ has a receiver. Each outbound full container $c_i \in C_{OF}$ has a shipper. The receivers and shippers, as well as the depots and the terminal, form a location set in this problem. Let L be this set of locations, i.e., $L = \{l_0, l_1, \ldots, l_m, l_{m+1}, \ldots, l_{m+p}\}$, where l_0 is the terminal location; $l_i(i = 1, \ldots, m)$ is the location of the depot $d_i \in D$; and $l_i(i = m + 1, \ldots, m + p)$ is the location of the receiver /shipper's container $c_i \in C_{IF} \cup C_{OF}$. The traveling time between locations l_i and $l_j(l_i, l_j \in L;\ i \neq j)$ is then denoted by $t_{ij}(t_{ij} \geqslant 0;\ t_{ij} = t_{ji})$.

For an inbound full container $c_i \in C_{IF}$, we have $\tau_{Di} - \tau_{Ai} \geqslant t + t_{0i}$. Otherwise, the origin and destination time windows cannot be satisfied simultaneously (see Fig. 1 for an example). Similarly, for an outbound full container $c_i \in C_{OF}$, we have $\tau_{Di} - \tau_{Ai} \geqslant t_i + t + t_{i0}$.

3. Construction of the graph model

The container truck transportation problem described in Section 2 is similar but very different to the pickup and delivery problem (PDP). There is another type of resource as containers besides trucks in this problem. An empty container should be delivered before a container of freight to be exported is picked up. Conversely, a new empty container will be released after a full container is delivered and unpacked. Therefore, a graph model is constructed in order to formulate the problem considered in this study.

3.1. Definition of the graph

Let $G = (V_D, V_C, A)$ be a graph to formulate the considered problem, where $V_D = \{1, 2, \ldots, m\}$ and $V_C = \{m + 1, m + 2, \ldots, m + p + q\}$ are the depot vertex set and the container vertex set, respectively; $A = \{(i, j) | i, j \in V_D \cup V_C;\ i \neq j;\ i \notin V_D \text{ or } j \notin V_D\}$ is a set of arcs.

Definition 1. Depot vertex $i \in V_D$ represents the initial start from and final return to the depot $d_i \in D$. The activities as picking up and dropping off containers at depots do not belong to depot vertices. It belongs to various arcs, which will be described below.

Definition 2. Container vertex $i \in V_C$ represents a series of continuous determined activities that must be taken in order to finish the corresponding container transportation $c_i \in C$. The corresponding types of container vertex sets, i.e., $V_C = V_{IF} \cup V_{OF} \cup V_{IE} \cup V_{OE}$ can be defined according to the different types of containers in set C. The activities of four types of container vertices are listed in Table 1. The mapping relationships between the given parameters and the vertex sets are shown in Fig. 2.

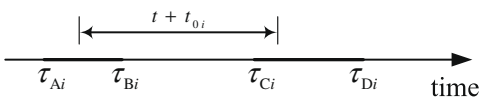

Fig. 1. Relationship between the origin and destination time windows.

Table 1
Activities of four types of container vertices.

Container vertex i	Activities
$i \in V_{IF}$	Pick up (load) the container at the terminal, deliver it to its receiver's location, drop off (unload), and unpack it
$i \in V_{OF}$	Pack the container at its shipper's location, pick up, deliver it to the terminal, and drop it off
$i \in V_{IE}$	Pick up the container at the terminal
$i \in V_{OE}$	Drop off the container at the terminal

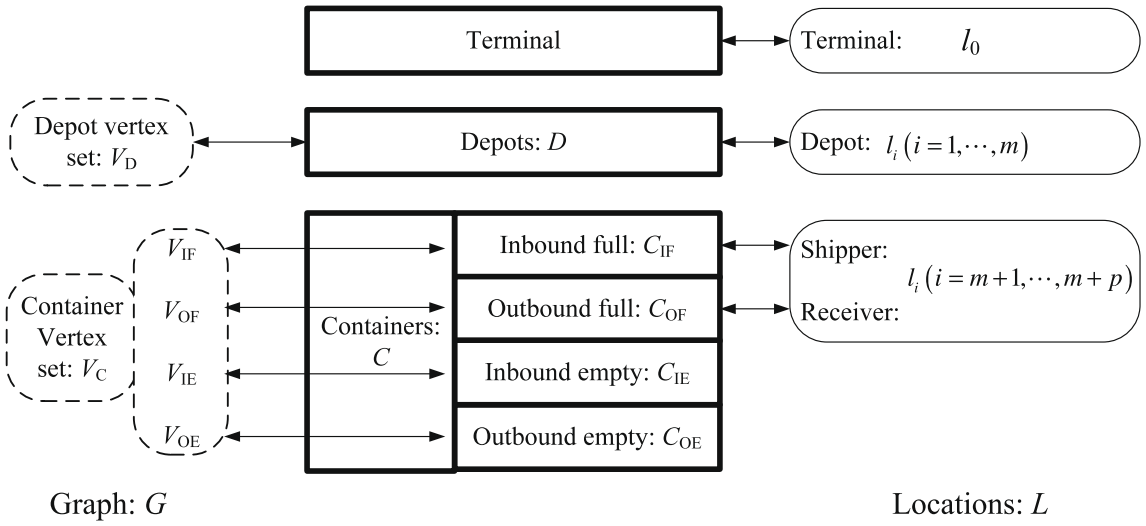

Fig. 2. Mapping relationships between given parameters and vertex sets.

Table 2
Activities of arcs.

Arc (i,j)	$j \in V_D$	$j \in V_{IF} \cup V_{IE}$	$j \in V_{OF}$	$j \in V_{OE}$
$i \in V_D$	–	Travel to the terminal	Pick up an empty container at the depot d_i, deliver it to the shipper of container c_j, and drop it off	Pick up an empty container at the depot d_i, and deliver it to the terminal
$i \in V_{IF}$	Pick up the recently emptied container c_i, deliver it to the depot d_j, and drop it off	Pick up the recently emptied container c_i, deliver it to a depot, drop it off, and travel to the terminal	If the receiver of container c_i is not the shipper of container c_j, pick up the recently emptied container c_i, deliver it to the shipper of container c_j, drop it off; otherwise, do nothing	Pick up the recently emptied container c_i, and deliver it to the terminal
$i \in V_{OF} \cup V_{OE}$	Travel to the depot d_j	Do nothing	Travel to a depot, pick up an empty container, deliver it to the shipper of container c_j, and drop it off	Travel to a depot, pick up an empty container, and deliver it to the terminal
$i \in V_{IE}$	Deliver the empty container c_i to the depot d_j, and drop it off	Deliver the empty container c_i to a depot; drop it off, and travel to the terminal	Deliver the empty container c_i to the shipper of container c_j, and drop off the empty container	Do nothing

Definition 3. Arc $(i,j) \in A$ is defined as the transfer from vertex i to vertex j. It represents various necessary activities. See Table 2 for detailed description of the arc activities.

3.2. Attributes of the graph

3.2.1. Attribute of depot vertices

The number of trucks, n_i, which are initially located at the depot $d_i \in D$, is the attribute of depot vertex $i \in V_D$.

3.2.2. Attribute 1 of container vertices

The amount of time consumed by container vertex $i \in V_C$ is called the *serving time* of this vertex. It is denoted by $T(i)$.

It is unavoidable for a truck to wait at the destination location under several certain conditions. Fig. 3 describes the relationships between the origin and destination time windows and the times of various activities, where inbound full containers are taken as an example. As shown in Fig. 3a, even if a truck is used to pick up an inbound full container at the terminal at the upper bound of a pick up time window, it will arrive before the lower bound of the delivery time window. Specifically, the waiting of this truck at the destination is unavoidable. This type of unavoidable waiting time is included in the serving time. In the case shown in Fig. 3b, it is not necessary for a truck to wait at the destination if the corresponding container is not picked up ahead of time.

The case of outbound full containers is somewhat similar to the case of inbound full containers. The only difference lies in the activities between the two time windows. The activities between the two time windows for inbound full containers are the pickup and transportation. However, for outbound full containers, they are pack, pickup and transportation.

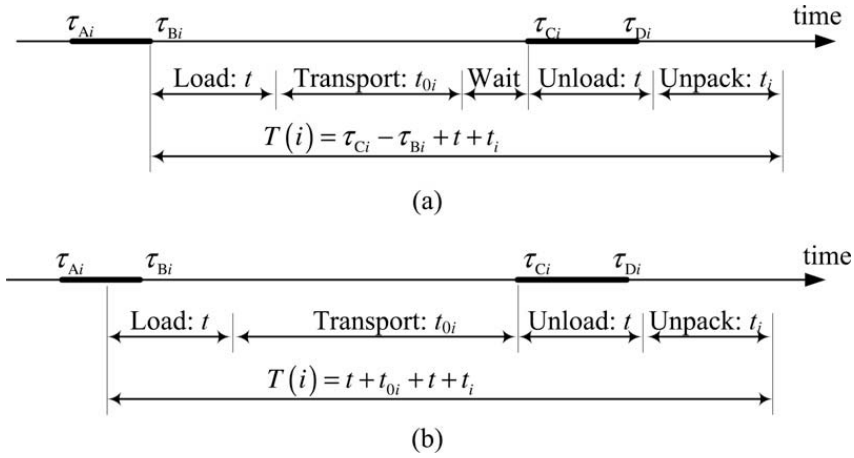

Fig. 3. Serving times of inbound full containers under different cases.

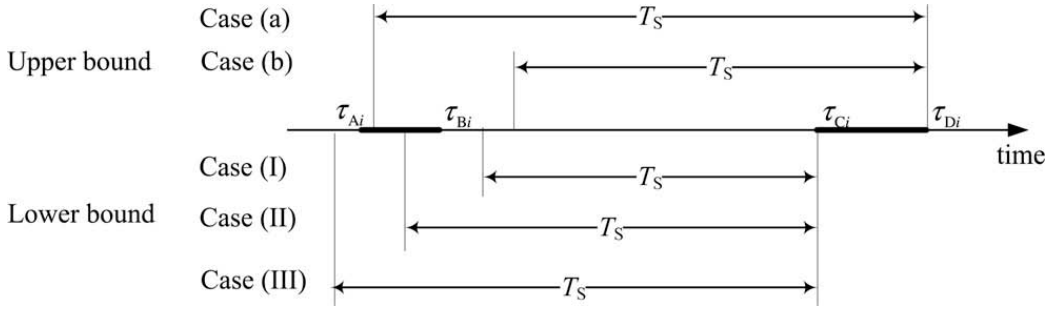

Fig. 4. Time windows of vertices under all cases.

The serving time of container vertex $i \in V_C$ is:

$$T(i) = \begin{cases} \max(\tau_{Ci} - \tau_{Bi}, t + t_{0i}) + t + t_i, & i \in V_{IF} \\ \max(\tau_{Ci} - \tau_{Bi}, t_i + t + t_{i0}) + t, & i \in V_{OF} \\ t, & i \in V_{IE} \cup V_{OE} \end{cases} \quad (1)$$

Attribute 2 of container vertices. The time period during which the container vertex $i \in V_C$ should be started is defined as the *time window* of the vertex. It is denoted by $[T_A(i), T_B(i)]$ $(T_A(i) \leqslant T_B(i))$.

It is known that each inbound or outbound full container involves two time windows, i.e., origin time window and destination time window. The two time windows should be combined according to the time consumed by the corresponding activities.

We shift the destination time window by the time T_S, where T_S is the amount of time consumed by the activities between the two time windows. The shifted destination time window and the origin time window are compared in Fig. 4. Several cases of the lower bound and upper bound are shown above and below the axis, respectively. In case (II) of Fig. 4, if a truck is used to pick up the corresponding container before the time $(\tau_{Ci} - T_S)$, it will wait at the destination; Otherwise, it will not. Therefore, it need not pick up the container before the time $(\tau_{Ci} - T_S)$. Of course, the lower bound of the time window of a vertex should be between the lower bound and upper bound of the pick up time window. Similarly, the truck cannot pick up the container later than the time $(\tau_{Di} - T_S)$, as shown in case (a) of Fig. 4. The case of $\tau_{Di} - T_S < \tau_{Ai}$ is infeasible and hence omitted in Fig. 4.

The time window of container vertex $i \in V_C$ is:

$$T_A(i) = \begin{cases} \min(\max(\tau_{Ai}, \tau_{Ci} - t - t_{0i}), \tau_{Bi}), & i \in V_{IF} \\ \min(\max(\tau_{Ai}, \tau_{Ci} - t_i - t - t_{i0}), \tau_{Bi}), & i \in V_{OF} \\ \tau_{Ai}, & i \in V_{IE} \cup V_{OE} \end{cases} \quad (2)$$

$$T_B(i) = \begin{cases} \min(\tau_{Bi}, \tau_{Di} - t - t_{0i}), & i \in V_{IF} \\ \min(\tau_{Bi}, \tau_{Di} - t_i - t - t_{i0}), & i \in V_{OF} \\ \tau_{Bi}, & i \in V_{IE} \cup V_{OE} \end{cases} \quad (3)$$

Table 3
Transfer times of arcs.

$T(i,j)$=	$j \in V_D$	$j \in V_{IF} \cup V_{IE}$	$j \in V_{OF}$	$v_j \in V_{OE}$
$i \in V_D$	–	t_{i0}	$t + t_{ij} + t$	$t + t_{i0}$
$i \in V_{IF}$	$t + t_{ij} + t$	$t + \min_{k=1,\ldots,m}(t_{ik} + t_{k0}) + t$	$\begin{cases} t + t_{ij} + t, & \text{if } t_{ij} \neq 0 \\ 0, & \text{if } t_{ij} = 0 \end{cases}$	$t + t_{i0}$
$i \in V_{OF} \cup V_{OE}$	t_{0j}	0	$\min_{k=1,\ldots,m}(t_{0k} + t_{kj}) + t + t$	$\min_{k=1,\ldots,m}(t_{0k} + t_{k0}) + t$
$i \in V_{IE}$	$t_{0j} + t$	$\min_{k=1,\ldots,m}(t_{0k} + t_{k0}) + t$	$t_{0j} + t$	0

3.2.3. Attribute of arcs

The amount of time consumed by the arc (i,j) is defined as the *transfer time* of the arc. It is denoted by $T(i,j)$.

Several arcs include the activities as traveling to a depot to deliver or pick up an empty container, and then traveling to another location, e.g., $i \in V_{OF} \cup V_{OE}$, $j \in V_{OF}$. In such cases, a suitable depot with the shortest traveling time must be chosen. The transfer time can be obtained as shown in Table 3 based on the corresponding activities. Any container vertex can be connected with any other container vertex or depot vertex if the time windows are suitable. This can help us to build a mathematical model for the problem.

In summary, a directed graph G with vertex set $V_D \cup V_C$ and arc set A has been obtained. Each depot vertex $i \in V_D$ has n_i trucks. Each container vertex $i \in V_C$ has a serving time $T(i)$ and a time window $[T_A(i), T_B(i)]$. Each arc $(i,j) \in A$ has a transfer time $T(i,j)$.

4. Mixed integer programming model

The problem is to find several routes with the minimal total transfer time. The first and last vertices of each route must be depot vertices and all the other vertices must be container vertices. The number of routes starting from each depot vertex cannot exceed its attribute n_i. Each container vertex must be visited exactly once during its time window.

The following decision variables are introduced.

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

y_i: time when a truck starts to serve the container vertex $i \in V_C$.

The problem can be formulated as the following mixed integer programming (MIP) model.

$$\min \sum_{(i,j)\in A} T(i,j)x_{ij} \tag{4}$$

Subject to

$$\sum_{j\in V_C} x_{ij} \le n_i, \quad \forall i \in V_D \tag{5}$$

$$\sum_{j\in V_D\cup V_C} x_{ij} = \sum_{j\in V_D\cup V_C} x_{ji} = 1, \quad \forall i \in V_C \tag{6}$$

$$\sum_{i\in Z, j\in Z} x_{ij} \le |Z| - 1, \quad \forall Z \subseteq V_C,\ Z \neq \Phi \tag{7}$$

$$T_A(i) \le y_i \le T_B(i), \quad \forall i \in V_C \tag{8}$$

$$y_i + T(i) + T(i,j) - y_j \le (1 - x_{ij})M, \quad \forall i \in V_C,\ j \in V_C \tag{9}$$

$$x_{ij} \in \{0,1\}, \quad \forall (i,j) \in A \tag{10}$$

$$y_i : \text{real variable}, \quad \forall i \in V_C \tag{11}$$

Here, the objective function (4) minimizes the total transfer time. The total transfer time (i.e., unprofitable operation time) involves the time of traveling with empty containers and traveling without containers. Constraint (5) indicates that the number of arcs that leave any depot vertex should not exceed the number of vehicles that are initially located at the corresponding depot. Constraint (6) implies that exact one arc enters and leaves any container vertex. Constraint (7) eliminates the sub-tours among container vertices, where Z is a sub set of V_C and Φ denotes an empty set. Constraints (6) and (7) indicate that any route initially starts from a depot vertex, and finally returns to a depot vertex. Actually, each route is followed by one vehicle. Constraint (8) indicates that the service of any container vertex should be started during its corresponding time window. Constraint (9) updates the start time along the route, where M is a sufficiently large number. Constraints (10) and (11) imply the types of variables.

If there is only one depot vertex, then all the vehicles should initially start from and finally return to the depot, and then the proposed problem falls into the multi- traveling salesman problem with time windows (m-TSPTW). Therefore, the proposed problem generalizes m-TSPTW after it is formulated as the graph. The proposed problem is called m-TSPTW with multiple depots. This problem is NP-hard since m-TSPTW is NP-hard (Toth and Vigo, 2002).

5. Heuristic algorithms

Since the considered problem is formulated as a linear mixed zero–one integer programming problem, it can be solved using various commercial tools such as CPLEX. However, such tools can only solve the instances with small size in considerable time. Some efficient heuristic algorithms should be developed in order to solve the real-world instances. Although m-TSPTW is a special case of the vehicle routing problem with time windows (VRPTW) (Chabrier, 2006), the best known solution approach to VRPTW is not well suited to solve m-TSPTW (Jula et al., 2005). Furthermore, the existing approaches to m-TSPTW cannot be directly used to solve this extension of m-TSPTW. Two heuristic algorithms are therefore developed to solve m-TSPTW with multiple depots.

5.1. Cluster method

5.1.1. Step 1: construct several sequences of container vertices

Firstly, the depot vertices and all the arcs associated with depot vertices are removed from G. An arc (i,j) is infeasible if $T_A(i) + T(i) + T(i,j) > T_B(j)$. All such infeasible arcs are removed. The obtained sub graph is denoted by $\underline{G}$, i.e., $\underline{G} = (\underline{V}, \underline{A}) = (V_C, \underline{A})$. A sequence initialized as $s_i = i$ is assigned to each vertex $i \in \underline{V}$.

Secondly, the arc with the shortest transfer time in $\underline{A}$ is denoted by (i,j). The tail vertex i and the head vertex j are combined into a new vertex and denoted by vertex k. The sequences of vertices i and j are connected. The sequence of vertex k is obtained as follows.

$$s_k = s_i \rightarrow s_j \tag{12}$$

The calculation of the serving time and time window of vertex k is similar to that of inbound full containers in Eqs. (1)–(3). Specifically,

$$T(k) = \max(T_A(j) - T_B(i), T(i) + T(i,j)) + T(j) \tag{13}$$

$$T_A(k) = \min(\max(T_A(i), T_A(j) - T(i) - T(i,j)), T_B(i)) \tag{14}$$

$$T_B(k) = \min(T_B(j) - T(i) - T(i,j), T_B(i)) \tag{15}$$

All the arcs that are related to the vertices i and j are removed and replaced by a number of new arcs that are related to vertex k. The transfer time of the new arcs can be formulated as:

$$T(i',k) = T(i',i), \quad \forall i' \in \underline{V} \tag{16}$$

$$T(k,j') = T(j,j'), \quad \forall j' \in \underline{V} \tag{17}$$

The feasibility of these new arcs is checked and then the infeasible arcs (if any) are removed. The number of vertices and the number of arcs in the sub graph $\underline{G}$ are decreased in this combination step. The two vertices of the arc with the shortest transfer time are combined again. Such combination step is continued until there is no feasible arc remains.

Finally, there are only several separated vertices in the sub graph $\underline{G}$. Let r be the number of vertices in $\underline{G}$. Each vertex has a sequence $s_i = (s_{i1}, s_{i2}, \ldots, s_{ih_i})$ $(i = 1, \ldots, r)$, where h_i is the length of sequence s_i. Each sequence means the containers which will be visited by one vehicle in their orders.

5.1.2. Step 2: assign depot vertices to the sequences

In this step, a start vertex s_{i0} and a return vertex $s_{i(h_i+1)}$ are assigned to each sequence s_i. Here, s_{i0} and $s_{i(h_i+1)}$ are depot vertices defined in Section 3. The vehicle that visits the container vertices in sequence s_i is initially located at depot s_{i0} and will finally be returned to depot $s_{i(h_i+1)}$.

First of all, a vehicle located at depot j^* is assigned to the sequence s_{i^*} according to the following selection criterion.

$$(j^*, i^*) = \arg\min_{j \in V_D, i \in \underline{V}} T(j, s_{i1}) \tag{18}$$

Specifically, all the arcs from depot vertices $j \in V_D$ to the first vertex of sequences $s_{i0}(i = 1, \ldots, r)$ are candidate arcs. The arc with the shortest transfer time is selected from the candidate arcs. Once depot j^* has been assigned to the sequence s_{i^*}, the number of unassigned vehicles at depot j^* is increased by one. If all the vehicles at a depot have already been assigned, the depot cannot be reassigned again. Furthermore, one sequence cannot be assigned for multiple times. Therefore, the candidate arcs should be updated after each assignment. Each sequence is assigned by a start depot vertex after such assignment is executed for r times.

When a vehicle finishes visiting all the containers in sequence s_i, the nearest depot to the current location is chosen. Therefore, the return depot vertex is:

$$s_{i(h_i+1)} = \arg\min_{j \in V_D} T(s_{ih}, j), i = 1, \ldots, r \tag{19}$$

5.2. Reactive tabu search algorithm

It has been proven that meta-heuristic methods such as the tabu search (TS), simulated annealing (SA), and genetic algorithm (GA) are able to solve many NP-hard problems efficiently (Wang et al., 2007). Among various meta-heuristic methods,

the TS method, first proposed by Glover (1989), appears to be the most efficient method for solving various combinatorial optimization problems including a number of VRPs (Nanry and Wesley Barnes, 2000; Toth and Vigo, 2002). Generally speaking, various parameters of meta-heuristic methods including TS should be carefully adapted in order to obtain a sound performance. The performance of the TS algorithm depends mainly on the length of the tabu list. The reactive tabu search (RTS) is an improved version of TS. It was first proposed by Battiti and Tecchiolli (1994). The tabu list length of RTS can be self-adapted to balance intensification and diversification. Furthermore, an escape mechanism is introduced to avoid recycling. The RTS method has been successfully applied in many fields (Castellani et al., 2007; da Silva et al., 2008; Blochliger and Zufferey, 2008).

An RTS algorithm is developed to solve the considered problem as follows.

5.2.1. Initial solution

An initial solution is generated using the cluster method described in Section 5.1.

5.2.2. Encoding

All the sequences of container vertices $s_i (i = 1, \ldots, r)$ and the corresponding start depot vertices $s_{i0} (i = 1, \ldots, r)$ are encoded into an array. If a certain depot has surplus vehicles that are not used in the solution, the depot still appears the times as the number of vehicles in the solution.

$$s = [s_{10}, s_{11}, \ldots, s_{1h_1}, \ldots s_{i0}, s_{i1}, \ldots s_{ih_i}, \ldots, s_{r0}, s_{r1}, \ldots s_{rh_r}, 1, 1, 2] \tag{20}$$

Eq. (20) is a typical formulation of a solution. The segment from s_{i0} to s_{ih_i} of the solution presents a route. A vehicle located at depot s_{i0} serves the container vertices from s_{i1} to s_{ih_i} in their orders. The elements 1, 1, and 2 at the end of the solution mean that two vehicles located at depot 1 and one vehicle located at depot 2 are not used in the current solution. It can be observed that the total length of array s is the number of containers plus the number of vehicles.

5.2.3. Variable-sized neighborhood structure

If two randomly selected elements of the current solution are exchanged, a neighborhood solution is obtained. In order to guarantee that there is a depot vertex in front of each sequence of container vertices, the first element of the array s is fixed during neighborhood exchanges. If the obtained neighbor solution is infeasible, it is discarded immediately. If it is superior to the best solution in history, it is accepted regardless of its tabu status. Otherwise, neighbor solutions are generated continuously until a sufficient number of feasible neighbors are obtained. The best non-tabu neighbor is accepted. The size of such neighborhood varies from 1 to its maximum number. This can save computation time compared to a fix-sized neighborhood mechanism.

5.2.4. Decoding, feasibility, and optimality of a solution

The feasibility and optimality of a solution can be verified along with the decoding of the solution. A vehicle from a depot serves the container vertices that follow the depot vertex orderly until another depot vertex is reached or until the end of the array. If no container vertex follows a depot vertex, the vehicle from this depot vertex is not used. The constraints are satisfied automatically with the exception of the time constraints (8) and (9). Therefore, an attempt to assign the start time y_i of a container vertex i along the route is made. If the assignment succeeds, the solution is feasible. Otherwise, it is infeasible. The objective value of the solution is the total transfer time of all segments $[s_{i0}, s_{i1}, \ldots, s_{ih_i}]$ plus the total transfer time from s_{ih_i} to the nearest depot vertex.

5.2.5. Adaptive tabu list

A pair of exchanged two elements of a solution is a cell of the tabu list. Two coefficients are given in the reactive tabu search. Let n_{INC} ($n_{\mathrm{INC}} > 1$) and n_{DEC} ($0 < n_{\mathrm{DEC}} < 1$) be the increase coefficient and decrease coefficient of the tabu list length, respectively. All the accepted solutions during iterations are recorded. After a solution is accepted, it is checked whether this solution has been previously visited or not. If the solution has been previously visited, it means that the search has fallen into a cycle. Therefore, the length of the tabu list should be increased. That is, $L_{\mathrm{T}} = L_{\mathrm{T}} n_{\mathrm{INC}}$, where L_{T} is the length of the tabu list. If no repeated solution has been found for a certain number of iterations, the length of the tabu list is decreased, whereby $L_{\mathrm{T}} = L_{\mathrm{T}} n_{\mathrm{DEC}}$.

5.2.6. Escape mechanism

If it is found that various solutions have been repeated for a certain number of times, an escape operation is triggered. The random exchange described in Section 5.2.3 is executed for several times. The ensuing search is then based on the new solution obtained. All the random exchanges are added to the tabu list in order to ensure that the search does not rapidly return to the cycle.

5.2.7. Record and retrieval of solutions

A large number of solutions are recorded and frequently retrieved in RTS. Therefore, a binary tree is introduced to store the solutions. If all the elements of two solutions are equal, then the solutions are equal. Otherwise, the comparison result is defined as the first unequal element of the solutions.

5.2.8. Stopping criterion

If the given maximum number of iterations is reached, the search stops.

6. Computational experiments

6.1. Generation of examples

The performance of the proposed solution methods should be tested on a large number of examples. However, it is very hard to obtain enough real-world data sets from trucking companies. Furthermore, there are no benchmark examples for the container truck transportation problem, to the authors' knowledge. Therefore, various examples are randomly generated in order to test the solution methods. The generated examples are as similar to the real-world case as they can be. Similar experiments based on randomly generated examples can be found in many existing papers such as Cheung et al. (2008), Imai et al. (2007) and Jula et al. (2005).

The examples are based on the Euclidean plane with a length and width equal to a 3-h's distance by truck. Several locations are randomly generated in the Euclidean plane firstly. These locations include the terminal, the depots and customers. Then, the shippers of the outbound full containers and the receivers of the inbound full containers are uniformly located at the customer locations. Several shippers of the outbound full containers and several receivers of the inbound full containers can occasionally be located at the same site, simulating the real-life case. A number of vehicles are initially located at the depots.

The load/unload time is assumed to be 5 min (Chung et al., 2007). The pack/unpack time of the full containers are distributed within the range of 5–60 min. The traveling time between any two locations can be calculated based on their coordinates. The lower bounds of pick up time windows of all the containers are distributed within the range of 8:00 am to 12:00 am. The interval lengths of pick up time windows are generated as random variables in the interval from 0 to 3 h. For inbound/outbound full containers, it is assumed that the lower bounds of the delivery time windows are the corresponding lower bounds of the pick up time windows plus the traveling times between the corresponding customer locations and the terminal. The interval lengths of the delivery time windows are generated as random variables in the interval from 2 to 5 h. Ten examples are generated to test the performance of the methods. General information for these examples is shown in Table 4.

The operation times and time windows in the examples are reasonable. Besides, the examples are very similar to the real-life case. All types of time windows, wide or narrow, can be found in the examples. For instance, the pickup time windows (in min) in Example 3 are listed in Table 5.

6.2. Comparison of the three algorithms

The linear MIP model is coded in ILOG CPLEX 10.0.0. The experiments are tested on a computer with Intel® Pentium CPU 3.40 GHz, 3.39 GHz, and 0.99 GB memory. The cluster method and the RTS algorithm are coded in Matlab 7.3 by Math Works, Inc. All examples are tested by the cluster method and the RTS algorithm on the same personal computer. The parameters of

Table 4
General information for the examples.

Example no.	m	$n_i (i = 1, \ldots, m)$	Number of containers				
			In-full	Out-full	In-empty	Out-empty	Total
1	2	2, 2	2	2	0	1	5
2	2	4, 3	5	4	0	1	10
3	4	2, 2, 2, 2	5	5	5	0	15
4	3	4, 4, 4	10	5	0	1	16
5	3	4, 4, 4	10	5	0	2	17
6	3	4, 4, 4	10	5	0	3	18
7	3	4, 4, 4	10	5	0	5	20
8	4	16, 6, 6, 6	30	15	0	5	50
9	4	10, 10, 10, 10	40	40	0	20	100
10	6	20, 20, 20, 20, 20, 20	100	80	20	0	200

Table 5
Pickup time windows in Example 3.

Container	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
LB	8	128	14	186	166	84	155	193	8	90	106	85	193	4	208
UB	135	223	161	336	230	229	268	215	43	240	108	91	293	60	296
Length	127	95	147	150	64	145	113	22	35	150	2	6	100	56	88

Table 6
Parameters of the RTS algorithm.

Parameters of the RTS algorithm	Values in the examples
Initial length of tabu list	5
Increase coefficient of the tabu list length (n_{INC})	1.01
Maximum number of iterations without finding repeated solutions	20
Decrease coefficient of the tabu list length (n_{DEC})	0.9
Maximum times of repeated solutions found	20
Maximum size of neighborhood	5 for Example 1, 20 for Example 2, 100 for Example 3 to Example 10
Maximum number of iterations	20 for Example 1, 100 for Example 2, 1000 for Example 3 to Example 6, 1500 for Example 7 to Example 10

Table 7
Comparison of the three solution methods.

Example no.	MIP		Cluster method		RTS algorithm	
	Objective value	CPU time	Objective value	CPU time	Objective value	CPU time
1	325	1.17	325	0.03	325	0.19
2	1096	3.48	1236	0.05	1096	4.25
3	682	23.9	949	0.06	682	289.77
4	598	64.98	761	0.08	598	220.55
5	1926	152.28	2067	0.11	1926	240.95
6	1417	1931.87	1854	0.17	1615	275.5
7	NA[a]	NA	1145	0.16	1105	572.63
8	NA	NA	4831	1.13	4556	1077.23
9	NA	NA	7836	9.61	6006	3428.30
10	NA	NA	16597	66.92	15342	5127.69

[a] NA: the result cannot be obtained because of too much computational time.

Table 8
The statistical information of the RTS algorithm.

Example	Index	Each repeated time										Statistical	
		1	2	3	4	5	6	7	8	9	10	Worst	Opt (%)
6	Obj	1417	1417	1417	1417	1754	1615	1417	1417	1417	1417	1754	80
	CPU	409	427	471	453	461	457	545	455	457	518	545	–
7	Obj	1105	1105	1105	1106	1105	1105	1105	1105	1105	1105	1106	90
	CPU	573	521	545	713	536	554	534	549	614	537	713	–

the RTS algorithm are shown in Table 6. The maximum size of the neighborhood and the maximum number of iterations increase along with the size of the examples.

The obtained objective values (in min), and the CPU times (in s) of the three methods are shown in Table 7.

As shown in Table 7, we found optimal solutions for the first six examples. However, as expected, the computation time of this method increases significantly as the number of containers increases. Therefore, the examples with medium or large size cannot be solved using this method. The examples with any size can be solved using the cluster method very quickly. In addition, the performance of the cluster method is relatively acceptable. The objective value gaps between the two methods are approximately 20%. The cluster method found the optimum solution for Example 1. The optimum solutions of most of the examples can be found by the RTS algorithm in a considerable amount of time. It can be observed that for the first five examples, the optimum solutions have been found by the RTS algorithm. Example 6 is the only example whereby the optimum solution has not been found by RTS with the given parameters. Therefore, this example is solved again by RTS with the maximum number of iterations being 1500. The optimum solution is found while the computation time is increased to 408.80 s. It can be found that the reactive mechanism of RTS can prevent the algorithm from cycling efficiently. The optimum solutions of the tested examples can be found only if the neighborhood size and the maximum iteration times are of a sufficient number. It is not necessary to vary the other parameters when the size of the problems changes. Therefore, among the three methods, the RTS algorithm is recommended.

6.3. Stability of the RTS algorithm

The developed RTS algorithm is a type of random search algorithm. Therefore, if the algorithm is re-executed, different results may perhaps be obtained. In order to test the stability of the RTS algorithm, Examples 6 and 7 are solved repeatedly for 10 times. The obtained results are shown in Table 8.

It can be observed that the optimum solutions are obtained at a high percentage (80% and 90% for the two examples). Furthermore, the objective values under the worst case are reasonable. The CPU times of the RTS algorithm are also stable. Therefore, the performance of the developed RTS algorithm is relatively stable and robust.

7. Conclusions

A container truck transportation problem is investigated in this paper. The problem involves multiple depots with time windows at both origins and destinations, including the reposition of empty containers. It is formulated as a multiple directed graph. Then, the problem is defined as a multi-traveling salesman problem with time windows (m-TSPTW) with multiple depots. The main contribution of the paper to the literature is to extend the existing studies on the container truck transportation problem (e.g., Imai et al., 2007; Cheung et al., 2008; Namboothiri and Erera, 2008) to the problem with multiple depots which is a more realistic problem. A cluster method and a reactive tabu search (RTS) algorithm are developed to solve the m-TSPTW with multiple depots. The two methods are tested using various randomly generated examples. The obtained results are compared with the optimum solutions obtained from the mixed integer program solved by CPLEX. The results indicate that the cluster method can rapidly solve large size problems in which the solutions are relatively acceptable. The RTS algorithm can find the optimum solutions of small size problems in a considerable amount of time. Furthermore, the developed RTS algorithm is sufficiently robust and stable to the large size problems.

Acknowledgements

The authors are grateful to the careful and constructive reviews from the editor-in-chief and anonymous referees. 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.
Castellani, U., Fusiello, A., Gherardi, R., Murino, V., 2007. Automatic selection of mrf control parameters by reactive tabu search. Image and Vision Computing 25 (11), 1824–1832.
Chabrier, A., 2006. Vehicle routing problem with elementary shortest path based column generation. Computers & Operations Research 33 (10), 2972–2990.
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.
Da Silva, L.G.W., Fernandes Pereira, R.A., Abbad, J.R., Sanches Mantovani, J.R., 2008. Optimised placement of control and protective devices in electric distribution systems through reactive tabu search algorithm. Electric Power Systems Research 78 (3), 372–381.
Glover, F., 1989. Tabu search: Part I. ORSA Journal on Computing 1 (3), 190–206.
Hsu, C.-I., Hsieh, Y.-P., 2007. Routing, ship size, and sailing frequency decision-making for a maritime hub-and-spoke container network. Mathematical and Computer Modelling 45 (7–8), 899–916.
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.
Nanry, W.P., Wesley Barnes, J., 2000. Solving the pickup and delivery problem with time windows using reactive tabu search. Transportation Research Part B: Methodological 34 (2), 107–121.
Shintani, K., Imai, A., Nishimura, E., Papadimitriou, S., 2007. The container shipping network design problem with empty container repositioning. Transportation Research Part E: Logistics and Transportation Review 43 (1), 39–59.
Steenken, D., Voß, S., Stahlbock, R., 2004. Container terminal operation and operations research – a classification and literature review. OR Spectrum 26 (1), 3–49.
Toth, P., Vigo, D., 2002. The Vehicle Routing Problem. SIAM (Society for Industrial and Applied Mathematics).
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, X., Regan, A.C., 2002. Local truckload pickup and delivery with hard time window constraints. Transportation Research Part B: Methodological 36 (2), 97–112.
Wang, D., Wang, J., Wang, H., Zhang, R., Guo, Z., 2007. Intelligent Optimization Methods. Higher Education Press, Beijing, China.
Yun, W.Y., Choi, Y.S., 1999. A simulation model for container-terminal operation analysis using an object-oriented approach. International Journal of Production Economics 59 (1–3), 221–230.