OR Spectrum (2010) 32:787–808
DOI 10.1007/s00291-010-0193-4

REGULAR ARTICLE

Heuristic-based truck scheduling for inland container transportation

Ruiyou Zhang · Won Young Yun · Herbert Kopfer

Published online: 26 January 2010

Abstract A truck scheduling problem for container transportation in a local area with multiple depots and multiple terminals including containers as a resource for transportation is addressed. Four types of movements of containers as inbound full, outbound full, inbound empty and outbound empty movements as well as the time windows at both the origin and the destination are considered. The total operating time of all trucks in operation is taken as the optimization criterion that has to be minimized. The problem is mathematically modeled based on a preparative graph formulation and falls into an extension of the multiple traveling salesman problem with time windows (m-TSPTW). The window partition based solution method for the m-TSPTW in Wang and Regan (Transp Res Part B: Methodol 36:97–112, 2002) is modified so that its computation time is reduced greatly. The experiments based on a number of randomly generated instances indicate that the modified method is quite fast and the quality of solutions is relatively high for the m-TSPTW. These experiments also demonstrate that our approach is able to generate high-quality results for the equivalent truck scheduling and inland container movement problem in container drayage operations.

R. Zhang
Institute of Systems Engineering, Northeastern University, 110004 Shenyang, China
e-mail: zhangruiyou@ise.neu.edu.cn

W. Y. Yun
Department of Industrial Engineering, Pusan National University, Busan 609735, Korea
e-mail: wonyun@pusan.ac.kr

H. Kopfer (✉)
Faculty of Business Studies and Economics, Chair of Logistics, University of Bremen, 28359 Bremen, Germany
e-mail: kopfer@uni-bremen.de

Keywords Container transportation · Traveling salesman problem (TSP) · Time window · Heuristics · Container drayage operation · Container as a resource in transportation planning

1 Introduction

Container transportation plays an increasingly important role in international logistics (see, e.g., Günther and Kim 2006; Stahlbock and Voß 2008). Freight transportation by containers can be briefly described as follows. First, cargo is packed within a container at a customer's place. Then, the container is transported to a terminal by truck and delivered to another terminal by train or by vessel. Finally, the container is delivered to its receiver by truck and unpacked.

The container movement by truck between a customer's location and a container terminal is also called drayage operation (Macharis and Bontekoning 2004). Usually, the container transportation by truck is necessary in containerized logistics since vessels and trains cannot provide a door-to-door service. Additionally, container drayage operation involves further activities such as packing/unpacking of containers. Drayage operations and especially container truck transportation account for a significant portion of the total transportation cost (Cheung et al. 2008). As a consequence, it is very important to improve the efficiency of container transportation between ports and customers' places. The optimization of such transportation processes leads to the truck scheduling problem in container drayage operation.

There are a number of papers about transportation optimization for container drayage operation. For example, Namboothiri and Erera (2008) studied a management problem of a fleet of trucks that provided transportation service of containers to a port with an appointment-access system. Cheung et al. (2008) built an attribute-decision model to investigate a cross-border drayage problem, using Hong Kong port as an example. Tjokroamidjojo et al. (2006) presented an optimization-based method to study the benefits and costs of sharing load information in advance of the pick up and delivery of containers. Wen and Zhou (2007) developed a genetic algorithm (GA) to solve a container vehicle routing problem (VRP) in a local area from the perspective of terminals. Coslovich et al. (2006) investigated a container drayage operation with the present and future operating costs minimized. See also Namboothiri (2006) and Ileri (2006) for further studies of container drayage operation problems.

In particular, truck routing and scheduling problems in container drayage operation have also been studied. Imai et al. (2007) formulated a container drayage problem as a pickup- and delivery problem (PDP) and solved the problem by using Lagrangian relaxation. Chung et al. (2007) built several mathematical models of container truck transportation problems with different cases, such as the case of single commodity and the case of multiple commodities. Jula et al. (2005) combined the pickup and delivery nodes to formulate the truck transportation problem as a multiple traveling salesman problem with time windows (m-TSPTW).

Most of the existing models did not consider empty containers as a transportation resource required to move freight. Recently, Caris and Janssens (2009) investigated the pre- and end-haulage of intermodal container terminals. Zhang et al. (2009)

considered a truck scheduling problem in which empty containers are considered as a transportation resource. However, Caris and Janssens (2009) and Zhang et al. (2009) assume that there is a single depot or a single terminal, respectively.

In this paper, a truck scheduling problem in container drayage operation with multiple depots and multiple terminals is studied. Empty containers are considered as a transportation resource. We deal with an optimization problem to determine efficient truck schedules satisfying all container transportation demands. The total operating time of all the trucks carrying containers is used as an optimization criterion to be minimized. The problem is graphically formulated and then falls into a multi-depot asymmetric m-TSPTW. We modify the solution method in Wang and Regan (2002) that introduces window partitions and that was inspired by the previous literature (Levin 1971; Appelgren 1971, 1969) to solve the truck scheduling problem. The minimum partitioning width is estimated based on the maximum number of sub loads. The over-constrained and under-constrained sub problems are solved only once, which can reduce the needed computation time greatly. The performance of the proposed method is investigated and compared with a reactive tabu search (RTS) algorithm by computational experiments.

The remaining part of this paper is organized as follows. The truck scheduling problem is described in Sect. 2, formulated as a directed graph in Sect. 3 and then mathematically modeled in Sect. 4. A heuristic based on the window partition method to obtain near optimal schedules is developed in Sect. 5, tested and compared with existing methods by numerical examples in Sect. 6. The paper is concluded in Sect. 7.

2 Truck scheduling problem

2.1 Problem definition

In this paper, a trucking company with trucks and empty containers is considered. The company handles a number of transportation tasks of containers between terminals and customers' places. We deal with the truck scheduling problem of the trucking company in a local area for a fixed time horizon, typically one day.

It is assumed that there are several depots in which empty containers can be stacked and where the trucks can be parked. Additionally, there are several terminals in the area and the terminals in this paper can be maritime ports or railway hub stations to which trucks transport full and empty containers from customer's places and vice versa.

There are two types of containers, *inbound* and *outbound* containers. The containers located at the terminals that need to be moved (to the depots or their receivers) are called *inbound* containers. Reversely, the containers located at the depots or customers' places that need to be delivered to the terminals are called *outbound* containers. Moreover, each type of containers can be divided into full and empty containers.

Thus, there are four types of containers demanding for transportation tasks that the company should carry out: inbound full (IF), outbound full (OF), inbound empty (IE) and outbound empty (OE) containers. First, an inbound full container is initially located at a terminal. It must be picked up by a truck at the terminal, delivered to its

receiver, dropped off, and unpacked. After an inbound full container is completely handled at its destination, we obtain an empty container and have to move it to a depot or another alternative location by a truck. Secondly, an outbound full container is actually some freight that will be transported in a container and is located at a customer's place. Thus, we should deliver an empty container to the customer's location, pack the freight with the container and deliver it to the specified terminal. Thirdly, an inbound empty container is also initially located at a terminal. It should be picked up at the terminal and transported to a depot or another alternative location. Finally, an outbound empty container means that an empty container should be delivered to the specified terminal.

In general, the inbound/outbound full containers cause two regular transportation tasks in container transportation but the transportation tasks of inbound/outbound empty containers happen because of trade imbalance between hub areas. In an export-dominant area, additional empty containers are needed. That is why empty containers are imported from import-dominant areas thus having a lot of inbound empty containers at such terminals. In the reverse case, we have a lot of outbound empty containers. All transportation tasks considered in this paper are based on the number of containers (loads) and we do not consider less than container load (LCL) freight.

Some transportation tasks may involve several containers. However, these transportation tasks are decomposed into loads so that each load corresponds to a single container.

We use *origin* and *destination* of transportation instead of terminals and customers' places in order to easily describe the problem. The destination of an inbound full load is its receiver and the origin of an outbound full load is its shipper. The origin of inbound full and empty loads is the corresponding terminal. The destination of outbound full and empty loads is also the corresponding terminal. The origins and destinations of the four types of container transportation tasks are summarized in Table 1. The origin of outbound empty loads and the destination of inbound empty loads are not defined by the problem data and their determination is part of the optimization process. Compared to a PDPTW, this means that some transportation requests might change their attributes as a result of the optimization process. For the truck scheduling problem in container drayage operation, not only optimal routes for the vehicles have to be found but also the origin and/or the destination of some transportation requests have to be chosen in a consistent and optimal way. This is one of the reasons why the problem at hand cannot be considered and solved as a usual PDPTW. Additional reasons will be given in Sect. 3.

All the terminals and customers' places (receivers/shippers) have a time window for each load but the depots have no time window. As a result, all inbound/outbound

Table 1 Origins and destinations of transportation types

	IF	OF	IE	OE
Origin	Terminal	Shipper	Terminal	?
Destination	Receiver	Terminal	?	Terminal

full loads have two time windows but inbound/outbound empty loads have only one time window.

A series of activities corresponding to the origin of a load should be started during its origin time window. For an inbound full load, these activities are the picking up of the container at the terminal and the transportation to the receiver. For an outbound full load, these activities are the dropping off of an empty container, packing the freight in it, picking up the packed container at the shipper's location and delivering it to the terminal. For an inbound/outbound empty load, this activity is picking up and dropping off the empty container, respectively. Similarly, another series of activities corresponding to the destination of a load should be started during its destination time window. For an inbound full load, these activities are dropping off the full container, unpacking it, and picking up of the emptied container. For an outbound full load, this activity is dropping off the container at the terminal.

We focus on the truck scheduling problem for a given period and want to determine: (a) where to deliver the empty containers released after inbound full/empty loads, (b) where to pick up the empty containers for outbound full/empty loads, and (c) in which order and by which truck the loads should be carried out. The objective is to minimize the total operating and waiting time at the shippers/receivers' locations of all trucks in operation.

Further assumptions and the parameters are stated in Sects. 2.2 and 2.3, respectively.

2.2 Assumptions

i. The trucking company personnel know all the transportation tasks in advance of the beginning of the time horizon.
ii. The traveling time between any two locations is given as a constant.
iii. All depots have enough empty containers and enough space for empty containers and trucks.
iv. All trucks are identical and all containers are homogeneous, i.e., we only consider 40-foot dry containers. One truck can be used to carry one container at one time.
v. All trucks are initially located at the depots. Each truck should be sent to a depot after all its loads are finished. However, it is unnecessary for a truck to be sent to its original depot.
vi. All containers should be delivered directly to their destinations. In other words, a truck cannot be interrupted while it has begun but not finished a load.

2.3 Parameters

m: number of depots
n: number of loads
K_i: initial number of trucks located at depot $i (i = 1, \ldots, m)$ at the start of the horizon
H_i: location of depot $i (i = 1, \ldots, m)$
$P_i \in \{$'IF', 'OF', 'IE', 'OE'$\}$: type of load $i (i = m + 1, \ldots, m + n)$. Types '$IF$', '$OF$', '$IE$' and '$OE$' represent inbound full, outbound full, inbound empty and outbound empty loads, respectively

O_i: origin location of load $i (i = m + 1, \ldots, m + n)$
D_i: destination location of load $i (i = m + 1, \ldots, m + n)$
$[a_{Oi}, b_{Oi}]$: origin time window of load $i (i = m + 1, \ldots, m + n)$ where a_{Oi} and b_{Oi} are the earliest and latest limits of the window, respectively. $a_{Oi} \leq b_{Oi}$
$[a_{Di}, b_{Di}]$: destination time window of inbound/outbound full load $i (P_i \in \{`IF', `OF'\}).a_{Di} \leq b_{Di}$
t_{Oi}: operation time of the series of activities corresponding to the origin of load $i (P_i \in \{`IF', `OF'\}).t_{Oi} \geq 0$
t_{Di}: operation time of the series of activities corresponding to the destination of load $i (P_i \in \{`IF', `OF'\}).t_{Di} \geq 0$. Furthermore, $a_{Oi} + t_{Oi} \leq b_{Di}$ so that the load i is feasible if $P_i \in \{`IF', `OF'\}$
$t(i, j)$: travel time between locations i and j. The locations can be the origin and destination of loads, or the depots
t: time to pick up or drop off a container

3 Directed graph formulation

3.1 Introduction of the graph

The truck scheduling problem in this paper is different from the classical VRPs and traveling salesman problems (TSPs) (see Toth and Vigo 2002). First, empty containers are a transportation resource in this paper and should be prepared to transport outbound freight in advance. Regarding inbound, we have new empty containers after the inbound full containers are delivered to their receivers. Secondly, the delivery location of the inbound empty containers and the pick-up location of the outbound empty containers are to be determined optimally. On the contrary, all starting and arrival locations in the classical VRPs or TSPs are assumed to be given.

Therefore, a directed graph $G = \{V, A\}$ is introduced to formulate the truck scheduling problem mathematically. The set of nodes is $V = V_D \bigcup V_C$. Here, V_D and V_C are the sets of start/return nodes and load nodes, respectively. $A = \{(i, j) | i \in V_D, j \in V_C;$ or $i \in V_C, j \in V_D \bigcup V_C\}$ is the arc set.

For a uniform formulation, we define the destination of inbound empty loads and the origin of outbound empty loads as the corresponding terminal. That means, for a given inbound or outbound empty load, the origin and the destination are the same.

The graph G is based on activities. All the determinate activities independent of the truck schedule are denoted by the nodes. All the indeterminate activities dependent on the truck schedule are denoted by the arcs. In the graph formulation, the process of solving the considered problem, which means finding the optimal arcs, simultaneously means to solve the transportation problem and to determine the indeterminate activities. Arcs connect one load node to another node, or one start/return node to one load node. However, there is no arc between two start/return nodes.

The detailed definition and attributes of the nodes and arcs are stated in Sects. 3.2 and 3.3, respectively. The graph formulation is modified in Sect. 3.4 and demonstrated by a simple example in Sect. 3.5 finally.

3.2 Nodes

Each start/return node $i \in V_D$ is related to a depot $i\,(i = 1, \ldots, m)$, corresponding to the initial start from and the final return to the corresponding depot. The only attribute of a start/return node is the initial number of trucks that are located at the depot.

A load node $i \in V_C$ means a series of activities of the corresponding transportation type. If $P_i = \text{`}IF\text{'}$ it means picking up the container $i$ at its origin, delivering it to its destination, dropping it off, unpacking, and again picking up the emptied container; if $P_i = \text{`}IE\text{'}$ it means picking up the inbound empty container i; if $P_i = \text{`}OF\text{'}$ it means dropping off the empty container, packing the freight with the container, picking up the packed container, delivering it to its destination, and dropping it off; if $P_i = \text{`}OE\text{'}$ it means dropping off the outbound empty container i.

The reason why we use the same index i for depots $(i = 1, \ldots, m)$ and loads $(i = m + 1, \ldots, m + n)$ is that both a depot and a load are related to a node.

According to the definition of load nodes, two time windows, i.e., the origin and destination time windows, have been imposed on the activities on each inbound/outbound full load node. Therefore, the two time windows of a load node $i\,(P_i \in \{\text{`}IF\text{'}, \text{`}OF\text{'}\})$ should be combined. First, the activities on the node $i\,(P_i \in \{\text{`}IF\text{'}, \text{`}OF\text{'}\})$ should be started during its own time window $[a_{Oi}, b_{Oi}]$ according to their definition. Secondly, these activities must be started before the time $b_{Di} - t_{Oi}$ so that there is enough time for the activities at the origin. Thirdly, these activities should be started after the time $a_{Di} - t_{Oi}$ if possible such that waiting at the destination before the time a_{Di} is avoided. As a result, the time window $[a_i, b_i]$, and service time τ_i, of the node $i \in V_C$, can be obtained as follows.

$$a_i = \begin{cases} \min\left(\max\left(a_{Oi}, a_{Di} - t_{Oi}\right), b_{Oi}\right), & \text{if } P_i \in \{\text{`}IF\text{'}, \text{`}OF\text{'}\} \\ a_{Oi}, & \text{if } P_i \in \{\text{`}IE\text{'}, \text{`}OE\text{'}\} \end{cases} \tag{1}$$

$$b_i = \begin{cases} \min\left(b_{Oi}, b_{Di} - t_{Oi}\right), & \text{if } P_i \in \{\text{`}IF\text{'}, \text{`}OF\text{'}\} \\ b_{Oi}, & \text{if } P_i \in \{\text{`}IE\text{'}, \text{`}OE\text{'}\} \end{cases} \tag{2}$$

$$\tau_i = \begin{cases} \max\left(a_{Di} - b_{Oi}, t_{Oi}\right) + t_{Di}, & \text{if } P_i \in \{\text{`}IF\text{'}, \text{`}OF\text{'}\} \\ t, & \text{if } P_i \in \{\text{`}IE\text{'}, \text{`}OE\text{'}\} \end{cases} \tag{3}$$

3.3 Arcs

An arc $(i, j) \in A$ is defined as the transfer from node i to node j and also consists of a variety of activities. These activities mainly include the movement from the destination of the previous load to the origin of the next load, corresponding to the time formulated in the form of $t(\bullet, \bullet)$. In some cases, these activities also include picking up or dropping off a container, corresponding to the time t. Furthermore, the truck should be sent to a depot to pick up (or drop off) an empty container if both the nodes i and j are outbound loads (or inbound loads), that is, $P_i \in \{\text{`}IF\text{'}, \text{`}IE\text{'}\}$, $P_j \in \{\text{`}IF\text{'}, \text{`}IE\text{'}\}$ or $P_i \in \{\text{`}OF\text{'}, \text{`}OE\text{'}\}$, $P_j \in \{\text{`}OF\text{'}, \text{`}OE\text{'}\}$. Because there are several depots, the depot with the minimum total traveling time is chosen. The amount of time consumed by these activities is called the transfer time of the arc. The transfer time of an arc (i, j), denoted by τ_{ij}, is shown in Table 2.

Table 2 Transfer time of arcs

From node	Transfer time τ_{ij}		
	To node $j \in V_D$	To node $j\,(P_j \in \{\text{'}IF\text{'}, \text{'}IE\text{'}\})$	To node $j\,(P_j \in \{\text{'}OF\text{'}, \text{'}OE\text{'}\})$
$i \in V_D$	–	$t(H_i, O_j)$	$t + t(H_i, O_j)$
$i(P_i \in \{\text{'}IF\text{'}, \text{'}IE\text{'}\})$	$t(D_i, H_j) + t$	$\min_{k=1,\ldots,m} (t(D_i, H_k) + t(H_k, O_j)) + t$	$t(D_i, O_j)$
$i(P_i \in \{\text{'}OF\text{'}, \text{'}OE\text{'}\})$	$t(D_i, H_j)$	$t(D_i, O_j)$	$\min_{k=1,\ldots,m} (t(D_i, H_k) + t(H_k, O_j)) + t$

Notice that the four types of loads are merged into two types during the calculation of the transfer time of arcs, as shown in Table 2. It can be found from Sect. 3.2 that both the first and the last activities of an inbound full load node is *picking up*, which is also the same as the only activity of an inbound empty load node. The previous state of these two types of load nodes is the same and the posterior state of these two types of load nodes is also the same. As a result, the types '*IF*' and '*IE*' can be merged when the transfer time of arcs is considered. Similarly, the types '*OF*' and '*OE*' can also be merged.

According to the definition, we pickup and immediately drop off an empty container at the same location if $P_i = \text{'}IF\text{'}$, $P_j = \text{'}OF\text{'}$, and if D_i is the same location as O_j. These activities as pickup and drop off are redundant. However, this is a very trivial problem since the probability of such case is quite tiny and the time to pick up (or drop off) a container is much shorter than the other operating time and travel time. Moreover, if the definition is changed to avoid this trivial problem, the four types of load nodes could not be merged. Therefore, the formulation of the transfer time of arcs would become much more complex; see Zhang and Yun (2008) for an example.

3.4 A modification of the graph

No time window is imposed on the arcs. Therefore, the activities on an arc (i, j) can be started immediately after all activities on the node i are finished. As a result, the service time of a node $i\,(i \in V_C)$ is transferred to and combined with the transfer time of the arc $(i, j) \in A$. The new transfer time of an arc (i, j) is denoted by T_{ij}. Hereafter, the service time of nodes is discarded.

$$T_{ij} = \begin{cases} \tau_{ij}, & \text{if } i \in V_D, j \in V_C \\ \tau_i + \tau_{ij}, & \text{if } i \in V_C, j \in V_D \bigcup V_C \end{cases} \tag{4}$$

3.5 A simple example

In this subsection, a simple example is introduced to illustrate the graph formulation.

The truck scheduling problem proposed in this paper is formulated as a directed graph. There are m start/return nodes and n load nodes in the graph G. Each start/return node $i \in V_D$ has K_i trucks initially. The activities on each load node $i \in V_C$ should

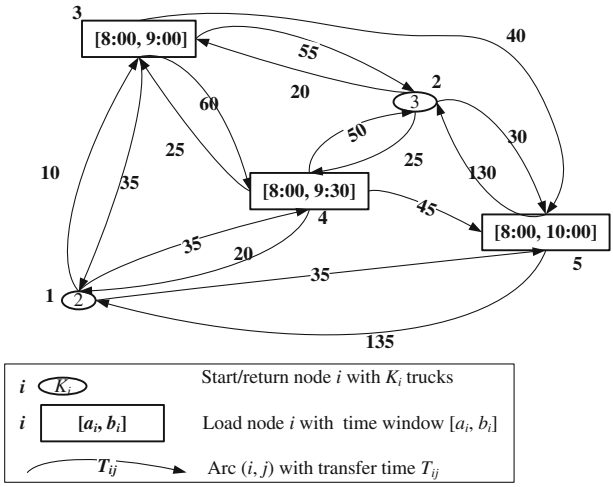

Fig. 1 An example for the graph formulation

be started during the window $[a_i, b_i]$. The transfer time of an arc $(i, j) \in A$ is T_{ij}. Each route starts from a start/return node, visits several load nodes and returns to any start/return node. The objective of the problem is to find a number of routes with minimum total length, so that each load node is visited once on a route of a truck (i.e., each load is performed by a truck) while the maximum number of trucks able to start in a start/return node $i\,(i = 1, \ldots, n)$ is limited by K_i. In the original transportation problem, an inbound full container generates an empty container at the receiver's location. If the next node is another inbound (full or empty) container, then the generated empty container will be delivered to a suitable depot. If the next node is an outbound container, then the generated empty container will be delivered to a terminal (outbound empty) or a shipper's location (outbound full). A simple example of the graph formulation is shown in Fig. 1. The example has two depots, corresponding to the two start/return nodes represented by ellipses in Fig. 1. Initially there are 2 trucks at depot 1 and 3 trucks at depot 2. Three containers need to be transported, corresponding to the three load nodes represented by rectangles in Fig. 1. The activities on the first load node (node 3) should be started between the time 8:00 and 9:00. The transfer time of the arc (3, 5) is 40. There is no arc from node 5 to node 3, which means that the transfer time is too long and hence the arc is infeasible.

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 the first activity on the node $i \in V_C$ is started

The problem can be formulated as the following mathematical model based on the graph G formulated in Sect. 3.

Model O:

$$\min \sum_{i \in V_C} \sum_{j \in V_D} \left(y_i + T_{ij}\right) x_{ij} - \sum_{i \in V_D} \sum_{j \in V_C} \left(y_j - T_{ij}\right) x_{ij} \tag{5}$$

Subject to

$$\sum_{j \in V_C} x_{ij} \leq K_i, \quad \forall i \in V_D \tag{6}$$

$$\sum_{j \in V_D \bigcup V_C} x_{ij} = \sum_{j \in V_D \bigcup V_C} x_{ji} = 1, \quad \forall i \in V_C \tag{7}$$

$$a_i \leq y_i \leq b_i, \quad \forall i \in V_C \tag{8}$$

$$y_i + T_{ij} - y_j \leq \left(1 - x_{ij}\right) M, \quad \forall i \in V_C, \ j \in V_C \tag{9}$$

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

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

In the objective function (5), $(y_i + T_{ij})x_{ij}$ means the time when a truck finally returns to a depot if $x_{ij} = 1 (i \in V_C, j \in V_D)$. $\sum_{i \in V_C} \sum_{j \in V_D} (y_i + T_{ij})x_{ij}$ means the summation of the times when the trucks have returned to the depots. Similarly, $\sum_{i \in V_D} \sum_{j \in V_C} (y_j - T_{ij})x_{ij}$ means the summation of the times when the trucks are initially starting from the depots. Therefore, the objective function (5) is equal to the total operating time of all the trucks in operation.

Constraint (6) means that the number of trucks that are starting from each depot should not exceed the number of the available trucks. Constraint (7) means that each load node should be entered and left exactly once. Constraint (8) is the time window constraint. In constraint (9), $y_i + T_{ij} - y_j \leq 0$ if $x_{ij} = 1$. There is no direct relationship between y_i and y_j if $x_{ij} = 0$. Therefore, constraint (9) updates the start time of load nodes along trips. Notice that the decision variable y_i increases along the route. Therefore, constraint (9) can also eliminate sub tours among load nodes. Additionally, constraints (7) and (9) guarantee that all trucks are initially located at a depot and finally return to a depot. Here, M is a sufficiently big constant. Constraints (10)–(11) state the types of the decision variables.

Using the formulation in this section the truck scheduling problem falls into an extension of the classical m-TSPTW (refer to Bektas 2006). This extension of m-TSPTW has the following features: (a) there are multiple start/return nodes in the m-TSPTW; (b) the trucks do not need to return to their initial start/return nodes; (c) the problem is asymmetric since $T_{ij} \neq T_{ji}$ for any two nodes i and j; and (d) the objective function differs from the objective functions of the classical m-TSPTW.

5 Solution method based on window partition

The truck scheduling problem in the previous section is NP-hard since it is an extension of the NP-hard problem m-TSPTW. A series of solution methods have been developed to solve some variants of VRPs (e.g., Pisinger and Ropke 2007). Even though the TSPTW is a special case of VRP with time windows (VRPTW), the best approach to the VRPTW is not always suitable to the TSPTW (refer to Jula et al. 2005).

Meta-heuristics such as GA and tabu search are widely used to solve similar problems. However, the computation time to obtain near-optimum solutions is usually a little long. For example, Jula et al. (2005) developed a hybrid GA method to solve a variant of m-TSPTW. The CPU times to solve the instances with 50 nodes and 100 nodes are 191.9 and 1,876 s, respectively.

In this paper, a window-partition based method (WPB method) inspired by Wang and Regan (2002) is developed to solve the truck scheduling problem. The equivalent model after window partition is formulated in Sect. 5.1. A feasible solution and a lower bound are described in Sects. 5.2 and 5.3, respectively. The implementation of the method is described in detail in Sect. 5.4.

5.1 Window partition

If the time windows of load nodes are partitioned into several parts, an equivalent model can be obtained (Wang and Regan 2002), which is suitable for the application of the WPB method. Model O in Sect. 4 can be reformulated as follows.

Let ω be the set of all the sub loads (partitioned loads). The load number of a sub load i is denoted by $\delta(i)$. The load set V_C in Model O is replaced by the sub load set ω. The time window $[a_i, b_i]$ and decision variables x_{ij} and y_i, which are defined in the set V_C in Model O, are redefined in the set ω. Furthermore, when referring to the transfer time of an arc (i, j), we use $\delta(i)$ or $\delta(j)$ if i or j belongs to the set ω.

The following model after window partition is equivalent to Model O.

Model WP:

$$\min \sum_{i\in\omega}\sum_{j\in V_D}\left(y_i + T_{\delta(i),j}\right)x_{ij} - \sum_{i\in V_D}\sum_{j\in\omega}\left(y_j - T_{i,\delta(j)}\right)x_{ij} \tag{12}$$

Subject to

$$\sum_{j\in\omega} x_{ij} \le K_i, \quad \forall i \in V_D \tag{13}$$

$$\sum_{j\in V_D \bigcup \omega} x_{ij} = \sum_{j\in V_D \bigcup \omega} x_{ji}, \quad \forall i \in \omega \tag{14}$$

$$\sum_{i\in V_D \bigcup \omega}\sum_{\substack{j\in\omega \\ \delta(j)=k}} x_{ij} = 1, \quad \forall k \in V_C \tag{15}$$

$$a_i \le y_i \le b_i, \quad \forall i \in \omega \tag{16}$$

$$y_i + T_{\delta(i),\delta(j)} - y_j \leq \left(1 - x_{ij}\right) M, \quad \forall i \in \omega,\ j \in \omega \tag{17}$$

$$x_{ij} \in \{0, 1\}, \quad \forall (i, j) \in A \tag{18}$$

$$y_i : \text{real variables}, \quad \forall i \in \omega \tag{19}$$

The objective function (12) in Model WP is equal to the objective function (5) in Model O. Constraint (13) is related to (6). Constraints (14) and (15) correspond to constraint (7). Constraint (14) means that an arc should leave a sub load node $i \in \omega$ if and only if an arc enters the sub load node. In other words, a sub load node $i \in \omega$ can neither be the first nor the last node of a route. Constraint (15) means that exact one sub load should be visited for each load. Constraints (16)–(19) are related to (8)–(11). In (18), $A = \{(i, j) | i \in V_D, j \in \omega;$ or $i \in \omega, j \in V_D \bigcup \omega\}$.

5.2 Feasible solution

If we only consider the latest limit of each time window of sub load nodes in Model WP, several feasible arcs are excluded and the problem becomes an over-constrained problem of Model WP (Wang and Regan 2002). A feasible solution of Model WP can be found by solving the following over-constrained model.

Because we only consider the latest limit of each time window, the decision variable $y_i (i \in \omega)$ is discarded and replaced by b_i. Therefore, Model WP becomes the following over-constrained model.

Model OC:

$$\min \sum_{i \in \omega} \sum_{j \in V_D} \left(b_i + T_{\delta(i),j}\right) x_{ij} - \sum_{i \in V_D} \sum_{j \in \omega} \left(b_j - T_{i,\delta(j)}\right) x_{ij} \tag{20}$$

Subject to

$$x_{ij} \left(b_i + T_{\delta(i),\delta(j)} - b_j\right) \leq 0, \quad \forall i \in \omega,\ j \in \omega \tag{21}$$

$$\text{Constraints (13)–(15), (18)} \tag{22}$$

The objective function (20) and constraint (21) are similar to the objective function (12) and constraint (17), respectively. Notice that they are automatically linearized. The constant M is useless hereafter.

5.3 Lower bound

If the earliest limit of the time window of the node $i (i \in \omega)$ and the latest limit of the time window of node $j (j \in \omega)$ are considered for each arc (i, j), several infeasible arcs of Model WP are included. Therefore, the problem becomes under-constrained (Wang and Regan 2002). It is possible to find a lower bound of Model WP by solving the following under-constrained model.

The decision variable $y_i (i \in \omega)$ and hence constraints (16) and (19) of Model WP are discarded. The decision variable $y_i (y_j)$ in the objective function (12) and

constraint (17) are replaced by a_i and b_j for the "*from*" node and "*to*" node of the arc (i, j), respectively. The following model, which is similar to Model OC, can be obtained.

Model UC:

$$\min \sum_{i \in \omega} \sum_{j \in V_D} \left(a_i + T_{\delta(i),j}\right) x_{ij} - \sum_{i \in V_D} \sum_{j \in \omega} \left(b_j - T_{i,\delta(j)}\right) x_{ij} \tag{23}$$

Subject to

$$x_{ij} \left(a_i + T_{\delta(i),\delta(j)} - b_j\right) \leq 0, \quad \forall i \in \omega, \ j \in \omega \tag{24}$$

$$\text{Constraints (13)–(15), (18)} \tag{25}$$

5.4 Implementation

The general idea of the WPB method is as follows. We can find a feasible solution of Model WP by solving Model OC and a lower bound of Model WP by solving Model UC, respectively. The ratio of the lower bound to the objective value of the feasible solution, so called *UC/OC ratio*, can indicate the level of the quality of the obtained solution. Additionally, Model WP is equivalent to Model O. Therefore, we can solve the truck scheduling problem in this paper by solving Model OC and test the solution quality by solving Model UC.

Model O, as well as Model WP, is a mixed integer programming model. However, Model OC and Model UC are pure integer linear programming models. As a result, it is much easier to solve Model OC and Model UC than to directly solve Model O. Additionally, if the width of windows is smaller, the quality of the obtained solution is better. Therefore, the windows of nodes are partitioned into smaller parts before Model OC is solved.

Wang and Regan (2002) implemented the solution method based on iteration. The partitioning width at each time of iteration is shown in Table 3. The windows are partitioned and then Model OC (and Model UC) is solved for several times. However, the result of the previous iteration does not affect the next iteration in the numerical experiment. In other words, iterations may be performed independently of each other and each iteration can be considered separately.

In general, the smaller the partitioning width, the higher the UC/OC ratio, and the better solution we can obtain. On the other hand, if the partitioning width is too small, the number of sub loads is too large and then, the computation time is too long and too much memory is required.

Table 3 Partitioning width in Wang and Regan (2002)

Iteration	1	2	3	4	5	6	7	8	9	10	11	12
Width (h)	2	1.75	1.5	1.25	1.00	0.75	0.6	0.5	0.4	0.3	0.2	0.1

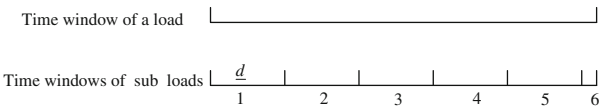

Fig. 2 An illustration of windows partition

Therefore, we modified the solution method in Wang and Regan (2002) as follows. We should estimate firstly the maximum number of sub loads in Model OC and Model UC within a reasonable time. Then, the minimum partitioning width can be roughly calculated from the total window width, i.e., the windows are partitioned only once and Model OC and Model UC are also solved once. As a result, we need much less computation time than the iteration method in Wang and Regan (2002). Let the maximum number of sub loads be $\overline{n}$, then the minimum partitioning width is as follows.

$$\underline{d} = \frac{1}{\overline{n}} \sum_{i \in V_C} (b_i - a_i) \tag{26}$$

Figure 2 illustrates how the windows are partitioned. The time window of the given load in the figure is partitioned into six parts based on the minimum partitioning width $\underline{d}$. Correspondingly, the load is replaced by the six sub loads. The time windows of all the loads are partitioned based on $\underline{d}$ one by one.

6 Numerical experiments and results

In this section, the performance of the proposed method is tested and its performance is compared with existing methods by numerical experiments. Even though it is very important for truck transportation problems in an inland area, relatively little research has been carried out on the m-TSPTW (Jula et al. 2005). Especially, it is not easy to find suitable benchmark instances about the m-TSPTW in the existing literature. Furthermore, the mathematical model built in this paper differs from the classical m-TSPTW models. Therefore, we generate a number of instances in order to test and compare the performance of the developed solution method with existing methods.

The generation of instances is briefly introduced in Sect. 6.1. The fundamental features of the solution method, including the relationship between the partitioning width and the performance are analyzed in Sect. 6.2. Further experiments and analyses about the problem are presented in Sect. 6.3. Finally the comparison with a tabu search approach is presented in Sect. 6.4.

6.1 Generation of instances

A Matlab function is developed in order to generate truck scheduling instances using Excel and Matlab R2006b by the Mathworks Inc. First, we set the values of the model parameters shown in Table 4. Then, a truck scheduling instance represented by the

Table 4 Parameters in the generation of instances

Parameter	Meaning of the parameter
num_D	Number of depots
num_T	Number of terminals
num_C	Number of customers (shippers and receivers)
num_IF	Number of inbound full loads
num_OF	Number of outbound full loads
num_IE	Number of inbound empty loads
num_OE	Number of outbound empty loads
width	Width of the pick up and delivery time windows

graph model is randomly generated and saved in an Excel file. The time windows are partitioned based on a developed macro in Excel file.

The generation of instances is briefly introduced. First of all, the locations of the depots, terminals and customers are generated uniformly in the Euclidean plane with a length and width equal to 180 minutes' distance by truck.

The loading/unloading time of a container is assumed to be 5 min (Chung et al. 2007). The pack/unpack time of an inbound/outbound container is generated uniformly in the range from 5 to 60 min. The traveling time between any two locations can be calculated by the distances in the time plane. We generate the lower bound of the pick up time window of each load uniformly in the range from 0 (8:00 a.m.) to 240 min and the upper bound according to the width. The delivery time window of each load is similarly generated but the lower bound should be reasonable according to the corresponding pick up window and the origin time.

6.2 Analysis of model parameters

6.2.1 Analysis of UC/OC ratio

In this subsection, the meaning of the UC/OC ratio is illustrated by a simple example. A small-sized instance (*Instance 1*) with 2 depots, 3 terminals, and 10 loads (containers) is generated using Matlab. The partitioning width is set to 5 min. We solve the Model OC and Model UC using Lingo 11.0 by LINGO Systems Inc on a personal computer with Pentium® double CPU (3.40 and 3.39 GHz) and 0.99 GB memory. The generator memory limit is set as 256 MB. Also, we directly solve the instance (Model O) using a RTS algorithm. The RTS algorithm (refer Sect. 6.3.2) is coded in Matlab R2006b.

The following result is obtained:

- For Model OC, the objective value = 2,438 min, the computation time = 1.87 s
- For Model UC, the lower bound = 2,388 min and the computation time = 1.97 s
- The UC/OC ratio is 0.979 (2,388/2,438).
- For the RTS algorithm, the best objective value is 2,431 min.

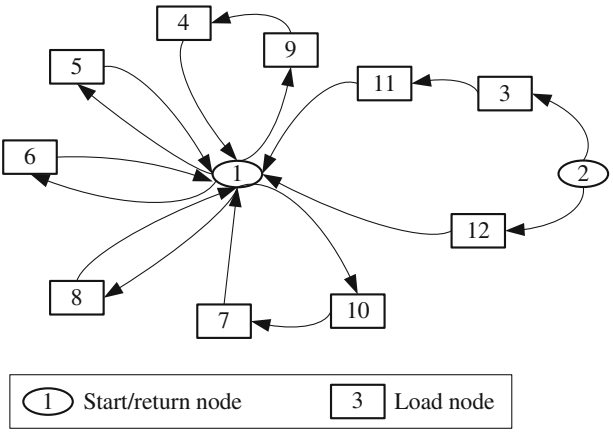

Fig. 3 The best solution of Instance 1

– In this case, the feasible solution obtained from the WPB method is the same as the best solution obtained from the RTS algorithm. The solution is shown in Fig. 3.

The numerical result can be analyzed as follows. On one hand, the lower bound in Model UC is less than (or equal to) the optimum value of the original problem (Model O) according to its definition. On the other hand, the objective value in Model OC is larger than (or equal to) the one in Model O. For example, the objective value of Model OC (2,438) is larger than that obtained from the RTS algorithm (2,431) in Instance 1. An efficient ratio of heuristics in performance is normally defined as the exact optimum value divided by the obtained objective value. Therefore, the following relations can be obtained

$$\begin{aligned} \text{efficient ratio} &= \frac{\text{optimum value of Model O}}{\text{obtained objective value of Model O}} \\ &\geq \frac{\text{optimum value of Model O}}{\text{optimum value of Model OC}} \\ &\geq \frac{\text{optimum value of Model UC}}{\text{optimum value of Model OC}} = \text{UC/OC ratio} \end{aligned} \tag{27}$$

Inequality (27) indicates that the efficient ratio is larger than (or equal to) the UC/OC ratio that is used to evaluate the solution quality of the WPB method. Thus, because we use the UC/OC ratio, the performance of the proposed method is underestimated.

6.2.2 Effect of the partitioning width

In this subsection, we investigate the effect of the partitioning width to the solution quality of the proposed algorithm. Two instances, *Instances* 2 and 3 with 30 loads and

50 loads, respectively, are developed. We obtain the solutions of these instances using the WPB method with different partitioning widths. The partitioning widths and the corresponding number of sub loads are shown in Table 5.

The computation times of Model OC for Instances 2 and 3 are shown in Fig. 4. The computation times of Model UC for Instances 2 and 3 are shown in Fig. 5. The figures show that as the partitioning width becomes smaller, the number of sub loads in both instances becomes larger and as the number of sub loads increases, the computation time of Model OC and Model UC also increases. If the number of sub loads is even more increasing, the generator memory limit in Lingo must be set to an even larger number.

The UC/OC ratio is shown in Fig. 6. Figure 6 shows the reasonable result that a smaller partitioning width leads to a better solution. Therefore, we only need to solve the instances once with the minimum partitioning width estimated well in a real world application. The result indicates that it is possible to find solutions of the truck

Table 5 Partitioning width and the number of sub loads used in Instances 2 and 3

Partitioning width (min)	10	9	8	7	6	5	4	3
Number of sub loads in Instance 2	279	311	346	392	457	544	674	894
Number of sub loads in Instance 3	499	557	623	703	822			

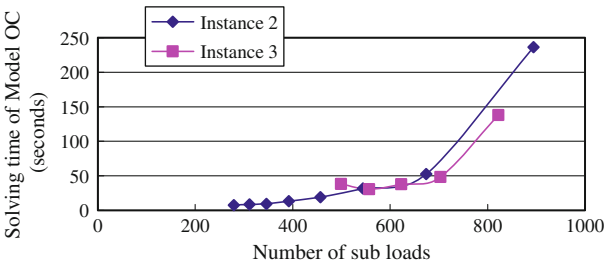

Fig. 4 Computation time of Model OC in Instances 2 and 3

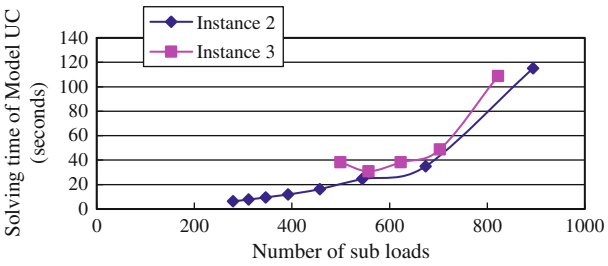

Fig. 5 Computation time of Model UC in Instances 2 and 3

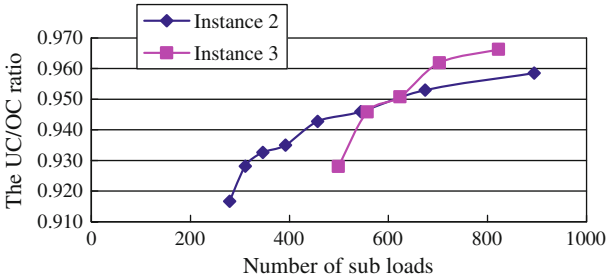

Fig. 6 The UC/OC ratio of Instances 2 and 3

scheduling problems solving the instances with about 800 sub loads within a reasonable time.

6.3 Further experiments and analyses

6.3.1 Analyses about the width of windows

According to Wang and Regan (2002), a sub-fleet of trucks typically consists of less than 20 trucks and is able to handle at most 75 containers in a day. Therefore, 20 instances (Instances from 4 to 23) with such realistic size are generated. All the 20 instances have 5 depots, 3 terminals and 75 loads. The window width used to generate the time windows is ranged from 1 to 4 h. The minimum partitioning width of each instance is estimated assuming that the maximum number of sub loads is about 800. All 20 instances are solved by means of the developed WPB method. The data of these instances and the numerical results are shown in Table 6.

Because the partitioning width is estimated, the actual number of sub loads is about 800.

Table 6 indicates that the WPB method generates the solutions in reasonable time. For all of the instances a feasible solution is found in 3 min. The computation times are less than 2 min in most of the cases. Model UC is also solved in about 2 min. The Model UC is only used to test the quality of the feasible solution and hence does not need to be solved in all cases. Thus, we can find reasonable solutions by the proposed WPB method within 2 min in most cases.

From the values of UC/OC ratio in Table 6, we can evaluate the solution quality of the proposed WPB method. In most cases the UC/OC ratio is greater than 0.95 (from 0.943 to 0.99) and the difference between the optimal objective values and the objective values from the proposed WPB method is almost less than 5%.

Thus, it can be concluded that the proposed method can be used to solve the truck scheduling problems in this paper with reasonable solution quality and computation speed. Additionally, Table 6 indicates that the performance of the WPB method depends on the window width of the loads. We can obtain better solutions if the window widths of loads are smaller. The UC/OC ratio of the five instances with 1 h of

Table 6 Results of 20 realistic-sized instances with WPB method

Window width (h)	Instance	Number of sub loads	Solving time of model OC (min:s)	Solving time of model UC (min:s)	The UC/OC ratio
1	4	719	01:24.4	01:09.9	0.989
	5	844	02:01.4	01:44.9	0.990
	6	756	02:12.6	02:21.5	0.991
	7	771	02:44.3	01:36.9	0.988
	8	810	01:24.8	01:45.4	0.983
2	9	738	01:48.5	01:47.6	0.973
	10	765	01:58.4	01:56.8	0.979
	11	803	02:06.7	01:20.8	0.980
	12	752	01:23.2	01:12.4	0.962
	13	740	01:08.5	01:10.1	0.975
3	14	895	02:30.6	03:15.3	0.960
	15	845	01:52.1	02:04.9	0.953
	16	840	02:20.1	02:30.6	0.943
	17	903	02:44.1	03:19.7	0.980
	18	881	01:57.4	02:03.4	0.974
4	19	824	01:26.9	01:35.3	0.952
	20	852	01:46.9	01:45.9	0.948
	21	804	02:07.0	02:10.8	0.954
	22	831	02:00.6	01:38.0	0.968
	23	895	02:29.3	03:32.2	0.948

window width is more than 0.98. It is greater than the UC/OC ratio (around 0.95) of the five instances with 4 h of window width.

6.3.2 Supersession of containers

It can be seen from the results of the experiments that the inbound and outbound containers usually supersede each other in the route of trucks. This can be found in both, small-sized and large-sized instances.

There are six inbound full containers, three outbound full containers and one outbound empty container in Instance 1 in Sect. 6.2.1.The start/return nodes are numbered from 1 to 2 in Fig. 3. Therefore, the load nodes numbered from 3 to 8 correspond to inbound containers, while the load nodes numbered from 9 to 12 correspond to outbound containers. Inbound containers and outbound containers supersede each other in all the routes in Fig. 3 which consist of two load nodes. For example, a truck initially located at depot 2 visits load node 3 (inbound full) and then load node 11 (outbound full), and returns to depot 1. The empty container generated at the receiver of load 3 is directly delivered to the shipper of load 11.

The detailed routes of the large-sized instances cannot be listed one by one as in Fig. 3 because of limited space. Five typical routes in the result of Instance 23 are

Table 7 Typical routes in the result of Instance 23

Number	Route
1	①-7(I)-③
2	①-56(O)-77(I)-④
3	①-17(I)-67(O)-19(I)-④
4	②-79(I)-52(O)-24(I)-75(O)-②
5	④-33(I)-61(O)-76(I)-64(O)-22(I)-①

shown in Table 7. There are five depots in Instance 23, denoted by the circled number from 1 to 5. There are 40 inbound full containers numbered from 6 to 45, there are 30 outbound full containers numbered from 46 to 75, and there are 5 inbound empty containers numbered from 76 to 80. The letters I and O in the brackets represent inbound containers and outbound containers, respectively.

The above result is quite natural because the inbound container followed by an outbound container, and vice versa, could result in much save in transportation time.

6.4 Comparison with other meta-heuristics

So far, we have analyzed the performance of the proposed WPB method by numerical examples. Now we compare the performance of the proposed method with that of existing methods. Many meta-heuristics such as tabu search and GA have also been widely used to solve such NP-hard problems as the TSPTW. Since it is not possible to compare the proposed method with all existing methods, we should select the best known method for a comparison. But there are only few research results studying optimization problems which are closely similar to the proposed problem of Sect. 4. Zhang et al. (2009) developed a RTS algorithm to solve the m-TSPTW. RTS, proposed by Battiti and Tecchiolli (1994), is an improved version of tabu search and has a mechanism to automatically take balance between intensification and diversification. Therefore, the performance of the WPB method is compared with the RTS algorithm of Zhang et al. (2009).

We select eight typical instances (Instances 7, 8, 12, 13, 17, 18, 22 and 23) from the 20 instances in Sect. 6.3.2. These eight instances have different window widths. We solve the eight instances using the RTS algorithm in Zhang et al. (2009) setting the maximum size of neighborhood to 100. The maximum iteration number is set to 1,500. All other parameters in the RTS algorithm are the same as in Zhang et al. (2009). The computation time and objective values of the two methods are shown in Figs. 7 and 8, together. The computation times of the WPB method in Fig. 8 are the computation times of Model OC.

Figure 8 indicates that the objective values found by the WPB method and the RTS algorithm are almost equal. For the instances 8, 12, 13, 17, 18 and 23, the WPB method obtains slightly better results. Additionally, Fig. 7 shows that the computation times of the WPB method are much shorter than those of the RTS algorithm. The direct comparison of the run-times of the two different approaches is admittedly difficult to

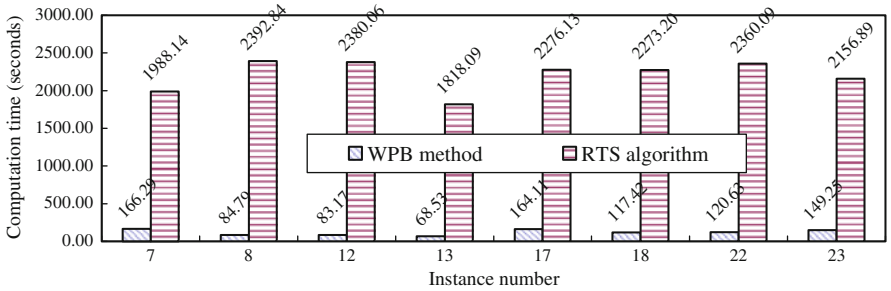

Fig. 7 Computation time of the RTS algorithm and WPB method

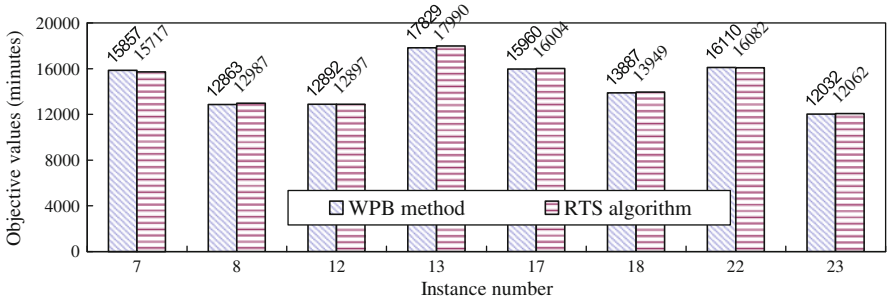

Fig. 8 Objective values of the RTS algorithm and WPB method

draw since the RTS approach is coded in Matlab and Matlab is much slower than an identical RTS algorithm written in C would be. But, as a total result, the proposed WPB method has proved its worth and can successfully be used to solve the complex and important truck scheduling problem in container drayage operation.

7 Conclusions

In this paper, we investigated a container transportation problem with truck deployment in a local area with shippers, receivers, multiple depots, and multiple terminals. Double-side time windows of loads at both the origin and the destination are considered. Empty containers are resources required for transportation and they are to be delivered to customer's locations or to one of the terminals. The optimization criterion minimized by the truck scheduling problem is the total operating time of all the trucks in operation. First, we built a mathematical model based on a preparative graph formulation. The optimization problem is an extension of the classical m-TSPTW. The method in Wang and Regan (2002) is modified, and a WPB method is proposed for solving the truck scheduling problem in this paper. The proposed method has been tested with randomly generated instances and has been compared with a RTS algorithm. The numerical results show that the presented method renders good quality solutions within reasonable computation times and that its total performance is better than that of the existing RTS algorithm.

For further studies, the proposed WPB methods can be applied to other optimization problems in container transportation area, for example to truck scheduling problems with non-homogeneous containers and trucks or to multimodal transportation problems for trains and ships including time windows.

Acknowledgments This study was partially supported by the Korean-German international symposium program of KOSEF in Korea and DFG in Germany, the Fundamental Research Funds for the Central Universities in China, and the National Natural Science Foundation of China (Nos. 70931001, 60821063, 70771021).

References

Appelgren LH (1969) A column generation approach for a ship scheduling problem. Transp Sci 3:53–68
Appelgren LH (1971) Integer programming methods for a vessel scheduling problem. Transp Sci 5:62–74
Battiti R, Tecchiolli G (1994) The reactive tabu search. ORSA J Comput 6:126–140
Bektas T (2006) The multiple traveling salesman problem: an overview of formulations and solution procedures. Omega 34:209–219
Caris A, Janssens GK (2009) A local search heuristic for the pre- and end- haulage of intermodal container terminals. Comput Oper Res 36:2763–2772
Cheung RK, Shi N, Powell WB, Simao HP (2008) An attribute-decision model for cross-border drayage problem. Transp Res Part E: Logist Transp Rev 44:217–234
Chung KH, Ko CS, Shin JY, Hwang H, Kim KH (2007) Development of mathematical models for the container road transportation in Korean trucking industries. Comput Ind Eng 53:252–262
Coslovich L, Pesenti R, Ukovich W (2006) Minimizing fleet operating costs for a container transportation company. Eur J Oper Res 171:776–786
Günther H-O, Kim K-H (2006) Container terminals and terminal operations. OR Spectr 28:437–445
Ileri Y (2006) Drayage optimization in truck/rail networks. PhD thesis, Georgia Institute of Technology
Imai A, Nishimura E, Current J (2007) A Lagrangian relaxation-based heuristic for the vehicle routing with full container load. Eur J Oper Res 176:87–105
Jula H, Dessouky M, Ioannou P, Chassiakos A (2005) Container movement by trucks in metropolitan networks: modeling and optimization. Transp Res Part E: Logist Transp Rev 41:235–259
Levin A (1971) Scheduling and fleet routing models for transportation systems. Transp Sci 5:232–255
Macharis C, Bontekoning YM (2004) Opportunities for OR in intermodal freight transport research: a review. Eur J Oper Res 153:400–416
Namboothiri R (2006) Planning container drayage operations at congested seaports. PhD thesis, Georgia Institute of Technology
Namboothiri R, Erera AL (2008) Planning local container drayage operations given a port access appointment system. Transp Res Part E: Logist Transp Rev 44:185–202
Pisinger D, Ropke S (2007) A general heuristic for vehicle routing problems. Comput Oper Res 34:2403–2435
Stahlbock R, Voß S (2008) Operations research at container terminals: a literature update. OR Spectr 30:1–52
Tjokroamidjojo D, Kutanoglu E, Taylor GD (2006) Quantifying the value of advance load information in truckload trucking. Transp Res Part E 42:340–357
Toth P, Vigo D (2002) The vehicle routing problem. SIAM (Society for Industrial and Applied Mathematics), Philadelphia
Wang X, Regan AC (2002) Local truckload pickup and delivery with hard time window constraints. Transp Res Part B: Methodol 36:97–112
Wen S, Zhou P (2007) A container vehicle routing model with variable traveling time. In: IEEE international conference on automation and logistics, 2007, pp 2243–2247
Zhang RY, Yun WY (2008) An optimum route modeling for container truck transportation. In: Asia conference on intelligent manufacturing & logistics systems, Kitakyushu, Japan, pp 356–362
Zhang RY, Yun WY, Moon I (2009) A reactive tabu search algorithm for the multi-depot container truck transportation problem. Transp Res Part E 45:904–914