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

Ruiyou Zhang\textsuperscript{a,b,⁎}, Chao Huang\textsuperscript{a}, Junwei Wang\textsuperscript{c}

\textsuperscript{a} College of Information Science and Engineering, Northeastern University, Shenyang 110819, China
\textsuperscript{b} State Key Laboratory of Rolling and Automation, Northeastern University, Shenyang 110819, China
\textsuperscript{c} Department of Industrial and Manufacturing Systems Engineering, The University of Hong Kong, Pokfulam Road, Hong Kong

\textbf{ARTICLE INFO}

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

\textbf{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.

1. Introduction

Short-haulage container transportation including the pre- and end-haulage is usually regarded as container drayage operation (Macharis & Bontekoning, 2004; Namboothiri & 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 & 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, Shi, Powell, & Simao, 2008; Kopfer, Schönberger, & Kopfer, 2014).

Trucks (or tractors, chassis), drivers, and containers can all be regarded as separable resources in container drayage operations (Smilowitz, 2006; Zhang, Yun, & Moon, 2011), generally speaking. Scheduling the transportation resources efficiently is vital for green and safe logistics in urban areas (Wang, Yu, & Tang, 2018; Zhou & 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. Efficient 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 \textit{et al.} (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 (CDTC). 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, Xing, Wang, & Zou, 2019; Yi et al., 2020). 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, Ko, Shin, Hwang, and Kim (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, Mahr, Strou, de Weerdt, and Zuidwijk (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, Muñuzuri, Guadix, and Arango (2013) presented a dynamic algorithm to solve a daily drayage problem with uncertain transit time. Recently, Zhang, Lu, and Wang (2013) developed an efficient two-phase deterministic annealing algorithm to solve the problem.

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, Dessouky, Ioannou, and Chassiakos (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 Braeckers et al. (2013), Braeckers, 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 programming model for a truck appointment system involving drayage operations and maritime container terminal operations. Furthermore, Di Francesco, Gentile, Schirra, Stecca, and Zuddas (2019) proposed three mathematical formulations for a CDT with each truck carrying one container.

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, Graćić, Di Francesco, and Zuddas (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, Popović, Ratković, and Radivojević (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, Ghezelsafii, Di Francesco, Frangioni, and Zuddas (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, Zhang, Lin, Miao, and Yang (2014) examined a local CDT problem considering the transportation of loaded containers under the D&P mode. Subsequently, Xue, Lin, Miao, and Zhang (2015) proposed a max-min ant colony optimization algorithm to deal with a similar problem. Recently, Song, Zhang, Liang, and Ye (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 et al. (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 et al. (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 et al. (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 et al. (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-dominated area such as China. Meanwhile, inbound full container tasks, outbound full container tasks, and outbound empty container tasks usually exist in an import-dominated 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-dominated area and an import-dominated area, individually.

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 et al., 2014; Funke & 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.

Table 1
Comparison between this study and the related literature.

<table>
<thead>
<tr>
<th>Article</th>
<th>Empty container constraints</th>
<th>Time windows</th>
<th>Reducing fixed costs of trucks</th>
<th>Minimizing variable costs of trucks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jula et al. (2005)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Coslovich et al. (2006)</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Imai et al. (2007)</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Cheung et al. (2008)</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Zhang et al. (2011)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Braekers et al. (2013)</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Lai et al. (2013)</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Nossack and Pesch (2013)</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Sterzik and Kopfer (2013)</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Braekers et al. (2014)</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Xue et al. (2014)</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Zhang et al. (2014)</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Sterzik et al. (2015)</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Xue et al. (2015)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Zhang et al. (2015)</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Funke and Kopfer (2016)</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Shiri and Huynh (2016)</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Song et al. (2017)</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Vidović et al. (2017)</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Ghezelsoflu et al. (2018)</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Torkjazi et al. (2018)</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Zhang et al. (2018)</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Di Francesco et al. (2019)</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>This study</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>


Fig. 2. Illustration of the source and destination of containers in drayage operations.
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
- $C = C_{IF} \cup C_{OF} \cup C_{IE} \cup C_{OE}$
- $N = C \cup \{0\}$: 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
- $k$: Amount of time for unpacking a full container $i \in C_{OF}$ or packing a container $i \in C_{IF}$
- $l_i \geq 0$: A truck's traveling time between two locations, $l$ and $l_j$. $i$ and $j$ belong to $C_{IF} \cup C_{OF} \cup \{0, G\}$, where $l$ represents a customer's location of the corresponding container task $i$ if $i \in C_{IF} \cup C_{OF}$, and $l_i$ and $l_j$ stand for the depot and the terminal, respectively.
- $t_G = t_f$
- $[t_L, t_U]:$ Origin time window of a container task $i \in C$, $t_L \leq t_U$
- $[t_L, t_R]:$ Destination time window of a full-container-related task $i \in C_{OF}$, $t_L \leq t_R$

### 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 et al., 2011, 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 set without confusions involved. Vertex 0 is a virtual start/return vertex, representing the initial 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 $i \in N$
- $[T_{L}, T_{U}]:$ Time window of a task vertex $i \in C$, during which the corresponding activities must be started, $T_{L} \leq T_{U}$
- $T_{ij}$: Transfer-time attribute of an arc $(i, j) \in A$
- $a_{ij} \in \{1, -1, 0\}$: Change-number attribute of an arc $(i, j) \in A$
- $\Delta_{ij} \geq 0$: Change-time attribute of an arc $(i, j) \in A$

The calculation procedure of the service time $T_i$ and the time window $[T_{L}, T_{U}]$ 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 et al. (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.

#### 4.2. Mathematical model

Several decision variables were introduced as follows:

- $x_{ij}$: 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:

\begin{align}
\text{min } f_1 &= \sum_{i \in C} x_{0i} \\
\text{min } f_2 &= \sum_{j \in C} (z_{j0} + T_{j0}x_{j0} - z_{0j}) \\
s. t. \sum_{j \in \mathcal{N}} x_{ij} = \sum_{j \notin \mathcal{N}} x_{ji} = 1, \quad \forall i \in C \\
T_{L} \leq y_{ij} \leq T_{U}, \quad \forall i \in C \\
T_{ij} + y_{ij} - z_{ij} \leq T_{ii} + T_{jj} + (1 - \chi_{ij}), \quad \forall j \in C \\
T_{ij} + z_{ij} - T_{ij} \leq T_{ii} + T_{jj} + (1 - \chi_{ij}), \quad \forall (i, j) \in A, i \neq j, j \neq 0 \\
T_{ij} + T_{ij} - y_{ij} \leq (T_{ii} + T_{jj} + T_{ij})(1 - \chi_{ij}), \quad \forall (i, j) \in A, i \neq j, j \neq 0 \\
-\chi_{ij} \leq T_{ij}x_{ij}, \quad \forall j \in C \\
\chi_{ij} \leq T_{ij}x_{ij}, \quad \forall i \in C \\
\chi_{ij} \leq T_{ii}x_{ij}, \chi_{ij} \leq T_{jj}x_{ij}, \quad \forall i \in C \\
\sum_{\{(p, q)|e_{ij}=1,\chi_{pq}=1\}} a_{pq} + m \geq 0, \quad \forall (i, j) \in A_i \\
x_{ij} \in [0, 1]. \quad \forall (i, j) \in A \\
y_{ij} \in \mathbb{R}. \quad \forall i \in C \\
z_{ij} \in \mathbb{R}. \quad \forall (i, j) \in A
\end{align}

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 $x_{ij}$ is 0. Similarly, Constraint (10) and the objective function (2) ensure that the value of $z_{ij}$ is 0 if the value of $x_{ij}$ is 0.
Constraint (11) will be explained in detail, where $A_1 = \{(i, j) | (i, j) \in A, \alpha_{ij} = -1\} \text{ and } A_2 = \{(i, j) | (i, j) \in A, \alpha_{ij} \neq 0\}$.

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 $(p, q) \in A_1$, which satisfy $x_{pq} = 1$, and $z_{pq} + \beta_{pq} \leq z_{ij} + \beta_{ij}$ should be summed. That is Constraint (11). Constraints (12)–(14) define the types of variables, where $R$ is the set of real numbers.

Main differences between the aforementioned model ((11)–(14)) and that presented by Zhang et al. (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 et al. (2011). Fourthly and finally, Constraint (11) is more specific and efficient than the corresponding constraint in Zhang et al. (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),

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

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

$$z_{ij} + \beta_{ij} - z_{pq} - \beta_{pq} \geq -M(1 - u_{ijpq}), \quad \forall (i, j) \in A_1, (p, q) \in A_2 \tag{16}$$

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

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

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

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

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

where $M$ is a large enough positive constant, and $\varepsilon$ 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 $x_{ij} = 0$ or $x_{pq} = 0$ or $z_{ij} = 0$ or $z_{pq} = 0$ 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

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

and

$$z_{ij} + \beta_{ij} - z_{pq} - \beta_{pq} + \varepsilon \leq Mu_{ijpq}, \quad \forall (i, j) \in A_1, (p, q) \in A_2 \tag{23}$$

respectively.

If $z_{ij} + \beta_{ij} - z_{pq} - \beta_{pq} \geq 0$, Constraint (22) indicates that $u_{ijpq} = 1$ and Constraint (22) always holds. On the contrary, if $u_{ijpq} = 1$, Constraint (22) indicates that $z_{ij} + \beta_{ij} - z_{pq} - \beta_{pq} \geq 0$ 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 $i \in C_{inr} \cup C_{or}$ while $j \in C_{of} \cup C_{of}$ for any arc $(i, j) \in A$. Similarly, the arcs $(p, q) \in A_1$ can be divided as: $p = 0$ and $q \in C_{of} \cup C_{of}$; $p \in C_{of} \cup C_{of}$ and $q = 0$; $p \in C, q \in C, \sigma_{pq} \neq 0$.

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

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

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

$$z_{ij} + \beta_{ij} - z_{pq} - \beta_{pq} \geq -M_{ijpq}(3 - u_{ijpq} - x_{ij} - x_{pq}), \tag{24}$$

$$z_{ij} + \beta_{ij} - z_{pq} - \beta_{pq} + \varepsilon \leq M_{ijpq}(2 + u_{ijpq} - x_{ij} - x_{pq}), \tag{25}$$

where

$$M_{ijpq} = T_{ij} + T_{ij} + \beta_{ij} + T_{ij} + T_{ij} + \beta_{ij} + \varepsilon. \tag{26}$$

Case 2: $i = 0, j \in C_{of} \cup C_{of}, p \in C_{of} \cup C_{of}, q = 0$.

Constraints (16) and (17) become

$$z_{ij} + \beta_{ij} - z_{pq} - \beta_{pq} \geq -M_{ijpq}(3 - u_{ijpq} - x_{ij} - x_{pq}), \tag{27}$$

$$z_{ij} + \beta_{ij} - z_{pq} - \beta_{pq} + \varepsilon \leq M_{ijpq}(2 + u_{ijpq} - x_{ij} - x_{pq}), \tag{28}$$

where

$$M_{ijpq} = T_{ij} + T_{ij} + \beta_{ij} + T_{ij} + T_{ij} + \beta_{ij} + \varepsilon. \tag{29}$$

Case 3: $i = 0, j \in C_{of} \cup C_{of}, p \in C, q \in C, \sigma_{pq} \neq 0$.

Constraints (16) and (17) become

$$z_{ij} + \beta_{ij} - z_{pq} - \beta_{pq} \geq -M_{ijpq}(3 - u_{ijpq} - x_{ij} - x_{pq}), \tag{30}$$

$$z_{ij} + \beta_{ij} - z_{pq} - \beta_{pq} + \varepsilon \leq M_{ijpq}(2 + u_{ijpq} - x_{ij} - x_{pq}), \tag{31}$$

where

$$M_{ijpq} = T_{ij} + T_{ij} + \beta_{ij} + T_{ij} + T_{ij} + \beta_{ij} + \varepsilon. \tag{32}$$

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

Constraints (16) and (17) become
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 \( \Omega_1 = C_{\text{in}} \cup C_{\text{dt}}, \Omega_2 = C_{\text{in}} \cup C_{\text{de}} \). Let \( X = (0) \) be an incomplete solution and \( n^0 = \) be the number of tasks 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 \( \|\vec{e}\| \in \Omega_1, y + T_{i(x)} + T_{j(x)} \leq T_{i(y)} \) = \( \varnothing \), let \( \Omega = \{\|\vec{e}\| \in \Omega_1, y + T_{i(x)} + T_{j(x)} \leq T_{i(y)} \} \) (44) denote the set of feasible candidate vertices currently, where \( e(X) \) is a function denoting the last item of the solution \( X \), and \( \varnothing \) denotes an empty set.

Step 3: If \( \Omega = \varnothing \), let \( \Omega = \{\|\vec{e}\| \in \Omega_1, y + T_{i(x)} + T_{j(x)} \leq T_{i(y)} \} \). (45)

Step 4: If \( \Omega = \varnothing \), go to Step 5; otherwise, go to Step 6.

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

Step 6: Let \( Y = \max(y + T_{i(x)} + T_{j(x)}, T_{i(y)}) \), \( \forall \vec{e} \in \Omega \).

\( f = \arg\min_{\vec{e} \in \Omega} Y \). (46)

Let \( X = X \cup f \) and \( y = Y_f \). If task \( f \) is an inbound task, let \( \eta = 1 \) and \( \Omega_1 = \Omega_1 \setminus \{f\} \); otherwise let \( \eta = 0 \) and \( \Omega_2 = \Omega_2 \setminus \{f\} \). 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 \( |\{\vec{e}\}| \), where \(|\cdot|\) denotes the number of items in the given set, and \(|\cdot|\) 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
\[ y_j = t_{i_j}. \]  
(48)

If \( j \in C \) is not the first container vertex handled by a truck, let
\[ y_j = \min(\{y_i + t_{i_j} + t_{i_0}, t_{i_j}\}), \]  
(49)

where \( i \) is the preceding container vertex of vertex \( j \). For any container vertex \( j \in C \), let
\[ z_{i_0} = y_j - t_{i_0}, \]  
(50)

where \( i \) is the preceding container vertex of vertex \( j \). For the last vertex \( i \in C \) handled by a truck, let
\[ z_{i_0} = y_j + t_{i_0}. \]  
(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 \( y_j < y_i - t_i - t_{i_0} \), let
\[ z_{i_0} = y_j - t_{i_0}, \]  
(52)
\[ y_j = \min(\{y_{i_0}, y_j - t_i - t_{i_0}\}) \]  
(53)

Given a solution \( X \) and the starting time of any vertex and arc, it is easy to check the 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 \( \Lambda \) 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 et al. (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 & Pesch, 2013; Shi & 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 h’ traveling distance by truck. The time for picking up or dropping off a container was set to be 5 min.

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 \( \alpha \), \( K \), and \( \Lambda \) 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
Table 4
Potential values of each parameter for different instance scales.

<table>
<thead>
<tr>
<th>Instance scale</th>
<th>α</th>
<th>K</th>
<th>Λ(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>0.2</td>
<td>5, 10, 15</td>
<td>0.1, 0.2, 1.0</td>
</tr>
<tr>
<td>Medium</td>
<td>0.2</td>
<td>20, 25, 30</td>
<td>2.0, 5.0, 10.0</td>
</tr>
<tr>
<td>Large</td>
<td>0.2</td>
<td>35, 40, 45</td>
<td>500, 600, 700</td>
</tr>
</tbody>
</table>

Table 5
Tuning results of parameters.

<table>
<thead>
<tr>
<th>Instance scale</th>
<th>α</th>
<th>K</th>
<th>Λ(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>0.4</td>
<td>15</td>
<td>0.2</td>
</tr>
<tr>
<td>Medium</td>
<td>0.3</td>
<td>30</td>
<td>5.0</td>
</tr>
<tr>
<td>Large</td>
<td>0.2</td>
<td>45</td>
<td>700.0</td>
</tr>
</tbody>
</table>

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.

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 s. 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 s, which was much faster than that of using CPLEX (the average running time was longer than 300.00 s). 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 s 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 s while that of the LNS algorithm was only 5.00 s. 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 & Regan, 2002; Zhang et al., 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 × 7) repeats runs for 700 s as the time limit reaches. Quite few exceptions are that the LNS algorithm stops before 700 s 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.

7.5. Comparison to Zhang et al. (2011)

The CDTRC problem studied in this research degenerates to be the problem presented by Zhang et al. (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 et al. (2011).

Table 6
Experiments for small-scale instances.

<table>
<thead>
<tr>
<th>Instance</th>
<th>m</th>
<th>No. of tasks</th>
<th>CPLEX</th>
<th>LNS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Obj.</td>
<td>CPU time (s)</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>5 (2, 2, 0, 1)</td>
<td>2362</td>
<td>0.23</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>9 (4, 4, 0, 1)</td>
<td>4923</td>
<td>0.27</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>15 (6, 7, 2, 0)</td>
<td>6136</td>
<td>1.01</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>17 (9, 6, 0, 2)</td>
<td>8615</td>
<td>1.23</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>19 (10, 6, 0, 3)</td>
<td>7572</td>
<td>19.84</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>20 (10, 6, 0, 4)</td>
<td>8690</td>
<td>17.58</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>22 (7, 11, 4, 0)</td>
<td>9237</td>
<td>9.77</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>25 (12, 9, 0, 4)</td>
<td>10,116</td>
<td>5.04</td>
</tr>
<tr>
<td>9</td>
<td>11</td>
<td>28 (11, 13, 4, 0)</td>
<td>11,248</td>
<td>18.45</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>30 (15, 11, 0, 4)</td>
<td>12,169</td>
<td>3312.00</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>30 (15, 11, 0, 4)</td>
<td>12,732</td>
<td>22.15</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td>22.15</td>
<td>18.45</td>
</tr>
</tbody>
</table>
We also solved the 17 instances using 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 et al. (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 (Fu, Wang, Tian, Li, & Hu, 2018; Wang, Fu, Huang, Huang, & Wang, 2017). 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 et al. (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 et al. (2011). That is, the null hypothesis is rejected. In other words, the LNS algorithm provides significantly better solutions than the algorithm of Zhang et al. (2011) in much shorter running time. **Table 10** presents the statistical results of the t-test.

**Table 10** indicates 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 et al. (2011) in much shorter running time. Again, this validates the LNS algorithm.

### 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 \( k_2 = k_{12} = k_{11} = k_{02} \) and increase the value of \( k_1 \) from 0 to 8000, and then set \( k_1 = 1 \) and \( k_2 \) remains, 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.

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.

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

![Fig. 3. Objective values under different values of m.](image-url)
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 equal to 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 et al. (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.

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 of China (No. N16040411), National Key R&D Program of China (2017YFB0306401), and the Open Research Fund from the State Key Laboratory of Rolling and Automation, Northeastern University, China (No. 2018RALKFRT008). 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 (Nos. 201611159213, 201811159163), and by the open project funded by State Key Laboratory of Synthesitical Automation for Process Industries of China (PAL-N201802).

References


