Transportation Research Part E 118 (2018) 225–239

Contents lists available at ScienceDirect

Transportation Research Part E

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

Range-based truck-state transition modeling method for foldable container drayage services

Ruiyou Zhang[a], Haishu Zhao[a], Ilkyeong Moon[b,*]

[a] College of Information Science and Engineering, Northeastern University, Shenyang 110819, China
[b] Department of Industrial Engineering and Institute for Industrial Systems Innovation, Seoul National University, Seoul 08826, Republic of Korea

ARTICLE INFO

Keywords:
Foldable container
Container transportation
Truck state transition
Reactive tabu search

ABSTRACT

Manufacturing technologies of foldable containers have almost matured. The use of foldable containers might save transportation costs; however, incorporating them into drayage services also creates great challenges. The foldable container drayage (FCD) problem is formulated as a sequence-dependent multiple traveling salesman problem with time windows using a range-based truck state transition method. An improved reactive tabu search algorithm is designed and validated to solve the FCD problem. The methodology is evaluated extensively on the basis of randomly generated instances. Compared to the use of standard containers, the use of four-in-one foldable containers can save approximately 10% on transportation costs.

1. Introduction

Container drayage is an important aspect of global container logistics. It refers to the pre- and end-haulage among initial shippers, final receivers, and container terminals, and it is typically carried out by trucks (Caris and Janssens, 2009). Drayage is necessary in the chain of container transportation because door-to-door services can be offered with it. However, other means of transportation, for example, vessels that sail between ports (Akyuz and Lee, 2016; Wang et al., 2016a), usually cannot provide direct delivery to end users. The transportation cost per container in drayage is relatively high despite the relatively short drayage distance (Cheung et al., 2008). Furthermore, drayage creates the main sources of road congestions, shipment delays, and emissions of greenhouse gases and particles (Wang et al., 2016b).

Recently, manufacturing technologies of foldable containers have become mature. Companies and research institutes such as Dutch Holland Container Innovations, American Staxxon, Korea Railroad Research Institute and Cargoshell have designed and produced variants of foldable containers. The typical length of these containers is 20 or 40 feet. The height of a folded container is either one-quarter or one-sixth the height when it is unfolded. Correspondingly, such foldable containers are called four-in-one or six-in-one containers. Some foldable containers, including 4FOLD containers, have been put into the commercial operations of shipping companies such as Samudera Indonesia and the Transworld Group.

Drayage services for containers have been investigated relatively extensively. However, the literature regarding container drayage has focused on standard (i.e., unfoldable) containers. The few published articles on foldable containers mainly offer discussions on general transportation (e.g., by vessels) or the repositioning of foldable containers. Section 2 offers a brief literature review on container drayage services and an exhaustive survey on the transportation and repositioning of foldable containers. However, to the best of our knowledge, the routing and scheduling of drayage services using foldable containers have not been publicly reported.

* Corresponding author.
E-mail addresses: zhangruiyou@ise.neu.edu.cn (R. Zhang), ikmoon@snu.ac.kr (I. Moon).

https://doi.org/10.1016/j.tre.2018.07.011
Received 13 March 2018; Received in revised form 11 July 2018; Accepted 31 July 2018

Therefore, in this research, we addressed a foldable container drayage (FCD) problem.

Introducing foldable containers into drayage services creates great challenges. In drayage services, the transportation of loaded containers, referred to as the *implementation of orders*, and the repositioning of empty containers must be considered simultaneously. This dual process is also a basic feature of drayage that is different from other means of container transportation. In addition, compared to the scenario in which standard containers are used, a truck in the FCD problem might carry several empty containers at a time. The number of empty containers carried by a truck needs to be decided. Moreover, both collection and distribution orders are dictated by time window constraints as determined by customers. The handling of an emptied container after a distribution order is delivered to a receiver and the source of the empty container for a collection order must be optimized simultaneously with the scheduling of trucks; this effort adds complexity to the model, making it more difficult to solve. To fully utilize resources, truck routes need to be well organized. We propose a range-based truck state transition (RTST) method to model the FCD problem and design an improved reactive tabu search (IRTS) algorithm to solve the problem.

The contributions made by this research can be summarized as follows. First, an FCD problem for which collection and distribution orders are considered with time windows explains the handling of the transportation of loaded foldable containers and the repositioning of empty containers, and it minimizes the total work time, including the waiting time of all involved trucks. Second, we propose an RTST method to model the range-based state of trucks and the coupled activities between the transportation of loaded containers and the repositioning of empty containers. Based on the RTST method, the FCD problem is formulated as an extension of the sequence-dependent multiple traveling salesman problem (SDMTSP) with time windows. Third, we design and validate an IRTS algorithm to solve the FCD problem. We evaluate the modeling and solution method on the basis of many randomly generated instances and offer several concluding remarks.

The remainder of this paper is organized as follows. Section 2 reviews two streams of literature related to container drayage and foldable containers. In Section 3, we formally define the FCD problem. Sections 4 and 5 show the formulation of the FCD problem using the RTST method and the number of empty containers assigned to a truck in the RTST method, respectively. Sections 6 and 7 present the description and evaluation of the IRTS algorithm, respectively. Finally, we conclude this paper in Section 8.

2. Literature review

In this section, we present a survey of the literature related to FCD services. In addition to an extensive literature review on the transportation and repositioning of foldable containers, we also present a brief survey on the research addressing container drayage.

2.1. Container drayage

Container types are seldom mentioned in the research of container drayage. Most articles in this field describe 40-foot dry standard containers because they are the container type that is most commonly used. Some articles do not account for container types but are based on the assumption that one truck can carry exactly one container; in other words, the researchers considered only truckload requirements. Hence, the formulations of container drayage problems under these scenarios are almost identical.

For example, Wang and Regan (2002) modeled a local truckload pickup and delivery problem as a multiple traveling salesman problem with time windows (m-TSPTW) and presented an iterative solution method for which time window partitions were used. Jula et al. (2005) formulated a similar problem in which work shifts of drivers were limited as an m-TSPTW with social constraints, and they designed a two-phase exact algorithm for which dynamic programming is used to solve it. Similarly, Caris and Janssens (2009) presented a full-truckload pickup and delivery problem with time windows and a local search heuristic to solve the pre- and end-haulage container transportation. Using the Hong Kong area as an example, Cheung et al. (2008) established an attribute-decision model to formulate a cross-border container drayage problem and implemented an adaptive labeling algorithm to solve it.

Research addressing container drayage has been enriched recently in various respects. For instance, Braekers et al. (2013) integrated the transportation of loaded and empty containers as a hierarchical programming model in which the flexible transport of empty containers is defined; that is, either the origin or the destination of a task was not predefined. Zhang et al. (2011) studied a similar problem with a limited number of empty containers available at depots and solved the problem using a reactive tabu search (RTS) algorithm. Xue et al. (2014) investigated a local container drayage problem under a new operation mode in which trailers (i.e., chassis) can be separated from tractors when loading or unloading freight. Ting and Liao (2013) presented a selective pickup and delivery problem, relaxing the constraint that all pickup nodes must be visited. A dynamic version of this problem has also attracted attention. Zhang et al. (2014) studied a container drayage problem with updatable logistic information, formulated it using a determined-activities-on-vertex graph, and solved it using a batch of re-optimization schemes. Escudero et al. (2013) introduced the real-time knowledge of vehicle positions, which was based on geographical information systems, into the daily drayage problem to take into account transportation time uncertainties.

Few articles reported drayage transportation problems with multiple container sizes. Chung et al. (2007) built several models for various container drayage problems. One of the problems studied by Chung et al. (2007) addressed both 20- and 40-foot containers and three types of trucks: 20-foot only, 40-foot only, and vehicles with a combined chassis. Lai et al. (2013) described a similar problem with heterogeneous trucks by assuming that the trucks had capacity to carry one or two containers, which served as the means to create truck types. Vidović et al. (2011) formulated a drayage problem with both 20- and 40-foot containers and a homogeneous fleet of trucks in which each vehicle had the capacity for two 20-foot equivalent units (TEUs) as a multiple matching problem. Similar research can also be found in Vidović et al. (2017).

Zhang et al. (2015) studied a drayage problem with multiple container sizes in which a truck with a combined chassis can carry

one 40-foot container or two 20-foot containers, and they proposed a state-transition-based method to model the problem. Furthermore, Funke and Kopfer (2016) studied a similar problem assuming that containers could be uncoupled from trucks, and they presented a mixed-integer linear programming model with two alternative objective functions: minimization of the total travel distance and minimization of the total operation time. The FCD problem considered in this paper significantly differs from the problem in Zhang et al. (2015) as follows: In the problem of Zhang et al. (2015), a truck might carry one 40-foot or two 20-foot, loaded or empty containers. However, in the FCD problem presented herein, a truck might carry one loaded container or several empty containers. The coupled transportation of loaded and empty containers challenges the modeling of the problem, making it distinct from the original Zhang et al. (2015) problem. Moreover, the number of empty containers that a truck carries in the FCD problem is uncertain unless all the drayage orders for the truck is given. Therefore, we designed a brand new range-based method, which differs from any existing method, to model the FCD problem.

2.2. Transportation and repositioning of foldable containers

Only a few researchers have reported on the transportation and repositioning of foldable containers. Moon et al. (2013) built mathematical models for repositioning empty containers among different ports using vessels carrying both standard and foldable containers. Their experiments indicated that decreasing the production cost of foldable containers plays a key role in increasing the use of foldable containers. Similar research can also be found in Myung and Moon (2014). Moon and Hong (2016) extended the research of Moon et al. (2013) by introducing the installation costs of folding and unfolding facilities for foldable containers and provided a series of sensitivity analyses on, for example, the costs of installing the facilities for folding and unfolding the containers.

Konings and Thijs (2001) analyzed several conditions for a successful application of foldable containers, including the costs for producing, folding, and unfolding them. Using a similar analysis, Konings (2005) verified that using foldable containers might prove profitable for the total transportation chain. Shintani et al. (2010) compared the repositioning costs of empty containers in the hinterland, where standard and foldable containers are used, under different scenarios. Zazgornik et al. (2012a) presented a mixed-integer programming model to formulate the scheduling and routing of vehicles by focusing on the transportation of wood products from forest sites to sawmills using foldable containers; Zazgornik et al. (2012b) presented a similar work.

The only article that addressed foldable containers in drayage services came from Zhang et al. (2018), who simplified the problem by assuming that a truck can carry containers that are either all loaded or containers that are all empty throughout the entire planning horizon. In other words, according to the model of Zhang et al. (2018), a truck can neither deliver a loaded container to a location and put empty containers on the truck for the return trip nor arrive to a location with empty containers and return with full containers. Under this assumption, the problem was decomposed into a loaded container sub-problem and an empty container sub-problem for which a mathematical model was built. The assumption limits the scheduling of trucks significantly such that the study of Zhang et al. (2018) differs from the research presented herein. Section 7.3 presents a description of the experiments we conducted and a comparison of our findings with those of Zhang et al. (2018).

3. Foldable container drayage problem

A trucking company provides drayage services in a local area using containers. For this study, the company possesses a depot for parking trucks and stacking containers. The depot can be visited by the trucks at any time. The containers are homogeneous; they can be folded and unfolded at either the depot or at customer locations. Furthermore, the containers can be folded and unfolded conveniently by workers at any location, including customer sites, without special equipment. The fleet consists of a number of homogeneous trucks, each of which can carry one or several folded containers.

According to the FCD problem, the drayage orders must be delivered as dictated by the planning horizon, which, as is typical, consists of one day for this study. All trucks start from the depot and return to the depot at the end of the delivery period. Drayage orders can be classified into two types: collection and distribution. In a collection order, an empty container needs to be transported to the shipper in the first step of the journey. After the freight is packed into the container, the loaded container is delivered to the depot. In contrast, in a distribution order, a loaded container is picked up at the depot and delivered to the receiver at the first stop. After the freight is unpacked, the empty container is picked up and delivered to the depot or to the shipper of a collection order. During the time that the freight is packed or unpacked, the truck can leave the customer. The truck picking up the loaded (or empty) container at a customer need not be the same truck that delivered the empty (or loaded) container at that location.

All information in the FCD problem was given in advance and remained static during the horizon. We assumed that the containers available were sufficient for completing the orders in the horizon. The times for loading or unloading a container on or off a truck were omitted. Each drayage order had a given time window, during which the activities, including packing and unpacking, must be started and ended.

The optimizing criterion is minimization of the total operating time, including the traveling and waiting times, of all involved trucks. The total operating time of trucks also reflects the total operation costs, at least to a certain extent, and has been widely used as the optimizing criterion in drayage problems (e.g., Sterzik et al., 2015; Zhang et al., 2014).

The following notation is used to describe the FCD problem:

m: number of trucks
$C (\geq 2)$: maximum number of folded containers that a truck can carry
O_C: set of collection orders

O_D: set of distribution orders
$p_o(\geq 0)$: time for unfolding a container and packing the freight, or unpacking the freight and folding a container, of order $o \in O_C \cup O_D$
$[T_o^L, T_o^U](T_o^L \leq T_o^U)$: time window of order $o \in O_C \cup O_D$
l_o: location of the depot if $o = s$ or the corresponding customer location of order o if $o \in O_C \cup O_D$
$t(l_{o1}, l_{o2}) = t(l_{o2}, l_{o1})(\geq 0)$: travel time between locations l_{o1} and l_{o2}

4. The range-based truck state transition modeling method

This section explains the RTST method used to model the FCD problem. The RTST method is introduced because the state of trucks changes after they are used to handle different types of drayage orders. Trucks in different states behave differently even when they are handling an identical type of order. Section 4.1 explains that orders are divided into sub-orders; Section 4.2 presents the definition of truck states. Section 4.3 shows an extensive explanation on the formulation of the transitions of truck states. Finally, Section 4.4 presents a mathematical formulation.

4.1. Division of orders

Each drayage order can be divided into two sub-orders because trucks can leave customers when the freight is being packed or unpacked. Specifically, for a collection order $o \in O_C$, *Phase I sub-order* denotes a truck delivery of an empty container to shipper l_o, and *Phase II sub-order* denotes a truck pick-up of a loaded container at shipper l_o and the delivery of it to the depot. Similarly, for Phases I and II sub-orders of a distribution order $o \in O_D$, a truck delivers the loaded container to receiver l_o and a truck picks up the emptied container at location l_o, respectively. A similar order-division method can also be found in Xue et al. (2014) and elsewhere.

In the extended set of sub-orders shown as

$$O = O_C^1 \cup O_C^2 \cup O_D^1 \cup O_D^2 \cup \{s\}, \tag{1}$$

O_C^1 and O_C^2 are the respective sets of Phase I and Phase II sub-orders of collection orders; O_D^1 and O_D^2 are the respective sets of Phase I and Phase II sub-orders of distribution orders; and s is a virtual sub-order defined as the initial start from and the final return to the depot. A sub-order $i \in O \backslash \{s\}$ is called a *real sub-order*. Any real sub-order can only be handled once, but a virtual sub-order can be handled more than once. The customer of sub-order i is called l_i.

4.2. Definition of truck states

Definition 1. A truck state is defined as $S(l, e^L, e^U)$, where l is the current location of the truck; e^L and e^U are the lower and upper bounds of the number of empty containers that the truck is carrying, respectively, $0 \leq e^L \leq e^U \leq C$.

We introduced truck states to formulate and then solve the FCD problem. A truck state expresses the location and the number of empty containers, reported as a range, that a truck is carrying. After it picks up a loaded container, the truck must deliver the container to the predetermined destination without deviations. As a result, a truck state representing a loaded container is unnecessary.

A truck can carry different numbers of empty containers. Furthermore, in some situations, the specific number of empty containers is not predetermined before a sub-order is handled. Therefore, a truck state does not refer to a specific number, but to a range of empty containers that it might be carrying. Such a definition of truck states can be used to formulate conveniently the transition of truck states in terms of the number of loaded and empty containers the truck is holding.

The initial state of each truck is $S(l_s, 0, C)$. Initially, each truck is parked at the depot. A truck in the initial state can handle a loaded container, for example, a sub-order $i \in O_D^1$. That is, the lower bound of the number of empty containers it carries is zero. Of course, it can also load a number of empty containers and deliver them to receivers; therefore, the upper bound of number of empty containers is C.

4.3. Transition of truck states

For a current truck state $S = S(l, e^L, e^U)$, and a sub-order that the truck subsequently handles $i \in O$, the new truck state is

$$f(S(l, e^L, e^U), i) = \begin{cases} S(l_i, 0, C-1), & \forall\, i \in O_C^1 \text{ and } e^U = 0, \\ S(l_i, \max(0, e^L-1), e^U-1), & \forall\, i \in O_C^1 \text{ and } e^U \geqslant 1, \\ S(l_s, 0, C), & \forall\, i \in O_C^2, \\ S(l_i, 0, 0), & \forall\, i \in O_D^1, \\ S(l_i, e^L+1, \min(e^U+1, C)), & \forall\, i \in O_D^2 \text{ and } e^L \leqslant C-1, \\ S(l_i, 1, C), & \forall\, i \in O_D^2 \text{ and } e^L = C, \\ S(l_s, 0, 0), & \text{if } i = s. \end{cases} \tag{2}$$

The state transition time is

$$g(S(l, e^L, e^U), i) = \begin{cases} t(l, l_s) + t(l_s, l_i), & \forall\, i \in O_C^1 \text{ and } e^U = 0, \text{ or } \forall\, i \in O_D^1, \text{ or } \forall\, i \in O_D^2 \text{ and } e^L = C, \\ t(l, l_i), & \forall\, i \in O_C^1 \text{ and } e^U \geqslant 1, \text{ or } \forall\, i \in O_D^2 \text{ and } e^L \leqslant C-1, \\ t(l, l_i) + t(l_i, l_s), & \forall\, i \in O_C^2 \text{ and } e^L = 0, \\ t(l, l_s) + 2t(l_s, l_i), & \forall\, i \in O_C^2 \text{ and } e^L \geqslant 1, \\ t(l, l_s), & \text{if } i = s. \end{cases} \tag{3}$$

First, we divided the transition of states into five cases according to the type of sub-order i. Then, as necessary, the cases were further divided according to e^L or e^U. The case-by-case explanation of the state transition is as follows:

Case 1. $i \in O_C^1$. The truck delivers an empty container to the shipper of sub-order i. If it is not carrying containers, the truck returns to the depot to pick up some containers before continuing.

Case 1-1. $e^U = 0$. The truck is not carrying any container. Therefore, it returns to the depot to pick up at least one and at most C empty containers, and then deliver one container to customer i. As a result, the new truck state is $S(l_i, 0, C-1)$. The transition time is $t(l, l_s) + t(l_s, l_i)$.

Case 1-2. $e^U \geq 1$. The truck directly delivers one empty container to customer i because it is carrying at least one container. Hence, the transition time is $t(l, l_i)$. Both the lower and upper bounds of the number of empty containers that the truck is carrying are decreased by one; however, the bounds cannot be negative. As a consequence, the new truck state is $S(l_i, \max(0, e^L-1), e^U-1)$.

Case 2. $i \in O_C^2$. The truck delivers the loaded container from customer i to the depot. As a result, the new truck state is $S(l_s, 0, C)$.

Case 2-1. $e^L = 0$. The number of empty containers on the truck can be zero. Therefore, the truck directly travels to customer i to pick up the loaded container. The state-transition time is $t(l, l_i) + t(l_i, l_s)$.

Case 2-2. $e^L \geq 1$. The truck is carrying at least one empty container. It delivers the empty container(s) to the depot before traveling to customer i. Therefore, the state-transition time is $t(l, l_s) + t(l_s, l_i) + t(l_i, l_s) = t(l, l_s) + 2t(l_s, l_i)$.

Case 3. $i \in O_D^1$. The truck returns to the depot and drops off the empty containers that it is carrying. Then, it delivers the loaded container from the depot to customer i. The new truck state and the state-transition time are $S(l_i, 0, 0)$ and $t(l, l_s) + t(l_s, l_i)$, respectively.

Case 4. $i \in O_D^2$. The truck picks up an empty container at customer i. The location for delivering the empty container remains undetermined. As a result, the truck stays at customer i.

Case 4-1. $e^L \leq C-1$. The truck may have extra space for an empty container. Therefore, it travels directly to customer i. The lower and upper bounds of the number of empty containers that it is carrying are increased by one; however, there can be at most Ccontainers. As a result, the new truck state is $S(l_i, e^L + 1, \min(e^U + 1, C))$. The state-transition time is $t(l, l_i)$.

Case 4-2. $e^L = C$. The truck is carrying C empty containers and has no extra space. It must return to the depot to drop off several (some or all) empty containers before traveling to customer i. The truck is carrying at least one empty container after it picks up the empty container at customer i. Therefore, the new truck state and state-transition time are $S(l_i, 1, C)$ and $t(l, l_s) + t(l_s, l_i)$, respectively.

Case 5. $i = s$. The truck finishes all real sub-orders assigned to it. It returns to the depot and drops off the empty containers that it carries, if any. As a result, the new truck state and the state-transition time can be described as $S(l_s, 0, 0)$ and $t(l, l_s)$, respectively.

Table 1 presents a summary of the structure of these divisions. Based on the RTST method, the FCD problem can be properly formulated. There are m homogeneous trucks in an identical initial state $S(l_s, 0, C)$. Each truck starts from sub-order s, handles a sequence of real sub-orders, and finally returns to sub-order s. Each real sub-order must be handled once. The activities of real sub-order $i \in O\backslash\{s\}$ at customer l_i must be carried out within the given time window $[T^L_{\theta(i)}, T^U_{\theta(i)}]$, where $\theta(i)$ is the corresponding order of sub-order i. The finishing time of any Phase I sub-order $i \in O_C^1 \cup O_D^1$ must be $p_{\theta(i)}$ earlier than the starting time of $\delta(i)$, where $\delta(i) \in O_C^2 \cup O_D^2$ is the Phase II sub-order corresponding to i. If a truck in state S handles sub-order i, the new state is $f(S, i)$ and the transition time is $g(S, i)$. The objective is to minimize the total duration of trucks in travel.

Table 1
Division of cases for state transition

Case	Type of sub-order$i$	Subcase	e^L or e^U
Case 1	$i \in O_C{}^1$	Case 1–1	$e^U = 0$
		Case 1–2	$e^U \geq 1$
Case 2	$i \in O_C{}^2$	Case 2–1	$e^L = 0$
		Case 2–2	$e^L \geq 1$
Case 3	$i \in O_D{}^1$		
Case 4	$i \in O_D{}^2$	Case 4–1	$e^U \leq C - 1$
		Case 4–2	$e^L = \mathrm{C}$
Case 5	$i = s$		

4.4. Mathematical formulation

Two decision variables were introduced as

$$x_{ij} = \begin{cases} 1, & \text{if a truck handles sub-order } j \in O \text{ after handling a sub-order } i \in O \\ 0, & \text{otherwise} \end{cases}$$

τ_i: time when the handling of real sub-order $i \in O\backslash\{s\}$ is finished

We formulated the FCD problem based on the RTST method as follows:

$$\min \omega = \sum_{i\in O\backslash\{s\}} (\tau_i + g(S_i, s))x_{is} - \sum_{i\in O\backslash\{s\}} (\tau_i - g(S(l_s, 0, C), i))x_{si}, \tag{4}$$

s.t.

$$\sum_{i\in O\backslash\{s\}} x_{si} \le m, \tag{5}$$

$$\sum_{j\in O} x_{ij}=1, \forall\, i \in O\backslash\{s\}, \tag{6}$$

$$\sum_{j\in O} x_{ji}=1, \forall\, i \in O\backslash\{s\}, \tag{7}$$

$$\tau_i + g(S_i, j) - \tau_j \le M(1-x_{ij}), \forall\, i \in O\backslash\{s\}, \forall\, j \in O\backslash\{s\}, i \ne j, \tag{8}$$

$$S_i = \begin{cases} S(l_s, 0, C), & i = s, \\ f(S_{j|\mathbf{x}ji=1}, i), & i \in O\{s\}, \end{cases} \tag{9}$$

$$\tau_i + p_{\theta(i)} - \tau_{\delta(i)} + t(l_i, l_s) \le 0, \forall\, i \in O_C^1, \tag{10}$$

$$\tau_i + p_{\theta(i)} - \tau_{\delta(i)} \le 0, \forall\, i \in O_D^1, \tag{11}$$

$$T^L_{\theta(i)} \le \tau_i \le T^U_{\theta(i)}, \forall\, i \in O_D^1 \cup O_D^2 \cup O_C^1, \tag{12}$$

$$T^L_{\theta(i)} \le \tau_i - t(l_i, l_s) \le T^U_{\theta(i)}, \forall\, i \in O_C^2, \tag{13}$$

$$x_{ij} \in \{0, 1\}, \forall\, i \in O, \forall\, j \in O, \tag{14}$$

$$\tau_i \in \mathbb{R}, \forall\, i \in O\{s\}. \tag{15}$$

In formulation (4)–(15), Objective Function (4) minimizes the total working time, including the traveling and waiting times, of all involved trucks. The term $\sum_{i\in O\backslash\{s\}} (\tau_i + g(S_i, s))x_{is}$ in Objective Function (4) summarizes the time points when trucks return to the depot for the final time; S_i defines a truck state (refer to Constraint (9)), and the function $g(\cdot,\cdot)$ is defined by Eq. (3). Similarly, $\sum_{i\in O\backslash\{s\}} (\tau_i - g(S(l_s, 0, C), i))x_{si}$ summarizes the time points when the trucks initially leave the depot.

Constraints (5)–(9) are similar to those of Zhang et al. (2015). In particular, Constraint (5) limits the number of trucks. Constraints (6) and (7) indicate that each sub-order must be handled exactly once. Constraint (8) denotes the time continuity for sub-orders that are handled continuously by the same truck and eliminates sub-tours for real sub-orders where M is a sufficiently large constant. Constraint (8) is $\tau_i + g(S_i, j) - \tau_j \le 0$ if $x_{ij} = 1$. Constraint (9) defines the truck-state transition, where the function $f(\cdot,\cdot)$ is defined in Eq. (2).

Constraints (10)–(13) refer to finishing times for sub-orders. Specifically, Constraint (10) explains collection orders and Constraint (11) addresses distribution orders. In Constraint (10), $\tau_i + p_{\theta(i)}$ is the time when the packing of a Phase I collection sub-order is finished, and $\tau_{\delta(i)} - t(l_i, l_s)$ is the time for initiating the pickup of the corresponding Phase II collection sub-order. As a result, $\tau_i + p_{\theta(i)} \le \tau_{\delta(i)} - t(l_i, l_s)$, which is Constraint (10). A Phase II distribution sub-order refers to a pickup of only an empty container. Therefore, $\tau_i + p_{\theta(i)} \le \tau_{\delta(i)}$ for $i \in O_D^1$, which is Constraint (11). Constraints (12) and (13) are the time window constraints for orders. Finally, Constraints (14) and (15) limit the type of variables for which $\mathbb{R}$ is the set of real numbers.

As a result, the FCD problem as described differs from the classical multiple traveling salesman problem with time windows (refer to Bektas (2006)) significantly. One major difference is that in the FCD problem, the traveling time between cities depends on the current state of the salesman according to the sequence of cities that the salesman has already visited. Furthermore, this problem differs from the SDMTSP (refer to Zhang et al. (2015)) as follows: Firstly, each city (sub-order) in the FCD problem has a time window constraint; Secondly and more importantly, the cities in the FCD problem consist of two types of sub-orders (Phase I and Phase II). The handling times of each pair of cities (two sub-orders of each order) are coupled with each other, making the problem particularly difficult to solve.

5. The number of empty containers on a truck and an example

The range-based modeling method, RTST, presented in Section 4 is sufficient for solving if we solve the FCD problem using meta-heuristics. In other words, given a sequence of sub-orders handled by a truck, the RTST method can formulate the state transition of the truck; hence, the solution can be evaluated. We discuss, in Section 5.1, the specific number of empty containers that a truck is carrying at any time, which is information required in application. Furthermore, we offer an example in Section 5.2 to explain both the RTST method and the determination process.

5.1. Determination process

The number of empty containers that a truck is carrying might change at either the depot or customer location. The change at customers is determined. For example, if a truck with three empty containers handles a sub-order $i \in O_C^1$, then the number of empty containers that it is carrying definitely becomes two. However, when it leaves the depot, the truck may carry one or more empty containers, a loaded container, or nothing. Therefore, we placed a definition on the depot-leaving case.

Definition 2. A case in which a truck leaves the depot is defined as a depot-leaving case.

Two types of depot-leaving cases are defined. In the first type, the current location of a truck is the depot: $l = s$. The truck leaves the depot before handling the next sub-order. A truck that starts from sub-order s is such a case because all trucks are initially parked at the depot. The follower of Case 2 is also a case of this type, as discussed in Section 4.3. In Case 2, a truck delivers a loaded container to the depot. Therefore, it is parked at the depot. In the second type, a truck is not at the depot, but it returns to the depot and then leaves the depot when handling a sub-order. Cases 1-1, 2-2, 3, and 4-2 are such cases.

Let the complete sequence of sub-orders that a truck handles be Y. Y is divided into segments by the depot-leaving cases. The number of segments is the times that the truck leaves the depot. For each segment, the truck leaves the depot once. In Case 2-2, the truck carries nothing when it leaves the depot. In Case 3, the truck carries a loaded container when it leaves the depot. In other cases, such as the initial case, Cases 1-1, 2-1 and 4-2, the truck may carry a number of empty containers when it leaves the depot. As a result, these cases are called *empty-container-carrying cases*.

Assume that a sub-order segment separated by the depot-leaving cases is

$$Z = (i_1, i_2, \cdots i_n). \tag{16}$$

The number of empty containers that the truck carries when leaving the depot in this segment is

$$e^{Initial} = e^{L\_Last} + n^{C1} - n^{D2}, \tag{17}$$

where e^{L_Last} is the lower bound of the number of empty containers in the truck state after handling sub-order i_n; n^{C1} is the number of such sub-orders in Z that belong to O_C^1; n^{D2} is the number of such sub-orders in Z that belong to O_D^2.

The calculation in Function (17) is suitable not only for empty-container-carrying cases but also for Cases 2-2 and 3. That is, in Cases 2-2 and 3, $e^{Initial} = 0$.

5.2. An example

We present an example here to illustrate the RTST method and the determination process. Set that $C = 2$ and a truck handles 8 real sub-orders:

$$Y = (i_{1C1}, i_{2C1}, i_{3C1}, i_{4C2}, i_{5D1}, i_{6D2}, i_{7D2}, i_{8D2}), \tag{18}$$

where the subscripts refer to the type of sub-orders. For example, i_{1C1} refers to the Phase I sub-order of a collection order. Initially, a truck can handle either a loaded container or an empty container; the number range of empty containers is $[0, C] = [0, 2]$. Therefore, the initial truck state is $S(l_s, 0, 2)$. If $i_1 \in O_C^1$, then Case 1-2 describes the situation because $e^U = 2 \geq 1$. That is, if the truck initially carries a container, it can directly deliver an empty container to l_{i_1}. The truck state is $S(l_{i_1}, 0, 1)$. If $i_2 \in O_C^1$, then Case 1-2 describes the situation because $e^U = 1 \geq 1$. The truck can directly deliver an empty container from l_{i_1} to l_{i_2}. Thus, the truck carries two empty containers when it leaves the depot. The truck state is $S(l_{i_2}, 0, 0)$. If $i_3 \in O_C^1$ and $e^U = 0$, then Case 1-1 describes the situation, which is a depot-leaving case. The truck must return to the depot before handling sub-order i_3. Hence, the first segment is (i_1, i_2). In this segment, $e^{L\_Last} = 0$, $n^{C1} = 2$, and $n^{D2} = 0$. Therefore, $e^{Initial} = 0 + 2 - 0 = 2$.

The truck returns to the depot to pick up empty containers before handling i_3. However, the number of empty containers that it carries is unknown. It can pick up one or two empty containers. As a result, the truck state is $S(l_{i_3}, 0, 1)$ after handling sub-order i_3. If $i_4 \in O_C^2$ and $e^L = 0$, then Case 2-1 describes the situation. The truck carries no empty container. Therefore, it directly travels to l_{i_4} to pick up the loaded container and delivers it back to the depot. The state is $S(l_s, 0, 2)$. Now, it is known that the truck carries one empty container when it handles sub-order i_3. This is a depot-leaving case; hence, the segment is (i_3, i_4). In (i_3, i_4), $e^{L\_Last} = 0$, $n^{C1} = 1$, and $n^{D2} = 0$; hence, $e^{Initial} = 0 + 1 - 0 = 1$.

Cases 3, 4-1, 4-1, and 4-2 reflect handling of sub-orders i_5 to i_8; and the new state for each sub-order is $S(l_{i_5}, 0, 0)$, $S(l_{i_6}, 1, 1)$, $S(l_{i_7}, 2, 2)$, and $S(l_{i_8}, 1, 2)$. The detailed description of the state transition is omitted here because of the limited space. Overall, the sequence Y is divided into four segments as

$$\underline{i_1}, \underline{i_2}, i_3, i_4, \underline{i_5}, \underline{i_6}, \underline{i_7}, \underline{i_8}. \tag{19}$$

For segment (j_5, j_6, j_7), the truck carries a loaded container when it leaves the depot. For segment (i_8), the truck carries 0(=1 + 0−1) empty container.

6. The improved reactive tabu search algorithm

The FCD problem with the RTST-based formulation cannot be solved using optimization software such as CPLEX because of the sequence-dependent feature. As a consequence, we designed a meta-heuristic algorithm, IRTS, to solve this problem.

Glover (1989) proposed a tabu search (TS) algorithm to solve hard problems. Battiti and Tecchiolli (1994) introduced a reactive mechanism into TS, resulting in a reactive tabu search (RTS) algorithm. Unlike the TS, the main features of an RTS algorithm are that the length of the tabu list can be adjusted automatically and that an escape mechanism prevents the search from falling into local optimum points. The RTS algorithm has been successfully applied to solve a large family of hard problems including the traveling salesman problem and variants of it: for example, Nanry and Barnes (2000), Wassan et al. (2008), Wang and Wang (2009), Zhang et al. (2009), Zhang et al. (2015), and Shiri and Huynh (2016). Recently, Zhang et al. (2015) improved the RTS algorithm slightly. In the RTS algorithm of Zhang et al. (2015), the length of the tabu list is increased or decreased by one when necessary but not multiplied by a coefficient.

The following parameters are introduced in the IRTS algorithm:

n_{ITER}: maximum number of iterations
n_{NBHD}: maximum size of a neighborhood
n_{TABU}: initial length of the tabu list
n_{NOR}: maximum number of iterations without finding repeated solutions
n_{ESC}: number of repeated solutions for triggering the escape

6.1. Encoding and decoding

The IRTS algorithm uses an order-encoding scheme with separators. A solution for the FCD problem can be formulated as a permutation of $|O|-1$ sub-orders with $m-1$ separators, where $|\cdot|$ is the size of the set. The separators split a solution into m sub-order sequences. Each sub-order sequence denotes the sub-orders that are handled by a truck in the order. A solution of the IRTS algorithm consists of no information about the handling time for sub-orders. We determined the sub-order handling time when checking the feasibility of solutions.

For example, a valid solution of an FCD instance with three trucks and four orders can be encoded as

$$X_1 = 4, \delta(3), \delta(4), 0, 1, 2, \delta(2), 0, 3, \delta(1), \tag{20}$$

where 0 is the separator. The separators split X_1 into three sub-order sequences as $Y_1 = (4, \delta(3), \delta(4))$, $Y_2 = (1, 2, \delta(2))$, and $Y_3 = (3, \delta(1))$. For instance, the sequence Y_1 denotes that a truck handles the Phase I sub-order of Order 4, and then handles the Phase II sub-orders of Orders 3 and 4.

6.2. The initial solution

The initial solution of the IRTS algorithm is generated greedily. To generate an initial solution, sub-orders are assigned to trucks in series. Initially, only Phase I sub-orders are possible candidate sub-orders. Once a Phase I sub-order is assigned, the corresponding Phase II sub-order becomes a possible candidate. The sub-order with the earliest time window among possible candidates is assigned to the current truck. If no sub-order is feasible for the truck, sub-orders are assigned to a new truck in a similar manner. Such a procedure continues until all sub-orders are assigned.

The step-by-step formulation of this procedure is as follows.

Step 1: Let $m^U = 0$, $\Omega = O_C^1 \cup O_D^1$, and $X = \varnothing$, where $\varnothing$ denotes an empty list.

Step 2: Let

$$i^C = \underset{i \in \Omega}{\operatorname{argmin}} T^L_{\delta(i)}, \tag{21}$$

let $\Omega = \Omega \backslash \{i^C\}$, $X = X \oplus i^C$, where $\oplus$ denotes appending an item to a list. Furthermore, let $j = i^C$ and $\tau_j = T^L_{\delta(j)}$, where τ_j is the finishing time of sub-order j. If i^C belongs to $O_C^1 \cup O_D^1$, let $\Omega = \Omega \cup \{\delta(i^C)\}$.

Step 3: If Ω is not empty, go to Step 4. Otherwise, the assignment is finished, let $m^U = m^U + 1$ and go to Step 7.

Step 4: Let the set of sub-orders that can be appended to X be

$$\Omega' = \left\{ i \mid i \in \Omega,\ T^U_{\delta(i)} \geq \begin{cases} \tau_j + g(S_j, i), & \text{if } i \in O_C^1 \cup O_D^1 \\ \max(\tau_j + g(S_j, i) - t(l_i, l_s),\ \tau_{\delta(i)} + p_{\theta(i)}), & \text{if } i \in O_C^2 \\ \max(\tau_j + g(S_j, i),\ \tau_{\delta(i)} + p_{\theta(i)}), & \text{if } i \in O_D^2 \end{cases} \right\}. \tag{22}$$

If Ω' is not empty, go to Step 5; otherwise, go to Step 6.

Step 5: Let the sub-order with the earliest finishing time among Ω' be

$$i^C = \underset{i\in\Omega'}{\operatorname{argmin}}\begin{cases}\max(\tau_j + g(S_j, i), T^L_{\theta(i)}), & \text{if } i \in O^1_C \cup O^1_D.\\ \max(\tau_j + g(S_j, i), \tau_{\delta(i)} + p_{\theta(i)} + t(l_i, l_s)), & \text{if } i \in O^2_C,\\ \max(\tau_j + g(S_j, i), \tau_{\delta(i)} + p_{\theta(i)}), & \text{if } i \in O^2_D.\end{cases} \tag{23}$$

Let $\Omega = \Omega\backslash\{i^C\}$, $X = X \oplus i^C$,

$$\tau_{i^C} = \begin{cases}\max(\tau_j + g(S_j, i^C), T^L_{\theta(i^C)}), & \text{if } i^C \in O^1_C \cup O^1_D,\\ \max(\tau_j + g(S_j, i^C), \tau_{\delta(i^C)} + p_{\theta(i^C)} + t(l_{i^C}, l_s)), & \text{if } i^C \in O^2_C,\\ \max(\tau_j + g(S_j, i^C), \tau_{\delta(i^C)} + p_{\theta(i^C)}), & \text{if } i^C \in O^2_D.\end{cases} \tag{24}$$

and $j = i^C$. If i^C belongs to $O^1_C \cup O^1_D$, let $\Omega = \Omega \cup \{\delta(i^C)\}$. Go to Step 3.

Step 6: No more sub-orders can be assigned to the truck; therefore, one more truck is required. Let $X = X \oplus 0$, $m^U = m^U + 1$, and go to Step 2.

Step 7: If $m^U > m$, the number of involved trucks exceeds the limit; therefore, no feasible solution is generated. Otherwise, append surplus zeroes (separators) to X to obtain a feasible solution X with m^U involved trucks. Stop.

6.3. Neighborhood solutions, the tabu list, and an escape mechanism

Selecting two items in a current solution randomly and exchanging them create a neighborhood solution. A pair of positions of the two exchanged items is a cell in the tabu list. A number (up to n_{NBHD}) of neighborhood solutions are created. If a neighborhood solution is better than the best-so-far solution or is the best among the non-tabu solutions, it is accepted as the new current solution. The length of the tabu list is increased by one if an accepted solution has been visited before, or it is decreased by one if no repeated solution has been found for n_{NOR} iterations. If solutions have been repeated for n_{ESC} times, random exchange is executed for a number of times as the escape mechanism. See Zhang et al. (2015) for a more detailed description of the adjustment of the tabu list length and the escape mechanism.

6.4. Feasibility and optimality of solutions

For the neighborhood solutions generated as described in Section 6.3, the truck number constraints always hold. We determined the service finishing time, checked the feasibility, and calculated the objective value for each sub-order simultaneously.

The separators divide a solution X into a number of sub-order sequences, each of which corresponds to a truck. For each sub-order sequence, the truck state after handling each sub-order and the corresponding transition time are determined according to Functions (2) and (3), respectively. The service finishing time of each sub-order is determined in two stages. In the first stage, the service finishing time τ_i of each sub-order i is temporarily given according to the idea of topological sorting. That is, before giving the service-finishing time of a sub-order, the service-finishing time of the preceding sub-order must be given. Furthermore, the service-finishing time of a Phase I sub-order must be given before the service-finishing time of the corresponding Phase II sub-order can be given. The feasibility-checking of solutions is finished. A solution X is feasible if the service-finishing time of each sub-order in X is given successfully or it is infeasible otherwise.

In the second stage, the service-finishing time of each sub-order is adjusted to avoid unnecessary waiting time. The service-finishing time of the last sub-order for each sequence is fixed. The service -finishing time τ_i of another sub-order i is delayed in the reverse order of topological sorting as follows:

$$\tau_i = \begin{cases}\min(\tau_j - g(S_i, j), \tau_{\delta(i)} - t(l_i, l_s) - p_{\theta(i)}), & \text{if } i \in O^1_C,\\ \min(\tau_j - g(S_i, j), T^U_i + t(l_i, l_s)), & \text{if } i \in O^2_C,\\ \min(\tau_j - g(S_i, j), \tau_{\delta(i)} - p_{\theta(i)}), & \text{if } i \in O^1_D,\\ \min(\tau_j - g(S_i, j), T^U_i), & \text{if } i \in O^2_D,\end{cases} \tag{25}$$

where j is the succeeding sub-order in the same sequence as sub-order i. When the service-finishing time of each sub-order in a solution is determined, its objective value can be calculated using Function (4).

6.5. Recording of solutions and the stopping criterion

All accepted solutions are recorded in a binary tree. Two solutions are equivalent to each other if they can be divided into the same set of sub-order sequences because the trucks are homogeneous. To avoid recording equivalent solutions, the sub-order sequences in a solution are sorted according to the head sub-order in each sequence before a solution is recorded.

For example, a solution

$$X_2 = 3, \delta(1), 0, 4, \delta(3), \delta(4), 0, 1, 2, \delta(2) \quad (26)$$

is equivalent to X_1 as shown in Eq. (20). Therefore, both solutions are recorded as

$$X = 1, 2, \delta(2), 0, 3, \delta(1), 0, 4, \delta(3), \delta(4) \quad (27)$$

in the binary tree and regarded as the same solution.

The algorithm stops after n_{ITER} is reached.

7. Validation and evaluation

In this section, we show the validation and evaluation of the solution method of the FCD problem. In particular, in Section 7.1, we describe our experiments. Section 7.2 features the tuning of the main parameters of the IRTS algorithm. Section 7.3 shows the comparative results between the problems presented in Zhang et al. (2018) and this study. Section 7.4 presents an evaluation of the stability of the IRTS algorithm. Finally, in Sections 7.5 and 7.6, we compare the results to a scenario in which standard containers were used, and analyze key parameters in the FCD problem, respectively.

7.1. Setting of experiments and instances

All experiments were executed on a personal computer with Intel (R) Core (TM) i5-6200U 4 CPUs @ 2.30 GHz and 4.0 GB of memory. The IRTS algorithm was coded using Matlab R2014a.

We set $C = 4$ hereafter unless explicitly stated because four-in-one foldable containers are the most commonly used according to Myung and Moon (2014) and Moon and Hong (2016). Two types of instances, S (scattered) and C (clustered), were randomly generated. For each S-type instance, the customer of each order was randomly and independently generated. For each C-type instance, 15 customers were generated randomly, and a customer might be the shipper or receiver of multiple orders. In both types of instances, the locations of the depot and customers were randomly generated on a Euclidean plane with a length and width equal to a two-hour distance by trucks. The ratio of the collection and that of the distribution orders was 0.6 and 0.4, respectively. The packing and unpacking times of orders were generated randomly in the range of [5, 120]. The left side and the width of each time window was generated randomly in the range of [0, 240] and of [300, 600], respectively. The unit for packing and unpacking times and for time windows in all experiments was the minute. The number of orders was set as 75 and the number of trucks was 40 in each instance unless explicitly stated, because a realistic-sized fleet of trucks could handle at most 75 containers each day according to, for example, Wang and Regan (2002).

7.2. Parameter tuning

Six instances (three for each type), S1-S3 and C1-C3, were generated to tune main parameters of the IRTS algorithm. Five parameters were selected to be tuned: n_{ITER}, n_{NBHD}, n_{TABU}, n_{NOR}, and n_{ESC}.

We first set a group of potential values for a parameter while keeping the values of the other parameters fixed. For each potential value, all six instances were solved using the IRTS algorithm. The value at which the IRTS algorithm performs best (i.e., the value at which the average objective among the six instances is minimal) was chosen and fixed. The other parameters were tuned similarly one by one. Such a parameter-tuning method has been widely used recently, for example, by Keskin and Çatay (2016). It is noteworthy that n_{ITER} and n_{NBHD} were tuned as a twin parameter because the running time of the IRTS algorithm depends mostly on the product of them. The initial values of the parameters were $n_{ITER} = 1000$, $n_{NBHD} = 30$, $n_{TABU} = 20$, $n_{NOR} = 20$, and $n_{ESC} = 40$. The final values of the parameters that were used thereafter were $n_{ITER} = 1000$, $n_{NBHD} = 30$, $n_{TABU} = 20$, $n_{NOR} = 20$, and $n_{ESC} = 60$.

7.3. Comparison to Zhang et al. (2018)

Recall that Zhang et al. (2018) is the only article that presents a similar problem as this research but there was an additional assumption in their problem. Their mathematical model was solved using CPLEX 12.6.1 embedded in Visual Studio 2010. The coding language was C++. The time limit for CPLEX to solve each instance was set as 2 h.

7.3.1. Experiments on small instances

Nine small S-type instances, Sm1 to Sm-9, were randomly generated and solved using both the method in Zhang et al. (2018) and the IRTS algorithm. Five trucks were given in each instance. The obtained objective values and the running times are shown in Table 2.

Table 2 indicates that, for five of nine instances, the running time of CPLEX was less than 2 h. That is, CPLEX reached the optimality of the model. For each of the other four instances, CPLEX did not provide the optimum solution within the given time limit. The objective values shown in Table 2 are those of the near-optimum solutions when the running time was 2 h. In contrast, the IRTS algorithm stopped within 80 s for each of the nine instances. Furthermore, the IRTS algorithm provided better solutions than their method for all nine instances. The average improvement on the objective values was 10.26%.

Instance Sm5 was used as an example to show the differences between the two methods. The solution provided by the method of Zhang et al. (2018) was

Table 2
Comparison results for small instances.

Instance	No. of orders	Zhang et al. (2018)		IRTS		Obj. Improvement (%)
		Obj. (min)	CPU time (s)	Obj. (min)	CPU time (s)	
Sm1	5	934	5.42	830	46.52	11.13
Sm2	5	853	6.63	747	47.65	12.43
Sm3	5	923	5.14	834	44.27	9.64
Sm4	6	814	996.13	686	52.16	15.72
Sm5	6	1067	358.20	996	55.95	6.65
Sm6	6	811	7200.00	644	47.69	20.59
Sm7	7	1081	7200.00	987	53.34	8.70
Sm8	7	1132	7200.00	1076	78.09	4.95
Sm9	7	1220	7200.00	1189	65.37	2.54
Average						10.26

$$4_{D1},\ \delta(3)_{C2},\ 0,\ 5_{D1},\ \delta(1)_{C2},\ 0,\ 6_{D1},\ \delta(2)_{C2},\ 0,\ 3_{C1},\ 2_{C1},\ 1_{C1},\ \delta(5)_{D2},\ \delta(6)_{D2},\ \delta(4)_{D2} \tag{28}$$

with an objective value of 1067 min. None of the four trucks carried both loaded and empty containers. As a comparison, the solution provided by the IRTS algorithm was

$$3_{C1},\ 2_{C1},\ 1_{C1},\ \delta(1)_{C2},\ 0,\ 4_{D1},\ \delta(4)_{D2},\ 0,\ 6_{D1},\ \delta(2)_{C2},\ 0,\ 5_{D1},\ \delta(3)_{C2},\ 0,\ \delta(5)_{C2},\ \delta(6)_{D2} \tag{29}$$

with an objective value of 996 min. In this solution, some trucks, such as the first truck, carry both loaded and empty containers.

7.3.2. Experiments on realistic-sized instances

Twelve more S-type instances, Instances S4-S15, and 12 more C-type instances, say Instances C4-C15, were generated. All 30 instances including those generated in Section 7.2, were solved by the two methods. For each of the 30 instances, CPLEX stopped at 2 h; hence, the near-optimum solution is reported for the method of Zhang et al. (2018). Regarding to the IRTS algorithm, the initial and final objective values are also reported using the proposed method. The results are shown in Figs. 1 and 2.

We reached similar conclusions by looking at Figs. 1 and 2 and the results reported in Section 7.3.1. The average running time of the IRTS algorithm for the 15 S-type instances was 963 s, which was 13.38% of the computation time of the method of Zhang et al. (2018) (7200 s). Furthermore, as shown in Fig. 1, for each of the 15 S-type instances, the IRTS algorithm provided much better solutions than the method of Zhang et al. (2018). The average improvement for the objective values was 44.75%. The highest improvement for the objective values was 52.93% (Instance S15).

The results of C-type instances were quite similar to those of the S-type instances. The average running time of the IRTS algorithm over the 15 C-type instances was 863 s, which was 11.99% of the method of Zhang et al. (2018). Compared to the method of Zhang et al. (2018), the average objective value improvement for the IRTS algorithm was 45.98%. The highest objective value improvement was 60.50% (Instance C5). Overall, C-type instances were a little easier to solve than the S-type instances were; that is, the running times were slightly shorter and the results were slightly better for the C-type instances.

We also compared the IRTS algorithm to the initial solution. As shown in Figs. 1 and 2, the final solutions provided by the IRTS algorithm were much better than the initial solutions. The average objective improvement over the S-type and C-type instances was 20.70% and 23.68%, respectively. This finding validates the searching operations of the IRTS algorithm. Furthermore, as shown in

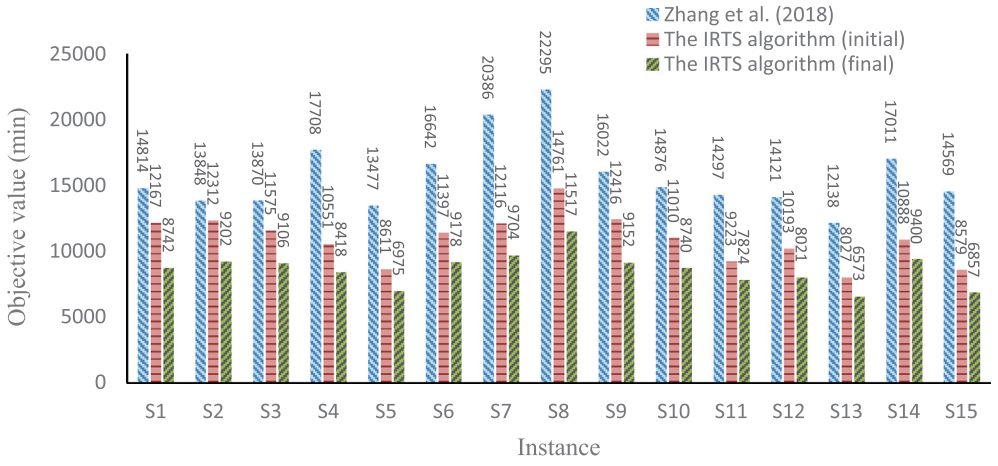

Fig. 1. Comparative results for S-type instances.

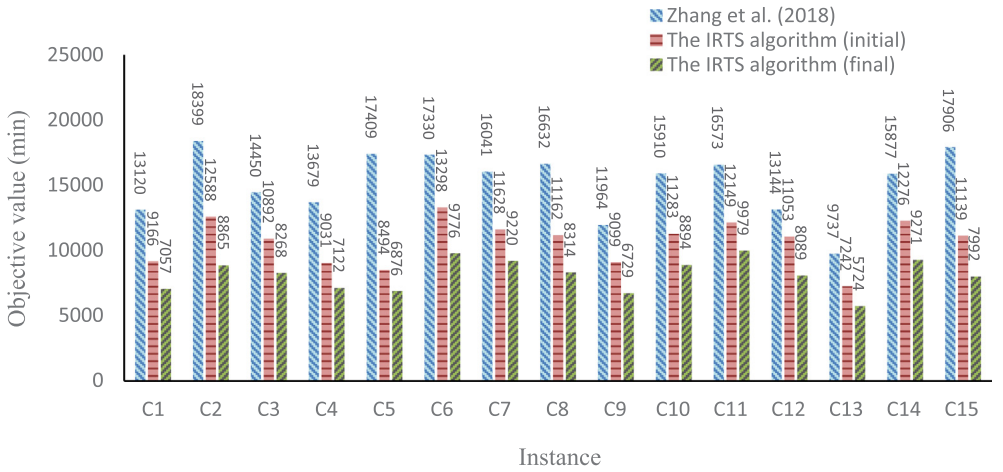

Fig. 2. Comparative results for C-type instances.

Figs. 1 and 2, the initial solutions were much better than those provided by their method. This finding also validates the modeling method.

7.4. Stability of the IRTS algorithm

An experiment was used to test the stability of the IRTS algorithm. Instances S1–S4 and C1–C4 were each solved three times independently by the IRTS algorithm. The objective values provided by and the computation times of the IRTS algorithm are shown in Figs. 3 and 4, respectively.

Figs. 3 and 4 indicate that the IRTS algorithm is very stable, especially when considering the quality of solutions. The maximum relative difference of objective values among the three repeated processes was 3.64%, which was found at Instance S2. The minimum relative difference of objective values among the three repeats was 0.59%. The deviations of the computation times were a little larger than those of the objective values. However, they were still acceptable (Fig. 4).

7.5. Comparison to the standard-container scenario

The results from the FCD problem were compared to those from a scenario in which standard containers were considered. For $C = 1$, a truck can carry only one container, either empty or loaded. Thus, the mathematical formulation and the IRTS algorithm show the scenario in which standard containers were used. On the basis of Instances S1-S4 and C1-C4, we solved the standard-container scenario using the IRTS algorithm with the same parameters we used for solving the cases of foldable containers. The comparative results are shown in Table 3, where the difference (improvement of objective value and saved computation time) is defined as the relative difference of results of the foldable-container scenario from the standard-container scenario.

For the eight instances, as expected, each objective value for the foldable-container scenario was lower than that in the standard-container scenario. The largest relative difference was 17.05%, and the average relative difference was 13.52%. That is, compared to the scenario in which standard containers were used, using four-in-one foldable containers can save, on average, approximately 10% on transportation costs.

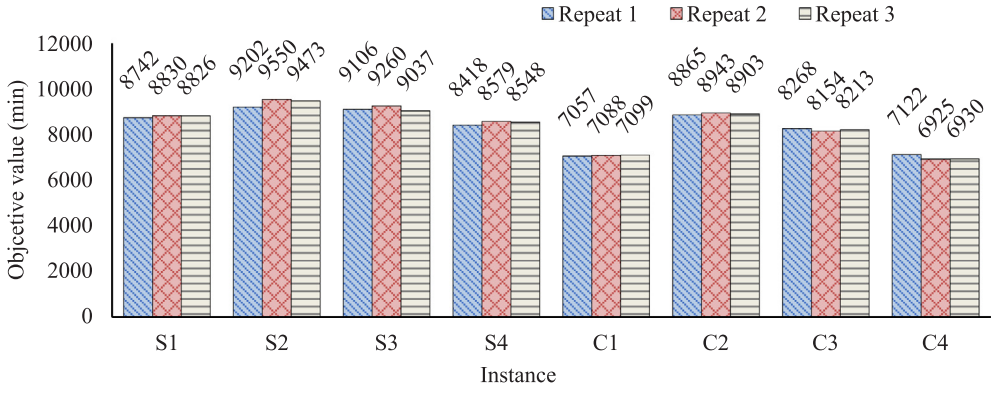

Fig. 3. Objective values in the stability testing experiment.

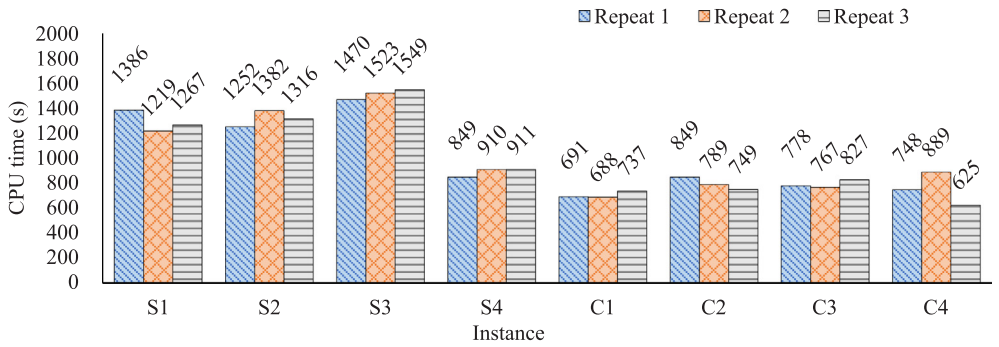

Fig. 4. Computation times in the stability testing experiment.

Table 3
Results compared to the standard-container scenario.

Instance	Objective value			CPU time		
	Standard (min)	Foldable (min)	Improvement (%)	Standard (s)	Foldable (s)	Saving (%)
S1	10,539	8742	17.05	1465	1386	5.39
S2	10,493	9202	12.30	1472	1252	14.95
S3	10,125	9106	10.06	1542	1470	4.67
S4	9422	8418	10.66	1081	887	17.95
C1	8431	7057	16.30	826	691	16.34
C2	9922	8865	10.65	1084	849	21.68
C3	9678	8268	14.57	942	778	17.41
C4	8535	7122	16.56	811	748	7.77
Average			13.52			13.27

In addition, the results in Table 3 indicate that the IRTS algorithm runs slightly slower for the standard-container scenario. The run time was longer because we solved the instances of the standard-container scenario using the IRTS algorithm with the same parameters as we used for the FCD problem. Under the standard-container scenario, the possibility of generating infeasible solutions increased, and hence, the run time for a specific number of iterations also increased.

7.6. Sensitivity analyses

7.6.1. The effect of changes on parameter m

Six typical instances (S1–S3 and C1–C3) were selected for a sensitivity analysis on the number of trucks. We modified the value of parameter m from 40 to 38 and solved the six instances using the IRTS algorithm. The running time of the IRTS algorithm remained almost the same for both values of m. Fig. 5 shows the obtained objective values.

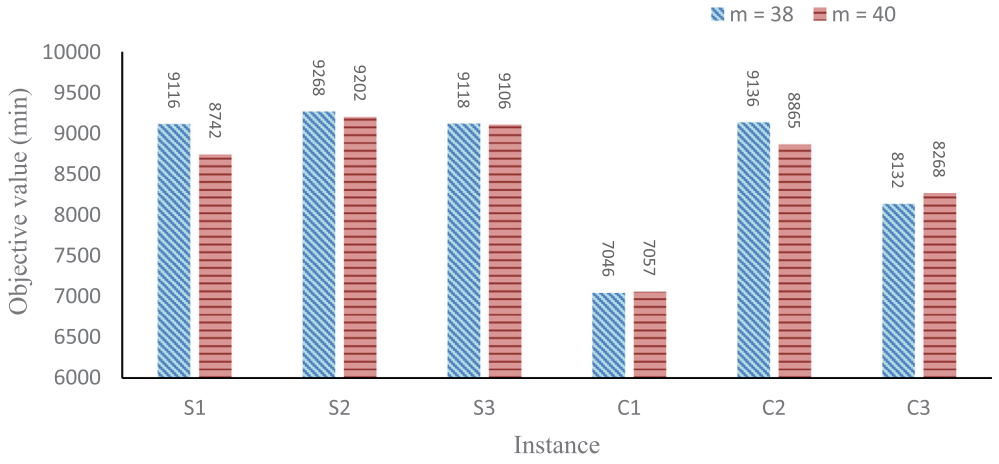

Fig. 5. Objective values under different values of m.

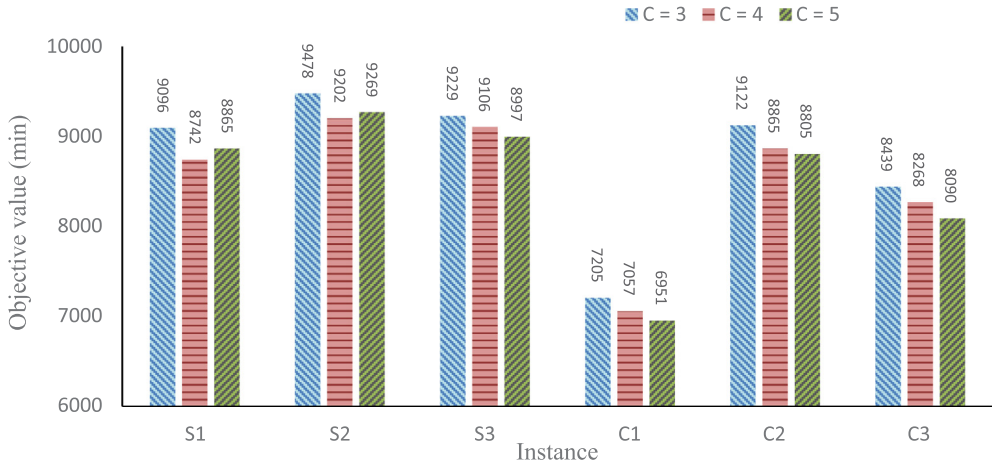

Fig. 6. Objective values under different values of *C*.

As expected, Fig. 5 shows that the objective value decreased as the number of trucks increased from 38 to 40 for most instances (S1–S3 and C2). However, we also found that the objective value increased slightly as the number of trucks increased from 38 to 40 for Instances C1 and C3. This outcome might be the result of an inherent feature of a meta-heuristic; that is, optimality cannot be guaranteed.

7.6.2. The effect of changes on parameter C

A sensitivity analysis experiment on parameter C was carried out using the six instances (S1–S3 and C1–C3). We modified the value of C from 4 to 3 and then to 5. The instances were solved using the IRTS algorithm. The results shown in Fig. 6 indicate that the objective value decreased as C incrementally increased from 3 to 5 for most instances. In other words, with the specific distribution of customers and other information about the instances, the total operating time decreased if a truck could carry more empty containers. This is expected. However, the results for Instances S1 and S2 indicate that the objective value increased slightly as C increased from 4 to 5. The objective value did not decrease as expected when C increased. Even if a truck could carry more empty containers, the constraints of the time windows and the traveling time among customers can limit the objective function to decrease.

8. Conclusions and future research

This paper describes a type of FCD problem in which containers can be folded and unfolded conveniently at any location, including customer sites. Each truck can carry a loaded container or multiple folded containers. Drayage orders include collection and distribution orders. The optimizing criterion was minimization of the total working time, including the time for waiting, of all involved trucks. We proposed a unique RTST method to model the FCD problem and an IRTS algorithm to solve the problem. The model and solution methods were extensively evaluated based on randomly generated instances.

The results indicate that the IRTS algorithm is quite stable and can provide much better solutions than the method of Zhang et al. (2018) designed to solve a similar problem, in much shorter time. The average improvement of the objective values was approximately 40% meanwhile the running time of the IRTS algorithm was just about 10% of the benchmark approach. Furthermore, the comparison to a standard-container scenario showed that the introducing of four-in-one foldable containers can save approximately 10% on transportation costs. The effects of key parameters as the number of trucks, m, and the maximum number of folded containers that a truck can carry, C, were also analyzed.

One limitation of this research was that we included only foldable containers in the FCD problem. As a result, we plan to introduce both standard and foldable containers into future problems of drayage services because standard containers will be used for a long time. However, the model and solution methods for these future studies are expected to be even more complex than those presented herein.

The research presented in this paper can be expanded in many ways. For example, the repositioning of empty containers between areas has been an issue for years. One may introduce inbound and outbound empty containers into the FCD problem as variants of drayage orders. Also, more than one depot may serve an area, and the trucks parked at different depots and containers stacked at depots are limited. The present research can be useful to researchers considering a study of FCD services with uncertainties or different dynamics. Furthermore, different solution methods for the FCD problem might be presented in future studies.

Acknowledgements

This work was partially supported by the National Natural Science Foundation of China (No. 71471034), Fundamental Research Funds for the Central Universities (No. N160404011), and the 111 Project (B16009). This research was also supported by the

National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning [Grant No. 2015R1A2A1A15053948].

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at https://doi.org/10.1016/j.tre.2018.07.011.

References

Akyuz, M.H., Lee, C.-Y., 2016. Service type assignment and container routing with transit time constraints and empty container repositioning for liner shipping service networks. Transp. Res. Part B: Methodol. 88, 46–71.
Battiti, R., Tecchiolli, G., 1994. The reactive tabu search. ORSA J. Comput. 6 (2), 126–140.
Bektas, T., 2006. The multiple traveling salesman problem: an overview of formulations and solution procedures. Omega 34 (3), 209–219.
Braekers, K., An, C., Janssens, G.K., 2013. Integrated planning of loaded and empty container movements. OR Spectrum 35 (2), 457–478.
Caris, A., Janssens, G.K., 2009. A local search heuristic for the pre- and end-haulage of intermodal container terminals. Comput. Oper. Res. 36 (10), 2763–2772.
Cheung, R.K., Shi, N., Powell, W.B., Simao, H.P., 2008. An attribute–decision model for cross-border drayage problem. Transp. Res. Part E: Logist Transp. Rev. 44 (2), 217–234.
Chung, K.H., Chang, S.K., Shin, J.Y., Hwang, H., Kim, K.H., 2007. Development of mathematical models for the container road transportation in Korean trucking industries. Comput. Ind. Eng. 53 (2), 252–262.
Escudero, A., Muñuzuri, J., Guadix, J., Arango, C., 2013. Dynamic approach to solve the daily drayage problem with transit time uncertainty. Comput. Ind. 64 (2), 165–175.
Funke, J., Kopfer, H., 2016. A model for a multi-size inland container transportation problem. Transp. Res. Part E: Logist. Transp. Rev. 89, 70–85.
Glover, F., 1989. Tabu search—Part I. ORSA. J. Comput. 1 (3), 190–206.
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 (3), 235–259.
Keskin, M., Çatay, B., 2016. Partial recharge strategies for the electric vehicle routing problem with time windows. Transp. Res. Part C: Emerg. Technol. 65, 111–127.
Konings, R., 2005. Foldable containers to reduce the costs of empty transport? A cost–benefit analysis from a chain and multi-actor perspective. Marit. Econ. Logist. 7 (3), 223–249.
Konings, R., Thijs, R., 2001. Foldable containers: a new perspective on reducing container-repositioning costs. Eur. J. Transp. Infrastruct. Res. 1 (4), 333–352.
Lai, M., Crainic, T.G., Di Francesco, M., Zuddas, P., 2013. An heuristic search for the routing of heterogeneous trucks with single and double container loads. Transp. Res. Part E: Logist. Transp. Rev. 56 (1), 108–118.
Moon, I., Do Ngoc, A.-D., Konings, R., 2013. Foldable and standard containers in empty container repositioning. Transp. Res. Part E: Logist.Transp. Rev. 49 (1), 107–124.
Moon, I., Hong, H., 2016. Repositioning of empty containers using both standard and foldable containers. Marit. Econ. Logist. 18 (1), 61–77.
Myung, Y.-S., Moon, I., 2014. A network flow model for the optimal allocation of both foldable and standard containers. Oper. Res. Lett. 42 (6–7), 484–488.
Nanry, W.P., Barnes, J.W., 2000. Solving the pickup and delivery problem with time windows using reactive tabu search. Transp. Res. Part B: Methodol. 34 (2), 107–121.
Shintani, K., Konings, R., Imai, A., 2010. The impact of foldable containers on container fleet management costs in hinterland transport. Transp. Res. Part E: Logist. Transp. Rev. 46 (5), 750–763.
Shiri, S., Huynh, N., 2016. Optimization of drayage operations with time-window constraints. Int. J. Prod. Econ. 176, 7–20.
Sterzik, S., Kopfer, H., Yun, W.-Y., 2015. Reducing hinterland transportation costs through container sharing. Flex. Serv. Manuf. J. 27 (2–3), 382–402.
Ting, C.K., Liao, X.L., 2013. The selective pickup and delivery problem: formulation and a memetic algorithm. Int. J. Prod. Econ. 141 (1), 199–211.
Vidović, M., Popović, D., Ratković, B., Radivojević, G., 2017. Generalized mixed integer and VNS heuristic approach to solving the multisize containers drayage problem. Int. Trans. Oper. Res. 24 (3), 583–614.
Vidović, M., Radivojević, G., Raković, B., 2011. Vehicle routing in containers pickup up and delivery processes. Proc. Soc. Behav. Sci. 20 (20), 335–343.
Wang, S., Meng, Q., Lee, C.Y., 2016a. Liner container assignment model with transit-time-sensitive container shipment demand and its applications. Transp. Res. Part B: Methodol. 90, 135–155.
Wang, X., Meng, Q., Miao, L., 2016b. Delimiting port hinterlands based on intermodal network flows: model and algorithm. Transp. Res. Part E: Logist. Transp. Rev. 88, 32–51.
Wang, X., Regan, A.C., 2002. Local truckload pickup and delivery with hard time window constraints. Transp. Res. Part B: Methodol. 36 (2), 97–112.
Wang, Z., Wang, Z., 2009. A novel two-phase heuristic method for vehicle routing problem with backhauls. Comput. Math. Appl. 57 (11–12), 1923–1928.
Wassan, N.A., Wassan, A.H., Nagy, G., 2008. A reactive tabu search algorithm for the vehicle routing problem with simultaneous pickups and deliveries. J. Combust. Optim. 15 (4), 368–386.
Xue, Z., Zhang, C., Lin, W.-H., Miao, L., Yang, P., 2014. A tabu search heuristic for the local container drayage problem under a new operation mode. Transp. Res. Part E: Logist. Transp. Rev. 62 (1), 136–150.
Zazgornik, J., Gronalt, M., Hirsch, P., 2012a. The combined vehicle routing and foldable container scheduling problem: a model formulation and tabu search based solution approaches. Inf. Syst. Oper. Res. 50 (4), 147–162.
Zazgornik, J., Gronalt, M., Hirsch, P., 2012b. A comprehensive approach to planning the deployment of transportation assets in distributing forest products. Int. J. Rev. Manage. 6 (6), 45–61.
Zhang, R., Lu, J.-C., Wang, D., 2014. Container drayage problem with flexible orders and its near real-time solution strategies. Transp. Res. Part E: Logist. Transp. Rev. 61 (1), 235–251.
Zhang, R., Yun, W.Y., Kopfer, H., 2015. Multi-size container transportation by truck: modeling and optimization. Flex. Serv. Manuf. J. 27 (2), 403–430.
Zhang, R., Yun, W.Y., Moon, I., 2009. A reactive tabu search algorithm for the multi-depot container truck transportation problem. Transp. Res. Part E: Logist. Transp. Rev. 45 (6), 904–914.
Zhang, R., Yun, W.Y., Moon, I.K., 2011. Modeling and optimization of a container drayage problem with resource constraints. Int. J. Prod. Econ. 133 (1), 351–359.
Zhang, R., Zhao, H., Liu, S., 2018. Modeling and optimization of a drayage problem with foldable containers. Syst. Eng. Theory Pract. 38 (4), 1013–1023.