Container drayage problem with flexible orders and its near real-time solution strategies

Ruiyou Zhang\textsuperscript{a,}*, Jye-Chyi Lu\textsuperscript{b}, Dingwei Wang\textsuperscript{a}

\textsuperscript{a}College of Information Science and Engineering, Northeastern University, Shenyang 110819, China
\textsuperscript{b}School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA

\textbf{A B S T R A C T}

This article studies a container drayage problem with flexible orders defined by using \textit{requiring} and \textit{releasing} attributes as a unified formulation of various order types. A determined-activities-on-vertex (DAOV) graph introduces a temporary vertex set to formulate different truck statuses. The problem is formulated as a mixed-integer nonlinear programming model based on the DAOV graph. Four strategies including a window partition based (WPB) strategy are presented and evaluated extensively to solve the problem. Results indicate that the WPB method could solve the problem effectively and efficiently. Furthermore, this method is robust considering the operating time biases compared to other algorithms.

\textsuperscript{*} This work was supported in part by National Natural Science Foundation of China (Nos. 71001019, 70931001, 71171039, 71001018, and 71021061), Fundamental Research Funds for the Central Universities (No. N120404016), and China Postdoctoral Science Foundation (No. 2012M520642).

\textsuperscript{\textcopyright} 2013 Elsevier Ltd. All rights reserved.

\section{1. Introduction}

Container logistics has grown rapidly in the last two decades. Correspondingly, there have been many substantial literature solving operational research problems in container logistics. A series of review articles such as those by Vis and de Koster (2003), Steenken et al. (2004), and several special issues including those by Chew et al. (2010, 2011) about seaport container terminal operations and freight transportation logistics have been published.

The long distance of container transportation usually requires vessels or trains as main transportation tools. However, a short distance of transportation by truck is necessary in most scenarios since the transportation by vessel or by train lacks door-to-door services (Zhang et al., 2010). As shown in Fig. 1, both the transportation from an initial consignee to a terminal (seaport or railway hub station) and that from a terminal to a final receiver use trucks. This segment of transportation by truck is usually called a drayage. Container drayage service in a local area is a critical issue in container logistics. It accounts for a significant portion of the total transportation costs and is the key source for shipment delays, road congestions, and disruptions in international logistics (Cheung et al., 2008).

Container drayage problems are usually considered as pickup-and-delivery problems (PDPs). See Parragh et al. (2008a, 2008b) for a comprehensive review and a classification of normal PDPs. However, container drayage problems differ from normal PDPs in several ways. On one hand, containers are another type of transportation resources besides trucks in container drayage problems. Freight must be packed within containers before it is transported. On the other hand, some drayage
orders (tasks) are flexible. Some properties of flexible orders are not given in advance. Orders of empty containers are typically flexible orders in container drayage problems. For example, in certain orders, an empty container located at a terminal should be picked up and delivered to a depot or another alternative location, e.g., a shipper of some freight (Braekers et al., 2013).

Quite recently, a few articles, e.g. Zhang et al. (2010, 2009), study container drayage problems with both well defined and flexible orders. In their problems, containers are not only transportation resources but also goods to be transported, which is a more realistic viewpoint compared to that of most existing articles. However, real-life scenarios in container drayage are more complicated. For example, some containers could be immediately ready for future use after they are delivered to their shippers and emptied. On the other hand, some containers might be returned to a depot and cleaned (and/or repaired) before further packing of freight. This paper defines two binary attributes of drayage orders which reflect whether an order requires and releases an empty container. The definition of orders using these attributes as well as the origin and the destination of orders covers all the situations mentioned above.

Container drayage problems are static problems in most existing articles such as those by Jula et al. (2005). In such articles, container drayage schedulers collect information about drayage orders, truck maintenances, and road conditions once in each time horizon (typically each day). This information is used when schedulers make decisions (e.g., at eight am every morning) and the decision does not change until the next decision-making point.

Recently, the concept and use of Internet of Things (IoT) have spread rapidly (Atzori et al., 2010). In the era of IoT, almost everyone and possibly every device can communicate and connect with each other based on the technologies of radio frequency identification (RFID) and communication (de Saint-Exupery, 2009; Ngai et al., 2008). For example, certain containers have RFID tags to provide schedulers with their location and freight information. Various sensors might monitor trucks’ health or maintenance conditions. This information would be critical to alter drayage decisions to limit the impact of logistic disruptions. Thus, some container drayage problems in real-life are becoming or will become quite dynamic.

However, few articles now consider container drayage problems as dynamic problems. For example, Yang et al. (2004) studied a dynamic multivehicle truckload pickup-and-delivery problem. Escudero et al. (2013) also proposes a dynamic approach to solve a daily drayage problem with time uncertainties. Certain information about the future, e.g., probabilistic information, is required in most of these articles. But such information is usually not available. Furthermore, the definition of drayage orders in such articles differs from that which is used in this research. See Section 2 for a detailed review and a comparison of related literature.

This research investigates a dynamic container drayage problem with flexible orders. Firstly, a unified definition of drayage orders is proposed. The container drayage problem with this definition of orders involves both fixed and flexible orders. The double properties of containers (as transportation resources and goods) are considered in this problem. This dynamic container drayage problem is first formulated as a determined-activities-on-vertex (DAOV) graph inspired by Zhang et al. (2009) with temporary vertices introduced to describe the trucks working at the decision epoch. Then, based on the DAOV graph, a mixed-zero-one nonlinear programming model as an extension of the asymmetric multiple-traveling salesman problem with time windows (am-TSPTW) formulates this problem. Secondly, a number of strategies are presented to solve the drayage problem. One strategy handles only the updated information when interruptions occur. The other strategies re-optimize the drayage problem each time when interruptions occur. These re-optimization strategies include solving the problem using commercial software directly, or after a simple discretization scheme, or based on a partition of time windows. Finally,
these strategies are validated and evaluated extensively. They are compared with each other and with a lower bound of the problem. Results indicate that the re-optimization strategies, especially the one based on window partitions, could solve the problem in a short period (2 min or shorter) with a high precision. Furthermore, the window partition based strategy has been proven to be very robust considering the bias of operational time.

The remainder of this paper is organized as follows. Section 2 surveys the literature related to drayage problems. The container drayage problem is formally defined in Section 3 and then mathematically formulated using the DAOV graph in Section 4. The solution strategies are presented and validated extensively in Sections 5 and 6, respectively. Finally, Section 7 concludes this paper.

2. Literature review

This section surveys the literature related to this research. Specifically, Sections 2.1 and 2.2 review static container drayage problems and dynamic pickup-and-delivery problems, respectively. The latter survey scope is a little wider than the former one since there are few articles addressing dynamic container drayage problems. The differences between this research and that from most related articles are also presented.

2.1. Static container drayage problems

Recently, a number of articles began to focus on hinterland container transportation systems. For example, Namboothiri and Erera (2008) study a container truck transportation problem given an appointment control system. Cheung et al. (2008) and Loo (2010) investigate cross-boundary container drayage problems with the Hong Kong-Pearl River Delta region as the considered area. Stahlbock and Voß (2008) summarize and update the literature about operational problems at container terminals. Several international journals, e.g., OR Spectrum, have published special issues about drayage operations (Chew et al., 2011; Günther and Kim, 2006).

PDPs are most direct to formulate container drayage problems since the containers are usually required to be picked up from somewhere and delivered to somewhere else. Imai et al. (2007) formulates a drayage problem with full container loads as a zero-one linear programming model and designs a Lagrangian relaxation-based heuristic to solve it. Caris and Janssens (2009) present a local search heuristic to solve a similar pre- and end-haulage container transportation problem. Ting and Liao (2013) relax the constraint that all pickup nodes must be visited and formulate a drayage problem as a selective PDP. A drayage problem falls into the variances of the multiple-Traveling Salesman Problem (m-TSP) if the pickup and delivery nodes of an order are merged. Therefore, Jula et al. (2005) formulates a drayage problem as an am-TSPTW with social constraints and solves it using a hybrid method of dynamic programming and genetic algorithm. Chung et al. (2007) develops several similar mathematical models for drayage problems. Vidović et al. (2011) and Popović et al. (2012) focus on modeling drayage problems with both 40 ft and 20 ft containers. Moreover, attribute-decision models are also used to formulate container drayage problems (Cheung et al., 2008; Powell et al., 2007).

Empty containers in container drayage problems are required before packing freight and released after unpacking freight. From this viewpoint, container transportation by truck differs from other modes of transportation significantly. Ileri et al. (2006) models container drayage transportation as an assignment problem considering the implicitly required movements of empty containers. Reinhardt et al. (2012) combines import orders and export orders into pairs and models a similar problem as a linear integer programming model. Sterzik and Kopfer (2013) define a comprehensive mathematical formulation which considers the routing and scheduling of vehicles and the repositioning of empty containers simultaneously. Quite recently, Meisel and Kopfer (in press) extend the abovementioned research and define two types of transportation resources to formulate the drayage operation.

A few later articles consider flexible orders in container drayage problems. Orders of empty container are typically flexible orders. Either the pickup or delivery location of an empty container in such an order is not given but should be optimized with the routing and scheduling of trucks. Smilowitz (2006) applies a multi-resource routing problem to container drayage operations. Considering flexible orders, Zhang et al. (2009) defines a container drayage problem with a single terminal and Zhang et al. (2010) extends it to a multi-terminal multi-depot case. Zhang et al. (2011b) further extend the problem by introducing a limited number of empty containers at depots. Braekers et al. (2013) also have proposed an efficient integrated planning scheme of loaded and empty container movements very recently. Similarly, Lai (2013) studies an integrated problem of truck routing and empty container repositioning. Sterzik et al. (in press) investigates the cost reduction in container drayage by sharing empty containers among trucking companies. See Braekers et al. (2011) for an excellent review of empty container movements. The container drayage problem studied in this research involves double properties of empty containers and flexible orders based on a unified definition of orders. Furthermore, containers might belong to either the customers or the trucking company.

2.2. Dynamic pickup-and-delivery problems

Container drayage problems belong to the large family of vehicle routing problems (VRPs). Goel and Gruhn (2008) present a general vehicle routing problem that consists of numerous components such as time windows and vehicle capacities. Pillac
et al. (2013) focus on dynamic VRPs and give a comprehensive review about their solution methods. A few articles address dynamic PDPs. For example, Yang et al. (1999) develop several online algorithms to assign and schedule truck fleets. Later, Yang et al. (2004) present re-optimization strategies of truckload PDPs with real-time information. Powell et al. (2007) also develop an attribute-decision model for dynamic freight transportation problems. See Berbeglia et al. (2010) for an extensive survey of dynamic PDPs.

There are few articles focusing on dynamic container drayage problems. Wang and Regan (2002) mention that local truckload drayage problems could be dynamic and that the problems could be resolved when more information becomes available. However, they do not mention how to model the problem when there is information updates. Coslovich et al. (2006) build an integer programming model of a drayage problem in which both the operational costs in a current day and the container repositioning costs in upcoming days are considered. However, logistic information never changes over a given day in their problems. Quite recently, Mähr et al. (2010) compare an agent-based method and an online optimization for a drayage problem with uncertainties. Similarly, Escudero et al. (2013) study the re-optimization of picking up and delivering containers with an uncertainty of transit times. Furthermore, Zhang et al. (2011a) and Long et al. (2012) study a dynamic version of a drayage problem with flexible orders and an empty container repositioning problem, respectively.

Most articles about dynamic drayage problems do not consider either the transportation resource attribute of containers or flexible orders. Although certain formulations of drayage problems, e.g., am-TSPTW, are special cases of more general formulations such as VRP with time windows (VRPTW), good algorithms for generic formulations might not be optimum algorithms to solve special formulations (Escudero et al., 2013; Jula et al., 2005). Furthermore, some formulations and algorithms for dynamic drayage problems depend on probabilistic information of the future while such information is not available or at least not accurate in real-life scenarios. Therefore, this research presents a number of near real-time solution strategies for drayage problems that do not require such probabilistic information. See the next section for a detailed problem definition of our research.

3. Problem definition

3.1. The drayage system

A typical inland container transportation system consists of a trucking company, several intermodal terminals, and industrial customers. The trucking company provides drayage services in the local region. It possesses a number of depots, trucks, and containers. The depots are warehouses for stacking their containers and parking the trucks. We assume that any depot has enough empty containers and enough space for their containers and trucks. Let \( V_D \) and \( L_i \) be the set of depots and the location of a depot \( i \in V_D \), respectively. Trucks can visit the depots at any time because the depots are properties of the trucking company.

Draayage services include the transportation of freight (within containers) and also some empty containers between the customers and terminals in a local region. Some customers do not have their own containers while others do. Containers can be used either within the local region or between different regions. However, the trucks are used in the local region only. Each truck can carry only one container at a time.

3.2. A unified formulation of drayage orders

A drayage order is decomposed into several orders if it involves more than one container so that there is only one container in each order (Sterzik and Kopfer, 2013). Drayage orders could be classified into different types. We first list the types of drayage orders that have been reported in literature. Drayage orders are usually classified into inbound and outbound orders in terms of order directions, which are derived from the well-known research field of logistics. Transportation of loaded containers is most widely studied in drayage problems. We have two types of loaded container orders as (i) inbound loaded container order and (ii) outbound loaded container order. Two basic assumptions for these two order types exist in literature. One assumption (a) is that the trucks always wait at customers’ sites for the containers to be (un)packed (Srour et al., 2010). The other assumption (b) is that the trucks leave the customers’ sites while the containers are being (un)packed and later other trucks come back to pick up the already (un)packed containers (Braekers et al., 2013).

There are empty container orders besides loaded container orders in container drayage problems. Logistic companies need to transport empty containers between regions periodically due to trade imbalances between regions (Zhang et al., 2009). Therefore, in certain drayage orders, only empty containers need to be transported. For example, empty containers should be picked up at a terminal in some orders. The destinations of these empty containers are irrelevant to the corresponding shippers and could be chosen by the decision maker. On the other hand, empty containers should be delivered to a terminal with their origins undefined in other orders. We get two empty container order types as (iii) inbound empty container order and (iv) outbound empty container order (Braekers et al., 2013). Either the origin or the destination of such an order is not defined in advance. Therefore, these two types of orders are typically flexible orders.

The following order types have not been reported but actually exist in real-life scenarios. First, loaded containers need to be transited between two sites within the local region in some orders. Such orders need neither packing nor unpacking empty containers. Second, freight needs to be transported between two sites within the local region in some orders. Empty
Table 1
Formulation of various order types.

<table>
<thead>
<tr>
<th>Type</th>
<th>Main activities, ( O_i ) and ( D_i ) of the order ( i )</th>
<th>( p_i^L )</th>
<th>( p_i^R )</th>
</tr>
</thead>
<tbody>
<tr>
<td>i-a</td>
<td>Pick up a loaded container at a terminal ( O_i ), deliver it to its receiver ( D_i ), unpack it, and remove the emptied container.</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>i-b1</td>
<td>Pick up a loaded container at a terminal ( O_i ) and deliver it to its receiver ( D_i ).</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>i-b2</td>
<td>Pick up an empty container at its receiver ( O_i ) and deliver it to a terminal ( D_i ).</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>ii-a</td>
<td>Deliver an empty container to its shipper ( O_i ), pack the freight, and deliver it to a terminal ( D_i ).</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>ii-b1</td>
<td>Delivery an empty container to the shipper ( O_i ) and another site ( D_i ).</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>ii-b2</td>
<td>Pick up the loaded container at its shipper ( O_i ) and deliver it to a terminal ( D_i ).</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>iii</td>
<td>Pick up an empty container at a terminal ( O_i ) and its receiver ( D_i ).</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>iv</td>
<td>Deliver an empty container to a terminal ( O_i ) and another site ( D_i ).</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>v</td>
<td>Pick up a loaded container at a site ( O_i ) and deliver it to another site ( D_i ).</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>vi</td>
<td>Deliver an empty container to a site ( O_i ), pack the freight, deliver the loaded container to another site ( D_i ), unpack it, and remove the emptied container.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>vii</td>
<td>Transport a customer-owned container from a site ( O_i ) to another site ( D_i ).</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

* The ‘a’ or ‘b’ following ‘i’ or ‘ii’ in order types denotes the former and latter assumptions for loaded containers orders. Furthermore, each loaded container order (type i or ii) under the latter assumption (b) is decomposed into two orders. The first one is labeled with ‘1’ and the second one is labeled with ‘2’.

Containers should be delivered to the origins before packing the freight and should be picked up after they are unpacked. Thus, we get two special types of orders numbered as (v) and (vi). The empty containers in order types (i) through (vi) belong to the trucking company. However, some customers have their own containers and could pack and unpack the containers. The trucking company does not have to worry about the handling of these empty containers. Therefore, we get the last order type numbered as (vii).

The authors introduce a unified formulation of order types. An order \( i \) (a container or a container of freight) has an origin (pickup site), denoted by \( O_i \), and a destination (delivery site), denoted by \( D_i \), as shown in Fig. 2. The origin and the destination might be different or the same site. The requiring attribute of an order \( i \), denoted by \( p_i^L \), is either 1 (if the order requires an empty container at its origin) or 0 (otherwise). The releasing attribute of an order \( i \), denoted by \( p_i^R \), is either 1 (if the order releases an empty container at its destination) or 0 (otherwise). Such a formulation of order types in itself is a contribution. It could cover all types of orders mentioned above. See Table 1 for the main activities and the formulation of different order types. Note that all orders become well defined in this research. We will not use the commonly used concepts of inbound and outbound hereafter. See Section 4 especially, e.g., Section 4.1.4, for the model of the problem under this formulation of order types.

Furthermore, a series of activities correspond to the origin and another series of activities correspond to the destination of an order. Let \( t_i^L \) and \( t_i^U \) be the operating times of the series of activities at the origin and the destination of an order \( i \), respectively. The origin (customer or terminal) usually provides a time window, denoted by \( [a_i^L, b_i^L] \) or \( [a_i^U, b_i^U] \), to limit the starting time of the origin activities of an order \( i \). Similarly, the destination of an order \( i \) might also have a time window, denoted by \( [a_i^D, b_i^D] \). The two windows cannot have any conflict, i.e., we have \( a_i^L + t_i^U \leq b_i^D \). Furthermore, a constant represents the traveling time between any two locations \( i \) and \( j \), denoted by \( l(i, j) \).

3.3. The dynamic container drayage problem

This research studies the container drayage problem from the perspective of a scheduler of a sub fleet of trucks. It is assumed that all trucks are located at depots at the beginning of a time horizon. Each truck should return to a depot after finishing all its orders. However, the starting depot and returning depot of a truck could be different. The final number of trucks might also differ from the initial number of trucks at each depot.

The scheduler should reroute and reschedule the trucks to finish the drayage orders each time interruptions occur. The scheduling criterion minimizes the total operating time of all involved trucks since it also reflects the total operating cost (at least to a certain degree). The total operating time includes the waiting time during operations. This is the dynamic container drayage problem. This problem generalizes the initial routing and scheduling of trucks at the beginning of the time horizon obviously.

4. Mathematical formulation

It is relatively hard to build mathematical programming models for container drayage problems directly. One major reason is that such problems involve not only routing and scheduling of trucks but also repositioning of empty containers. Zhang et al. (2009) propose a DAOV graph to model a static container drayage problem. It falls into a generalized
multiple-traveling salesman problem. This research extends the DAOV graph to model the dynamic container drayage problem with more generic types of drayage orders as discussed in Section 3.

4.1. The DAOV graph

Let \( G = (V, A) \) be a DAOV graph to formulate the drayage problem. Here,
\[
V = V_D \cup V_O \cup V_T,
\]
is the vertex set, where \( V_D, V_O, \) and \( V_T \) are the start/return vertex set, order vertex set, and temporary vertex set, respectively. Moreover,
\[
A = \{(i, j) | i \in V_D, j \in V_O; \text{ or } i \in V_O \cup V_T, j \in V_O \cup V_D; i \neq j\}.
\]
is the arc set.

4.1.1. Start/return vertex

A start/return vertex \( i \in V_D \) consists of the initial start and final return to the depot \( i \) (Zhang et al., 2009). It has a single attribute \( n_i \), as the number of trucks parking at the depot at the decision epoch (decision-making time).

4.1.2. Order vertex

An order vertex \( i \in V_O \) consists of the continuous determinate activities corresponding to the order \( i \) (Zhang et al., 2009). Note that the newly arrived information could include appending of priority orders, modifying or canceling of existing orders, or modifying of road or truck conditions. The order vertex set \( V_O \) includes appended orders but excludes canceled orders. It corresponds to all orders that have not been visited at the decision epoch. The activities of an order vertex include both the origin activities and destination activities. An order vertex has two attributes: the service time as the amount of time that its activities last, and the time window as a period when its origin activities should begin.

At first glance, the time window and service time of an order vertex seem to be its origin time window and the sum of operating times at both the origin and the destination, respectively. However, we need to improve this formulating method since an order vertex has also a time window at its destination. Jula et al. (2005) reported that if a truck starts too late at its origin, it will violate the latest limit of its destination time window. As a result, they push back the latest limit of the time window of an order vertex according to its destination time window. Furthermore, Zhang et al. (2009) found that the starting time at the origin should be neither too late nor too early. Therefore, they delay the earliest limit of the time window of an order vertex so that there is no avoidable wait at its destination. We modify both limits of the time window similarly to how it is done in Zhang et al. (2009).

If we shift back the destination time window of an order \( i \) in time by \( t_i^0 \), it becomes \([a_i^0 - t_i^0, b_i^0 - t_i^0] \). Fig. 3 shows different relationships between the two time windows \([a_i^0 - t_i^0, b_i^0 - t_i^0] \) and \([a_i^0, b_i^0] \). If the two windows overlap, as shown in Fig. 3(a), the overlapped part is the time window of the order vertex. A truck starting during such a time window at the origin could start the destination activities without waiting after it arrives at the destination. The service time of the order vertex is the sum of the operating times at its origin and destination. If \([a_i^0 - t_i^0, b_i^0 - t_i^0] \) is ahead of \([a_i^0, b_i^0] \), as shown in Fig. 3(b), the time window of the order vertex degenerates into a single point \( b_i^0 \). The gap between the two windows is the unavoidable waiting time at the destination. Adding it to the sum of the operating times at the origin and destination is the service time of the order vertex. If \([a_i^0 - t_i^0, b_i^0 - t_i^0] \) is behind \([a_i^0, b_i^0] \), as shown in Fig. 3(c), it is not possible to finish the order. This is the reason that Section 3.2 introduces the basic assumption that \( a_i^0 + t_i^0 \leq b_i^0 \), i.e., \([a_i^0 - t_i^0, b_i^0 - t_i^0] \) cannot be behind \([a_i^0, b_i^0] \). In summary, the time window and service time of an order vertex \( i \), denoted by \([a_i, b_i]\) and \( t_i \), respectively, could be formulated as:

\[
a_i = \min \{\max(a_i^0, a_i^0 - t_i^0), b_i^0\}, \quad \forall i \in V_O,
\]

\[
b_i = \min(b_i^0, b_i^0 - t_i^0), \quad \forall i \in V_O,
\]

\[
t_i = \max(a_i^0 - b_i^0, t_i^0) + t_i^0, \quad \forall i \in V_O.
\]

\[
\begin{array}{c}
[a_i^0 - t_i^0, b_i^0 - t_i^0]: \\
[a_i^0, b_i^0]: \\
[a_i, b_i]:
\end{array}
\]

(a) (b) (c)

Fig. 3. Different relationships between two time windows. (a) The two time windows overlap. (b) \([a_i^0 - t_i^0, b_i^0 - t_i^0] \) is ahead of \([a_i^0, b_i^0] \). (c) \([a_i^0 - t_i^0, b_i^0 - t_i^0] \) is behind \([a_i^0, b_i^0] \).
4.1.3. Temporary vertex

The trucks working at the decision epoch, say \( u \), might be in various statuses. Neither start/return vertices nor order vertices could formulate them. Therefore, a temporary vertex \( i \in V_T \) is introduced for each of such trucks. A truck working at time \( u \) is either visiting an order vertex or transferring between vertices. A temporary vertex is defined as the remaining activities of the order being visited.

The following three attributes of a temporary vertex \( i \in V_T \) formulate the different statuses of trucks: (i) service time denoted by \( t_i \), (ii) location at the end of its service time denoted by \( D_i \), and (iii) releasing attribute denoted by \( p_i^e \). If a truck \( i \in V_T \) is visiting an order vertex at the time \( u \), the attributes \( t_i, D_i \) and \( p_i^e \) are the remaining service time, the destination location, and the releasing attribute of the order vertex, respectively. Otherwise, \( t_i = 0, D_i \) is the current location of the truck, and \( p_i^e \) is the current status of the truck. Specifically, \( p_i^e \) is one if the truck is carrying an empty container or zero otherwise.

4.1.4. Arc

An arc \((i, j) \in A\) includes the transfer between vertices. An order vertex could be either the head or the tail of an arc. A start/return vertex could also be either the head or the tail of an arc. However, the head and the tail of an arc cannot be start/return vertices simultaneously. According to the definition of temporary vertices, a temporary vertex must be the tail of an arc. In other words, a working truck at the decision epoch must either visit an order vertex or directly return to a depot.

An arc has a single attribute, namely the transfer time. It is the operating time of the activities during the transfer between vertices. If an arc transfers from a start/return vertex to an order vertex, its transfer time is the traveling time from the depot to the origin location of the order. If an arc transfers to a start/return vertex, its transfer time is the traveling time from the destination location of its tail vertex to the depot. The transfer from an order vertex or a temporary vertex to another order vertex is a little complicated. If the releasing attribute of the tail vertex is equal to the requiring attribute of the head vertex, the truck could travel from the destination of the tail vertex to the origin of the head vertex directly (see Section 3.2 for definition of these two attributes). However, if the two attributes are different, the truck should travel to a depot to pick up or drop off an empty container first and then travel to the origin of the head vertex. The truck chooses the nearest depot for the empty container since multiple depots exist. Here, we omit the time of loading a container onto/off a truck to keep the presentation brief. See Zhang et al. (2009) for an example of transfer time formulation including the loading time.

The following equation summarizes the transfer time, say \( \tau_{ij} \), of an arc \((i, j)\).

\[
\tau_{ij} = \begin{cases} 
  l(i, O_i), & \forall i \in V_O, j \in V_O \\
  l(D_i, L_i), & \forall i \in V_O \cup V_T, j \in V_O \\
  l(D_i, O_j), & i \in V_O \cup V_T, j \in V_O, p_i^e = p_j^e \\
  \min_{k \in V_O} (l(D_i, L_k) + l(L_k, O_j)), & i \in V_O \cup V_T, j \in V_O, p_i^e \neq p_j^e 
\end{cases}
\]

In summary, the DAOV graph \( G \) formulates the drayage problem. The object is to present a series of routes with the minimum total length. Each route starts from a start/return vertex or a temporary vertex, visits several order vertices within their time windows, and then finally returns to a start/return vertex. Specially, a DAOV graph without any temporary vertex formulates the drayage problem at the beginning of the time horizon.

4.2. The mixed zero-one programming model

Based on the DAOV graph, the drayage problem is formulated as a mixed zero-one programming model, say Model O. The decision variables are as follows:

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

\[y_i: \text{time when order vertex } i \in V_O \text{ is started.}\]

4.2.1. Objective

The objective of Model O is to minimize the total operating time of all involved trucks. For a given truck, its operating time is

\[
f_{\text{OPERATING}} = f_{\text{RETURN}} - f_{\text{START}},
\]

where \( f_{\text{RETURN}} \) and \( f_{\text{START}} \) are the time points when the truck finally returns to a depot and when the truck initially starts, respectively. Such a formulation of operating time includes traveling time, with or without a container, packing/unpacking time, and waiting time, at the origin or the destination of orders. If a truck returns from order vertex \( i \) to start/return vertex \( j \), it arrives at the depot at time \( y_i + t_i + \tau_{ij} \). Remember that the trucks corresponding to temporary vertices are working at the time \( u \). If a truck returns from the temporary vertex \( i \) to the start/return vertex \( j \) directly, it reaches the depot at the time \( u + t_i + \tau_{ij} \). Therefore, we get two parts of the sum of time points when the trucks finally return to the depots. One part is \( \sum_{i \in V_O} \sum_{j \in V_O} (y_i + t_i + \tau_{ij}) x_{ij} \), which corresponds to the trucks that return from order vertices. The other part is \( \sum_{i \in V_T} \sum_{j \in V_O} (u + t_i + \tau_{ij}) x_{ij} \), which corresponds to the trucks returning from temporary vertices.
If a truck starts from depot \( i \) to order vertex \( j \), the starting time is \( y_j - \tau_{ij} \). Note that all the trucks corresponding to temporary vertices must start at time \( u \) since they have already been working at that time. Therefore, we get two parts of the sum of time points when the trucks initially start. One part is \( \sum_{i \in V_D} \sum_{j \not\in V_D} (y_j - \tau_{ij})x_{ij} \), which relates to the trucks that start from depots. The other part is \( |V_J|u \), where \( |\cdot| \) is the size of the set. Therefore, the objective function of Model O is

\[
\min f = \sum_{i \in V_D} \sum_{j \not\in V_D} (y_j - \tau_{ij})x_{ij} + \sum_{i \in V_D} \sum_{j \in V_O} (u + t_i + \tau_{ij})x_{ij} - \sum_{i \in V_D} \sum_{j \not\in V_D} (y_j - \tau_{ij})x_{ij} - |V_J|u. \tag{1}
\]

4.2.2. Constraints

Solutions of Model O must satisfy some basic constraints. The number of trucks starting from any depot is at most the number of trucks available at the depot at time \( u \). Thus, these basic constraints are (2)–(4).

\[
\sum_{j \in V_O} x_{ij} \leq n_i, \quad \forall i \in V_D. \tag{2}
\]

\[
\sum_{j \in V_O \setminus V_D} x_{ij} = 1, \quad \forall i \in V_T. \tag{3}
\]

\[
\sum_{j \in V} x_{ji} = \sum_{j \in V_O \setminus V_D} x_{ij} = 1, \quad \forall i \in V_0. \tag{4}
\]

Furthermore, this model has several constraints regarding time. First, the visiting time of an order vertex should be within its time window. The visiting time of vertices on a route must satisfy a continuity constraint. That is, a truck could visit a vertex only after it has arrived at the vertex. Of course, the earliest starting time of trucks is at time \( u \). Finally, the decision variables should be of certain data types. These constraints are as follows:

\[
a_i \leq y_i \leq b_i, \quad \forall i \in V_0. \tag{5}
\]

\[
y_i + t_i + \tau_{ij} - y_j \leq (1 - x_{ij})M, \quad \forall i \in V_O, \ j \in V_O. \tag{6}
\]

\[
u + t_i + \tau_{ij} - y_j \leq (1 - x_{ij})M, \quad \forall i \in V_T, \ j \in V_O. \tag{7}
\]

\[
u + \tau_{ij} - y_j \leq (1 - x_{ij})M, \quad \forall i \in V_D, \ j \in V_0. \tag{8}
\]

\[
x_{ij} \in \{0, 1\}, \quad \forall (i, j) \in A. \tag{9}
\]

\[
y_i \in \mathbb{R}, \quad \forall i \in V_0 \tag{10}
\]

In constraints (6)–(8),

\[M = \max_{i \in V_D} \left( u \max_{j \in V_O} b_j \right) + \max_{i \in V_T} t_i + \max_{i \in V_D, j \not\in V_D} \tau_{ij} - \min_{i \in V_D} a_i\]

is a sufficiently large constant. Note that constraints (5)–(8) could also eliminate subtours among the set \( V_0 \).

Model O (formulation (1)–(10)) differs from the classical multi-depot m-TSPTW (refer to Bektas (2006)) in several ways. First, the final number of trucks could differ from the initial number of trucks at each depot. Second, the objective function differs from the normal objective functions. Third, this model has a so-called temporary vertex set. One truck could start from each temporary vertex at a given time but no truck could return to any temporary vertex. Finally, all trucks in this problem have a limit on the earliest starting time.

5. Solution strategies

This section presents several strategies to solve the drayage problem. Specifically, Section 5.1 introduces a simple strategy that just appends newly arrived orders to the existing routes. Section 5.2 introduces three strategies, each of which resolves the problem every time interruptions occur.

5.1. A simple appending strategy

A simple strategy, named **APPENDING**, is designed as a benchmark strategy for other strategies. It reflects what a scheduler might do without any aid of scheduling systems, as mentioned similarly in Yang et al. (2004). If a priority order arrives, the strategy checks whether assigning this order to any truck is possible. If several orders have already been assigned to a
truck, the new order could only be appended to the end of the existing order sequence. Among those feasible assignments, the one with the lowest marginal operating time is selected. Such an assignment is repeated if multiple priority orders arrive at the same time. This repeat is in the ascending order of the earliest limit of their time windows, which is \( a_i \) of an order \( i \). For comparability with other strategies, the objective value of this strategy is calculated as

\[
\text{Appending} = f_{\text{previous}} - f_{\text{elapsed}} + f_{\text{marginal}}.
\]

where \( f_{\text{previous}} \), \( f_{\text{elapsed}} \), and \( f_{\text{marginal}} \) are the objective value of the previous scheduling before this interruption occurs, total elapsed operating time of all involved trucks until the interruption occurs, and the total marginal operating time, respectively.

5.2. Resolution strategies

5.2.1. Solving Model O using commercial software

Model O ((1)–(10)) is a mixed-integer nonlinear programming (MINLP) model. Therefore, it could be solved using certain commercial optimization software such as Lingo. Local optimal solutions can be reached if solving time permits. A strategy that solves model O directly using commercial software is called a MINLP strategy.

If we discretize the real-value decision variable \( y_i \) into integers, Model O becomes Model D with the objective function (1) and the constraints (2)–(9) and

\[
y_i \in \mathbb{Z}, \quad \forall i \in V_O.
\]

Feasible solutions of Model D are feasible for Model O. Model D is a pure-integer nonlinear programming (PINLP) model. As a result, the third strategy used in this research, named a PINLP strategy, solves Model D ((1)–(9) and (11)) using commercial software. The strategies MINLP and PINLP could also be considered as benchmark strategies for the last strategy that is presented in the next section.

5.2.2. A robust window partition based strategy

The idea of discretizing time windows when solving scheduling problems with time windows appeared very early. Appelgren (1969, 1971) and Levin (1971) used a time discretization strategy to solve a ship scheduling problem and a flight assignment problem, respectively. However, such discretization methods have been rarely used for many years.

Recently, Wang and Regan (2002) developed an iteration method to solve a m-TSPTW. In their method, they partition time windows and solve two models, one to get a feasible solution and the other to evaluate the solution. They narrow the partitioning width and resolve the two models until the performance of the solution is satisfactory. They further prove that this method guarantees convergence to the optimal solution while traditional methods for feasible solutions do not hold such a guarantee (Wang and Regan, 2009).

More recently, Zhang et al. (2010) revise the iteration method of Wang and Regan (2002). They first estimate a smallest partitioning width according to the details of instances and computational ability of computers. Then, they partition time windows using the partitioning width and solve the two models once instead of many times, which makes their method much faster than that of Wang and Regan (2002). The experiments of Zhang et al. (2010) also indicate that their solution method could reach almost the same solutions but much more quickly than through the reactive tabu search algorithm that is a widely used metaheuristic algorithm.

We further modify the method of Zhang et al. (2010) slightly in this research. This method gives a partitioning width according to precision requirements. A smaller partitioning width results in a better solution usually, as learned from Wang and Regan (2002) and Zhang et al. (2010). Furthermore, the partitioning width in this method itself could be considered as the precision of the feasible solutions that this method provides to a certain degree. As a result, the quality of such solutions could be evaluated using the partitioning width even if a lower bound is not calculated. A simple example of TSPTW with five order vertices, as shown in Fig. 4, might help us understand the meaning of the partitioning width in this method. In this example, the feasible solution provided by this method has an identical visiting sequence of vertices to the optimum solution. Even visiting times of vertices in these two solutions are quite close to each other. The visiting time of an order vertex in the optimum solution could be at any point within its time window. The one in the feasible solution can only be the latest limit of one partitioned time window. However, the gap between them is at most the partitioning width.

This window partition based (WPB) method is applied to solve the drayage problem. This strategy is more robust compared to other strategies. The schedule holds if actual traveling times of trucks differ slightly from those given when making the schedule. See Section 6.2.2 for validation of this robustness.

The partitioning width, denoted by \( d \), divides each order vertex with a time window and replaces it with a number of suborder vertices. Specifically speaking, for an order vertex \( i \) with a time window \([a_i, b_i]\) and a service time \( t_i \), the time windows of the suborder vertices are \([a_i + d, a_i + 2d], \ldots, [a_i + (\phi_i - 2) d, a_i + (\phi_i - 1) d], [a_i + (\phi_i - 1) d, b_i]\), where \( \phi_i = \left\lfloor \frac{b_i - a_i}{d} \right\rfloor \) is the number of suborder vertices of order vertex \( i \) and \( x \) is the largest integer that is not larger than \( x \). Each of these \( \phi_i \) suborder vertices has a service time \( t_i \). For each order vertex, one and only one of its suborder vertices must be visited. Thus, a model that is equivalent to Model O is generated.

Let \( \omega \) be the set of suborder vertices for all order vertices. Also, let \( \phi(i) \) be the order number of a suborder \( i \in \omega \). The time window \([a_i, b_i]\), which is originally defined in \( V_O \), is now redefined in \( \omega \). The set \( V_O \) used in the definitions of \( x_{ij} \) is also replaced...
by the set $\omega$. The WPB strategy considers only the latest limit of the time window of suborder vertices. The decision variable $y_i$ is hence replaced by $b_i$. The optimum solution of this model, say Model WPB, is a suboptimum solution of Model O.

Model WPB could be formulated as:

\begin{align}
\text{minimize } & f_{WPB} = \sum_{i \in \omega} \sum_{j \in V_D} (b_i + t_{d(j)} + \tau_{d(i),j})x_{ij} + \sum_{i \in V_T} (u + t_i + \tau_j)x_{ij} - \sum_{i \in V_D} \sum_{j \in \omega} (b_j - \tau_{l,i,j})x_{ij} - |V_T|u \\
\text{Subject to} & \\
\sum_{j \in \omega} x_{ij} & \leq n_i, \quad \forall i \in V_D \tag{13} \\
\sum_{j \in V_D} x_{ij} & = 1, \quad \forall i \in V_T \tag{14} \\
\sum_{j \in V_D} x_{ij} & = \sum_{j \in V_D} x_{ji}, \quad \forall i \in \omega \tag{15} \\
\sum_{i \in V_D, \omega} \sum_{j \in \omega, k} x_{ij} & = 1, \quad \forall k \in V_O \tag{16} \\
(b_i + t_{d(j)} + \tau_{d(i),j} - b_j)x_{ij} & \leq 0, \quad \forall i, j \in \omega \tag{17} \\
(u + t_i + \tau_{l,i,j} - b_j)x_{ij} & \leq 0, \quad \forall i \in V_T, j \in \omega \tag{18} \\
(u + \tau_{l,i,j} - b_j)x_{ij} & \leq 0, \quad \forall i \in V_D, j \in \omega \tag{19} \\
x_{ij} & \in \{0, 1\}, \quad \forall (i, j) \in A \tag{20}
\end{align}

The nonlinear objective function (1) of Model O is automatically linearized as (12) of Model WPB. Constraints (13)–(16) corresponds to constraints (2)–(4). One and only one of the suborders that belong to an order must be visited. Furthermore, a suborder vertex is the tail of an arc if and only if it is the head of another arc. Constraint (5) of Model O is not needed in Model WPB. Constraints (17)–(19) corresponds to constraints (6)–(8), where $M$ is not needed. Model WPB is a pure-zero-one linear programming model and could be solved using commercial software such as Lingo much quickly.

6. Validation and evaluation

6.1. Experiments setup

All validatory and evaluative experiments of this research are implemented on a Hewlett-Packard (HP) personal computer with Intel(R) Core(TM) 2 Quad CPU Q9400 @ 2.66 GHz and 2.66 GHz and 3.49 GB of RAM memories. Randomly generated instances of the dynamic container drayage problem are built in Microsoft Office Excel 2007 using VBA programs. Generation of the DAOV graph, partition of time windows, and the APPENDING strategy are also implemented using VBA programs. Lingo 11.0 of Lindo Systems Inc is the solver of Models O, D, and WPB in the three resolving strategies. The memory limit of a model generator is set as 512 MB. The limit of the running time is set as 600 s. The output level is set as Terse. Although Lingo allows using an initial solution when solving pure integer linear programming models, we do not use initial
solutions when resolving the drayage problem in this research since they do not improve the solution speed of Lingo significantly.

6.1.1. General setting of instances

First, a large number of instances are randomly generated in order to validate the strategies. Randomly generated instances have been widely used to evaluate similar models in existing work such as in Jula et al. (2005), Imai et al. (2007), Cheung et al. (2008), and Zhang et al. (2009). One major reason might be that real-life operational data especially temporal data in drayage is difficult to obtain. Therefore, some certain temporal data in real-world instances derived from the Port of Rotterdam are generated randomly in Srour et al. (2010). Furthermore, randomly generated instances that are similar to real-world scenarios are enough, generally speaking, to evaluate algorithmic performance, e.g., running speed and robustness.

It is appreciated that Srour et al. (2010) share their instances publicly. However, they focus their research on fixed orders. Terminals are also depots for empty containers in their problem. However, this research considers more generic types of orders (see Section 3.2), which makes the problem more complicated. Besides, the instances of Srour et al. (2010) are in the format of SCIP (refer to Achterberg (2009)). Therefore, they cannot be used directly in this research. However, the generation of instances in this research is partially based on that of Srour et al. (2010).

The generated instances are as similar to real-world scenarios as possible. Depots are randomly generated on a two-dimensional Euclidean plane. The number of trucks parking at each depot at the beginning of each day ranges uniformly in a given range. The attributes of orders such as origin locations are generated randomly. Specially, if an order involves an empty container, its destination location and time window are the same as its origin location and time window, respectively. Its origin service time is zero. This work gives a time point as the elapsed amount of time since the beginning of the day, i.e., 8 am. If an order becomes available later then the beginning of the day, its origin time window is set considering the information updating time and the traveling time from depots to its origin to make the order feasible. Note that the origin time window and service time of an order can deduce its earliest limit of its destination time window. See Table 2 for detailed values of parameters used when generating instances.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value or value range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length and width of the Euclidean plane (in unit of traveling time, min)</td>
<td>180</td>
</tr>
<tr>
<td>Number of depots</td>
<td>5</td>
</tr>
<tr>
<td>Range of truck numbers at each depot at the beginning of the time horizon</td>
<td>[1,15]</td>
</tr>
<tr>
<td>Probability of an order i with $p^i_0 = 1$</td>
<td>0.5</td>
</tr>
<tr>
<td>Probability of an order i with $p^i_0 = 1$</td>
<td>0.5</td>
</tr>
<tr>
<td>Probability of an order i with $O_i = D_i$</td>
<td>0.1</td>
</tr>
<tr>
<td>Maximum value of earliest limit of origin time windows (min)</td>
<td>420</td>
</tr>
<tr>
<td>Maximum length of time windows (min)</td>
<td>120</td>
</tr>
<tr>
<td>Range of operating time (min)</td>
<td>[0,60]</td>
</tr>
</tbody>
</table>
6.1.3. Lower bound

Wang and Regan (2002) and Zhang et al. (2010) use a lower bound based on window partitions to evaluate the solutions generated by their methods. We apply this lower bound to Model O for validation of the strategies although the partitioning width could indicate the quality of the solutions generated by the WPB strategy. The optimum value of the following model (Model LB) is a lower bound of Model O.

\[
\begin{align*}
\text{minimize } & \quad f_{LB} = \sum_{i \in \omega} \sum_{j \in V_D} (a_i + t_{i,j}) x_{ij} + \sum_{i \in V_P} \sum_{j \in V_D} (u + t_i + \tau_j) x_{ij} - \sum_{i \in V_P} \sum_{j \in \omega} (b_j - \tau_{i,k,j}) x_{ij} - |V_T| u
\end{align*}
\]

Subject to

\[
(a_i + t_{i,j} + \tau_{i,k,j} - b_j) x_{ij} \leq 0, \quad \forall i \in \omega, j \in \omega
\]

Constraints (13)–(16) and (18)–(20).

Model LB is solved using the same solver with the same setting as Model WPB.

6.1.4. Setting of the partitioning width

Truck traveling time and container handling time in real world involve many uncertainties. Estimating such time with a high precision in advance is quite hard. For example, traveling time between Suyoung and Umgung might be 1 h or 45 min although it is given as 50 min by Chung et al. (2007). A precision of 5 min is usually enough according to Zhang et al. (2010) and our experiences. Furthermore, such time is often given with a precision of 5 min or even lower (longer than 5 min) in literature such as in Chung et al. (2007) and Srour et al. (2010). The partitioning width in Models WPB and LB is set as 4 min in all experiments throughout this article. Therefore, the precision of solutions might be a little higher than it is required.

6.2. Evaluation of the strategies

6.2.1. Comparison of the four strategies

The four strategies are validated and compared based on a primary experiment first. Five instances of the drayage problem where information is available at the beginning of a day are generated randomly and solved using the WPB strategy. Based on them, five groups of representative instances with interruptions are randomly generated and solved using the strategies and the lower bound. For each instance, each strategy is run only once since the strategies have no randomness. Table 3 shows brief information of these instances. The trucks could not handle more orders than up to the last row within each group. A sub-fleet of trucks is able to handle at most 75 containers in one day according to Wang and Regan (2002) and Zhang et al. (2009). The number of containers involved in each instance of Srour et al. (2010) is 65. The number of orders required.

Figs. 6 and 7 display the running time and obtained objective values, respectively. Note that all objective values in Fig. 7 have been divided by the lower bound of the corresponding instance in order to normalize them. The results indicate that the APPENDING strategy runs fastest among the four strategies. The running time is less than one second for each instance. For the instances with a small number of newly arrived orders, e.g., instances P1, P2, and P7, the solutions provided by this simple strategy are almost as good as those by other strategies. However, for the instances with a larger number of newly arrived orders, e.g., instances P6, P14, and P21, the APPENDING strategy could not provide as good solutions as the other strategies. This strategy provides no feasible solution for instance P11. This result is natural.

The MINLP and PINLP strategies could reach local optimum solutions of Models O and D, respectively. They cannot guarantee to provide the global optimum solution because the models involve nonlinear formulation of decision variables. Local optimum solutions of most instances could be arrived in a short running time. The average running times of the strategies MINLP and PINLP are 67.2 and 25.1 s, respectively. The local optimal solution of only one instance (P5) has not been reached by the MINLP strategy within 10 min, as shown in Fig. 6. Furthermore, the PINLP strategy could provide better solutions for most instances than the MINLP strategy in almost the same or even shorter running time. Recall that the only difference between these two strategies is that the PINLP strategy discretizes the decision variables. This is an interesting discovery.

The WPB strategy, among the four strategies, could provide solutions with a high precision quickly and stably. The average running time of this strategy is 54.6 s. Model WPB of all instances could be solved to global optimality within 2 min. The average and maximum objective values divided by the corresponding lower bound over all the instances are only 1.021 and 1.038, respectively. That is to say, the gap between these solutions and the global optimum solutions is only about 3%. Such a
performance should be acceptable in application. Furthermore, the average objective values of solutions provided by the APPENDING and WPB strategies are 12,416 and 12,211 min. The saved 205 min of the truck operating time indicates the economic effect of the WPB strategy.

Table 3
Brief information of instances in the primary experiment.

<table>
<thead>
<tr>
<th>Group</th>
<th>Initial information</th>
<th>Ins.</th>
<th>Information when the interruption occur</th>
<th># of trucks</th>
<th># of orders</th>
<th>Interrupting time (min)</th>
<th># of temporary vertices</th>
<th># of newly arrived orders</th>
<th>Total # of orders</th>
<th># of suborders</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>36 50</td>
<td>P1</td>
<td></td>
<td>60</td>
<td>8</td>
<td>1</td>
<td>51</td>
<td>572</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>P2</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>P3</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>P4</td>
<td></td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>P5</td>
<td></td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>P6</td>
<td></td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>32 50</td>
<td>P7</td>
<td></td>
<td>60</td>
<td>3</td>
<td>1</td>
<td>51</td>
<td>521</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>P8</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>P9</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>P10</td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>P11</td>
<td></td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>42 50</td>
<td>P12</td>
<td></td>
<td>60</td>
<td>1</td>
<td>3</td>
<td>53</td>
<td>650</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>P13</td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>P14</td>
<td></td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>38 50</td>
<td>P15</td>
<td></td>
<td>40</td>
<td>1</td>
<td>3</td>
<td>53</td>
<td>547</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>P16</td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>P17</td>
<td></td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>P18</td>
<td></td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>P19</td>
<td></td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>40 5</td>
<td>P20</td>
<td></td>
<td>120</td>
<td>3</td>
<td>40</td>
<td>44</td>
<td>443</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>P21</td>
<td></td>
<td>45</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 6. Running time of the strategies in the primary experiment.

Fig. 7. Obtained objective values of the strategies in the primary experiment.
6.2.2. Robustness of the WPB strategy

Here we design an experiment to evaluate the robustness of the WPB strategy. First, three instances with brief information shown in Table 4 are generated similarly as in Section 6.2.1. It is assumed that the transfer time of each arc has a probability of 0.01 that there is a bias between the given and actual transfer time. The bias level of instances R1, R2, and R3 is set as ±1, ±2, and ±4 min, respectively. The bias level means that the actual transfer time of an arc might be 4 min longer or shorter than the given transfer time, where ‘±4’ is taken as an example. Five specific biases for each instance are randomly generated. Note that such biases of transfer time are irrelevant to the information updates discussed outside of this section.

The PINLP strategy is used as a comparing strategy since it appears better than the other two strategies. The results shown in Table 5 again validate that the WPB strategy could provide solutions with a high precision fast and stably. Three indexes are used to evaluate the robustness of strategies. They are the changes of objective values, whether the visiting sequences of orders change, and the number of orders which visiting time changes compared to the corresponding base case. The results shown in Table 5 tell us that for most instances, the scheduling results under transfer time biases provided by the PINLP strategy differ from the corresponding base case even if the bias level is only ±1 min. The number of orders with a visiting time different from the base cases is very large, with an average of 23 (34.2% of the total number of orders). The objective values also differ from the base case significantly. The solution under only one bias (bias 3 for instance R2) is identical to that under the base case. However, the sequences of orders under most cases (13 out of 15) given by the WPB strategy remain as the base cases. The average number of orders, which visiting time changes, is very small especially under a low level of bias. The average change of objective values provided by the WPB strategy is also much smaller than that provided by the PINLP strategy. Therefore, the WPB strategy is very robust considering estimating biases of operational time. As analyzed in Section 5.2.2, this robustness is the merit of the short wait at a large number of customers in the solutions given by the WPB strategy. Other solution algorithms including most metaheuristics have not been reported to have such robustness.

6.2.3. Further evaluation of the WPB strategy

The information update could be of any type. The above experiments assume that a number of orders become available when an interruption occurs. However, the WPB strategy and other resolving strategies could also handle other types of interruptions including canceling and modifying of orders and modifying of road and truck conditions. The reason is that the graph-based formulation could formulate the problem when the information updates. Hence, these strategies could solve it.

Table 4
Brief information of instances in the robustness evaluating experiment.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Interrupting time (min)</th>
<th># of orders</th>
<th># of sub orders</th>
<th># of temporary orders</th>
<th># of arcs</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>15</td>
<td>75</td>
<td>880</td>
<td>3</td>
<td>6640</td>
</tr>
<tr>
<td>R2</td>
<td>60</td>
<td>68</td>
<td>726</td>
<td>5</td>
<td>5694</td>
</tr>
<tr>
<td>R3</td>
<td>120</td>
<td>57</td>
<td>689</td>
<td>13</td>
<td>4650</td>
</tr>
</tbody>
</table>

Table 5
Results of the robustness evaluating experiment.

<table>
<thead>
<tr>
<th>Level (Ins.)</th>
<th>Bias</th>
<th>PINLP</th>
<th>WPB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Time (s)</td>
<td>CO&lt;sup&gt;a&lt;/sup&gt;</td>
</tr>
<tr>
<td>±1(R1)</td>
<td>Base</td>
<td>27 0 (14,877)</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>95 –182</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>37 250</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>42 –192</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>25 –89</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>29 64</td>
<td>Yes</td>
</tr>
<tr>
<td>±2(R2)</td>
<td>Base</td>
<td>18 0 (14,844)</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>44 –133</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>37 86</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>29 0</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>19 45</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>17 –39</td>
<td>Yes</td>
</tr>
<tr>
<td>±4(R3)</td>
<td>Base</td>
<td>124 0 (13,353)</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>24 24</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>27 42</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>29 43</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>23 –2</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>48 31</td>
<td>Yes</td>
</tr>
</tbody>
</table>

<sup>a</sup> CO: change of objective values; WVSC: whether the visiting sequences of orders change or not; #VTC: the number of orders which visiting time is changed.
Interruptions might occur more than once in one day. We generate five groups of instances to illustrate the application of the WPB strategy. In each of the five groups, an instance with only the information available at the beginning of the day is generated first. An interruption occurs resulting in a second instance. Then, another interruption occurs resulting in a third instance. Table 6 shows brief information and results of these instances.

Table 6 indicates that this method could solve the problem when the information updates for a second time. Of course, information could update for more times in this method. Performance of the WPB strategy depends mainly on the number of decision variables and constraints. Table 7 shows the statistical information of Model WPB of a typical instance.

Finally, the authors find that the schedule after an interruption occurs is quite similar as the initial schedule provided by the WPB strategy although this strategy completely resolves the problem. This similarity is even obvious if the number of appended orders is not large. The following is a typical example. The initial schedule of instance P10 (refer to Section 6.2.1) involves 50 orders (numbered from 1 to 50) and 29 trucks. Five orders numbered from 51 to 55 become available at time \( u = 60 \) min. Table 8 shows the similarity between the two schedules obtained from the WPB strategy. One can observe that 22 out of the 29 routes remain identical in the new schedule. Four out of the five similar pairs of routes are the “same” as

<table>
<thead>
<tr>
<th>Group</th>
<th>Instance</th>
<th>Decision epoch (min)</th>
<th>Predecessor instance</th>
<th># of orders</th>
<th># of sub orders</th>
<th># of temporary orders</th>
<th>Solving time (s)</th>
<th>Obj. (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>M1–1</td>
<td>0</td>
<td>–</td>
<td>40</td>
<td>601</td>
<td>0</td>
<td>43.3</td>
<td>8717</td>
</tr>
<tr>
<td></td>
<td>M1–2</td>
<td>60</td>
<td>M1–1</td>
<td>42</td>
<td>636</td>
<td>4</td>
<td>49.5</td>
<td>9141</td>
</tr>
<tr>
<td></td>
<td>M1–3</td>
<td>120</td>
<td>M1–2</td>
<td>44</td>
<td>674</td>
<td>12</td>
<td>60.5</td>
<td>9059</td>
</tr>
<tr>
<td>2</td>
<td>M2–1</td>
<td>0</td>
<td>–</td>
<td>40</td>
<td>402</td>
<td>0</td>
<td>18.8</td>
<td>8551</td>
</tr>
<tr>
<td></td>
<td>M2–2</td>
<td>60</td>
<td>M2–1</td>
<td>42</td>
<td>426</td>
<td>0</td>
<td>22.1</td>
<td>9037</td>
</tr>
<tr>
<td></td>
<td>M2–3</td>
<td>120</td>
<td>M2–2</td>
<td>44</td>
<td>463</td>
<td>2</td>
<td>27.0</td>
<td>9407</td>
</tr>
<tr>
<td>3</td>
<td>M3–1</td>
<td>0</td>
<td>–</td>
<td>45</td>
<td>498</td>
<td>0</td>
<td>31.2</td>
<td>9681</td>
</tr>
<tr>
<td></td>
<td>M3–2</td>
<td>20</td>
<td>M3–1</td>
<td>55</td>
<td>555</td>
<td>0</td>
<td>42.7</td>
<td>10,469</td>
</tr>
<tr>
<td></td>
<td>M3–3</td>
<td>150</td>
<td>M3–2</td>
<td>50</td>
<td>556</td>
<td>6</td>
<td>43.2</td>
<td>10,417</td>
</tr>
<tr>
<td>4</td>
<td>M4–1</td>
<td>0</td>
<td>–</td>
<td>50</td>
<td>531</td>
<td>0</td>
<td>37.8</td>
<td>11,668</td>
</tr>
<tr>
<td></td>
<td>M4–2</td>
<td>100</td>
<td>M4–1</td>
<td>52</td>
<td>569</td>
<td>3</td>
<td>45.3</td>
<td>11,939</td>
</tr>
<tr>
<td></td>
<td>M4–3</td>
<td>180</td>
<td>M4–2</td>
<td>49</td>
<td>546</td>
<td>19</td>
<td>41.2</td>
<td>11,727</td>
</tr>
<tr>
<td>5</td>
<td>M5–1</td>
<td>0</td>
<td>–</td>
<td>60</td>
<td>626</td>
<td>0</td>
<td>61.5</td>
<td>12,625</td>
</tr>
<tr>
<td></td>
<td>M5–2</td>
<td>40</td>
<td>M5–1</td>
<td>62</td>
<td>631</td>
<td>0</td>
<td>64.8</td>
<td>13,097</td>
</tr>
<tr>
<td></td>
<td>M5–3</td>
<td>240</td>
<td>M5–2</td>
<td>39</td>
<td>389</td>
<td>31</td>
<td>18.9</td>
<td>11,080</td>
</tr>
</tbody>
</table>

Table 7
Brief statistical information of Model WPB for Instance M1–3.

<table>
<thead>
<tr>
<th>Statistical index</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of variables (all are 0–1 integers)</td>
<td>469,189</td>
</tr>
<tr>
<td>Number of constraints (all are linear)</td>
<td>466,470</td>
</tr>
<tr>
<td>Total nonzero coefficients</td>
<td>1,879,518</td>
</tr>
</tbody>
</table>

Table 8
Similarity between the initial and updated scheduling.

The number of identical routes is 22 (We omit the specific routes here)
The number of similar routes is 5

<table>
<thead>
<tr>
<th>Initial scheduling at the beginning of the day</th>
<th>Updated scheduling at time ( u = 60 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>T2 (60) – 9 (127) – 22 (232) – D3 (422)</td>
<td>D2 (30) – 9 (127) – 22(232) – D3 (422)</td>
</tr>
<tr>
<td>D3 (102) – 46 (132) – 12 (259) – D2 (470)</td>
<td>D3 (110) – 46 (140) – 12(267) – D2 (478)</td>
</tr>
<tr>
<td>D1 (276) – 48 (381) – D4 (521)</td>
<td>D2 (302) – 48 (381) – D4 (521)</td>
</tr>
</tbody>
</table>

The number of different routes is 5

<table>
<thead>
<tr>
<th>Initial scheduling at the beginning of the day</th>
<th>Updated scheduling at time ( u = 60 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>D2 (257) – 36 (322) – D2 (631)</td>
<td>D1 (76) – 39 (112) – D2 (399)</td>
</tr>
<tr>
<td>D1 (276) – 54 (327) – D2 (546)</td>
<td>D1 (276) – 54 (327) – D2 (546)</td>
</tr>
</tbody>
</table>

Note: The number following “D” is the number of the depot. The number following “T” is the number of the temporary vertex. The number in a bracket is the visiting time (in minutes) of the order that is followed by the bracket.
each other actually. The two temporary vertices T1 and T2 are generated in the corresponding routes. The third pair of similar of routes in Table 8 has the identical visiting sequence and the total working time between the two routes. So does the fourth pair of similar routes. Such similarity of solutions is just what a scheduler expects.

7. Conclusions

This article studies a container drayage problem with flexible orders where logistic information could be updated during one time horizon. A unified definition of drayage orders based on the requiring and releasing attributes could flexibly formulate various types of drayage orders, some of which exist in literature while the others have not been reported. The DAOV graph is modified to introduce a temporary vertex set to formulate the truck statuses when interruptions occur. Based on the graph, the drayage problem is formulated as a mixed-integer nonlinear programming model as an extension of the multi-depot asymmetric m-TSPTW. Four strategies are presented to solve the dynamic container drayage problem and evaluated extensively. The comparative results to the other three strategies and a lower bound indicate that the WPB strategy learned from Wang and Regan (2002) could provide solutions with a high precision quickly and stably. The WPB strategy is robust considering how it handles time biases compared to other algorithms and could save operating costs greatly. Therefore, performance of the WPB strategy should be acceptable in application. Possible future research might include studies of more general logistics and transportation problems with updatable information.

Acknowledgements

The authors are grateful to the valuable and constructive comments from the editor and anonymous referees.

References


