A novel mathematical model and a large neighborhood search algorithm for container drayage operations with multi-resource constraints

Ruiyou Zhang [1], Chao Huang [1], Junwei Wang [2]

1. *College of Information Science and Engineering, Northeastern University, Shenyang 110819, China*
2. *Department of Industrial and Manufacturing Systems Engineering, The University of Hong Kong, Pokfulam Road, Hong Kong*

Emails: zhangruiyou@mail.neu.edu.cn (R Zhang), xiachjeng.2012@163.com (C Huang), jwwang@hku.hk (J Wang)

Acknowledgements

The work of Ruiyou Zhang and Chao Huang was partially supported by the National Natural Science Foundation of China (Nos. 71971050, 71831006, and 71471034), the Fundamental Research Funds for the Central Universities (No. N160404011), and National Key R&D Program of China (2017YFB0306401). The work of Junwei Wang was partially supported by the National Natural Science Foundation of China (No. 71571156), by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (No. T32-101/15-R), by the University of Hong Kong through the Seed Fund for Basic Research (No. 201611159213), and by the open project funded by State Key Laboratory of Synthetical Automation for Process Industries (PAL-N201802).

Abstract

Multi-resource constraints in container drayage operations exist widely in real-life scenarios but have been seldom reported in literature. This study mainly addresses a container drayage problem with a limited number of empty containers at depots, with the goal of minimizing both the number of trucks in operation and the total working time of trucks. The problem is firstly formulated as a bi-objective mathematical model based on a so-called determined-activities-on-vertex graph. Three schemes are proposed to handle the developed mathematical model as: (a) a nonlinear constraint is linearized; (b) a parameter that is a large enough constant in the model is analyzed and tuned deeply; and (c) the bi-objective model is converted into a single-objective model. Furthermore, a large neighborhood search algorithm is designed to solve the problem. The two solution methods are validated and evaluated based on randomly generated instances as well as instances from literature. Numerical experimental results indicate that the methods can provide optimal or near-optimal solutions for medium- and large-scale instances in a short running time.

Keywords: Container drayage scheduling; Linearization; Mathematical modeling; Multi-resource constraint; Large neighborhood search

1. Introduction

Short-haulage container transportation including the pre- and end-haulage is usually regarded as container drayage operation (Macharis and Bontekoning, 2004; Namboothiri and Erera, 2008). In container drayage operations, trucks carrying full or empty containers travel among shippers' locations, consignees' locations, terminals and depots. As shown in Fig. 1, trucks provide drayage services in a local area while the long-haul transportation of containers usually requires vessels as transportation tools. Despite of the relatively short distance of truck transportation, the expense of drayage operations accounts for 25%-40% of the whole intermodal container transportation chain (Macharis and Bontekoning, 2004). Moreover, container drayage activities have aggravated a series of ecological and social issues such as greenhouse gas emissions, road congestions and traffic accidents (Cheung et al., 2008; Kopfer, Schönberger, and Kopfer, 2014).

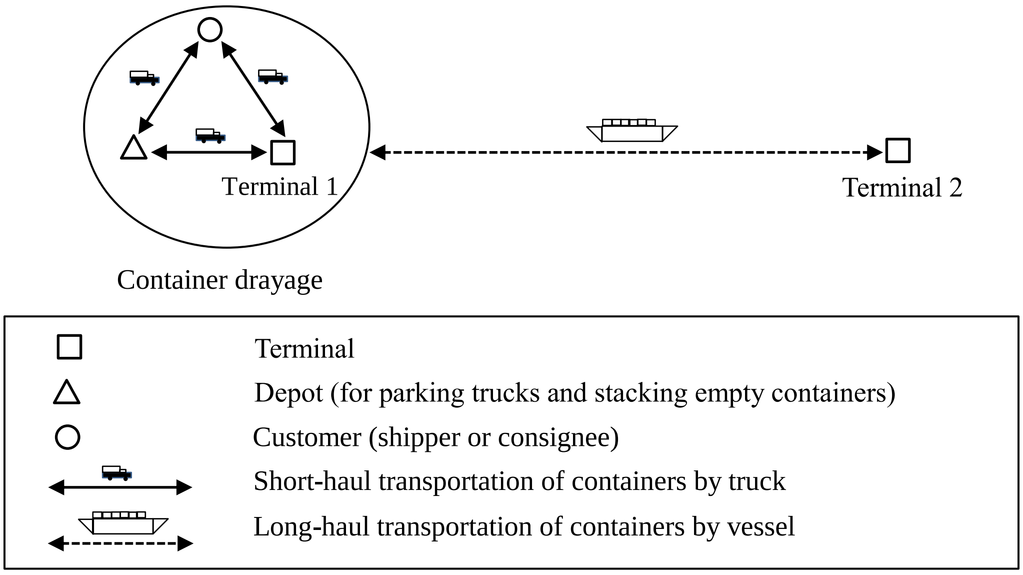
Terminal 1
Terminal 2
Container drayage
Terminal
Depot (for parking trucks and stacking empty containers)
Customer (shipper or consignee)
Short-haul transportation of containers by truck
Long-haul transportation of containers by vessel

Fig. 1. Illustration of container drayage transportation in a local area

Trucks (or tractors, chassis), drivers, and containers can all be regarded as separable resources in container drayage operations (Smilowitz, 2006; Zhang, Yun, and Moon, 2011), generally speaking. Scheduling the transportation resources efficiently is vital for green and safe logistics in urban areas (Wang, Yu, and Tang, 2018; Zhou and Wang, 2018). Trucks with drivers can provide door-to-door services for customers including shippers and consignees. Containers are not only freight to be transported, but also transportation resources. When a container of freight has been delivered to a consignee's location, the container is unpacked, and the recently emptied container has to be transported to a depot or a shipper's location for future use. On the contrary, for a shipper, an empty container should be firstly transported from a depot or a consignee's location to the shipper's location. Then, the full container will be delivered to a terminal after the container is packed.

The container drayage transportation (CDT) problem is a variant of the vehicle routing problem (VRP), and has been studied extensively. Vehicles (including trucks, tractors and chassis) and drivers are two basic types of resources in the VRP. Regarding to the impact of transportation resources, especially about empty containers in drayage operations, most researchers assume that the empty containers at a storage place (e.g., a depot or a terminal) is enough. Sufficient empty containers at a depot can be fluently scheduled to satisfy the requests of customers when necessary. However, the number of empty containers at a storage place is usually limited in real-life scenarios.

Very few articles consider the CDT problem with empty container constraints, especially about a limited number of empty containers at a depot. Zhang, Yun, and Moon (2011) studied a CDT problem with a limited number of empty containers at a depot in an export-dominant area. In their study, three types of containers (i.e., inbound full containers, outbound full containers, and inbound empty containers) and constraints of time windows at a terminal and customers' locations are considered. The objective is to minimize the total working time of all trucks in operation. The problem was mathematically modeled based on a determined-activities-on-vertex (DAOV) graph. The introduction of a nonlinear constraint ensures that the number of empty containers at the depot meets the drayage operations during working time, but makes the problem much difficult to be solved. As a result, a meta-heuristic based on a reactive tabu search (RTS) algorithm was proposed to solve large-scale instances.

This research studies a CDT problem with resource constraints (CDTRC). Four types of containers, i.e., inbound full containers, outbound full containers, inbound empty containers and outbound empty containers, are considered. The focus is to minimize not only the number of involved trucks but also the trucks' total working time. Evolutionary algorithms have successfully solved variant combinational optimization problems especially for large-scale instances (Wang et al., 2018; Xiang et al., 2019; Yi et al., in press). As a result, we not only mathematically solve the problem but also design an evolutionary algorithm based on neighborhood to solve the problem, especially for the large-scale instances in this study.

The main contributions made in this study can be summarized as follows: (i) A container drayage transportation problem with a limited number of empty containers at depots and four types of container tasks is introduced formally. The optimizing criterion is to minimize the number of trucks in operation and the total working time of trucks. The problem is mathematically formulated as a mixed binary nonlinear programming model based on a DAOV graph. (ii) Three schemes are introduced to handle the mathematical model. The nonlinear constraint of the number of empty containers is linearized resulting in a mixed-integer linear programming model. A parameter that is a large enough constant is deeply analyzed and specified. The bi-objective model is converted into a single-objective model using the popular weighted summation method. (iii) A large neighborhood search (LNS) algorithm is designed to solve the CDTRC problem, especially for large-scale instances. The aforementioned modeling and solving methods are validated and evaluated based on a large number of instances (randomly generated instances and benchmark instances from literature) with several concluding remarks proposed.

The remainder of this paper is organized as follows. An extensive review on the literature related to container drayage operations especially for transportation resources is presented in Section 2. The CDTRC problem is formally described in Section 3 and mathematically formulated based on the DAOV graph in Section 4. Sections 5 and 6 handle the mathematical model and formulate the LNS algorithm, respectively. Section 7 presents a series of computational experiments. Finally, Section 8 concludes this paper.

2. Literature review

This section surveys the literature related to the topics studied in this research. Specifically, Section 2.1 summarizes the researches on container drayage operations briefly. Section 2.2 reviews the literature on drayage from the perspective of transportation resources.

2.1. Container drayage operation

The CDT problem has been studied somehow extensively during the recent two decades. For example, Imai, Nishimura, and Current (2007) developed a Lagrangian relaxation-based heuristic to solve a pick-up and delivery problem considering the transportation of full containers. Following that, Caris and Janssens (2009) studied a similar problem by introducing time windows determined by customers and depots. Chung et al. (2007) designed three mathematical models for several types of drayage operations considering some trucking companies in Korean. Recently, Braekers, Caris, and Janssens (2013) presented a problem with integrated transportation of loaded and empty containers, and proposed a hierarchical objective function and an efficient two-phase deterministic annealing algorithm to solve the problem. In order to maximize the total profit, Caballini, Sacone, and Saeednia (2016) introduced a collaboration scheme and a compensation mechanism to drayage operations in a seaport environment.

Dynamic versions of the CDT problem have also drawn much attention from academic fields. For instance, Máhr et al. (2010) investigated a CDT problem with time windows under several cases of uncertainties involving service time and job-arrival, and compared an agent-based method with an online optimization approach to address the problem. Escudero et al. (2013) presented a dynamic algorithm to solve a daily drayage problem with uncertain transit time. Recently, Zhang, Lu, and Wang (2014) described a CDT problem with updatable information and flexible orders and proposed a batch of re-optimization schemes.

2.2. Transportation resources in drayage

Containers, vehicles (e.g., trucks, tractors), and drivers are three main types of transportation resources in drayage. The resources are operated in different modes, with or without time window constraints. Jula et al. (2005) formulated a CDT problem as an asymmetric multiple traveling salesman problem with time windows (m-TSPTW), and proposed a hybrid method of dynamic programming and genetic algorithm. Coslovich, Pesenti, and Ukovich (2006) presented an integer programming model for a CDT problem to minimize fleet operating costs. Based on Braekers, Caris, and Janssens (2013), Braekers, Caris, and Janssens (2014) further studied a CDT problem with a bi-objective function minimizing both the number of involved vehicles and the total traveling distance. Shiri and Huynh (2016) formulated a CDT problem with time windows and truck appointment systems as an extension of the m-TSPTW. Sterzik and Kopfer (2013) proposed a CDT problem with multi-depot and time windows, in which both the repositioning of empty containers and the scheduling of vehicles need to be considered simultaneously. In addition, Sterzik, Kopfer, and Yun (2015) extended the problem in Sterzik and Kopfer (2013) by introducing empty containers shared among different trucking companies. Nossack and Pesch (2013) formulated a CDT problem as a full-truckload pickup-and-delivery problem with time windows, and proposed an efficient two-stage heuristic. Quite recently, Torkjazi, Huynh, and Shiri (2018) developed a mixed integer nonlinear programing model for a truck appointment system involving drayage operations and maritime container terminal operations. Furthermore, Di Francesco et al. (2019) proposed three mathematical formulations for a CDT with each truck carrying one container.

Table 1

Comparison between this study and the related literature

Article	Empty container constraints	Time windows	Reducing fixed costs of trucks	Minimizing variable costs of trucks
Jula et al. (2005)		√		√
Coslovich, Pesenti, and Ukovich (2006)			√	√
Imai, Nishimura, and Current (2007)				√
Cheung et al. (2008)		√		√
Zhang, Yun, and Moon (2011)	√	√		√
Braekers, Caris, and Janssens (2013)		√	√	√
Lai et al. (2013)				√
Nossack and Pesch (2013)		√		√
Sterzik and Kopfer (2013)		√		√
Braekers, Caris, and Janssens (2014)		√	√	√
Xue et al. (2014)		√	√	√
Zhang, Lu, and Wang (2014)		√		√
Sterzik, Kopfer, and Yun (2015)		√		√
Xue et al. (2015)			√	√
Zhang, Yun, and Kopfer (2015)				√
Funke and Kopfer (2016)		√		√
Shiri and Huynh (2016)		√		√
Song et al. (2017)		√		√
Vidović et al. (2017)		√		√
Ghezelsoflu et al. (2018)		√		√
Torkjazi, Huynh, and Shiri (2018)		√		√
Zhang, Zhao, and Moon (2018)		√		√
Di Francesco et al. (2019)				√
This study	√	√	√	√

Recently, the scenarios in which a truck can carry more than one container at a time have appeared in the CDT problems. For instance, Lai et al. (2013) built an integer linear programming model for a truck routing problem involving two types of trucks, in which a truck could carry one container or two containers at a time, and proposed a meta-heuristic algorithm to solve the problem. Zhang, Yun, and Kopfer (2015) formulated a CDT problem with multi-size containers based on a state-transition method, and designed three tree search procedures and an improved RTS algorithm. Similarly, Funke and Kopfer (2016) presented a mixed-integer linear programming model for a multi-size inland container transportation problem. However, only small-scale instances could be solved in their research. Vidović et al. (2017) introduced several types of vehicles to a CDT with time windows, during which a vehicle could carry a number of 20- or 40-foot containers simultaneously. A mixed integer linear program model was developed and a variable neighborhood search heuristic was proposed by Vidović et al. (2017). Similarly, Ghezelsoflu et al. (2018) proposed a set-covering formulation for a CDT, in which a truck could carry one or two (loaded or empty) containers at a time. Quite recently, Zhang, Zhao, and Moon (2018) introduced foldable containers into a CDT problem with time windows, and formulated the problem with a range-based truck-state transition method mathematically.

A so-called drop-and-pull (D&P) mode, in which a tractor (or a truck) can be separated from a trailer (or a container) during container's packing or unpacking time, has also begun to attract researchers' attention. Cheung et al. (2008) formulated a cross-boundary CDT problem with a composite of individual resources (driver, tractor, and chassis) as an attribute-decision model, and designed a labeling algorithm. Xue et al. (2014) examined a local CDT problem considering the transportation of loaded containers under the D&P mode. Subsequently, Xue et al. (2015) proposed a max-min ant colony optimization algorithm to deal with a similar problem. Recently, Song et al. (2017) presented an arc-flow formulation and proposed a branch-and-price-and-cut algorithm to solve a CDT problem under the D&P mode. However, they assumed that the same truck had to come back to pick up the container that it dropped off before.

Zhang, Yun, and Moon (2011) is the most similar article to this research. They presented a nonlinear mathematical model for a CDT problem with limited number of empty containers at a depot, and relaxed the nonlinear constraints resulting in a linear programming model. Based on the relaxed linear model, optimum objective values that were lower bound of the original model were obtained for small-scale instances by using CPLEX. This research differs from Zhang, Yun, and Moon (2011) mainly in three aspects. Firstly, four types of container tasks as inbound full, outbound full, inbound empty and outbound empty, are considered, which is more general than the case in Zhang, Yun, and Moon (2011). Secondly, the optimizing criteria include not only the variable costs (the working time of trucks) but also the fixed costs (the number of involved trucks). Thirdly and most importantly, we propose a novel linearization formulation for the problem and design an efficient LNS heuristic to solve it, and evaluate the solution methods extensively.

In summary, container drayage operations have been studied relatively extensively considering variants of modes. However, transportation resources (specifically, the empty containers available at depots) are usually assumed to be enough. This article presents a drayage problem with a limited number of empty containers at depots and with four types of tasks, which is more general. Moreover, the solution methods presented in this research outperforms the method in Zhang, Yun, and Moon (2011) which is the most similar one to this research. Table 1 summarizes the literature addressing transportation resources in drayage from the perspectives of time windows and operating modes. In Table 1, "Empty container constraints" specifically stands for the constraints on a limited number of empty containers at depots (or terminals).

3. Problem description

A trucking company provides container drayage services for a terminal and a number of customers consisting of shippers and consignees in an area. The company owns a depot, a limited number of empty containers and a number of trucks. The depot is used to store empty containers and park trucks, and can be visited by trucks during working time. Both trucks and containers are homogeneous. Trucks with containers are used to transport freight for customers. Any truck and any container are compatible. Each truck can carry at most one container at a time. A truck's travelling speed remains the same when carrying a full container, an empty container, or no container. The company deals with a number of container tasks within the area in a planning horizon. The planning horizon is usually a working day.

Each transportation task corresponds to the transportation of one container (or a container of freight). If a task consists of the transportation of more than one container, it is decomposed into several tasks so that each task consists of only one container. During a container's packing or unpacking time, the truck handling the container has to stay at the customer's location. The information of all tasks is known in advance. Trucks are located at the depot initially, and have to return to the depot after the tasks assigned to them are finished.

Four types of tasks including inbound full container tasks, outbound full container tasks, inbound empty container tasks, and outbound empty container tasks need to be distributed in the CDTRC problem. Inbound full container tasks, outbound full container tasks, and inbound empty container tasks usually exist in an export-dominant area such as China. Meanwhile, inbound full container tasks, outbound full container tasks, and outbound empty container tasks usually exist in an import-dominant area such as USA. As a result, inbound empty container tasks and outbound empty container tasks do not exist simultaneously in a real-life area. For a container task, the source and destination of a container may be flexible. In order to schedule containers efficiently, it is important and critical to schedule the source and destination of containers in drayage operations. Therefore, Fig. 2 illustrates the source and destination of containers in drayage services in an export-dominant area and an import-dominant area, individually.

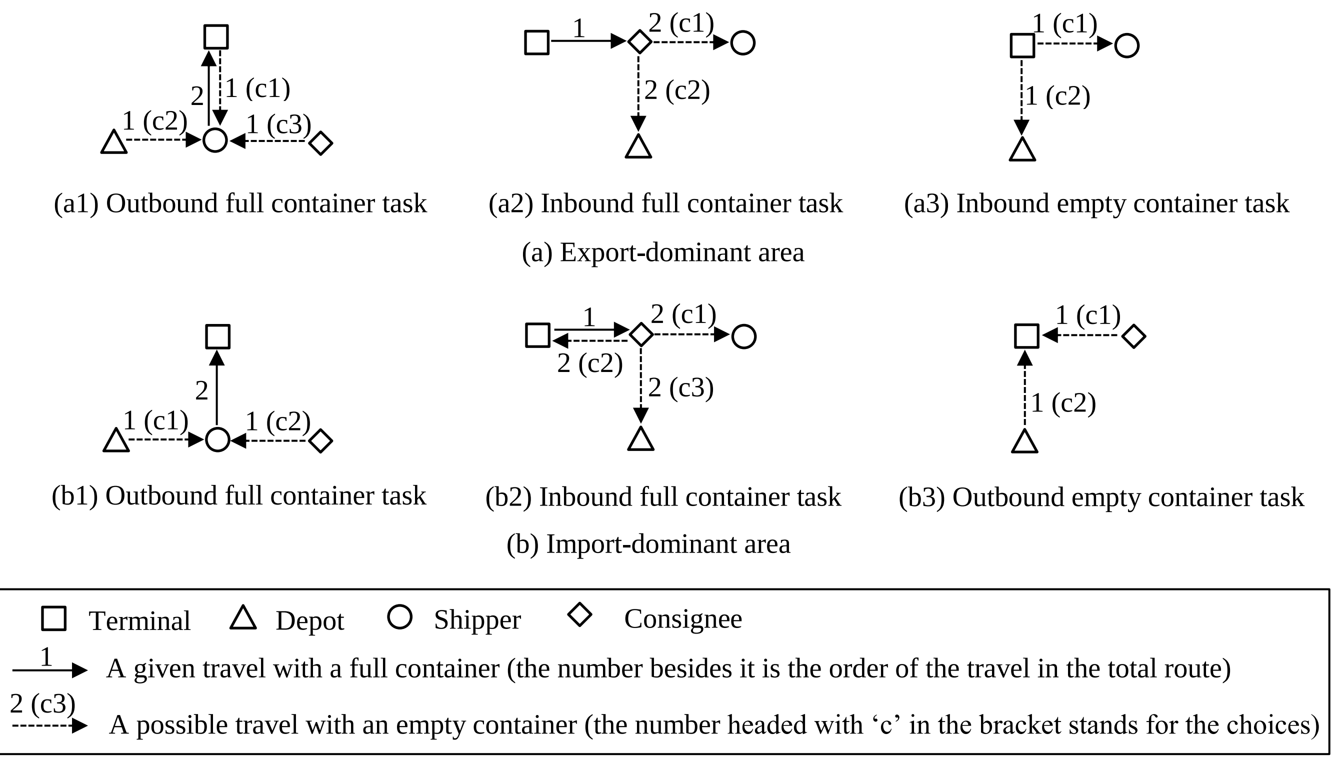
2
1 (c1)
1 (c2)
1 (c3)
(a1) Outbound full container task
1
2 (c1)
2 (c2)
(a2) Inbound full container task
1 (c1)
1 (c2)
(a3) Inbound empty container task
(a) Export-dominant area
2
1 (c1)
1 (c2)
(b1) Outbound full container task
1
2 (c1)
2 (c2)
2 (c3)
(b2) Inbound full container task
1 (c1)
1 (c2)
(b3) Outbound empty container task
(b) Import-dominant area
Terminal
Depot
Shipper
Consignee
1
A given travel with a full container (the number besides it is the order of the travel in the total route)
2 (c3)
A possible travel with an empty container (the number headed with 'c' in the bracket stands for the choices)

Fig. 2. Illustration of the source and destination of containers in drayage operations

Fig. 2(a) illustrates a container's source and destination in three types of tasks in an export-dominant area. For an outbound full container task (Fig. 2(a1)), a truck firstly transfers an empty container to a shipper's location. Note that, the empty container may come from the terminal (corresponding to an inbound empty container task), the depot, or a consignee's location. After packing the freight into the container at the shipper's location, the truck will transfer the full container to the terminal, and drop it off. With respect to an inbound full container task (Fig. 2(a2)), a truck firstly delivers a full container from the terminal to a consignee's location, and drops it off. After the container has been unpacked at the consignee's location, the same truck delivers the empty container to a shipper's location or the depot for future use. For an inbound empty container task (Fig. 2(a3)), a truck picks up an empty container at the terminal, and transfers it to a shipper's location or the depot for future use. The operations of containers in an import-dominant area are in Fig. 2(b) with more details omitted here because of the limited space.

Not only customers but also the terminal provides time windows to limit the starting time of the activities of a container task (packing, unpacking, picking up, and dropping off). Each empty container task has only an origin time window determined by the terminal, whereas each full container task is given an origin time window and a destination time window simultaneously. We assume that the amount of time for picking up a container or dropping off a container is the same.

Two optimizing objectives exist in the CDTRC problem, similarly as in common drayage operations (Braekers, Caris, and Janssens, 2014; Funke and Kopfer, 2016). One is to minimize the number of trucks scheduled. The other is to minimize the total working time that includes the traveling and waiting time of all the trucks in operation. The number of trucks reflects the fixed costs. The total working time of the trucks reflects the variable costs for a trucking company, at least to a certain degree.

The following notations are used to describe the CDTRC problem:

C_{IF}:	Set of inbound full container tasks
C_{OF}:	Set of outbound full container tasks
C_{IE}:	Set of inbound empty container tasks
C_{OE}:	Set of outbound empty container tasks
$C = C_{\mathrm{IF}} \cup C_{\mathrm{OF}} \cup C_{\mathrm{IE}} \cup C_{\mathrm{OE}}$	
$N = C \cup \{0\}$:	Set of tasks, where task 0 is a virtual start/return task
m:	Initial number of empty containers at the depot in a planning horizon
t:	Amount of time for picking up or dropping off a container
t_i:	Amount of time for unpacking a full container $i \in C_{\mathrm{IF}}$ or packing a container $i \in C_{\mathrm{OF}}$
$t_{ij} \geq 0$:	A truck's traveling time between two locations, l_i and l_j. i and j belong to $C_{\mathrm{IF}} \cup C_{\mathrm{OF}} \cup \{0, \mathrm{G}\}$, where l_i represents a customer's location of the corresponding container task i if $i \in C_{\mathrm{IF}} \cup C_{\mathrm{OF}}$, and l_0 and l_{G} stand for the depot and the terminal, respectively. $t_{ij} = t_{ji}$
$[\tau_{1\mathrm{L}i}, \tau_{1\mathrm{R}i}]$:	Origin time window of a container task $i \in C$, $\tau_{1\mathrm{L}i} \leq \tau_{1\mathrm{R}i}$
$[\tau_{2\mathrm{L}i}, \tau_{2\mathrm{R}i}]$:	Destination time window of a full-container-related task $i \in C_{\mathrm{IF}} \cup C_{\mathrm{OF}}$, $\tau_{2\mathrm{L}i} \leq \tau_{2\mathrm{R}i}$

4. Mathematical formulation

Differently from the classical VRP, both the scheduling of trucks and repositioning of empty containers need to be considered in the CDTRC problem. It is difficult to construct a mathematical model for the problem directly. Therefore, the CDTRC problem is mathematically formulated based on a DAOV graph (Zhang, Yun, and Moon, 2011; Zhang, Lu, and Wang, 2014).

4.1. The DAOV graph

Let $G = (N, A)$ be a DAOV graph formulating the CDTRC problem, where $N = C \cup \{0\}$ is the vertex set.

Note that the set N stands for not only the task set but also the task vertex set without confusions involved. Vertex 0 is a virtual start/return vertex, representing the initial start from and final return to the depot. $A = \{(i,j) | i \in N, j \in N, i \neq j\}$ is the arc set, in which an arc $(i,j) \in A$ is defined as the transfer from a vertex $i \in N$ to another vertex $j \in N$, $i \neq j$.

The following parameters are introduced in the DAOV graph:

T_i:	Service time consumed by activities of a task vertex $i \in N$
$[T_{Li}, T_{Ui}]$:	Time window of a task vertex $i \in C$, during which the corresponding activities must be started, $T_{Li} \leq T_{Ui}$
T_{ij}:	Transfer-time attribute of an arc $(i,j) \in A$
$\alpha_{ij} \in \{1, -1, 0\}$:	Change-number attribute of an arc $(i,j) \in A$
$\beta_{ij} \geq 0$:	Change-time attribute of an arc $(i,j) \in A$

The calculation procedure of the service time T_i and the time window $[T_{Li}, T_{Ui}]$ of a task vertex $i \in C$, and that of the transfer-time T_{ij} of an arc $(i,j) \in A$, are similar to that presented in Zhang, Yun, and Moon (2011) and hence are omitted here. $T_0 = 0$ is defined for a uniform formulation. The change-number and change-time attributes of an arc $(i,j) \in A$ are presented in Tables 2 and 3, respectively.

Table 2
Values of α_{ij} for arc $(i,j) \in A$

i	$j = 0$	$j \in C_{\mathrm{IF}}$	$j \in C_{\mathrm{OF}}$	$j \in C_{\mathrm{IE}}$	$j \in C_{\mathrm{OE}}$
$i = 0$	–	0	-1	0	-1
$i \in C_{\mathrm{IF}}$	1	1	0	1	0
$i \in C_{\mathrm{OF}}$	0	0	-1	0	-1
$i \in C_{\mathrm{IE}}$	1	1	0	1	–
$i \in C_{\mathrm{OE}}$	0	0	-1	–	-1

Table 3
Values of β_{ij} for arc $(i,j) \in A$

i	$j = 0$	$j \in C_{\mathrm{IF}}$	$j \in C_{\mathrm{OF}}$	$j \in C_{\mathrm{IE}}$	$j \in C_{\mathrm{OE}}$
$i = 0$	–	0	0	0	0
$i \in C_{\mathrm{IF}}$	$t_{i0} + 2t$	$t_{i0} + 2t$	0	$t_{i0} + 2t$	0
$i \in C_{\mathrm{OF}}$	0	0	t_{G0}	0	t_{G0}
$i \in C_{\mathrm{IE}}$	$t_{\mathrm{G0}} + t$	$t_{\mathrm{G0}} + t$	0	$t_{\mathrm{G0}} + t$	–
$i \in C_{\mathrm{OE}}$	0	0	$\mathrm{t_{G0}}$	–	t_{G0}

4.2. Mathematical model

Several decision variables were introduced as follows:

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

y_i: time when the activities of task vertex $i \in C$ are started

z_{ij}: time when the activities of arc $(i,j) \in A$ are started

The CDTRC problem can be formulated as the following mathematical model based on the DAOV graph:

$$\min \ f_1 = \sum_{i \in C} x_{0i} \tag{1}$$

$$\min\ f_2 = \sum_{j\in C}\left(z_{j0} + T_{j0}x_{j0} - z_{0j}\right) \tag{2}$$

$$\text{s.t.}\quad \sum_{j\neq i, j\in N} x_{ij} = \sum_{j\neq i, j\in N} x_{ji} = 1,\ \ \forall i \in C \tag{3}$$

$$T_{\mathrm{L}i} \le y_i \le T_{\mathrm{U}i},\ \ \forall i \in C \tag{4}$$

$$y_i + T_i - z_{i0} \le (T_{\mathrm{U}i} + T_i)(1 - x_{i0}),\ \ \forall i \in C \tag{5}$$

$$z_{0j} + T_{0j} - y_j \le \left(T_{\mathrm{U}j} + T_{0j}\right)\left(1 - x_{0j}\right),\ \ \forall j \in C \tag{6}$$

$$y_i + T_i - z_{ij} \le \left(T_{\mathrm{U}i} + T_i + T_{\mathrm{U}j}\right)\left(1 - x_{ij}\right),\ \ \forall (i,j) \in A, i \neq 0, j \neq 0 \tag{7}$$

$$z_{ij} + T_{ij} - y_j \le \left(T_{\mathrm{U}i} + T_{ij} + T_{\mathrm{U}j}\right)\left(1 - x_{ij}\right),\ \ \forall (i,j) \in A, i \neq 0, j \neq 0 \tag{8}$$

$$z_{0j} \le T_{\mathrm{U}j}x_{0j},\ \ \forall j \in C \tag{9}$$

$$-z_{i0} \le (T_{\mathrm{U}i} + T_i)x_{i0},\ \ \forall i \in C \tag{10}$$

$$\sum_{(p,q)\in A_2,\ x_{pq}=1,\ z_{pq}+\beta_{pq}\le z_{ij}+\beta_{ij}} \alpha_{pq} + m \ge 0,\ \ \forall (i,j) \in A_1 \tag{11}$$

$$x_{ij} \in \{0,1\},\ \ \forall (i,j) \in A \tag{12}$$

$$y_i \in \mathbb{R},\ \ \forall i \in C \tag{13}$$

$$z_{ij} \in \mathbb{R},\ \ \forall (i,j) \in A \tag{14}$$

The objective functions (1) and (2) minimize the number of involved trucks, and the total working time of the trucks, respectively. The term $z_{j0} + T_{j0}x_{j0}$ in (2) represents the time point when a truck finally returns to the depot, and z_{0j} represents the time point when a truck initially leaves the depot. Constraint (3) ensures that any container vertex $i \in C$ has to be visited by a truck exactly once. Constraint (4) states that the activities of a container vertex $i \in C$ have to be started during its time window. Constraints (5)-(8) not only claim the relationship among vertices and arcs, but also remove sub-tours among container vertices. Constraint (9) and the objective function (2) collectively ensure that the value of z_{0j} is 0 if the value of x_{0j} is 0. Similarly, Constraint (10) and the objective function (2) ensure that the value of z_{i0} is 0 if the value of x_{i0} is 0.

Constraint (11) will be explained in detail, where

$$A_1 = \left\{(i,j)\middle|(i,j) \in A, \alpha_{ij} = -1\right\}, \text{ and } A_2 = \left\{(i,j)\middle|(i,j) \in A, \alpha_{ij} \neq 0\right\}.$$

Note that the number of empty containers at the depot should not be smaller than zero at any time. If $\alpha_{ij} \neq -1$, arc (i,j) will not result in a decrease on the number of empty containers at the depot. Therefore, such arcs (i,j) that satisfy $\alpha_{ij} = -1$ are focused on in (11). If arc (i,j) is included in the solution, i.e., $x_{ij} = 1$, the moment when the number of empty containers at the depot is changed by the arc (i,j) is $z_{ij} + \beta_{ij}$. If $\alpha_{ij} = 0$, the arc (i,j) does not change the number of empty containers at the depot. As a result, for each arc $(i,j) \in A_1$, the change-number attribute of such arcs $(p,q) \in A_2$, which satisfy $x_{pq} = 1$, and $z_{pq} + \beta_{pq} \le z_{ij} + \beta_{ij}$ should be summed. That is Constraint (11). Constraints (12)-(14) define the types of variables, where $\mathbb{R}$ is the set of real numbers.

Main differences between the aforementioned model ((1)-(14)) and that presented by Zhang, Yun, and Moon (2011) are as follows. Firstly, double (but not single) objective functions exist in this study. Secondly, the number of trucks at the depot is not restricted in constraints because it is minimized in the objective function. Thirdly, Constraints (5)-(8) automatically remove sub-tours among container vertices when describing time continuity, which differ from and are more efficient than the corresponding constraints in Zhang, Yun, and Moon (2011). Fourthly and finally, Constraint (11) is more specific and efficient than the corresponding constraint in Zhang, Yun, and Moon (2011).

5. Handling of the model

Partially because of the existence of the special nonlinear Constraint (11), it is difficult to solve the model in Section 4 directly by using optimization software such as CPLEX and BARON. Therefore, three strategies are introduced to handle this model. Specifically, Section 5.1 presents a linearized formulation of the model. Section 5.2 further analyzes and tunes a key parameter used in the model. Finally, the two objective functions are converted into a single one in Section 5.3.

5.1. Linearization of the model

A binary variable u_{ijpq} was defined to describe the relationship between arcs (i,j) and (p,q) in Constraint (11).

$$u_{ijpq} = \begin{cases} 1, & \text{if } x_{ij} = 1, x_{pq} = 1, \text{and } z_{pq} + \beta_{pq} \le z_{ij} + \beta_{ij} \\ 0, & \text{otherwise} \end{cases}, \quad \forall (i,j) \in A_1, (p,q) \in A_2 \tag{15}$$

Proposition 1. Constraint (11) is equivalent to the following linear constraints.

$$z_{ij} + \beta_{ij} - z_{pq} - \beta_{pq} \ge -M(3 - u_{ijpq} - x_{ij} - x_{pq}), \qquad \forall (i,j) \in A_1, (p,q) \in A_2 \tag{16}$$

$$z_{ij} + \beta_{ij} - z_{pq} - \beta_{pq} + \varepsilon \le M(2 + u_{ijpq} - x_{ij} - x_{pq}), \qquad \forall (i,j) \in A_1, (p,q) \in A_2 \tag{17}$$

$$u_{ijpq} \le x_{ij}, \quad \forall (i,j) \in A_1, (p,q) \in A_2 \tag{18}$$

$$u_{ijpq} \le x_{pq}, \quad \forall (i,j) \in A_1, (p,q) \in A_2 \tag{19}$$

$$\sum_{(p,q)\in A_2} \alpha_{pq} u_{ijpq} + m \ge 0, \forall (i,j) \in A_1 \tag{20}$$

$$u_{ijpq} \in \{0,1\}, \quad \forall (i,j) \in A_1, (p,q) \in A_2 \tag{21}$$

where M is a large enough positive constant, and ε is a small enough positive constant.

Proof. The definition of u_{ijpq} shows that u_{ijpq} makes sense if and only if both of the arcs $(i,j) \in A_1$ and $(p,q) \in A_2$ are selected in the solution, i.e., $x_{ij} = 1$ and $x_{pq} = 1$. In other words, $u_{ijpq} = 0$ if x_{ij} or x_{pq} equals to zero. Thus, Constraints (18) and (19) are concluded. Therefore, $x_{ij} = 1$ and $x_{pq} = 1$ is assumed hereafter. Because of $x_{ij} = 1$ and $x_{pq} = 1$, Constraints (16) and (17) become

$$z_{ij} + \beta_{ij} - z_{pq} - \beta_{pq} \ge -M(1 - u_{ijpq}), \qquad \forall (i,j) \in A_1, (p,q) \in A_2 \tag{22}$$

and

$$z_{ij} + \beta_{ij} - z_{pq} - \beta_{pq} + \varepsilon \le M u_{ijpq}, \qquad \forall (i,j) \in A_1, (p,q) \in A_2 \tag{23}$$

respectively.

If $z_{ij} + \beta_{ij} - z_{pq} - \beta_{pq} \ge 0$, Constraint (23) indicates that $u_{ijpq} = 1$ and Constraint (22) always holds. On the contrary, if $u_{ijpq} = 1$, Constraint (22) indicates that $z_{ij} + \beta_{ij} - z_{pq} - \beta_{pq} \ge 0$ and Constraint (23) always holds.

Similarly, if $z_{ij} + \beta_{ij} - z_{pq} - \beta_{pq} < 0$, Constraint (22) indicates that $u_{ijpq} = 0$ and Constraint (23) always holds. On the contrary, if $u_{ijpq} = 0$, Constraint (23) indicates that $z_{ij} + \beta_{ij} - z_{pq} - \beta_{pq} < 0$ and Constraint (22) always holds.

In summary, Constraint (11) is equivalent to Constraints (16)-(21). □

5.2. Specifying the parameter M

The parameter M used in Constraints (16) and (17) is further specified in this subsection. According to Table 2, we have $i = 0$ or $i \in C_{\text{OF}} \cup C_{\text{OE}}$ while $j \in C_{\text{OF}} \cup C_{\text{OE}}$ for any arc $(i,j) \in A_1$. Similarly, the arcs $(p,q) \in A_2$ can be divided as: $p = 0$ and $q \in C_{\text{OF}} \cup C_{\text{OE}}$; $p \in C_{\text{IF}} \cup C_{\text{IE}}$ and $q = 0$; $p \in C, q \in C, \alpha_{pq} \neq 0$.

Therefore, Constraints (16) and (17) are rewritten and M is specified correspondingly according to different cases:

Case 1: $i = 0, j \in C_{\text{OF}} \cup C_{\text{OE}}, p = 0, q \in C_{\text{OF}} \cup C_{\text{OE}}$.

Constraints (16) and (17) can be rewritten as:

$$z_{0j} + \beta_{0j} - z_{0q} - \beta_{0q} \geq -M^1_{0j0q}\big(3 - u_{0j0q} - x_{0j} - x_{0q}\big), \tag{24}$$
$$z_{0j} + \beta_{0j} - z_{0q} - \beta_{0q} + \varepsilon \leq M^1_{0j0q}\big(2 + u_{0j0q} - x_{0j} - x_{0q}\big), \tag{25}$$

where

$$M^1_{0j0q} = T_{0j} + T_{\mathrm{U}j} + \beta_{0j} + T_{0q} + T_{\mathrm{U}q} + \beta_{0q} + \varepsilon. \tag{26}$$

Case 2: $i = 0, j \in C_{\mathrm{OF}} \cup C_{\mathrm{OE}}, p \in C_{\mathrm{IF}} \cup C_{\mathrm{IE}}, q = 0.$

Constraints (16) and (17) become

$$z_{0j} + \beta_{0j} - z_{p0} - \beta_{p0} \geq -M^2_{0jp0}\big(3 - u_{0jp0} - x_{0j} - x_{p0}\big), \tag{27}$$
$$z_{0j} + \beta_{0j} - z_{p0} - \beta_{p0} + \varepsilon \leq M^2_{0jp0}\big(2 + u_{0jp0} - x_{0j} - x_{p0}\big), \tag{28}$$

where

$$M^2_{0jp0} = T_{0j} + T_{\mathrm{U}j} + \beta_{0j} + T_{\mathrm{U}p} + T_p + T_{p0} + \beta_{p0} + \varepsilon. \tag{29}$$

Case 3: $i = 0, j \in C_{\mathrm{OF}} \cup C_{\mathrm{OE}}, p \in C, q \in C, \alpha_{pq} \neq 0.$

Constraints (16) and (17) become

$$z_{0j} + \beta_{0j} - z_{pq} - \beta_{pq} \geq -M^3_{0jpq}\big(3 - u_{0jpq} - x_{0j} - x_{pq}\big), \tag{30}$$
$$z_{0j} + \beta_{0j} - z_{pq} - \beta_{pq} + \varepsilon \leq M^3_{0jpq}\big(2 + u_{0jpq} - x_{0j} - x_{pq}\big), \tag{31}$$

where

$$M^3_{0jpq} = T_{0j} + T_{\mathrm{U}j} + \beta_{0j} + T_{\mathrm{U}p} + T_{\mathrm{U}q} + \beta_{pq} + \varepsilon. \tag{32}$$

Case 4: $i \in C_{\mathrm{OF}} \cup C_{\mathrm{OE}}, j \in C_{\mathrm{OF}} \cup C_{\mathrm{OE}}, p = 0, q \in C_{\mathrm{OF}} \cup C_{\mathrm{OE}}.$

Constraints (16) and (17) become

$$z_{ij} + \beta_{ij} - z_{0q} - \beta_{0q} \geq -M^4_{ij0q}\big(3 - u_{ij0q} - x_{ij} - x_{0q}\big), \tag{33}$$
$$z_{ij} + \beta_{ij} - z_{0q} - \beta_{0q} + \varepsilon \leq M^4_{ij0q}\big(2 + u_{ij0q} - x_{ij} - x_{0q}\big), \tag{34}$$

where

$$M^4_{ij0q} = T_{\mathrm{U}i} + T_{\mathrm{U}j} + \beta_{ij} + T_{0q} + T_{\mathrm{U}q} + \beta_{0q} + \varepsilon. \tag{35}$$

Case 5: $i \in C_{\mathrm{OF}} \cup C_{\mathrm{OE}}, j \in C_{\mathrm{OF}} \cup C_{\mathrm{OE}}, p \in C_{\mathrm{IF}} \cup C_{\mathrm{IE}}, q = 0.$

Constraints (16) and (17) become

$$z_{ij} + \beta_{ij} - z_{p0} - \beta_{p0} \geq -M^5_{ijp0}\big(3 - u_{ijp0} - x_{ij} - x_{p0}\big), \tag{36}$$
$$z_{ij} + \beta_{ij} - z_{p0} - \beta_{p0} + \varepsilon \leq M^5_{ijp0}\big(2 + u_{ijp0} - x_{ij} - x_{p0}\big), \tag{37}$$

where

$$M^5_{ijp0} = T_{\mathrm{U}i} + T_{\mathrm{U}j} + \beta_{ij} + T_{\mathrm{U}p} + T_p + T_{p0} + \beta_{p0} + \varepsilon. \tag{38}$$

Case 6: $i \in C_{\mathrm{OF}} \cup C_{\mathrm{OE}}, j \in C_{\mathrm{OF}} \cup C_{\mathrm{OE}},\ p \in C, q \in C, \alpha_{pq} \neq 0.$

Constraints (16) and (17) become

$$z_{ij} + \beta_{ij} - z_{pq} - \beta_{pq} \geq -M^6_{ijpq}\big(3 - u_{ijpq} - x_{ij} - x_{pq}\big), \tag{39}$$
$$z_{ij} + \beta_{ij} - z_{pq} - \beta_{pq} + \varepsilon \leq M^6_{ijpq}\big(2 + u_{ijpq} - x_{ij} - x_{pq}\big), \tag{40}$$

where

$$M^6_{ijpq} = T_{\mathrm{U}i} + T_{\mathrm{U}j} + \beta_{ij} + T_{\mathrm{U}p} + T_{\mathrm{U}q} + \beta_{pq} + \varepsilon. \tag{41}$$

5.3. Transformation of the objectives

Similarly as in Xue et al. (2014), the two objective functions f_1 and f_2 are multiplied by two cost coefficients,

$k_1(\geq 0)$ and $k_2(\geq 0)$, respectively. k_1 is the fixed cost of using a truck. k_2 is the cost of a truck per unit of working time. As a result, the final objective function becomes

$$\min\ k_1 \sum_{i\in C} x_{0i} + k_2 \sum_{j\in C} \left(z_{j0} + T_{j0}x_{j0} - z_{0j}\right). \tag{42}$$

In summary, the initial model ((1)-(14)) has been converted as a single-objective mixed-binary linear programming model (3)-(10), (12)-(14), (18)-(21), (24)-(42). Hereafter, when talking about solving the model, we refer to this namely *improved* model.

6. A large neighborhood search algorithm

The handling schemes presented in Section 5 make the model possible to be solved by using existing software. However, it is still hard to solve the improved model by using common optimization software such as CPLEX, because of too much memories and/or too long computational time required when the size of instances is large. Accordingly, an algorithm based on LNS is designed to solve the problem. LNS was firstly proposed by Shaw (1998) to solve a VRP. It has been successfully applied to solve variants of VRP recently (Hintsch and Irnich, 2018; Hojabri et al., 2018; Mancini and Stecca, 2018; Zhu and Sheu, 2018). The basic idea of LNS is to explore the solution space iteratively by removing a large number of vertices of the current solution and re-inserting them into different positions. In the LNS algorithm designed in this study, an initial solution is generated based on a greedy algorithm. Some key factors of the CDTRC problem including time windows and constraints between two adjacent task vertices in a current solution are considered to re-insert the removed tasks.

Three parameters are introduced in the LNS algorithm:

α: Remove coefficient, $0 \leq \alpha \leq 1$
K: Maximum number of iterations
Λ: Given CPU time limit

6.1. Encoding and decoding

Similarly as in, e.g., Zhang, Zhao, and Moon (2018), an order-encoding scheme with separators was used in the LNS algorithm. For example, a valid solution of an instance with five container vertices and three trucks can be encoded as

$$X_1 = (0, 1, 4, 0, 5, 2, 0, 3, 0), \tag{43}$$

where 0 is the separator, and 1, 2, 3, 4, 5 are five different container tasks, respectively. Taking $(0, 1, 4, 0)$ for an illustration, a truck initially starting from the depot firstly handles Task 1, then handles Task 4, and returns to the depot finally.

6.2. Generation of the initial solution

The initial solution of the LNS algorithm is generated based on a greedy idea. All container vertices are assigned to trucks in series. Three main principles are used in the generation of the initial solution. Firstly, an inbound (full or empty) container vertex is preferred to be the first task for a truck. Secondly, a truck is preferred to handle inbound and outbound vertices alternately. Specifically, an inbound container vertex is preferred if a truck just handles an outbound container vertex, or vice versa. Thirdly, a container vertex with the earliest activity-beginning time among the feasible candidates is preferred for a current truck. If no container vertex is feasible for a current truck, a new truck is arranged. Such a procedure continues until all tasks are assigned.

The step-by-step procedure of the generation of the initial solution is as follows.

Step 1: Initialization. Let $\Omega_1 = C_{\text{IF}} \cup C_{\text{IE}}$ and $\Omega_2 = C_{\text{OF}} \cup C_{\text{OE}}$. Let $X = (0)$ be an incomplete solution and $n^u = 0$ be the number of trucks involved. Let $y = -M$ and $\eta = 0$, where M is still a large enough positive constant. The flag $\eta = 0$ if an inbound task is preferred or $\eta = 1$ otherwise.

Step 2: If $\eta = 1$ or $\{i | i \in \Omega_1, y + T_{e(X)} + T_{e(X)i} \le T_{\text{U}i}\} = \emptyset$, let

$$\Omega = \{i | i \in \Omega_2, y + T_{e(X)} + T_{e(X)i} \le T_{\text{U}i}\} \tag{44}$$

denote the set of feasible candidate vertices currently, where $e(X)$ is a function denoting the last item of the solution X, and $\emptyset$ denotes an empty set.

Step 3: If $\Omega = \emptyset$, let

$$\Omega = \{i | i \in \Omega_1, y + T_{e(X)} + T_{e(X)i} \le T_{\text{U}i}\}. \tag{45}$$

Step 4: If $\Omega = \emptyset$, go to Step 5; otherwise, go to Step 6.

Step 5: If $\Omega_1 = \emptyset$ and $\Omega_2 = \emptyset$, the procedure has generated an initial solution $X \oplus 0$ with $n^u + 1$ trucks and stops, where $\oplus$ denotes appending an item to a list; otherwise, let $n^u = n^u + 1$, $X = X \oplus 0$, $\eta = 0$, and $y = -M$, and go to Step 2 (a new truck is required).

Step 6: Let

$$Y_i = \max(y + T_{e(X)} + T_{e(X)i}, T_{\text{L}i}), \ \forall i \in \Omega. \tag{46}$$

$$f = \operatorname*{argmin}_{i \in \Omega} Y_i. \tag{47}$$

Let $X = X \oplus f$ and $y = Y_f$. If task f is an inbound task, let $\eta = 1$ and $\Omega_1 = \Omega_1 \backslash \{f\}$; otherwise let $\eta = 0$ and $\Omega_2 = \Omega_2 \backslash \{f\}$. Go to Step 2.

6.3. Remove and insertion operators

For a current solution, a number of container vertices are removed randomly and inserted back resulting in a new solution. The number of container vertices that are removed from a solution is $\lfloor \alpha |C| \rfloor$, where $|\cdot|$ denotes the number of items in the given set, and $\lfloor \cdot \rfloor$ denotes the largest integer that is not larger than the given value. If two or more continuous separators exist in a destroyed solution, redundant separators are removed.

The starting time of any remaining container vertex and that of any arc are calculated in the *forward* order as follows. If $j \in C$ is the first container vertex handled by a truck, let

$$y_j = T_{\text{L}j}. \tag{48}$$

If $j \in C$ is not the first container vertex handled by a truck, let

$$y_j = \max(y_i + T_i + T_{ij}, T_{\text{L}j}), \tag{49}$$

where i is the preceding container vertex of vertex j. For any container vertex $j \in C$, let

$$z_{ij} = y_j - T_{ij}, \tag{50}$$

where i is the preceding container vertex of vertex j. For the last vertex $i \in C$ handled by a truck, let

$$z_{i0} = y_i + T_i. \tag{51}$$

All removed container vertices are inserted back one by one, which is the so-called repair operator. For any to-be-inserted vertex $j \in C$, the best position among all feasible ones is selected. If a vertex j is appended to the end of a solution X, an additional separator is added after j, which means that one more truck is involved. If a vertex j is inserted at another position, that is, an existing truck will handle the vertex, then the starting time of vertices and that of arcs from the insertion position through the next separator must be checked and re-determined similarly as in formula (48)-(51). If such a checking procedure succeeds, the insertion position is said to be a feasible position. A *best* position is defined as the position at which the increased objective value reaches its minimum. See Section 6.4 for the further check on the number of empty containers.

6.4. Feasibility and optimality of solutions and the stopping criterion

For a newly generated solution described in Section 6.3, the starting time of vertices and arcs is adjusted in order to avoid unnecessary waiting time. For each truck, the starting time of the last container vertex is fixed, and the starting time of the other container vertices and arcs is adjusted in the *backward* order as follows. For an arc (i,j), if $y_i < y_j - T_i - T_{ij}$, let

$$z_{ij} = y_j - T_{ij}, \tag{52}$$

$$y_i = \min(T_{\mathrm{U}i}, y_j - T_i - T_{ij}). \tag{53}$$

Given a solution X and the starting time of any vertex and arc, it is easy to check the empty container constraints according to Constraint (11). If a solution is infeasible, it is discarded and another solution is generated using the remove and insertion operators. For a feasible solution, its objective value is calculated according to Function (42). If a better solution is found, update the best-so-far solution, and the number of iterations of the algorithm increases by one; otherwise, generate another solution in the similar way. The algorithm stops if the maximum number of iterations K or the predetermined time limit Λ is reached.

6.5. Discussion

The algorithm starts from a given feasible solution, and repeats remove and insertion steps until a stopping criterion terminates it. The degree of destruction in the remove operator, i.e., the remove coefficient, is key for the performance of the algorithm. Removing a proper percentage of vertices from a current solution results in a good balance between exploring a large search space and reducing computational time. Therefore, parameter-tuning experiments of the remove coefficient together with the other parameters for different scales of instances are carried out (in Section 7.2). With respect to the insertion operator, the starting times of activities on both container vertices and arcs are calculated considering time windows and other constraints between two adjacent task vertices in a current solution, which contributes to reducing the total working time of trucks and the number of trucks in operation. In addition, the newly generated solution in each iteration of the LNS is improved by eliminating unnecessary waiting time to the greatest extent. Experimental results in Section 7 reveal the effectiveness and efficiency of the LNS algorithm.

7. Computational experiments

The mathematical model, as well as the LNS algorithm, is validated and evaluated based on numerical experiments in this section. Specifically, Section 7.1 presents the setting of experiments and the generation of instances. Section 7.2 formulates the tuning of main parameters of the LNS algorithm. Section 7.3 analyzes the computational results for small- and medium-scale instances. Section 7.4 validates the LNS algorithm based on large-scale instances. In Section 7.5, the results are compared to a similar research done by Zhang, Yun, and Moon (2011). Finally, sensitivity analyses for three key parameters in the CDTRC problem are presented in Section 7.6.

7.1. Experiments setting and instances generation

All experiments were carried out on a personal computer with Intel Core i7-2600 CPU 3.40 GHz, 3.40 GHz, and 7.83 GB memory available under the operating system of Windows 7. The improved model was solved using IBM ILOG CPLEX 12.6.1. The C++ language in Visual Studio 2010 was used to call the CPLEX solver and to code the LNS algorithm.

The CPLEX solver was configured as follows: The MIP node selection strategy was *Depth-first* search. The memory available for working storage was set to be 3072 MB. The *node file* parameter was set to be three. The CPU time limit was set to be one hour.

The CDTRC instances were generated randomly similarly as in many existing researches (Nossack and Pesch, 2013; Shiri and Huynh, 2016; Song et al., 2017). The locations of the terminal, depot, shippers, and consignees were generated randomly on a Euclidean plane with a length and width equal to 3 hours' traveling distance by truck. The time for picking up or dropping off a container was set to be 5 minutes. The time for packing or unpacking a container was uniformly distributed in the range of $[5, 60]$ minutes. Similarly, the left bound of the origin time window of each task was distributed in the range of $[0, 240]$ minutes. For each full container task, the left bound of the destination time window was set to be equal to the summation of the left bound of corresponding origin time window and the traveling time between the corresponding customer's location and the terminal. The widths of the origin time window and destination time window for a task were distributed in the range of $[0, 240]$ and $[0, 300]$ minutes, respectively. The cost coefficients were set as $k_1 = 400$ and $k_2 = 1$ (per minute), respectively, unless explicitly stated.

7.2. Parameter tuning of LNS

Performance of the LNS algorithm depends on the key parameters α, K, and Λ and scales of instances. Therefore, we performed a series of experiments to tune the parameters for different instance scales. The numbers of container tasks for small-, medium- and large-scale instances are in the range of [5, 30], [31, 55], and [56, 80], respectively. Fifteen instances (five for each instance scale) were randomly generated.

The tuning method was similar as in, e.g., Ropke and Pisinger (2006), Keskin and Çatay (2016) and Hintsch and Irnich (2018). We change the value of one parameter while keep the other parameters fixed. For each potential value of a parameter, all five instances were solved using the LNS algorithm. The value at which the LNS algorithm performed best (i.e., the average objective value among the five instances was minimal) was chosen and fixed. All the three parameters were tuned one by one in the same way. Table 4 presents the potential values of each parameter for different instance scales. The tuning results are shown in Table 5.

Table 4

Potential values of each parameter for different instance scales

Instance scale	α	K	Λ (s)
Small	0.2, 0.3, 0.4	5, 10, 15	0.1, 0.2, 1.0
Medium	0.2, 0.3, 0.4	20, 25, 30	2.0, 5.0, 10.0
Large	0.2, 0.3, 0.4	35, 40, 45	500, 600, 700

Table 5

Tuning results of parameters

Instance scale	α	K	Λ (s)
Small-scale	0.4	15	0.2
Medium-scale	0.3	30	5.0
Large-scale	0.2	45	700

7.3. Experiments on small- and medium-scale instances

Eleven small-scale instances (i.e., Instances 1-11) and twenty-eight medium-scale instances (i.e., Instances 12-39) were generated. All the instances were solved using both CPLEX (based on the improved model) and the LNS algorithm. Tables 6 and 7 present the results for small- and medium-scale instances, respectively. The four numbers in the brackets of the third column of Tables 6 and 7 represent the numbers of inbound full container tasks, outbound full container tasks, inbound empty container tasks, and outbound empty container tasks, respectively.

Table 6

Experiments for small-scale instances

Instance	m	No. of tasks	CPLEX		LNS	
			Obj.	CPU time (s)	Obj.	CPU time (s)
1	2	5 (2, 2, 0, 1)	2362	0.23	2362	0.20
2	3	9 (4, 4, 0, 1)	4923	0.27	4923	0.20
3	6	15 (6, 7, 2, 0)	6136	1.01	6136	0.20
4	7	17 (9, 6, 0, 2)	8615	1.23	8615	0.20
5	8	19 (10, 6, 0, 3)	7572	19.84	7572	0.20
6	8	20 (10, 6, 0, 4)	8690	17.58	8690	0.20
7	8	22 (7, 11, 4, 0)	9237	9.77	9237	0.20
8	8	25 (12, 9, 0, 4)	10116	5.04	10116	0.20
9	11	28 (11, 13, 4, 0)	11248	18.45	11248	0.20
10	10	30 (15, 11, 0, 4)	12169	3312.00	12169	0.20
11	10	30 (15, 11, 0, 4)	12732	22.15	12732	0.20
Average				309.78		0.20

Table 7

Experiments for medium-scale instances

Instance	m	No. of tasks	CPLEX			LNS		Obj. Imp (%)*
			Obj.	Gap (%)*	CPU time (s)	Obj.	CPU time (s)	
12	10	32 (16, 11, 0, 5)	11415	0.00	1901.69	11415	5.00	0.00
13	15	32 (16, 12, 0, 4)	12057	1.82	3600.00	12057	5.00	0.00
14	13	33 (17, 11, 0, 5)	14639	0.00	67.92	14639	5.00	0.00
15	13	35 (18, 12, 0, 5)	16638	0.09	3600.00	16638	5.00	0.00
16	13	35 (18, 12, 0, 5)	12948	0.08	3600.00	12948	5.00	0.00
17	12	38 (19, 14, 0, 5)	13129	7.20	3600.00	13129	5.00	0.00
18	16	40 (14, 20, 6, 0)	15598	3.71	3600.00	15598	5.00	0.00
19	16	40 (14, 20, 6, 0)	16353	2.05	3600.00	16353	5.00	0.00
20	15	42 (16, 20, 6, 0)	14298	6.13	3600.00	14274	5.00	0.17
21	15	44 (21, 17, 0, 6)	15660	6.10	3600.00	15660	5.00	0.00
22	13	45 (17, 22, 6, 0)	16839	0.00	2771.67	16839	5.00	0.00
23	16	45 (22, 17, 0, 6)	20649	6.77	3600.00	20649	5.00	0.00
24	17	45 (22, 17, 0, 6)	18391	7.55	3600.00	18391	5.00	0.00
25	17	47 (23, 18, 0, 6)	16012	3.81	3600.00	16012	5.00	0.00
26	17	47 (23, 18, 0, 6)	17812	7.02	3600.00	17812	5.00	0.00
27	17	47 (23, 18, 0, 6)	18964	6.23	3600.00	18964	5.00	0.00
28	19	50 (24, 20, 0, 6)	19768	3.90	3600.00	19768	5.00	0.00
29	19	50 (24, 20, 0, 6)	18552	6.64	3600.00	18510	5.00	0.23
30	19	50 (24, 20, 0, 6)	17409	5.38	3600.00	17409	5.00	0.00
31	20	52 (25, 21, 0, 6)	19114	4.87	3600.00	19114	5.00	0.00
32	23	52 (25, 21, 0, 6)	23509	2.01	3600.00	23509	5.00	0.00
33	21	52 (25, 21, 0, 6)	23298	4.06	3600.00	23298	5.00	0.00
34	21	53 (25, 22, 0, 6)	17157	16.03	3600.00	17045	5.00	0.65
35	21	53 (25, 22, 0, 6)	21269	6.80	3600.00	21269	5.00	0.00
36	21	53 (25, 22, 0, 6)	18994	9.61	3600.00	18994	5.00	0.00
37	23	55 (22, 27, 6, 0)	27933	0.72	3600.00	27933	5.00	0.00
38	23	55 (22, 27, 6, 0)	22360	4.74	3600.00	22355	5.00	0.02
39	23	55 (22, 27, 6, 0)	22958	1.91	3600.00	22958	5.00	0.00
Average				4.47	3383.62		5.00	0.04

*: "Gap" is the relative difference between the objective value and the lower bound, provided by CPLEX. "Obj. Imp" is defined as the relative difference of objective values obtained by using CPLEX and LNS.

Firstly, the results in Tables 6 and 7 validate the model. For all the small-scale instances and three medium-scale instances (i.e., Instances 12, 14 and 22), CPLEX provided the optimum solutions in the given time limit. Particularly, the average solving time over the 11 small-scale instances was only 309.78 seconds. In addition, for the other 25 medium-scale instances, CPLEX was capable of providing high-quality solutions within the specified time limit. It was worth mentioning that the average gap between the objective value and the lower bound over the 28 medium-scale instances was only 4.47%. The largest gap was 16.03%, which was reached at Instance 34 with 53 container tasks. Such a performance of the mathematical model is acceptable in application.

Secondly, performance of the LNS algorithm is also validated according to the results in Tables 6 and 7. For each of the 11 small-scale instances, which were solved to optimality by CPLEX, the LNS algorithm was capable of providing the optimum solutions within 0.20 seconds, which was much faster than that of using CPLEX (the average running time was longer than 300.00 seconds). Furthermore, for each of 4 instances among 28 medium-scale instances (i.e., Instances 20, 29, 34 and 38), the solution provided by the algorithm during the time limit of 5.00 seconds was better than that provided by CPLEX in one hour. In addition, for each of the other 24 medium-scale instances, the algorithm provided the same solution within a shorter running time compared to CPLEX. For medium-scale instances, optimal or near-optimal solutions were obtained by using CPLEX. It was worth mentioning that the average running time of CPLEX was 3383.62 seconds while that of the LNS algorithm was only 5.00 seconds. Remember that the mathematical model solved by CPLEX is an efficient improved model. The LNS algorithm can still provide better or at least the same solutions in a much shorter running time than solving the model. This validates the LNS algorithm.

7.4. Experiments on large-scale instances and stability tests for the LNS algorithm

A sub-fleet of trucks can handle at most 75 container tasks in one day (Wang and Regan, 2002; Zhang, Zhao, and Moon, 2018). Therefore, eight large-scale instances (i.e., Instances 40-47), each of which consists of 80 container tasks, were generated randomly in this section. It is hard to provide even a feasible solution for such large –scale instances using CPLEX. Therefore, these instances were solved by using the LNS algorithm only. Each of the eight instances was solved seven times independently using the LNS algorithm. Most of the $56(= 8 \times 7)$ repeats runs for 700 seconds as the time limit reaches. Quite few exceptions are that the LNS algorithm stops before 700 seconds as the given number of iterations reaches.

Table 8 presents statistical information over the seven repeats for the eight instances. The results in Table 8 indicate that, regarding to objective values, the LNS algorithm is also very stable. For two out of the eight instances (Instances 45 and 47), the LNS algorithm provided the same solutions in each of the eight repeats. The last column of Table 8 presents the relative difference between the maximum and minimum objective values over the seven repeats. The largest relative difference is only 0.49%, which reaches at Instance 46. The average relative difference over the eight instances is only 0.11%. This is acceptable in application.

Table 8

Statistical information of objective values over the seven repeats

Instance	Minimum	Maximum	Average	Difference (%)*
40	32566	32579	32572	0.04
41	27729	27746	27741	0.06
42	36130	36172	36141	0.12
43	32118	32170	32161	0.16
44	31789	31795	31794	0.02
45	34290	34290	34290	0.00
46	32386	32546	32486	0.49
47	33024	33024	33024	0.00

*: “Difference” is the relative difference between maximum value and minimum value.

7.5. Comparison to Zhang, Yun, and Moon (2011)

The CDTRC problem studied in this research degenerates to be the problem presented by Zhang, Yun, and

Moon (2011) if we make the following modifications: Firstly, the outbound empty container tasks are deleted while the other three types of tasks remain. Secondly, the cost coefficients are set as $k_1 = 0$ and $k_2 = 1$. Thirdly, the constraint on the number of trucks is introduced. In this subsection, we compare the results to the method of Zhang, Yun, and Moon (2011).

Zhang, Yun, and Moon (2011) generated 17 small-scale instances and solved them using a meta-heuristic. For 15 out of the 17 instances, the algorithm of Zhang, Yun, and Moon (2011) provided the optimum solutions. We solved these 17 instances using CPLEX based on the improved model. The results are shown in Table 9, in which the number in bracket of the first column is the instance name used in Zhang, Yun, and Moon (2011). Again, the results validate the mathematical model. For all 17 instances, CPLEX reached the optimum solutions in a much shorter running time. The average running time of the algorithm in Zhang, Yun, and Moon (2011) over the 17 instances is 598 seconds. As a contrast, the average running time of CPLEX is only 17 seconds.

We also solved the 17 instances using the LNS algorithm with the results presented in Table 9. For each of these instances, the LNS algorithm also provided the optimum solutions. The running time of the LNS algorithm is only 0.2 seconds, which is even much shorter than CPLEX.

Zhang, Yun, and Moon (2011) generated four realistic-scale instances, each of which consists of 70 or 75 tasks, randomly, solved each of them three times independently, and presented the repeat with the smallest objective value among the three repeats. We also solved each of the four instances by the LNS algorithm three times independently and provided the best repeat in Table 9. It can be easily found that, for all the realistic-scale instances, the LNS algorithm can provide better solutions than the algorithm of Zhang, Yun, and Moon (2011) in much shorter running time.

Furthermore, we carried out more experiments and evaluated the LNS algorithm based on *t*-test similarly as in literature (Wang et al., 2017; Fu et al., 2018). Each of the four realistic-scale instances was solved seven more times independently by the LNS algorithm. Thus, each of the instances was solved for ten times including the three times carried out before. We employed a one-tailed one-sample *t*-test with 9 degrees of freedom at a significance level of 0.05 based on the four instances. The best objective value among the three repeats provided by Zhang, Yun, and Moon (2011) was regarded as the hypothesized value for each instance. The null hypothesis is that the objective values provided by the LNS algorithm are not smaller (i.e., better) than those provided by Zhang, Yun, and Moon (2011). Table 10 presents the statistical results of the *t*-test.

Table 10 indicate that the *t*-test results are in the rejection region (i.e., smaller than -1.8331) for all instances and are marked as "+". That is, the null hypothesis is rejected. In other words, the LNS algorithm provides significantly better solutions than the algorithm of Zhang, Yun, and Moon (2011) in much shorter running time. Again, this validates the LNS algorithm.

Table 9

Comparison to Zhang, Yun, and Moon (2011)

Instance	Objective value					CPU time (s)		
	Zhang, Yun, and Moon (2011) (min)	CPLEX (min)	Imp[1] (%)*	LNS (min)	Imp[2] (%)*	Zhang, Yun, and Moon (2011)	CPLEX	LNS
48 (1)	771	771	0.00	771	0.00	0.34	0.19	0.20
49 (2)	1529	1529	0.00	1529	0.00	9.25	1.31	0.20
50 (3)	3225	3225	0.00	3225	0.00	622.27	2.44	0.20
51 (4)	2912	2912	0.00	2912	0.00	546.77	12.94	0.20
52 (5)	4120	4120	0.00	4120	0.00	725.02	6.61	0.20
53 (6)	3718	3718	0.00	3718	0.00	695.88	25.49	0.20
54 (7)	4926	4926	0.00	4926	0.00	589.34	7.41	0.20
55 (8)	4037	4037	0.00	4037	0.00	684.70	8.52	0.20
56 (9)	3965	3965	0.00	3965	0.00	746.38	3.58	0.20
57 (10)	5077	5024	1.04	5024	1.04	1038.00	10.00	0.20
58 (11)	4488	4488	0.00	4488	0.00	765.36	8.76	0.20
59 (12)	3809	3809	0.00	3809	0.00	559.19	13.19	0.20
60 (13)	3375	3230	4.30	3230	4.30	422.97	26.94	0.20
61 (14)	2373	2373	0.00	2373	0.00	453.53	126.77	0.20
62 (15)	4775	4775	0.00	4775	0.00	821.20	3.48	0.20
63 (16)	4767	4767	0.00	4767	0.00	675.33	17.67	0.20
64 (17)	4190	4190	0.00	4190	0.00	814.83	12.33	0.20
65 (18)	15128	-	-	14879	1.65	6607.00	-	700.00
66 (19)	11064	-	-	10941	1.11	4386.00	-	700.00
67 (20)	11536	-	-	11334	1.75	5229.00	-	700.00
68 (21)	11780	-	-	11470	2.63	4952.00	-	700.00
Average			0.31		0.59			

*: Imp[1] and Imp[2] are the relative improvements of objective values between CPLEX and Zhang, Yun, and Moon (2011), and those between the LNS algorithm and Zhang, Yun, and Moon (2011), respectively.

"-" indicates that CPLEX is not used to solve the instance because of its too large size.

Table 10

Statistical results of objective values for each realistic-scale instance

Instance	Average	Standard Deviation	*t*-test
65	14911	46.21	-14.88 (+)
66	10981	32.26	-8.16 (+)
67	11414	44.52	-8.66 (+)
68	11505	45.32	-19.16 (+)

7.6. Sensitivity analyses

In this subsection, we analyze the sensitivities of several key parameters in the problem description. Instance 40, as a large-scale instance, was selected in the following experiments.

7.6.1. The effect of k_1 and k_2

We keep $k_2 = 1$ and increase the value of k_1 from 0 to 8000, and then set $k_1 = 1$ and $k_2 = 0$, and re-solve Instance 40 using the LNS algorithm. Table 11 shows the objective values as well as the two components. The results indicate that as k_1 increases, the number of trucks in the solutions decreases and the total working time of trucks increases, generally. Of course, as k_1 increases and k_2 remains, the objective value increases also. This is natural. We can adjust the values of these weights according to the actual cost information in applications.

Table 11

Results under different values of k_1 and k_2 for Instance 40

k_1	k_2	Objective value	Number of involved trucks	Total working time of trucks (min)
0	1	18166	36	18166
400	1	32566	36	18166
8000	1	298844	35	18844
1	0	35	35	18844

7.6.2. The effect of parameter m

We increase the initial number of empty containers at the depot, i.e., m, in Instance 40, from 20 to 50 at a step of 5. Each of the new instances was solved using the LNS algorithm with results presented in Fig 3. As expected, the objective value decreases as m increases from 20 to 40, but remains as m continues increasing. That is, more empty containers can do good to scheduling of trucks if the empty containers are not enough. In Instance 40, a certain number of empty containers, e.g., 40, are enough for fluently scheduling the trucks to finish the drayage tasks. This is unsurprising.

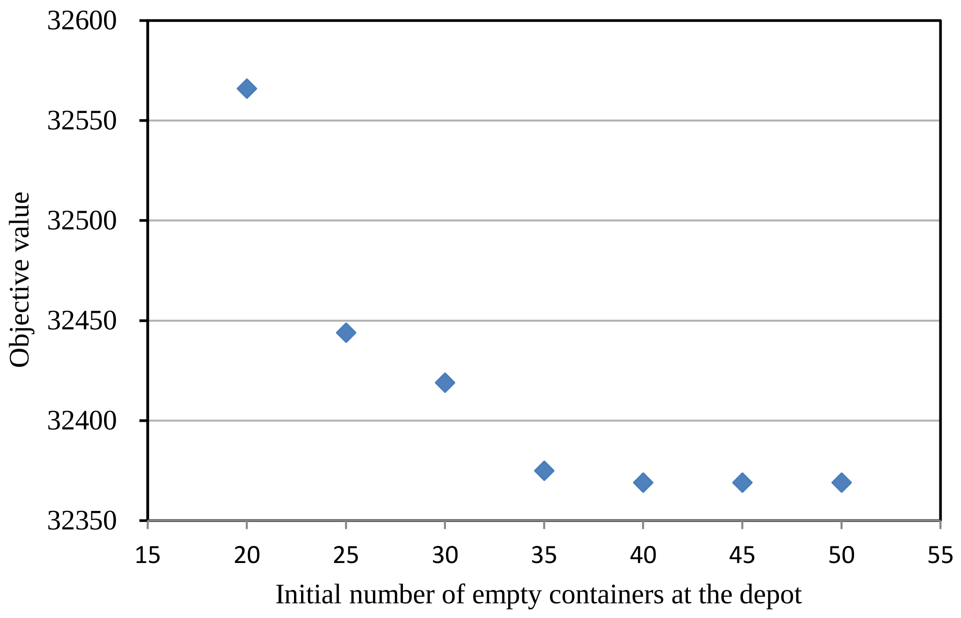

Fig. 3. Objective values under different values of m

8. Conclusions and future research

This study presents a so-called bi-objective CDTRC problem focusing on the constraints of the number of empty containers at depots in container drayage services. Four types of container tasks as inbound full, outbound full, inbound empty and outbound empty container tasks with time windows given by the shippers, consignees and the terminal are considered. One objective is to minimize the number of involved trucks and the other minimizes the total working time of trucks. The CDTRC problem is mathematically formulated as a nonlinear programming model based on a DAOV graph. The model is linearized and the parameter M is analyzed and specified. It is converted into a single objective mixed-integer linear programming model using the weighted summation method. A LNS algorithm is further designed to solve the problem.

Numerical experiments based on a large number of instances validate the aforementioned research. Based on the mathematical model, small-scale instances of the problem can be solved optimally by using popular optimization software such as CPLEX. High-quality solutions with an average gap to the corresponding lower bound as 4.47% of medium-scale instances including 32 through 55 container tasks can be provided by CPLEX in one hour. The LNS algorithm outperforms the mathematical model solved by CPLEX especially considering large-scale instances. It can provide the same or much better solutions in much shorter running time and performs very stably. Furthermore, both the mathematical model and the LNS algorithm perform much better than the algorithm presented for a similar drayage problem by Zhang, Yun, and Moon (2011).

This research can be followed in several ways in the future. For example, a large trucking company may own more than one depot to provide drayage services for more customers. The modeling and solving of drayage services with multiple depots considering limited number of empty containers must be much complicated. Secondly, the problem considered in this study is under the stay-with mode, i.e., a truck waits at the customers when the container is being packed or unpacked. If the packing or unpacking activities last a long time, the truck that ever carries the container can choose to leave the customer to handle some other container tasks, and another truck may come back to pick up the packed or unpacked container when it is ready. Thirdly, if foldable containers are introduced, a truck can handle one full container or several empty folded containers at a time. Handling of the constraints on the number of containers as well as the number of trucks in such scenarios is interesting but definitely more complicated.

References

Braekers, K., Caris, A., & Janssens, G. K. (2013). Integrated planning of loaded and empty container movements. *OR Spectrum*, *35*(2), 457-478.

Braekers, K., Caris, A., & Janssens, G. K. (2014). Bi-objective optimization of drayage operations in the service area of intermodal terminals. *Transportation Research Part E: Logistics and Transportation Review*, *65*, 50-69.

Caballini, C., Sacone, S., & Saeednia, M. (2016). Cooperation among truck carriers in seaport containerized transportation. *Transportation Research Part E: Logistics and Transportation Review*, *93*, 38-56.

Caris, A., & Janssens, G. K. (2009). A local search heuristic for the pre- and end-haulage of intermodal container terminals. *Computers & Operations Research*, *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. *Transportation Research Part E: Logistics and Transportation Review*, *44*(2), 217-234.

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

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

Di Francesco, M., Gentile, C., Schirra, S., Stecca, G., & Zuddas, P. (2019). An integral LP relaxation for a drayage problem. *Discrete Optimization*, *31*, 93-102.

Escudero, A., Muñuzuri, J., Guadix, J., & Arango, C. (2013). Dynamic approach to solve the daily drayage problem with transit time uncertainty. *Computers in Industry*, *64*(2), 165-175.

Fu, Y., Wang, H., Tian, G., Li, Z., & Hu, H. (2018). Two-agent stochastic flow shop deteriorating scheduling via a hybrid multi-objective evolutionary algorithm. *Journal of Intelligent Manufacturing*, *30*(5), 2257-2272.

Funke, J., & Kopfer, H. (2016). A model for a multi-size inland container transportation problem. *Transportation Research Part E: Logistics and Transportation Review*, *89*, 70-85.

Ghezelsoflu, A., Di Francesco, M., Frangioni, A., & Zuddas, P. (2018). A set-covering formulation for a drayage problem with single and double container loads. *Journal of Industrial Engineering International*, *14*(4), 665-676.

Hintsch, T., & Irnich, S. (2018). Large multiple neighborhood search for the clustered vehicle-routing problem. *European Journal of Operational Research*, *270*(1), 118-131.

Hojabri, H., Gendreau, M., Potvin, J.-Y., & Rousseau, L.-M. (2018). Large neighborhood search with constraint programming for a vehicle routing problem with synchronization constraints. *Computers & Operations Research*, *92*, 87-97.

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

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

Keskin, M., & Çatay, B. (2016). Partial recharge strategies for the electric vehicle routing problem with time windows. *Transportation Research Part C: Emerging Technologies*, *65*, 111-127.

Kopfer, H. W., Schönberger, J., & Kopfer, H. (2014). Reducing greenhouse gas emissions of a heterogeneous vehicle fleet. *Flexible Services and Manufacturing Journal*, *26*(1-2), 221-248.

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. *Transportation Research Part E: Logistics and Transportation Review*, *56*, 108-118.

Máhr, T., Srour, J., de Weerdt, M., & Zuidwijk, R. (2010). Can agents measure up? A comparative study of an agent-based and on-line optimization approach for a drayage problem with uncertainty. *Transportation Research Part C: Emerging Technologies*, *18*(1), 99-119.

Macharis, C., & Bontekoning, Y. M. (2004). Opportunities for OR in intermodal freight transport research: A review. *European Journal of Operational Research*, *153*(2), 400-416.

Mancini, S., & Stecca, G. (2018). A large neighborhood search based matheuristic for the tourist cruises itinerary planning. *Computers & Industrial Engineering*, *122*, 140-148.

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

Nossack, J., & Pesch, E. (2013). A truck scheduling problem arising in intermodal container transportation. *European Journal of Operational Research*, *230*(3), 666-680.

Ropke, S., & Pisinger, D. (2006). An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. *Transportation Science*, *40*(4), 455-472.

Shaw, P. 1998. Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems, Berlin, Heidelberg.

Shiri, S., & Huynh, N. (2016). Optimization of drayage operations with time-window constraints. *International Journal of Production Economics*, *176*, 7-20.

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

Song, Y., Zhang, J., Liang, Z., & Ye, C. (2017). An exact algorithm for the container drayage problem under a separation mode. *Transportation Research Part E: Logistics and Transportation Review*, *106*, 231-254.

Sterzik, S., & Kopfer, H. (2013). A Tabu Search Heuristic for the Inland Container Transportation Problem. *Computers & Operations Research*, *40*(4), 953-962.

Sterzik, S., Kopfer, H., & Yun, W.-Y. (2015). Reducing hinterland transportation costs through container sharing. *Flexible Services and Manufacturing Journal*, *27*(2-3), 382-402.

Torkjazi, M., Huynh, N., & Shiri, S. (2018). Truck appointment systems considering impact to drayage truck tours. *Transportation Research Part E: Logistics and Transportation Review*, *116*, 208-228.

Vidović, M., Popović, D., Ratković, B., & Radivojević, G. (2017). Generalized mixed integer and VNS heuristic approach to solving the multisize containers drayage problem. *International Transactions in Operational Research*, *24*(3), 583-614.

Wang, H., Fu, Y., Huang, M., Huang, G. Q., & Wang, J. (2017). A NSGA-II based memetic algorithm for multiobjective parallel flowshop scheduling problem. *Computers & Industrial Engineering*, *113*, 185-194.

Wang, J., Yu, Y., & Tang, J. (2018). Compensation and profit distribution for cooperative green pickup and delivery problem. *Transportation Research Part B: Methodological*, *113*, 54-69.

Wang, R., Lai, S., Wu, G., Xing, L., Wang, L., & Ishibuchi, H. (2018). Multi-clustering via evolutionary multi-objective optimization. *Information Sciences*, *450*, 128-140.

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

Xiang, S., Xing, L., Wang, L., & Zou, K. (2019). Comprehensive learning pigeon-inspired optimization with tabu list. *Science China Information Sciences*, *62*(7).

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. *Transportation Research Part E: Logistics and Transportation Review*, *62*, 136-150.

Xue, Z., Lin, W.-H., Miao, L., & Zhang, C. (2015). Local container drayage problem with tractor and trailer operating in separable mode. *Flexible Services and Manufacturing Journal*, *27*(2-3), 431-450.

Yi, J.-H., Xing, L.-N., Wang, G.-G., Dong, J., Vasilakos, A. V., Alavi, A. H., & Wang, L. (in press). Behavior of crossover operators in NSGA-III for large-scale optimization problems. *Information Sciences*, http://doi.org/10.1016/j.ins.2018.1010.1005.

Zhang, R., Yun, W. Y., & Moon, I. K. (2011). Modeling and optimization of a container drayage problem with resource constraints. *International Journal of Production Economics*, *133*(1), 351-359.

Zhang, R., Lu, J.-C., & Wang, D. (2014). Container drayage problem with flexible orders and its near real-time solution strategies. *Transportation Research Part E: Logistics and Transportation Review*, *61*, 235-251.

Zhang, R., Yun, W. Y., & Kopfer, H. (2015). Multi-size container transportation by truck: modeling and optimization. *Flexible Services and Manufacturing Journal*, *27*(2-3), 403-430.

Zhang, R., Zhao, H., & Moon, I. (2018). Range-based truck-state transition modeling method for foldable container drayage services. *Transportation Research Part E: Logistics and Transportation Review*, *118*, 225-239.

Zhou, Y., & Wang, J. (2018). Critical Link Analysis for Urban Transportation Systems. *IEEE Transactions on Intelligent Transportation Systems*, *19*(2), 402-415.

Zhu, L., & Sheu, J.-B. (2018). Failure-specific cooperative recourse strategy for simultaneous pickup and delivery problem with stochastic demands. *European Journal of Operational Research*, *271*(3), 896-912.