The multi-trip container drayage problem with synchronization for efficient empty containers re-usage

Fazi, Stefano; Choudhary, Sourabh Kumar; Dong, Jing Xin

DOI
10.1016/j.ejor.2023.02.041

Publication date
2023

Document Version
Final published version

Published in
European Journal of Operational Research

European Journal of Operational Research 310 (2023) 343–359

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Innovative Applications of O.R.

The multi-trip container drayage problem with synchronization for efficient empty containers re-usage

Stefano Fazi[a,*], Sourabh Kumar Choudhary[b], Jing-Xin Dong[c]

[a] Delft University of Technology, Department of Technology, Policy and Management, PO Box 5015, GA Delft 2600, The Netherlands
[b] Georgia Institute of Technology, Industrial and Systems Engineering Department, Atlanta, GA 30313, USA
[c] Business School, Newcastle University, 5 Barrack Road, Newcastle upon Tyne, NE1 4SE, UK

ARTICLE INFO

Article history:
Received 18 March 2022
Accepted 28 February 2023
Available online 4 March 2023

Keywords:
OR in maritime industry
Daily truck scheduling
Synchronization
Empty containers management

ABSTRACT

We study a typical daily drayage problem concerning the last-mile logistics at seaports for inland container supply chains. A set of trucks available at an inland container terminal must fulfil shippers' requests of transporting containers within time windows and, to do so, can perform multiple daily trips. A request may entail picking up or delivering containers either at the shippers' premises, the inland terminal or the seaport. Demand for empty containers can be satisfied by either using the available limited stock at the inland terminal, by street-turning or, ultimately, by retrieving them at a local depot for empties resulting in extra mileage. Hence, the minimization of routing costs also entails synchronizing trucks' trips that retrieve and add empty containers to the inland terminal stock to avoid unnecessary visits to the empty depot. After modelling the problem mathematically, we develop an exact column-and-row generation approach embedded in a branch-and-price framework. To accelerate the solving process of the pricing problem, we propose effective strategies by combining a set of tailored pricing algorithms. These strategies perform well on a set of adapted Solomon's instances up to 100 nodes and against a standard branch-and-cut solver. Finally, experiments on real-world instances, inspired by a case study of an inland terminal at the Port of Rotterdam region, provide insights into current planning practices.

1. Introduction

Truck transport plays a vital role in container supply chains. Despite the efforts in several regions worldwide to reduce its use, trucking is still pivotal for short and even medium-range transportation. Reasons are the flexibility of this modality due to almost unnecessary consolidation on a container level, fast transport times, and direct connections with all inland destinations. Also, in the context of empty container management in the maritime industry, trucks are crucial to reposition empties quickly at the seaport side to avoid detention penalties from shipping lines and to meet the requests from the shippers (Fazi & Roodbergen, 2018; Zhang, Huang, & Wang, 2020). Despite the merits of using trucks in container supply chains, it is challenging for truck operators to limit the costs and devise schedules that fit with the highly diverse requirements of inland container supply chains.

A common problem in drayage systems is the management of finite resources such as trucks and empty containers. Trucks are typically bound to short routes since each container is related to a single shipping request (Bruglieri, Mancini, Peruzzini, & Pisacane, 2021). This entails that each truck is required to perform multiple trips throughout the day, especially when the size of the available fleet is limited (Imai, Sasaki, Nishimura, & Papadimitriou, 2006). Properly scheduling trucks' trips is required in case of a limited fleet and time windows at the shippers' premises. With respect to empty containers, they are considered a scarce resource, especially given the current shortage in several container supply chains (Toygar, Yildirim, & Gani Mustafa, 2022). Transport operators strive to re-use the empty containers available in the network and to avoid retrieving new ones from local depots for empties. However, accomplishing re-usage is challenging from an operational perspective given that shippers have different time requirements and, therefore, "street-turning" may not always be feasible. Hence, temporary storage of empty containers at the transport operators' premises may provide a further solution to this. Still, it requires careful planning to synchronize trucks' trips to make sure that

* Corresponding author.
E-mail addresses: s.fazi@tudelft.nl (S. Fazi), schoudhary66@gatech.edu (S.K. Choudhary), jingxin.dong@ncl.ac.uk (J.-X. Dong).

https://doi.org/10.1016/j.ejor.2023.02.041

empty containers are timely available, especially when trucks are a finite resource too.

In this paper, we tackle a multi-trip container drayage problem concerning an inland container terminal managing a fleet of trucks and a stock of empty containers to serve a set of shippers located in its hinterland. In particular, a shipper may request full or empty containers to be delivered or picked up within time windows. Besides the shippers' premises, the containers' origins and destinations include the inland terminal and the seaport. Shippers' requests for empty containers can be satisfied by using the available stock at the inland terminal or retrieving them at a local empty container depot. Alternatively, if some shippers release empty containers, these can be street-turned or re-positioned at the inland terminal to fill the available stock for future deliveries. The goal is to minimize routing costs. Due to the fact that empty depots are typically closer to seaport locations rather than inland ones, the goal aligns with making the best use of the available empties in the network and, consequently, limiting extra mileage to retrieve new empties from the local empty depot. The added complexity lies in timing the availability of the empty containers at the inland terminal, taking into account shippers' time windows, and properly scheduling the multiple trips of each truck.

In the literature, several studies have addressed the drayage problem, mainly in the last two decades. Several variants have been developed by considering single or multiple starting depots, empty container repositioning, time windows, multiple container sizes, etc. (Chen, Meng, & Jia, 2022). However, to our knowledge, no study has combined empty container inventory management and multi-trip scheduling problems and provided a comprehensive framework for the proposed drayage problem. Distinct efforts can be found in the work of Zhang et al. (2020) for empty containers, and of Bruglieri et al. (2021) for multi-trips. While the former did not consider several elements such as, for example, the contemporary presence of import and export flows, the latter developed a multi-period setting without explicitly considering the multi-trip scheduling component. With respect to routing problems with synchronization constraints, this problem arises when vehicles compete for scarce resources. As discussed in the review paper of Paraskevopoulos, Laporte, Repoussis, & Tarantilis (2017), the topic has not been thoroughly explored, both in terms of practical models and methodologies. The main applications are, for example, in cross-docking systems (Grangier, Gendreau, Lehuédé, & Rousseau, 2021), airport cargo systems (Bombelli & Fazi, 2022), and workers scheduling (Fink et al., 2019; Nasir & Kuo, 2020). In the former two examples, the problem of resource conflict is tackled with a parallel machine scheduling approach, in the latter the routing variables are defined for both workers and vehicles to make sure they follow each other. Because empty containers are not a "renewable" resource, a different approach is required.

This paper aims to tackle and solve the proposed problem, which is inspired by the cases of several independent inland terminals active in the Port of Rotterdam region. After modelling the problem mathematically with a mixed-integer linear programming (MILP) formulation, we develop an exact approach by tailoring a column-and-row generation algorithm embedded in a branch-and-price framework. The row generation adds constraints to the restricted master problem to ensure that the simultaneous choice of paths (i.e., trips) that retrieve empty containers at the inland terminal does not exceed the available stock. The multitrip component of the problem is tackled in two different ways. In the first, multitrip routes are directly generated in the pricing stage, while in the second, single-trip routes are generated and then combined within the master problem to form multi-trip paths. In the latter case, further constraints (i.e., rows) are added to the master problem to ensure that the number of available trucks is not exceeded. Compared to a basic column generation algorithm, the proposed novel approach does not require preliminary discretization of the time horizon and reduces the effort in solving the restricted master problem. Furthermore, a set of tailored algorithms for the pricing problems are developed, along with strategies to reduce the computational effort. Numerical experiments on adapted Solomon's instances up to 100 nodes follow to assess the performance of the proposed method. Finally, a set of real-world instances, drawn from an available case study, have been generated to provide a practical setting and managerial insights into planning practices.

All in all, the contribution of this paper is fourfold:

- we develop a comprehensive novel framework for drayage problems for both import and export flows that treats both empty containers and trucks as a scarce resource
- to solve the problem we propose a column-and-row generation algorithm embedded in a branch-and-price framework that does not require preliminary discretization of the time horizon
- we contribute to the more general VRP literature on resource synchronization, considering non-renewable resources, i.e., the empty containers
- we apply the framework to real-world instances to generate managerial insights into planning practices and current bottlenecks in inland container transport chains.

This paper is structured as follows. In Section 2, we review the related literature on the subject and specify the contribution of the study. In Section 3, we formalize the problem description and develop the MILP formulation. Section 4 presents the methodology. Section 5 shows the numerical results, and finally, Section 6 closes the paper with our recommendations and indications for future research.

2. Literature review

This section first provides an overview of the literature related to the proposed drayage problem. Next, we review works related to the two main components of the problem, namely the presence of multi-trips and the synchronization of routes for empty containers.

2.1. Literature on drayage problems

The problem at hand is typically referred to as container drayage problem, full-truck transportation problem, and inland container transportation problem. Despite the difference in names, the treated problems share a basic pickup and delivery problem between one or multiple depots/terminals and shippers (i.e., customers). The main constraint is the trucks carrying one or at most two containers, leading to short routes. Research on the topic has been active mainly in the last two decades (Bruglieri et al., 2021).

The pickup and delivery problem is treated with variants of the vehicle routing problem or travelling salesman problem. Because trucks can typically handle one request at a time, the networks are reduced in size. In particular, only the associated delivery node can be reached from a pickup node; from a delivery node, only terminals and pickup nodes can be reached. Further reductions may be performed for time windows restrictions. Despite the simplification, the problem is still NP-hard (Imai, Nishimura, & Current, 2007). Therefore, the most common approach to solve these problems has been with heuristics and metaheuristics.

Imai et al. (2007) studied a basic version of the problem with a single depot, maximum trips' length, and trucks visiting at most two nodes for pickup and delivery. Although each truck can be associated with multiple trips, the model does not explicitly sequence them in the time horizon since time windows constraints are not considered. A Lagrangian relaxation-based solution was developed to identify near-optimal solutions. Caris & Janssens

(2009) extend the work by adding time windows at the shippers and the depot and propose a local search heuristic, but, like in a regular VRP model, trucks are associated with single routes. Reinhardt, Pisinger, Spoorendonk, & Sigurd (2016) extend these models by considering an unlimited fleet of trucks and tackling balancing constraints of container levels at the terminals. However, similarly to other papers, the level is not tracked through time. Zhang, Yun, & Moon (2009) develop a reactive Tabu Search algorithm for a multi-depot setting, with time windows and empty container repositioning. The empty containers are used to satisfy shippers' requirements, and some can be re-positioned to tackle the imbalance in the network, but shortage and surplus are determined a priori. Zhang, Yun, & Kopfer (2010) further extend their previous paper by considering multiple terminals where trips originate or end. Braekers, Caris, & Janssens (2013) and Braekers, Caris, & Janssens (2014) formulate the problem as an asymmetric multiple vehicle Travelling Salesman Problem with Time Windows, where empty containers can be used to satisfy requests. Similarly to our problem, their origins and destinations are not determined a priori.

In terms of resource constraints, very few works have been carried out. In this category, we see a separation between the resources and the trucks (Benantar, Abourraja, Boukachour, Boudebous, & Duvallet, 2020). Ileri et al. (2006) tackle a variant of the pickup and delivery problem with time windows and include trailers repositioning in intermediate stops. A column generation approach is developed. Zhang, Yun, & Moon (2011) model a one seaterminal and one inland terminal, named depot in the paper, setting where empty containers are stored. The empty containers are limited, and the inland terminal must never stock out. A non-linear formulation is developed, and a Tabu Search approach is proposed to solve the problem. In contrast, our paper also considers in a linear model the presence of an empty depot to retrieve empty containers when the system is not able to supply them. In a recent article, Zhang et al. (2020) propose an extension to Zhang et al. (2011) by considering four types of container tasks and fixed costs for the number of used trucks and total working time. A Large Neighborhood Search heuristic is developed. The setting described in Zhang et al. (2020) is the most closely related to our research. However, it does not consider the supply of empty containers from an empty depot as a decision but can only occur due to predetermined tasks. Besides, we further extend their work by considering additional shipper types (see Section 3.1), the inland terminal as a source and destination of full containers, the multi-trip component and the consideration of a hybrid system that deals with both import and export container flows. Finally, we provide an exact algorithm, unlike previous heuristic-based approaches, that can tackle both the multi-trip scheduling component and the management of empty containers.

2.2. Multi-trip and synchronized vehicle routing problems

The multi-trip aspect of our problem has received moderate attention in the literature. One of the first attempts is the study of Taillard, Laporte, & Gendreau (1996). A solution approach based on Tabu Search was developed. Successive works have focused on developing heuristics and metaheuristics; see Şen & Bülbül (2008) for a survey up to 2008. The first attempt to solve the problem exactly is provided by Mingozzi, Roberti, & Toth (2013). They propose two set-partitioning like formulations. The first requires an a priori generation of all feasible routes, whereas the second is based on the generation of all feasible schedules. Instances with 120 customers could be processed with the first approach outperforming the second one. Hernandez, Feillet, Giroudeau, & Naud (2016) develop two branch-and-price frameworks for the variant with time windows. Similar to Mingozzi et al. (2013), in the first, columns represent sequences of trips, whereas in the second, single trips. Experiments on adapted Solomon's instances have shown a more consistent performance of the latter. Other works on the subject in the last decade focus mainly on local search heuristics; for example, see Cattaruzza, Absi, & Feillet (2016). Finally, the only explicit attempt regarding drayage-related literature is the recent work of Bruglieri et al. (2021). However, they consider a setting where release and due dates are defined over whole periods, meaning that explicit multi-trip truck scheduling is not computed. Finally, the empty container management problem is not tackled. The authors develop an arc-based integer linear formulation and design different Combinatorial Benders' Cuts approaches to solve medium and large instances. Hence, with respect to multi-trips, former works (e.g., Imai et al., 2006 and Bruglieri et al., 2021) do not explicitly model the sequence of trips for each truck, which fits well only with the assumption of an infinite fleet. In our problem, we remove this assumption.

The second aspect of the proposed problem concerns the synchronization between vehicle routes. Synchronization requirements may concern spatial, temporal, and load factors. A survey on this class of problems is offered by Drexl (2012) and Paraskevopoulos et al. (2017). Based on the proposed classification in Drexl (2012), our problem falls in the category of resource synchronization, which is the case when vehicles compete for scarce resources. Paraskevopoulos et al. (2017) further extends the definition of resources as renewable (e.g., personnel, machinery) or non-renewable (e.g., money, raw materials). In this category, a few works are available and mainly focus on renewable resources (Paraskevopoulos et al., 2017). Hempsch & Irnich (2008) propose a generic non-linear model for rich VRPs by means of intertour resource constraints. From this model, efficient solution procedures for local search are developed. Ebben, van der Heijden, & van Harten (2005) study the scheduling of automated guided vehicles (AGVs), competing for four scarce resources at an airport. A scheduling method is developed and tested in a discrete event simulation framework. Grangier et al. (2021) study a routing problem where the number of cross-docking stations is limited and propose a metaheuristic approach. Bombelli & Fazi (2022) develop two extensive linear mathematical formulations with precedence constraints to synchronize trucks using cross-docks at a cargo airport. An Adaptive Large Neighborhood Search algorithm is also developed. Both Grangier et al. (2021) and Bombelli & Fazi (2022) integrate a parallel machine scheduling problem in their formulation to tackle the conflicts at the cross-docks. Finally, several papers model synchronized visits, where different resources (e.g., vehicles and personnel) need to meet at specific locations at the same time window. See for example, Bredström & Rönnqvist (2008) and Liu, Tao, & Xie (2019).

2.3. Conclusions

All in all, the contribution of this paper is as follows. About the drayage literature, our paper tackles simultaneously limited stock of empty containers at the inland terminal, an empty depot to support the operations and a multi-trip component. Some of these components can be found singularly in Zhang et al. (2011), Zhang et al. (2020), and (Bruglieri et al., 2021). However, to the best of our knowledge, a comprehensive and more realistic framework is missing. Regarding the multi-trip component, this paper is the first to address an explicit scheduling approach in the drayage literature. Concerning empty containers, the setting in Zhang et al. (2020) is the closest to our research. Still, it does not consider a more general definition of the type of shippers, the multi-trip component, the role of the empty depot as a flexible supply of empty containers, and the role of the inland terminal (named depot in their paper) as a source and destination of full containers. Further-

more, the availability of real-world data in our study provides opportunities for managerial insights and comparisons.

About the methodological aspect of our study, to our knowledge, this is the first work giving an exact column and row generation method embedded in branch-and-price framework for such limited resource cases of a vehicle routing problem, without discretizing the time horizon. The proposed method can be used in other routing problems involving synchronizing trucks' routes and sharing limited resources at the starting depot, where time synchronization is a critical component. Finally, our review, along with the review paper of Drexl (2012), showed that the number of contributions on resource synchronization is limited. Hence, this paper aims to further develop the research area on rich VRPs on resource synchronization and their applications.

3. Problem formulation

In this section, we define the problem at hand. We first present the setting and the main assumptions. Next, we show the mathematical formulation.

3.1. Problem setting

We consider the problem of the transportation of containers between an inland terminal and several inland locations. These locations consist of shippers, a major seaport, and an empty container depot. Shippers generate transport demand and may either receive or send containers. The inland terminal is typically a private and independent entity that acts as a truck operator with a homogeneous and finite fleet of trucks. Its role has become more and more vital in several container supply chains to facilitate hinterland access and provide a competitive advantage to the reference seaport (Rodrigue, Debrie, Fremont, & Gouvernal, 2010).

From the hinterland perspective, the seaport is the primary source and destination for import and export containers, respectively. The inland terminal is the origin and final destination of the trucks. It holds a finite supply of empty containers that can be used to satisfy shippers' demands. Finally, the empty container depot is an important element in inland container systems as the main supplier of empty containers. In general, due to the scarcity of empty containers in several supply chains, inland terminals may request a set of these containers to the empty depot to support their operations. However, this entails an extra cost for the inland terminal as it requires extra mileage to reach the empty depot, which are typically closer to seaport locations than inland ones. Hence, efficiently using the already available containers is key to gaining a competitive advantage (Fransoo & Lee, 2013).

As mentioned, shippers are the entities generating transport movements and may have different requests to be satisfied within a time window. Early arrival at the shippers' premises before the time windows is allowed. We define six types of shippers depending on whether they request an inbound or outbound full or empty container. In particular, a shipper type is identified with the symbol $\Diamond^{ab}$, where a stands for the inbound request and b for the outbound.

We list here the considered scenarios:

- A shipper requests an empty container ($\Diamond^{E\varnothing}$);
- A shipper requests an empty container and immediately releases a full one ($\Diamond^{EF}$);
- A shipper only releases a full container ($\Diamond^{\varnothing F}$);
- A shipper receives a full container and releases immediately another one ($\Diamond^{FF}$);
- A shipper receives a full container and releases immediately an empty one ($\Diamond^{FE}$);
- A shipper only receives a full container ($\Diamond^{F\varnothing}$).

If released by the shipper, full containers may be destined for the inland terminal or the seaport. Likewise, full containers destined for the shippers may originate at the inland terminal or the seaport. Empty containers originating from the shippers may be re-positioned to the inland terminal to re-fill the stock or used on the fly to satisfy other requests (i.e., street-turn). However, in some cases, a shipper may be obliged to return an empty container to the seaport due to the upcoming end of the rental period, known as detention. For conciseness, these cases will be identified as $\Diamond^{FF}$ or $\Diamond^{\varnothing F}$, with the final destination being the seaport. We note that these scenarios may implicitly cover other cases. For example, $\Diamond^{\varnothing F}$ may also represent a request to pick up a container at the seaport and move it to the inland terminal. In such a case, the "shipper's" location coincides with the seaport one. Finally, each delivery or pickup at the shippers' premises must be carried out within a time window. This restriction is relevant since shippers typically schedule containers' arrivals based on their handling capacity. Also, in the case of pickup or delivery at the seaport, the inland terminal is generally requested to book a time slot in advance and respect demurrage and detention time windows (Fazi & Roodbergen, 2018). Compared to the paper of Zhang et al. (2020), we consider $\Diamond^{FF}$, $\Diamond^{\varnothing F}$, and $\Diamond^{F\varnothing}$, and the inland terminal as possible origin and destination of full containers. This is relevant since inland terminals can directly connect with the seaport via rail or barge to transport full containers.

Regarding trucks, we consider a finite fleet that can perform multiple trips in a day and that can carry one container at a time; the latter is a common assumption in the drayage literature, given that 40 ft is the most common size and that the industry is progressively transitioning to a single size (Cui, Chen, Chen, & Meng, 2022; de Ricqlès, 2019). An individual trip is a route starting and finishing at the inland terminal. A truck can perform multiple trips in a sequence, or in other words, it may be assigned to a set of trips as long as these are not overlapping.

The goal is to schedule the available trucks and satisfy all shippers' requests while minimizing the total routing cost. The main constraints are the time windows, the limited number of trucks, and the stock of empty containers at the inland terminal. Along with the latter constraint, the terminal must not exceed its stock by properly synchronizing the inflow and outflow of empties. Retrieval of empty containers from the empty depot is not explicitly penalized, but the potential penalty is incorporated in the extra mileage needed to reach it.

3.2. Mathematical formulation

We consider a network $G(N, \mathcal{A})$ with $\mathcal{A}$ the set of arcs and with N the set of nodes that consists of the inland terminal (node 0), the depot for empties (node 1), and a node for each shipper. $\mathcal{S}$ identifies the subset of shippers within N. C_{ij} is the time distance between node i and j. We reduce the number of arcs in the network by creating hyperarcs that directly connect shippers or a shipper with the inland terminal in case the connection requires the visit of either the seaport or the empty depot between them. In particular, the seaport node is "bypassed" in our network, meaning that if a shipper requires an inbound or outbound connection to it, the distance from/to the other nodes incorporates the detour through the seaport. The depot for empties is "bypassed" if the truck is scheduled to go from a shipper to another one requesting an empty container, given that the first does not require sending a full container to the inland terminal or does not release an empty container. In the case of an empty release from the first visited shipper, the request for an empty for the second shipper may simply be satisfied with a direct link. Note that we cannot bypass the empty depot from the inland terminal since we can satisfy $\Diamond^{E*}$ shippers by either using the available stock or by making a de-

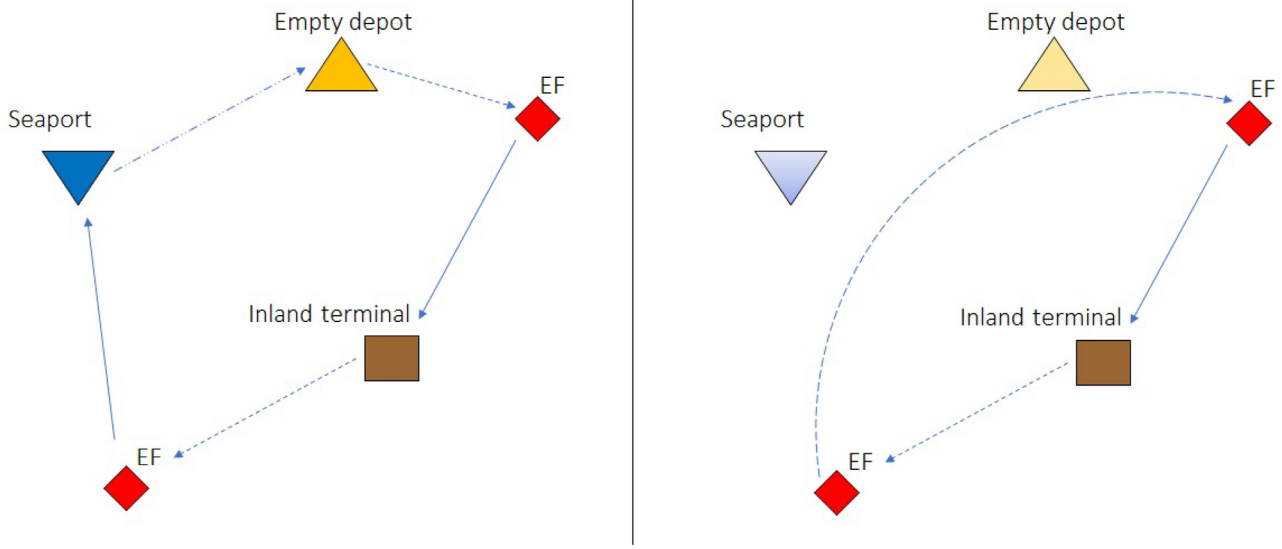

Fig. 1. Simplification of the network by creating hyperarcs that bypass either the seaport or the empty depot and that directly connect two shippers.

tour to reach the empty depot. See Fig. 1 for a graphical example. Here, we show an extreme example of a route that visits two $\Diamond^{EF}$ shippers in a row, with the first sending the full container to the seaport and the second requesting an empty container. Both the seaport and the empty depot are bypassed, and the hyperarc accounts for the actual distance.

Next, we define the set of trucks K, a maximum number of trips R and the related numeric set of trips $\mathcal{R} = 1 \dots R$. For each shipper $i \in \mathcal{S}$, the time window is represented by $[A_i, D_i]$. Finally, E empty containers are initially available at the inland terminal.

We now define the decision variables. x_{ij}^{kr} is the binary routing variable for truck k in trip r and arc (i, j). Time variable t_i^{kr} $(\in \mathbb{R}^+)$ is the departure time for node $i = 0$ and the arrival time for $i \neq 0$. t_{end}^{kr} $(\in \mathbb{R}^+)$ is the end time of trip r of truck k. Concerning variables keeping track of the inventory available at the inland terminal, binary variable $z^{\alpha\beta,kr}$ equals 1 if truck/trip "$\alpha\beta$" departs after "kr" and is relevant when "$\alpha\beta$" departs with an empty container removal from the inland terminal, and "kr" ends by filling the inland terminal with an empty container. Finally, binary variable $y^{\alpha\beta,kr}$ is 1 if truck/trip "$\alpha\beta$" starts after "kr" and is relevant if both trips began with an empty container removal from the inland terminal.

All sets, parameters, and variables are reported in Table 1.

We formulate the problem as follows:

$$\min \sum_{k\in K}\sum_{r\in\mathcal{R}}\sum_{(ij)\in\mathcal{A}} x_{ij}^{kr} C_{ij} \tag{1}$$

$$\sum_{(0i)\in\mathcal{A}} x_{0i}^{kr} \leq 1 \qquad \forall k \in K, r \in \mathcal{R} \tag{2}$$

$$\sum_{j:(ji)\in\mathcal{A}} x_{ji}^{kr} = \sum_{j:(ij)\in\mathcal{A}} x_{ij}^{kr} \qquad \forall k \in K, r \in \mathcal{R}, i \in N \tag{3}$$

$$\sum_{k\in K}\sum_{r\in\mathcal{R}}\sum_{j:(ji)\in\mathcal{A}} x_{ji}^{kr} = 1 \qquad \forall i \in \mathcal{S} \tag{4}$$

$$t_i^{kr} + C_{ij} - M(1 - x_{ij}^{kr}) \leq t_j^{kr} \qquad \forall k \in K, r \in \mathcal{R}, i \in N, j \in N/\{0\} \tag{5}$$

$$(A_i + C_{ij}) x_{ij}^{kr} \leq t_j^{kr} \qquad \forall k \in K, r \in \mathcal{R}, i \in \mathcal{S}, j \in N/\{0\} \tag{6}$$

$$t_i^{kr} + C_{i0} - M(1 - x_{i0}^{kr}) \leq t_{end}^{kr} \qquad \forall k \in K, r \in \mathcal{R}, i \in \mathcal{S} \tag{7}$$

Table 1
Sets, parameters, and variables.

Sets	
N	Set of docks, index 0 for the inland terminal, 1 for the empty depot
$\mathcal{A}$	Set of arcs
$\mathcal{S}$	Set of shippers (indexed from 2 to $\|N\|$ in set N)
$\Diamond^{ab}$	Shipper type: "a" and "b" refer to the type of inflow and outflow
K	Set of trucks
$\mathcal{R}$	Numeric set for trips
Parameters	
E	Initial inventory of empties at the inland terminal
R	Maximum number of trips
$[A_i, D_i]$	Time windows of shipper i
C_{ij}	Transportation time between nodes i and j
M	A large value
Variables	
$x_{i,j}^{kr}$	Binary variable that equals 1 if truck k in trip R goes from node i to j, 0 otherwise
t_i^{kr}	Arrival time of truck k in trip r at shipper i $(\mathbb{R}^+)$
t_0^{kr}	Departure time of trip r of truck k $(\mathbb{R}^+)$
t_{end}^{kr}	End time of trip r of truck k $(\mathbb{R}^+)$
$y^{\alpha\beta,kr}/z^{\alpha\beta,kr}$	Binary variables used to compute the stock of empties at the inland terminal

$$(A_i + C_{i0}) x_{i0}^{kr} \leq t_{end}^{kr} \qquad \forall k \in K, r \in \mathcal{R}, i \in \mathcal{S} \tag{8}$$

$$t_i^{kr} \leq D_i \qquad \forall k \in K, r \in \mathcal{R}, i \in \mathcal{S} \tag{9}$$

$$t_0^{kr+1} \geq t_{end}^{kr} \qquad \forall k \in K, r \in 1 \dots R-1 \tag{10}$$

$$t_0^{kr} \leq t_{end}^{kr} \qquad \forall k \in K, r \in \mathcal{R} \tag{11}$$

$$t_0^{\alpha\beta} - t_{end}^{kr} \leq M z^{\alpha\beta,kr} + M\left(1 - \sum_{i:(i0)\in\Diamond^{FE}} x_{i0}^{kr}\right) + M\left(1 - \sum_{i:(0i)\in\Diamond^{E\varnothing}\cup\Diamond^{EF}} x_{0i}^{\alpha\beta}\right) \qquad \forall k, \alpha \in K, r, \beta \in \mathcal{R} : \alpha \neq k \parallel \beta \neq r \tag{12}$$

$$z^{\alpha\beta,kr} \leq \sum_{i:(0i)\in \diamondsuit^{E\varnothing}\cup\diamondsuit^{EF}} x_{0i}^{\alpha\beta} \qquad \forall k,\alpha \in K, r,\beta \in \mathcal{R} : \alpha \neq k \parallel \beta \neq r \tag{13}$$

$$z^{\alpha\beta,kr} \leq \sum_{i:(i0)\in \diamondsuit^{E\varnothing}} x_{i0}^{kr} \qquad \forall k,\alpha \in K, r,\beta \in \mathcal{R} : \alpha \neq k \parallel \beta \neq r \tag{14}$$

$$t_{end}^{\alpha\beta} - t_0^{kr} \leq M(1 - z^{\alpha\beta,kr}) \qquad \forall k,\alpha \in K, r,\beta \in \mathcal{R} : \alpha \neq k \parallel \beta \neq r \tag{15}$$

$$t_0^{\alpha\beta} - t_0^{kr} \leq My^{\alpha\beta,kr} + M\left(1 - \sum_{i:(0i)\in \diamondsuit^{E\varnothing}\cup\diamondsuit^{EF}} x_{0i}^{\alpha\beta}\right) + M\left(1 - \sum_{i:(0i)\in \diamondsuit^{E\varnothing}\cup\diamondsuit^{EF}} x_{0i}^{kr}\right) \qquad \forall k,\alpha \in K, r,\beta \in \mathcal{R} : \alpha \neq k \parallel \beta \neq r \tag{16}$$

$$y^{\alpha\beta,kr} + y^{kr,\alpha\beta} = 1 \qquad \forall \alpha, k \in K, r,\beta \in \mathcal{R} : \alpha \neq k \parallel \beta \neq r \tag{17}$$

$$(2 - y^{kr,\alpha\beta} - y^{\alpha\beta,\gamma\delta}) \geq 1 - y^{kr,\gamma\delta} \qquad \forall \alpha,\gamma,k \in K,\beta,\delta,r \in \mathcal{R} : (\alpha \neq k \parallel \beta \neq r)\&(\alpha \neq \gamma \parallel \beta \neq \delta)\& \&(\gamma \neq k \parallel \delta \neq r) \tag{18}$$

$$x_{0i}^{kr} \leq E - \sum_{\alpha\in K}\sum_{\beta\in\mathcal{R}:\alpha\neq k\parallel\beta\neq r} y^{kr,\alpha\beta} + \sum_{\alpha\in K}\sum_{\beta\in\mathcal{R}:\alpha\neq k\parallel\beta\neq r} z^{kr,\alpha\beta} + (1 - x_{0i}^{kr}) * M \qquad \forall i \in \diamondsuit^{E\varnothing}\cup\diamondsuit^{EF}, k \in K, r \in \mathcal{R} \tag{19}$$

The objective function (1) minimizes the total routing cost. Constraints from (2) to (4) are classical VRP constraints and impose respectively: no more than one truck per trip, all shippers to be visited, and flow conservation. Inequalities (5) and (6) compute the arrival time of a truck at a node; likewise, (7) and (8) compute the end of the trip. Along with (9), these constraints also impose that the departure time from a shipper is within its time window. (10) imposes that a trip must start after the previous has ended, and with (11) the end of a trip is after its start. Inequalities (12) define variable $z^{\alpha\beta,kr}$. This takes value 1 if trip β of truck α starts after trip r of truck k, trip kr reduces the inventory of the inland terminal by one empty container, and trip $\alpha\beta$ increases the stock by one container from $\diamondsuit^{FE}$. From constraints (13) to (15), we prevent $z^{\alpha\beta,kr}$ from taking value 1 when unnecessary. Next, inequalities from (16) to (18) define variables $y^{\alpha\beta,kr}$ and avoid wrong values. $y^{\alpha\beta,kr}$ takes value 1 if trip β of truck α starts after trip r of truck k and if both trips reduce the inland terminal's inventory of 1 empty container. If the departure times of the two trips are the same, (17) imposes that one of the corresponding y variables takes value 1. (18) ensures the transitivity property. If three trips $(\alpha\beta, \gamma\delta, kr)$ take out the container at the same time, then it must hold, for example, that $\alpha\beta < \gamma\delta < kr$. In particular, this constraint makes sure that the following situation does not occur: $\alpha\beta < \gamma\delta$, $\gamma\delta < kr$ and $kr < \alpha\beta$. Finally, (19) imposes that the stock of empty containers E is not exceeded.

In practice, some companies may be conscious of some additional operational costs. Our model is flexible and allows to extend the objective function with extra costs. For example, minimizing the waiting times of trucks at a shipper's location may be considered by adding the cost $\sum_{k\in K}\sum_{i\in\mathcal{S}} W_{ki}$, where W_{ki} is the total waiting time and can be computed in the constraints with $W_{ki} \geq A_i - t_i^{kr} \forall k \in K, r \in \mathcal{R}, i \in \mathcal{S}$ and $W_{ki} \geq 0\ \forall k \in K, i \in \mathcal{S}$.

4. Solution framework

Column Generation has been widely used to solve vehicle routing problems. In this algorithm, the problem is framed as a set-covering problem named master problem. This formulation is often large, so a restricted master problem (with only a small subset of variables), which is also a linear relaxation of the original problem, is solved. So-called paths, which represent a subset of feasible routes, are linked to the variables of the master problem. The dual values of the master problem's constraints are then passed over to the pricing problem, which uses them to generate a new path. This process repeats until a new path (i.e., column) potentially improving the objective cannot be found (Feillet, 2010).

A basic column generation approach is inapplicable to our problem unless the time horizon is discretized and all times are represented in the constraints of the set covering model of the master problem, as seen in Hernandez et al. (2016). This is due to the empty container constraints (19) that define a limited resource whose availability at the inland terminal changes over time. Therefore, we propose a way to avoid discretizing the time interval while still satisfying the limited resource (empty containers) constraints at all points in time. This is done by employing the limited resource constraints only at critical times, which are the times when an empty container is removed from the inland terminal. Hence, for each new critical time, new constraints (i.e., rows) are added to the master problem.

A non-discretized time horizon has two advantages. First, it gives a more accurate decision of the truck's departure time by not limiting the possibilities only to the defined discrete times, thereby not sacrificing optimality. Secondly, it reduces the restricted master problem size regarding the number of constraints, ensuring faster convergence. In the approach we develop, new constraints are generated on the fly along with the new path-related information (related to new critical times) added to the restricted master problem. This approach is known as column-and-row generation and has been employed, to some extent, in different settings related to resource constraints; for example, see Maher (2016) and Li & Jia (2019). However, to our knowledge, this is the first attempt that generates constraints targeting critical times.

Additionally, we propose two approaches to framing the master and the pricing problems. In the first, the more intuitive for our problem, we generate routes as sequences of trips within a replicated network. More specifically, we explicitly construct multi-trip paths for the master problem. In the second, single trips are generated and then combined within the master problem. In such a case, additional constraints must be added to guarantee that every time a truck departs, the total number of operating trucks does not exceed the maximum. However, this extra effort in the master problem is expected to be compensated in the pricing problem, since the generation of single trip paths is faster. For the pricing problem, in both approaches, we tailor a set of algorithms from the literature and strategies to ensure the process comes to termination in a reasonable time. In particular, a heuristic approach named roll-out and a labelling dynamic programming algorithm with relaxation to dominance criteria are launched to find good negative reduced cost paths in a short time. When these cannot be found, an exact but efficient algorithm called Pulse is launched to prove the non-existence of these paths.

We embed this column-and-row generation algorithm in a branch-and-price framework for the problem at hand, which guarantees to reach optimality if the whole tree is explored. The same principles of column generation apply to a column-and-row generation algorithm, whose proof of correctness is provided in Sadykov & Vanderbeck (2013).

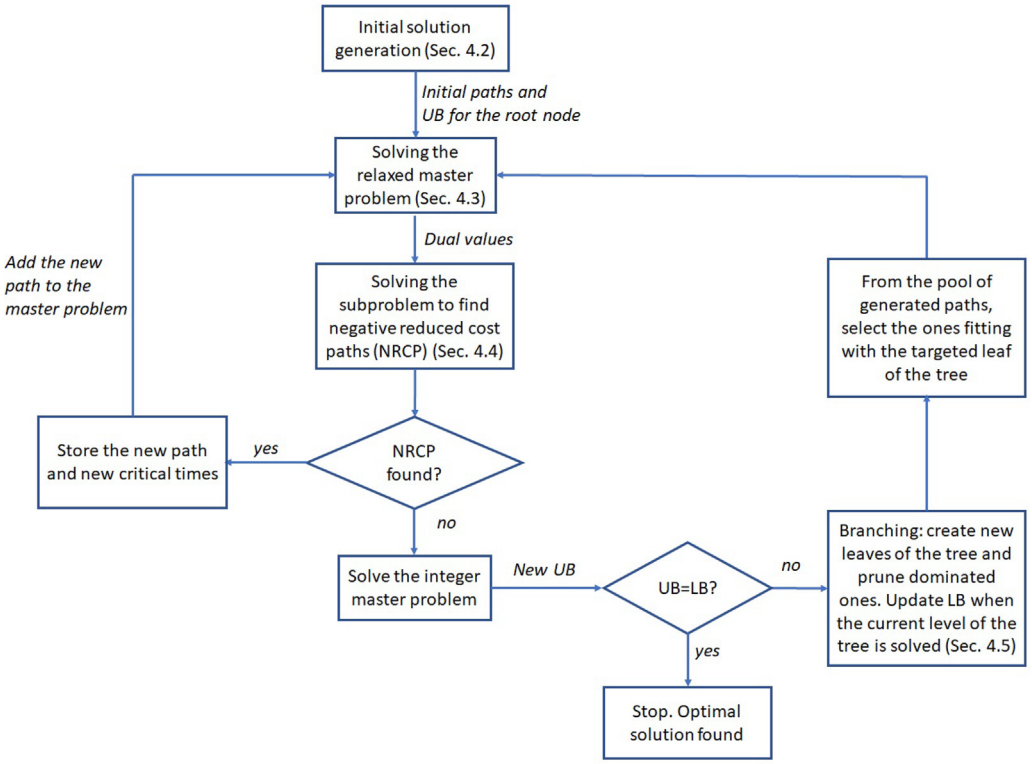

Fig. 2. A representation of the solution (Branch-and-Price) framework. In the boxes, we also indicate in which section the related content is described.

This section has the following organization. In 4.1, we describe the algorithmic framework with all its components and their interactions. Next, we describe each component in depth.

4.1. Structure of the algorithm

The logical representation of the algorithmic framework is shown in a flowchart in Fig. 2. The first step is to define a set of paths to compute an initial feasible solution to the problem (i.e., upper bound (UB)), which initializes the root node of the branch and bound (B&B) tree. This is performed by a greedy algorithm (see Section 4.2). Next, we start the column-and-row generation framework by feeding all paths to the relaxed set covering model (i.e., the restricted master problem), describing the routing problem at hand (see Section 4.3). Once solved, the dual values are fed to the subproblem, which is an elementary shortest path problem with resource constraints (ESPPRC). The aim is to find a new path with negative reduced cost. Since the ESPPRC is NP-hard, tailored algorithmic approaches are required, and these are described in Section 4.4. Note that because of the structure of the problem, the addition of paths can include additional constraints (row generation). This procedure is repeated iteratively until new paths with negative reduced cost are obtained.

When negative reduced cost paths cannot be generated, the targeted node of the B&B tree is considered explored, and the integer master problem can be solved to provide an UB. At the same time, the lower bound (LB) of the node is provided by the solution of the last solved relaxed master problem. Next, the algorithm enters the branching scheme, where new nodes (i.e., leaves of the tree) are generated, others are pruned, and the global UB and LB can be updated (See Section 4.5). Computation is stopped when $UB = LB$ or a computational time limit is reached.

4.2. Initialization

A greedy algorithm calculates an initial feasible solution, thereby providing initial columns for the restricted master problem. The algorithm starts by selecting an available truck and setting the inland terminal as initial node of the path. Among a set of closest nodes from the last node of the path, the one with the smallest due date will be selected, and in case of a tie, the one with the smallest release date. The aim is to include as many shippers as possible in a path. When no more shippers can be inserted, the inland terminal closes the trip and a new one can start. If no more shippers can be added after the inland terminal, the path is closed, and a new truck is opened if some shippers have not been allocated yet. While creating the paths, we store the information about the critical times when empty containers are removed or added from/to the inland terminal. If necessary, the procedure will include the empty depot when visiting either $\diamond^{E\varnothing}$ or $\diamond^{EF}$ nodes.

4.3. Master problems

We develop two distinct column and row generation approaches that differ in the master problem formulation and the subproblem. In the first, paths made of multiple trips are generated, whereas, in the second, paths made of single trips are generated and combined within the scope of the master problem.

Let us define Ω as the set of feasible vehicle routes (i.e., paths) made of one or multiple consecutive trips. λ_p is a decision variable taking value 1 if a path 'p' is selected, 0 otherwise. a_{ip} indicates whether path p contains shipper i. T^{crit} is the set of critical times and increases in size as new generated paths add new critical times in the restricted master problem. Finally, H_{pt} is an integer parameter representing the number of empty containers removed from or added to the inland terminal before a critical time 't.' For

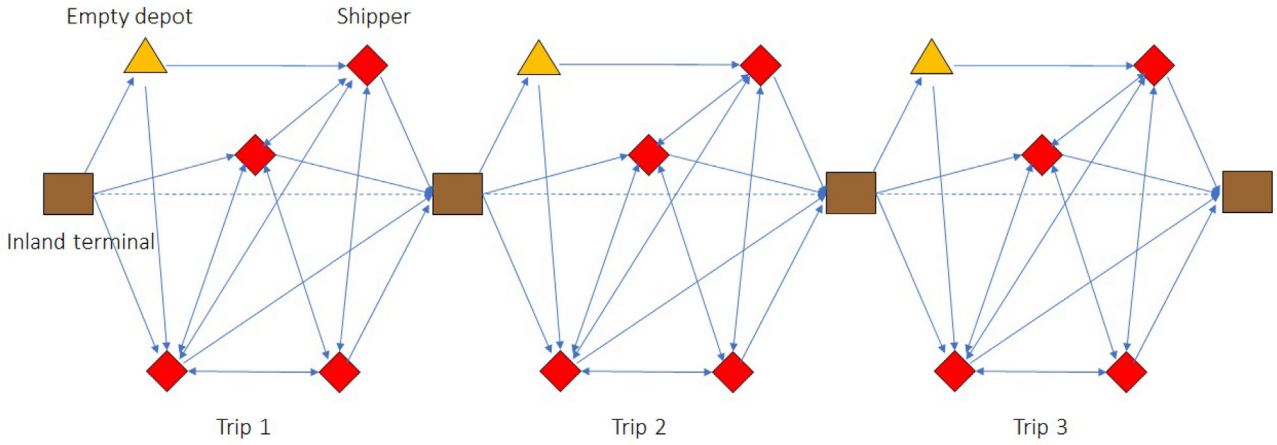

Fig. 3. Example of a replicated graph.

example, if a multi-trip path 'p' takes out two empty containers before 't,' $H_{pt} = 2$. On the other hand, if 'p' brings in two empty containers before 't,' $H_{pt} = -2$.

For the first Branch-and-Price algorithm, in which the restricted master problem is fed with routes/paths containing one or multiple trips, henceforth "B&P - Multi," we develop the following master problem:

$$\min \sum_{p\in\Omega} c_p \lambda_p \tag{20}$$

$$\sum_{p\in\Omega} a_{ip}\lambda_p \geq 1 \qquad \forall i \in \mathcal{S} \tag{21}$$

$$\sum_{p\in\Omega} \lambda_p \leq |K| \tag{22}$$

$$\sum_{p\in\Omega} H_{pt}\lambda_p \leq E \qquad \forall t \in T^{crit} \tag{23}$$

$$\lambda_p \geq 0 \quad \text{and integer} \qquad \forall p \in \Omega \tag{24}$$

Constraint (21) imposes that every customer is covered in at least one of the chosen paths. With constraint (22), the number of selected paths does not exceed the number of trucks. Finally, inequality (23) ensures that the stock of empty containers is not exceeded at the inland terminal. Note that this constraint must be defined only for the critical times when the paths take out an empty container. If feasibility is ensured at these times, the feasibility at other times will follow consequently. See Fig. 3 for a depiction of a replicated graph for the "B&P - Multi" procedure.

Similarly, for the Branch-and-Price algorithm in which the master problem is fed with routes/paths containing single trips, henceforth "B&P - Single," the master problem is similar, except for the following constraint, replacing (22):

$$\sum_{p\in\Omega} Z_{pt}\lambda_p \leq |K| \qquad \forall t \in R^{crit} \tag{25}$$

where R^{crit} is the set of critical times when paths start their trips. Z_{pt} takes value 1 if path p begins before or at critical time t and is still active, 0 otherwise. This constraint ensures that the number of single trips active at every critical time does not exceed the number of available trucks. Finally, also in this case, the size of R^{crit} increases if the created paths provide new departure times.

The described master problems represent the MILP model (1)–(19) because a path by definition includes a trip/ set of trips which satisfy the time windows, and the master problem enforces that all the customers are covered by the selected paths and that the empty container limitation is respected. These feasible paths are constructed in the pricing problem described in the following subsection. Contrary to these master problems, which can be easily solved with a commercial solver, solving the pricing problem requires a more sophisticated approach.

4.4. Pricing problem

For "B&P - Multi," the pricing problem entails finding a path with negative reduced cost, defined as $\Delta(p) = c_p - \sum_{i\in\mathcal{S}} a_{ip}\delta_i - \psi - \sum_{i\in T^{crit}} H_{pt}\lambda_i$, where δ_i, ψ and λ_i are the dual values corresponding to primal constraints (21), (22), and (23). Whereas, for "B&P - Single," the reduced cost is defined as $\Delta(p) = c_p - \sum_{i\in\mathcal{S}} a_{ip}\delta_i - \sum_{i\in R^{crit}} Z_{pt}\psi_i - \sum_{i\in T^{crit}} H_{pt}\lambda_i$, where ψ_i are dual variables related to primal constraints (25).

For both approaches, the pricing problem can be treated as an Elementary Shortest Path Problem with Resource Constraints (ESPPRC), which is NP-hard in the strong sense. Note that although a truck can visit the seaport and the empty container depot multiple times in a particular trip, it is still valid to formulate the subproblem as an Elementary Shortest Path Problem with Resource Constraints. This is because the special 'bypass' arcs allow this possibility implicitly.

Several algorithms have been developed in the literature to solve the ESPPRC. Due to NP-hardness, the required computational time may be unbearable, especially when a negative reduced cost path does not exist and the algorithms are required to search the whole solution space. Due to this, a common approach is to try to solve the problem heuristically and to run an exact algorithm when the heuristics fail to find a negative reduced cost path (Desaulniers, Lessard, & Hadjar, 2008; Guerriero, Di Puglia Pugliese, & Macrina, 2019).

In this paper, we adopt a Dynamic Programming algorithm, a Pulse algorithm, and a Roll-out algorithm. The latter is a greedy algorithm, whereas Dynamic programming and Pulse algorithms are exact methods that can guarantee the non-existence of a negative reduced cost path. The Pulse algorithm developed by (Lozano, Duque, & Medaglia, 2016) proved to be a very efficient method, whereas classical labelling algorithms such as dynamic programming are notoriously slow if the dominance criteria are not effective enough. Therefore, we will combine these algorithms to make the search more efficient. In particular, roll-out is launched first. If the given path does not have a negative reduced cost, a relaxed version of the dynamic programming algorithm with weak dominance criteria is run. When both Roll-out and Dynamic program-

ming fail at their task, the exact Pulse algorithm is launched. We refer to Section 5.2 for the fine-tuning of this strategy.

Before specifying the details of these algorithms, we present the structural properties of the pricing problem when dealing with both "B&P - Multi" and "B&P - Single".

Principles for generating multi-trips paths

For the "B&P - multi," we aim to generate paths on a replicated graph with the least negative reduced cost. The inland terminal can be visited indefinitely for as many trips as the path can allow. However, to bound the search for the latter case, we will limit the number of trips to R.

Concerning the decision on the departure time of each trip from the depot, one crucial aspect is whether to allow a truck to postpone its departure. In this case, assessing all different combinations may turn out to be unbearable from a computational point of view. The following lemma guarantees that creating multi-trip paths does not require this assessment.

Lemma 1. *There is always an optimal solution to the master problem with no truck waiting at the inland terminal. In other words, if the master problem is solved only on the set of the paths in which there is no waiting time at the inland terminal, an optimal solution to the problem is still obtained.*

Proof. We prove the lemma by contradiction. Let P_1 be the set of all paths, including the ones with waiting times at the inland terminal, and let P_2 be the set of paths with no waiting times at the inland terminal. Suppose there is no optimal solution that has the paths of all the trucks coming from P_2. Then consider any optimal solution to the problem. It will have at least one path where the truck waits at the inland terminal. The following two cases may occur:

Case 1: A truck (say $T1$) waits for an empty container to arrive at the inland terminal. Assume that after the container is brought by another truck (say $T2$), this truck visits a customer i requiring an empty container and let us define the future path of the truck after customer i as $R1$. In this case, a new solution without a waiting time at the inland terminal can be constructed with a better or equivalent cost. This is done by continuing the previous trip of the truck ($T2$) that brings in the empty container to the inland terminal with customer i and path $R1$. That is, truck $T2$, which had initially brought in the empty container for truck $T1$, serves the future customers of $T1$ without ending its current trip. $T1$ then serves the prospective customers of $T2$ without waiting at the inland terminal. The triangular inequality ensures that the cost of this solution is better or equal to the supposed optimal solution. See Fig. 4

Case 2: A truck waits for the opening time windows of the next customer. In this case, the truck can simply depart early from the inland terminal and wait at the customer's location for its time window to open.

Thus, we can modify the paths of the considered optimal solution to construct another feasible solution with no waiting time. The new solution has an equal or better objective value. This contradicts our assumption that there is no solution with all the selected paths coming from the set P_2. Therefore, there is at least one optimal solution with all the selected paths in P_2. □

Principles for generating single-trip paths

For the "B&P - single," we need to create paths composed of one trip. The starting time is an important feature to compute since starting all trips at time "0" will impede the creation of multi-trip paths. Hence, this approach requires finding paths starting after each critical moment belonging to sets R^{crit} and T^{crit}, whose dual value is not zero.

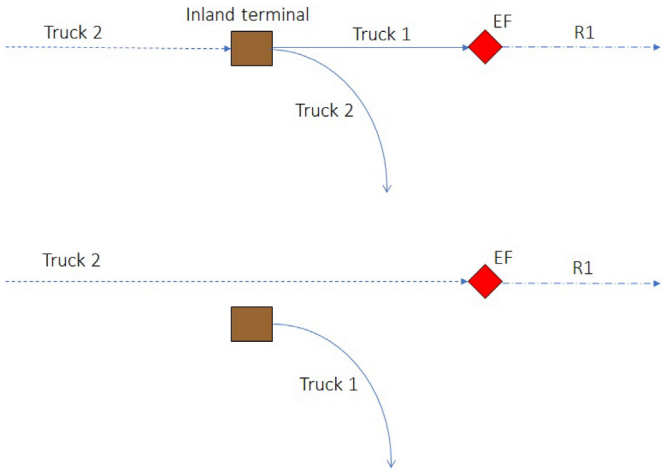

Fig. 4. Case 1: Paths of two trucks interchanged to construct a solution with no waiting time at the inland terminal.

4.4.1. Roll-out algorithm

To support the fast generation of negative reduced costs path in the first iterations of the column generation, a "Roll-out" heuristic is developed. Guerriero et al. (2019) proposed this algorithm to solve the ESPPRC, and they showed it outperformed a Tabu Search approach. The process starts with an initial trivial subpath P containing only the inland terminal. This is iteratively extended with additional nodes until the final destination node is reached. The new node is selected with the following procedure.

First, from every node i attainable from the last node in P, a greedy algorithm is launched to find the minimum cost path to the destination node (i.e., the inland terminal). The greedy algorithm, starting from i, computes paths from each neighbor of i, say j, to the inland terminal by selecting the best neighbor generating the largest decrement in reduced costs. The j whose path has the minimum reduced cost from i is selected. Starting from j, the same steps are repeated. Once j is the destination node, the search is stopped, and a final cost for the initial node i is computed. The node i, whose sum of the distance from the last node of the current path and the greedy heuristic objective cost for the neighbor is minimum, is added to the current path P. See 1 for the pseudocode of the algorithm.

4.4.2. Dynamic programming

The Dynamic Programming (DP) algorithm builds new paths starting from the inland terminal and iteratively extends them in all possible directions. This exhaustive search is generally made more efficient by discarding non-promising paths (i.e., dominated). We refer the reader to Irnich & Desaulniers (2005) for an excellent illustration of DP applied to solve shortest path problems with resource constraints.

The algorithm can be initialized by creating a pool U of subpaths with an initial trivial subpath consisting of the inland terminal only. For each created subpath p, we store the following information: arrival time at the last node $T(p)$, current path cost $C(p)$, current path partial reduced cost $\Delta(p)$, current number of trips effectuated $R(p)$, last visited node $v(p)$. For dominance purposes, we also store the subset of shippers $V(p)$ contained in the subpath, and the shippers that cannot be added anymore because of time windows restrictions.

The DP exhaustive search starts by selecting from the pool of subpaths U the one with the least reduced cost $\Delta(p)$. This selected path p is extended with one additional node for every possible node whose addition to the path is feasible. Every time the subpath is extended with a node $s \in \mathcal{S}$, the algorithm checks whether

Algorithm 1 Rollout Algorithm.

input: a problem instance data, dual values
Best negative reduced cost path $p^* = \varnothing$ with reduced cost $S^* \leftarrow M$ ▷ M is a large value
Create set $P = \{0\}$ ▷ 0 is the inland terminal
while P can be extended **do**
  Select last node i from P
  Set $i' = -1$ ▷ possible new node
  Set $S \leftarrow M$
  for $j = 0$ to N **do**
    if node j can be added to path P **then**
      Compute subpath from j till final inland terminal by selecting iteratively neighbours
      generating the largest decrement in reduced cost
      Compute reduced cost C of the subpath from j
      if $C < S$ **then**
        $S \leftarrow C$ and $i' \leftarrow j$
      end if
    end if
  end for
  if $i' \neq -1$ **then**
    $P \leftarrow P \cup \{i'\}$
    if $i' = 0$ **then**
      Compute current reduced cost C of path P
      if $C < S^*$ **then**
        $S^* \leftarrow C$ and $p^* \leftarrow P$ ▷ In case of "B&P - single" stop here
      end if
    end if
  else Stop while
  end if
end while
output: best solution found p^* with cost S^*

the new subpath is dominated by some path in U, ending with the same node s. At the end of the procedure, the algorithm updates pool U by adding non-dominated paths and deleting dominated ones. If a subpath is extended with the inland terminal, it is either closed if the maximum number of trips is reached, or it is copied directly into U. In any case, such a path can be considered complete, and it is compared with the best solution found so far, say S^*.

The dominance criteria for an ESPPRC, useful to limit the exhaustive search, are related to the number of consumed resources and costs of the subpaths. The first necessary (but not sufficient) set of dominance conditions, named DC1, entail that candidate dominated subpaths are paths q against p, such that $v(p) = v(q)$ and that the following conditions hold: $R(p) \leq R(q)$, $\Delta(p) \leq \Delta(q)$, $C(p) \leq C(q)$, and in at least one there is no equality. Next, if $\sum_{i \in T^{crit}} H_{pt}\delta_i = 0$ and if $T(p) \leq T(q)$ and $V_p \subseteq V_q$, then the dominance is established. On the other hand, if $\sum_{i \in T^{crit}} H_{pt}\delta_i \neq 0$, dominance cannot be easily established because $T(p) \leq T(q)$ implies that path p has more chances to retrieve a container before a critical time and thus be penalized.

Despite the usefulness of these dominance criteria to reduce the search space, the DP can be very slow in either finding the most negative path or demonstrating that no negative reduced cost path exists, especially for large-scale instances. Therefore, in the implementation, we will use a DP with relaxed dominance criteria and stop the algorithm after the solution is not improved after a set number of iterations. Concerning the relaxed dominance criteria, a path q is considered dominated by a path p if conditions DC1 hold and if $\sum_{i \in T^{crit}} H_{pt}\delta_i = 0$ or $T(p) \geq T(q)$. If the DP fails to yield a negative reduced cost path, the Pulse Algorithm is run to ensure that a negative reduced cost path is generated whenever such a path exists. The pseudocode for the DP algorithm is provided in 2.

Algorithm 2 Dynamic Programming Algorithm.

input: a problem instance data, dual values
Best negative reduced cost path $p^* = \varnothing$ with reduced cost $S^* \leftarrow M$ ▷ M is a large value
Create set $U = \varnothing$
Populate U with path $\{0\}$ ▷ Node 0 indicates the inland terminal
while $U \neq \varnothing$ **do**
  Select subpath $p \in U$ with least reduced cost
  for $s = 0$ to $N - 1$ **do** ▷ Range set of nodes $N = 0 \dots N - 1$
    if feasible to add s to subpath p **then**
      $p' \leftarrow p + \{s\}$
      Compute: $T(p')$, $C(p')$, $\Delta(p')$, $R(p')$, $V(p')$
      if $s \in \mathcal{S}$ (set of Shippers) **then**
        if p' dominates path $p'' \in U$ with final node s **then** ▷ Start check dominance criteria
          Replace $p'' \in U$ with p'
        else if p' is dominated by any path $p'' \in U$ with final node s **then**
          Delete p'
        else Add p' to U
        end if
      else if $s = 0$ & $\Delta(p') < S^*$ **then** ▷ $\Delta(p')$ is the reduced cost of path p'
        $S^* \leftarrow \Delta(p')$ and $p^* \leftarrow p'$
        Store information p' and add p' to U
      end if
    end if
  end for
  if path p cannot be extended with any node **then**
    Delete p from U
  end if
end while
output: best solution found p^* with cost S^*

4.4.3. Pulse algorithm

The pulse algorithm is an exact two-stage algorithm to solve ESPPRC, proposed by Lozano et al. (2016). In the first stage, bounds for subpaths are computed and are used in the second stage to efficiently prune the definition of complete paths in a branch-and-bound search.

The algorithm is initialized by defining a so-called bounding matrix B for subpaths, whose (i, j) entry is the best partial reduced cost that can be obtained starting from a node $j \in N/\{0\}$ and a time represented by row i and ending at the inland terminal. In particular, the rows of B define the set of possible starting times subpaths can start. The first row is the latest possible time, $T_{\max}$, decreased of a fixed quantity T row by row until the last one, indicating starting time 0. Evaluating more time steps will create tighter bounds for the second stage but will increase the number of computations required. The number of columns equals all possible nodes that can compose a subpath. In the case of "B&P - single," the first columns refer to the empty depot and the shippers, and the last one refers to the inland terminal for the end of the trip. In the case of "B&P - multi," there is a column for each node in the replicated graph.

In the first stage of the algorithm, we fill matrix B; we start from the first row and iteratively process each column. Given a column j, representing the starting node, and a row i, representing a starting time, the value of B_{ij} is the most negative reduced cost S of a "partial" subpath starting from this (root) node j at the time

of row i and ending to the inland terminal. In case the node of a selected column j has a due date smaller than the time of the reference row i, then we set B_{ij} equal to a large number M to indicate infeasibility and select the next column. On the other hand, if the due date is larger, we try all possible and feasible subpath extensions from the root node with a branch-and-bound algorithm. In particular, we start investigating possible paths by adding leaves (i.e., nodes) to the branch-and-bound tree starting from node j. Each leaf of the tree contains labels similar to those described in the DP algorithms to keep track of the subpaths' features. Once a new leaf is added, two procedures are activated to possibly prune the leaf. In the first, the algorithm closes the new leaf l, whose related node is p, if its partial reduced cost plus the cost $B_{hp} \neq M$ $\forall h \leq i$ is larger than the current best found partial path S of the tree. In other words, the leaf is pruned if the cost of the current subpath ending at p plus the cost of the best subpaths starting at p previously found is larger than current best S. In the second, a so-called rollback pruning rule is applied. If both the reduced cost of the new leaf l and the arrival time at the final node p are greater than or equal to the values of the other leaf ending at p at the preceding layer of the tree, leaf l is pruned. If the leaf survives, new leaves are created from it and explored.

In the second stage of the Pulse algorithm, we start a single branch-and-bound tree whose root node is, in this case, the inland terminal. We perform the same procedures to explore the tree as in the first stage, pruning the tree using matrix B. Finally, once the tree is fully explored, we select the minimum reduced cost path found S^*. See 3 for the pseudocode of the algorithm.

Preliminary experiments on the "B&P - multi" revealed that for some challenging instances, proving the non-existence of negative reduced cost paths in the last iteration turned out to be too slow even for the Pulse algorithm. In this case, we use the following technique, which works on a reduced graph size corresponding to just one trip (i.e. no replication of the graph) to prove the non-existence of a negative reduced cost path. A Pulse algorithm solving the problem for finding a minimum reduced cost just for a single trip case is launched. We solve this for all the starting times corresponding to every critical time in T^{crit} in the solutions whose dual is not zero. After the conclusion of the Pulse algorithm, we will have obtained the minimum possible reduced cost path for a single trip selected from among all possible critical starting times from the inland terminal. This yields information on the lower bound of the minimum reduced cost path possible for the entire replicated graph. The minimum possible reduced cost for the entire graph is obtained by multiplying this minimum partial reduced cost of a single trip with the maximum number of trips in the graph $|R|$ and finally subtracting the dual variable ψ corresponding to constraint 22. This is because the other trips' partial reduced cost will be at least equal or higher because of less time available to serve customers, reducing possible combinations of the visit sequence of the customers. Consequently, when this lower bound is non-negative, the non-existence of negative reduced cost paths will be guaranteed.

4.5. Branch-and-bound scheme

Since the column-and-row generation solves the relaxed version of the master problem, the solution is quite often not an integer solution. In such a case, we branch on arcs to create two subsequent "child" problems (nodes or leaves of the tree).

In particular, as soon as the pricing problem cannot generate a negative reduced cost path, the final relaxed solution is checked. Each arc (i, j) of paths p, for which λ_p is not integer (subset of paths P^F), is listed and its value computed as $\sum_{p \in P^F} \lambda_p e^p_{(i,j)}$ (where $e^p_{(i,j)} = 1$ if (i, j) is in path 'p', 0 otherwise). Among all arcs with fractional values, we choose the one whose absolute value of the difference with zero or one is minimum. In the first child problem, such an arc is enforced in one of the trucks' routes, and we modify the pricing problem such that any subpath reaching node 'i' can only be extended to node 'j.' On the other child node, we prohibit the arc (i, j) from the pricing problem. Finally, if a node does not generate arcs with fractional values, it is closed.

The branch-and-bound tree is searched in a breadth-first fashion. The following rules apply: a node is pruned if its lower bound (the solution of the final restricted master problem) is higher than the current upper bound of the tree; the paths generated in a parent node of the B&B tree are passed on to the appropriate child node to improve the computation of the children nodes; only when all nodes of a level are solved, the global UB and LB are updated. See Fig. 5.

Algorithm 3 Pulse Algorithm.

input: a problem instance data, dual values
Best negative reduced cost path $p^* = \varnothing$ with reduced cost $S^* \leftarrow M$ ▷ M is a large value
Time Step T
Define matrix B with $\#rows = int(Tmax/T) + 1$ ▷ Tmax maximum possible time $\#columns = N$ ▷ Case of "B&P - single"
Initialize all elements of matrix B with M
for $i = 0$ to $\#rows$ **do**
  for $j = 0$ to $\#columns$ **do**
    Set $S = M$
    Create root of the branch-and-bound tree with node j and start time $T_{\max} - (i + 1) * T$
    while The tree is not fully explored **do**
      Select a leaf of the tree
      Create new leaves by extending the subpath of the selected leaf to all feasible nodes
      Remove dominated leaves
      if a subpath ends to the final node & reduced cost is lower than S **then**
        Update S
      end if
    end while
    $B_{ij} \leftarrow S$
  end for
end for
Create root of the branch-and-bound tree with node 0 ▷ node 0 is the inland terminal
while The tree is not fully explored **do**
  Select a leaf of the tree
  Create new leaves by extending the subpath of the selected leaf to all feasible nodes
  Remove dominated leaves
  if a subpath p ends to the final node & its reduced cost C is lower than S^* **then**
    $S^* \leftarrow C$ and $p^* \leftarrow p$
  end if
end while
output: best solution found p^* with cost S^*

5. Numerical experiments

In this section, we test the performances of the proposed exact methods and compare them with the MILP formulation, processed in a standard branch-and-cut solver. Next, we perform experiments on real-world instances inspired by a Dutch case study and draw managerial insights.

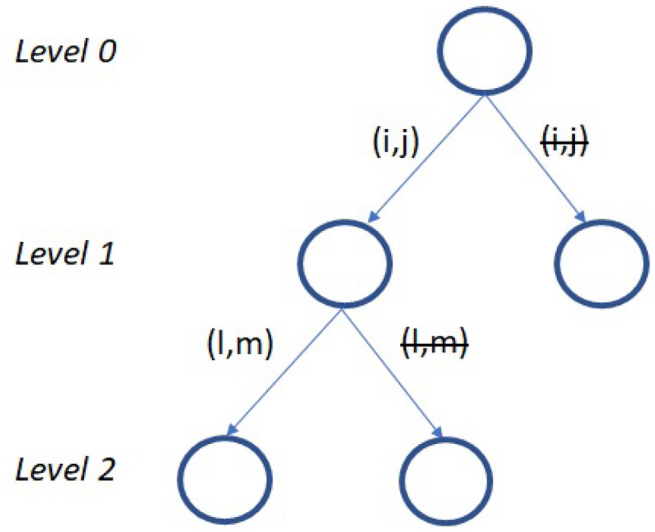

Fig. 5. Branching scheme.

This section is structured as follows. First, we describe the chosen parametrization and technical details. Second, we show the results on the classical Solomon's instances. Finally, the experiments on real-world instances are presented.

5.1. Parametrization and technical aspects

The initial solution generator selects a number of closest neighboring points to build the solution. This number was set to six to guarantee an initial feasible solution to all tried instances.

The selection of the algorithms to solve the pricing problem requires attention. The chosen strategy is based on preliminary experimentation that gave insights into the performances of the algorithms in terms of speed and solution quality. The outcome was that the Dynamic Programming algorithm with relaxed dominance criteria had overall the best average performance, followed by the Pulse algorithm and the Dynamic programming with complete dominance criteria. See Fig. 6 for the test results on a 25 nodes instance in eight sample iterations. DP with exact criteria was stopped after 15 seconds of computation, whereas the other two could continue until the end of the exploration of their respective search spaces. The graph on the left shows the solution quality, and the one on the right shows the computational time required. In general, DP with relaxed criteria is the most performing algorithm. The solution quality is comparable with the exact Pulse algorithm, and the speed is better overall. The DP with exact criteria is too slow to attain the most negative reduced cost path.

Therefore, we adopt the following strategies for the two B&P frameworks. We first run the roll-out algorithm for "B&P - multi." If the roll-out fails to find a negative reduced cost path (NRCP), the Dynamic Programming algorithm with relaxed dominance criteria is run and stopped after 5000 iterations without improvements. Next, if an NRCP is not found, the Pulse algorithm is launched to find a single-trip NRCP. The procedure is stopped if the NRCP is not found, and it is possible to prove that such a path cannot be attained. Otherwise, a final Pulse algorithm is launched. Finally, for "B&P - single," we first run the Dynamic Programming algorithm with relaxed dominance criteria and next, if unsuccessful, the Pulse algorithm.

In terms of technical settings, we ran the experiments on a high-performing machine with a 24 Intel Xeon 2.5 GHz cores and 128 GB of internal memory. All algorithms were coded in C++. The MILP formulation was integrated into CPLEX using the Concert Technology in C++ with twenty-four threads in default conditions. The time limit of the experiments was set to 3 hours for every instance.

5.2. Tests on classical instances

We chose the classical Solomon's instances for the Vehicle Routing Problem with Time Windows with 25, 50, and 100 nodes. We selected C1 and RC2 types of instances for a total of 17 instances per instance size. C1 instances, nine in total, have clustered customers and narrow time windows. RC, eight in total, have a combination of randomly placed and clustered customers and larger time windows. The reason is due to the common configuration of drayage systems, with shippers' warehouses typically located in clustered industrial areas. Additional coordinates of the seaport and the empty depot were set respectively to (70,80) and (0,0). The empty depot's coordinates were chosen to penalize the need to reach it. The customers' types and the origin and the destination of each container, be it the inland terminal or the seaport, were generated randomly.

Tables 2–4 report the results for the 25, 50, and 100 nodes instances, respectively. We compare the performances of the MILP formulation (CPLEX MILP), the B&P focusing on multi-trips (B&P - Multi), and the B&P focusing on single trips (B&P - Single). For the 100 nodes instances, we also provide an experiment, reported in Table 5, where we impose that in the "B&P - Single," the relaxed Dynamic Programming algorithm is stopped as soon as a negative path is found; hence, we do not let it run for 5.000 iterations until further improvements.

From Table 2, it emerges that our B&P frameworks can find all optimal solutions in a very short time. It is noticeable how the "B&P - Single" has a faster computation than "B&P - Multi," and more evidently, for instance RC203. The performance of the MILP formulation embedded in CPLEX is quite satisfactory for this set of instances, though the optimal solution was not found in five cases. The average gap is about 3%. Because results are very close in this experiment, we provide the timings when CPLEX achieved the best gap. Finally, we notice how CPLEX manages to compute, overall, reasonable upper bounds.

The second set of instances turns out to be already very challenging for CPLEX. In Table 3, we report the results. For six instances, an upper bound could not be computed. However, the lower bounds are acceptable and close to the optimal solutions found by the B&P algorithms. Again, although both methods solve almost all instances to optimality, the B&P focusing on single paths has the edge in terms of computational times, which are lower for most of the cases. For instance RC208, the "B&P - Single" required more time, but a close analysis showed that this resulted from an unfavourable branching. Overall, the B&P algorithms offer remarkable performance, with competitive computational times.

Finally, in Table 4, we show the results for the challenging 100 nodes instances. The "B&P - Single" algorithm has the edge also in this set, finding the optimal solution for eight instances out of seventeen and almost for another five instances. Both B&P algorithms could not completely solve the root node for four instances; hence, they could not provide a lower bound. The "B&P - Multi" proved to be quite competitive, but it required more attention in the parametrization phase to get acceptable results. However, the superiority of the "B&P - Single" is indisputable, confirming previous insights from the literature (Hernandez et al., 2016; Mingozzi et al., 2013). On the other hand, CPLEX could not handle any of the instances, showing the importance of developing tailored algorithms to solve larger instances.

Finally, Table 5 shows the experiment with "B&P - Single" accepting the first negative path found in the Dynamic Programming

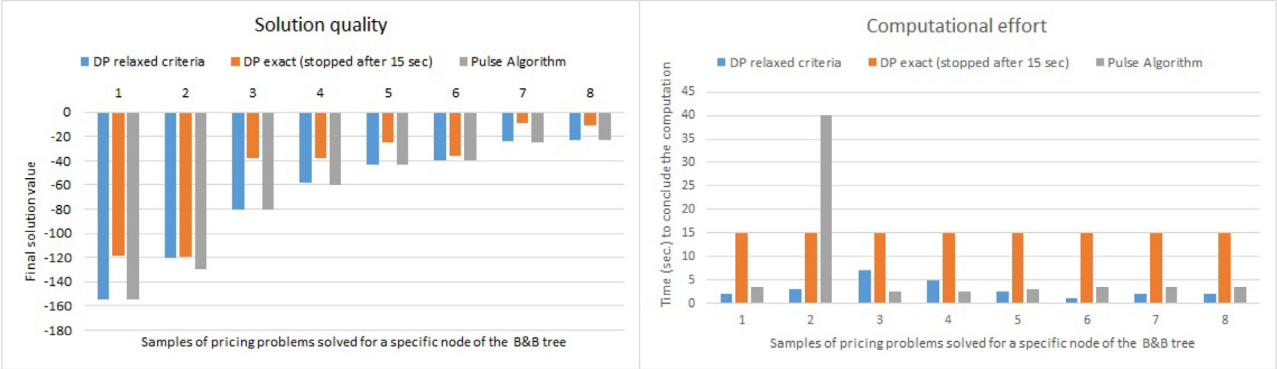

Fig. 6. Comparison of the performances of the three algorithms for a 25 nodes instance for a node of the B&B tree in eight sample iterations when solving the pricing problem.

Table 2
Results on 25 nodes adapted Solomon's instances. UB and LB stand respectively for the upper bound and lower bound of the minimization problem.

	CPLEX MILP				B&P - Multi				B&P - Single			
Instance	UB	LB	Gap(%)	Time (sec.)	UB	LB	Gap(%)	Time(sec.)	UB	LB	Gap(%)	Time(sec.)
C101	1698	1494	12	5310	1698	1698	0	307	1698	1698	0	1
C102	1118	1118	0	146	1118	1118	0	1	1118	1118	0	4
C103	1120	1120	0	50	1120	1120	0	5	1120	1120	0	1
C104	767	767	0	662	767	767	0	25	767	767	0	17
C105	1869	1361	27.1	2348	1869	1869	0	1	1869	1869	0	1
C106	1497	1375	8.1	419	1497	1497	0	4	1497	1497	0	12
C107	1264	1264	0	28	1264	1264	0	1	1264	1264	0	1
C108	938	938	0	18	938	938	0	4	938	938	0	1
C109	968	968	0	28	968	968	0	4	968	968	0	3
RC201	2092	2092	0	245	2092	2092	0	2	2092	2092	0	1
RC202	2169	2169	0	181	2169	2169	0	1	2169	2169	0	2
RC203	1924	1717	10.7	9363	1924	1924	0	1979	1924	1924	0	88
RC204	1423	1423	0	116	1423	1423	0	8	1423	1423	0	2
RC205	1921	1919	0.1	72	1921	1921	0	11	1921	1921	0	6
RC206	1912	1912	0	253	1912	1912	0	2	1912	1912	0	1
RC207	1500	1500	0	634	1500	1500	0	1	1500	1500	0	1
RC208	1876	1794	4.3	1768	1876	1876	0	60	1876	1876	0	117

Table 3
Results on 50 nodes adapted Solomon's instances.

	CPLEX MILP				B&P - Multi				B&P - Single			
Instance	UB	LB	Gap(%)	Time(sec.)	UB	LB	Gap(%)	Time(sec.)	UB	LB	Gap(%)	Time(sec.)
C101	2709	2709	0	6059	2709	2709	0	5463	2709	2709	0	10
C102	-	2489	-	10,800	2731	2731	0	447	2731	2731	0	220
C103	2801	2681	4.2	10,800	2729	2729	0	1278	2729	2729	0	734
C104	2104	2104	0	1445	2104	2104	0	176	2104	2104	0	188
C105	3178	2410	24.1	10,800	2981	2981	0	313	2981	2981	0	18
C106	2942	2841	3.4	10,800	2859	2859	0	4882	2859	2859	0	420
C107	3008	3001	0.1	10,800	3008	3008	0	54	3008	3008	0	10
C108	-	2504	-	10,800	3131	3122.5	0.1	10,800	3131	3122.5	0.1	10,800
C109	-	2471	-	10,800	3351	3351	0	1987	3351	3351	0	726
RC201	3699	3592	2.8	10,800	3606	3606	0	947	3606	3606	0	71
RC202	-	3825	-	10,800	3856	3856	0	786	3856	3856	0	306
RC203	3650	3445	5.6	10,800	3457	3457	0	291	3457	3457	0	466
RC204	-	2817	-	10,800	2821	2821	0	1754	2821	2821	0	900
RC205	3627	3405	6.1	10,800	3491	3491	0	874	3491	3491	0	585
RC206	3548	3476	2.1	10,800	3541	3541	0	264	3541	3541	0	42
RC207	-	3464	-	10,800	3751	3751	0	803	3751	3751	0	406
RC208	-	3052	-	10,800	3052	3052	0	1358	3052	3052	0	5051

algorithm. An in-depth analysis showed how this approach created several negative reduced cost paths, which were unnecessary for the final restricted master problem and thus required the algorithm to linger more than necessary at the root node.

5.3. Tests on real-world instances

This section shows tests on a set of instances inspired by a case study of an inland terminal in the Netherlands, providing data on a set of containers handled in 2021. The terminal is located in the neighborhood of Tilburg and s'Hertogenbosch and carries out regular transport of containers to and from the nearby seaport of Rotterdam. Planning and scheduling of truck transport is done manually by experienced planners, supported by data sheets. The terminal owns a fleet of trucks and can retrieve empty containers from a nearby depot owned by a shipping line. The exact location of this depot was not disclosed, so in the experiments, we considered one available in Antwerp. This site is chosen to penalize the retrieval

Table 4
Results on 100 nodes adapted Solomon's instances.

	CPLEX MILP				B&P - Multi				B&P - Single			
Instance	UB	LB	Gap(%)	Time (sec.)	UB	LB	Gap(%)	Time (sec.)	UB	LB	Gap(%)	Time (sec.)
C101	-	-	-	10,800	6013	6012.5	0.1	10,800	6013	6012.5	0.1	10,800
C102	-	-	-	10,800	6421	6416.5	0.1	10,800	6417	6416.5	0.1	10,800
C103	-	-	-	10,800	6720	5088	32.1	10,800	5089	5088	0.1	10,800
C104	-	-	-	10,800	7259	-	-	10,800	7259	-	-	10,800
C105	-	-	-	10,800	7594	6183	22.8	10,800	6183	6183	0	362
C106	-	-	-	10,800	5294	5285	0.1	10,800	5285	5285	0	767
C107	-	-	-	10,800	4771	4759	0.1	10,800	4759	4759	0	919
C108	-	-	-	10,800	6892	-	-	10,800	5533	5533	0	580
C109	-	-	-	10,800	7937	-	-	10,800	5823	5823	0	4322
RC201	-	-	-	10,800	7966	6177.28	28.9	10,800	6282	6281	0.1	10,800
RC202	-	-	-	10,800	6018	6014.33	0.1	10,800	6015	6014.5	0.1	10,800
RC203	-	-	-	10,800	7407	-	-	10,800	7407	-	-	10,800
RC204	-	-	-	10,800	7759	-	-	10,800	7759	-	-	10,800
RC205	-	-	-	10,800	5561	5552	0.1	10,800	5552	5552	0	644
RC206	-	-	-	10,800	7053	5884	19.8	10,800	5884	5884	0	422
RC207	-	-	-	10,800	5624	5613	0.1	10,800	5613	5613	0	2332
RC208	-	-	-	10,800	7660	-	-	10,800	7660	-	-	10,800

Table 5
Results on 100 nodes adapted Solomon's instances of the B&P algorithm accepting first generated negative reduced cost path.

	B&P - FNA			
Instance	Best Integer	Best Bound	Gap	Time (sec.)
C101	6013	6012.5	0.1	10,800
C102	8325	-	-	10,800
C103	5088	5088	0	1436
C104	7259	-	-	10,800
C105	6183	6183	0	863
C106	5285	5285	0	1899
C107	6446	-	-	10,800
C108	5533	5533	0	748
C109	8087	-	-	10,800
RC201	6282	6280.53	0.1	10,800
RC202	7837	-	-	10,800
RC203	7407	-	-	10,800
RC204	7759	-	-	10,800
RC205	5552	5552	0	1522
RC206	5884	5884	0	954
RC207	7507	-	-	10,800
RC208	7660	-	-	10,800

Table 6
Results on a set of instances inspired by a real-world case.

	Planner		B&P - single		
	Obj.	#trips	Obj.	#trips	Improvement
Inst1	1953	22	1559	16	−25.2%
Inst2	2279	21	1775	15	−28.3%
Inst3	1719	17	1230	9	−39.7%
Inst4	1429	13	1131	10	−26.3%
Inst5	2392	23	1670	14	−43.2%

Table 7
Results to analyze the impact of reducing the possibility of empty container re-usage. "Fewer $\Diamond^{FE}$ - B&P" heading refers to the transformed instances with fewer opportunities for empty containers' re-usage.

	Basic real-world instances - B&P		Fewer $\Diamond^{FE}$ - B&P		
	Obj.	visits empty depot	Obj.	visits empty depot	Obj. Deterioration
Inst1	1559	10	1589	12	+1.9%
Inst2	1775	15	1928	19	+8.6%
Inst3	1230	3	1365	9	+10.9%
Inst4	1131	3	1270	4	+12.2%
Inst5	1670	5	1813	14	+8.5%

of empty containers to a certain degree. A big part of the pickups and deliveries is located in the Netherlands with high density in the neighborhood. Sometimes containers are transported to Germany or Belgium, but typically not too far away from the border. In contrast to the setting proposed by Zhang et al. (2020), both import and export flows are present in the region.

To draw managerial insights, we process a set of real-world instances with our "B&P - single" algorithm, given its superiority in the previously reported tests. Some instances' data, such as locations, customer types, and time windows, when absent, was generated based on the information in the available data set and the insights from the planners.

We perform four distinct experiments. In the first, we compare the solution of our model with the one from the planners, highlighting the number of trips required to perform the tasks. In the second, we attempt to quantify the adverse effects of having less opportunity to reuse empty containers by reducing the number of shippers that freely release empty containers in the network. In the third, we analyze the impact of the initial stock of empty containers in the solution. Finally, we evaluate the impact of the distance of the empty depot from the inland terminal to the solution. Because the algorithm was able to solve all instances, we do not report the optimality gap in the tables.

5.3.1. Evaluating planners' solutions

We evaluate the decision-making of the planners based on their input and the information available in the data set. Table 6 shows the results emphasizing the difference in the number of trips required.

In the first part of the experiment in Table 6, we quantify the gain when optimizing. On average, the solutions could be improved by 30%, with an average saving of ≈ 900km per instance. The table also exhibits how the algorithm can consolidate more requests in single trips.

5.3.2. Impact of empty container re-usage

In this experiment, we consider the five basic instances from the data-set and reduce the number of shippers who could release empty containers for re-usage ($\Diamond^{FE}$). In particular, we transform customers of type $\Diamond^{FE}$, the one releasing empty containers, into customers of type $\Diamond^{F\varnothing}$ and $\Diamond^{FF}$. With the latter, we want to recreate the situation where an empty container must be returned to the seaport's empty depots to avoid so-called "detention" fees, or in general lack of coordination between the shipper and the transport operator. Results in Table 7 show an increase in covered mileage when there are fewer opportunities for empties' re-usage

Table 8
Results for the 50-nodes Solomon's instances to analyze the impact of reducing the possibility of empty container re-usage.

	Solomon's instances		Fewer $\Diamond^{FE}$		
Instance	Obj.	Visits empty depot	Obj.	Deterioration	Visits empty depot
C101	2709	9	3515	+29.8%	15
C102	2731	12	3577	+31.0%	18
C103	2729	12	3476	+27.4%	14
C104	2104	0	2183	+3.8%	4
C105	2981	19	4225	+41.7%	24
C106	2859	14	3221	+12.7%	16
C107	3008	5	3145	+4.6%	9
C108	3131	17	3651	+16.6%	20
C109	3351	19	4041	+20.6%	25
RC201	3606	0	4283	+18.8%	9
RC202	3856	10	4420	+14.6%	14
RC203	3457	11	4058	+17.4%	15
RC204	2821	6	3716	+31.7%	14
RC205	3491	2	3821	+9.5%	9
RC206	3541	13	4651	+31.3%	19
RC207	3751	12	4902	+30.7%	18
RC208	3052	2	3459	+13.3%	11

Table 9
Results to analyze the impact of the initial stock of empties on the planning. "E +30%" indicates the basic instance with an increase of empty containers of ≈ 30%. "E=0" indicates a null inventory. The number of empty containers in the basic instances is reported beside the objective value.

	Basic real-world instances - B&P	E + 30%		E = 0	
	Obj.	Obj.	Improvement	Obj.	Deterioration
Inst1	1559 (E=3)	1483	−4.8%	1699	+8.9%
Inst2	1775 (E=5)	1713	−3.5%	1818	+2.4%
Inst3	1230 (E=4)	1230	0%	1276	+3.7%
Inst4	1131 (E=3)	1131	0%	1203	+6.3%
Inst5	1670 (E=5)	1670	0%	1798	+7.6%

Table 10
Results for the 50-nodes Solomon's instances to analyze the impact of the initial stock of empties on the planning.

	Solomon's instances	E+30%		E=0	
Instance	Obj.	Obj.	Improvement	Obj.	Deterioration
C101	2709	2709	0%	3189	+17.7%
C102	2731	2611	−4.4%	3321	+21.6%
C103	2729	2681	−1.8%	3241	+18.7%
C104	2104	2104	0%	2104	0%
C105	2981	2772	−7.1%	3786	+27.1%
C106	2859	2841	−0.6%	3038	+6.2%
C107	3008	3008	0%	3008	0%
C108	3131	2920	−6.7%	3842	+22.7%
C109	3351	3147	−6.1%	4032	+20.3%
RC201	3606	3606	0%	3606	0%
RC202	3856	3833	−0.6%	4336	+12.4%
RC203	3457	3445	−0.3%	3547	+2.6%
RC204	2821	2821	0%	2868	+1.6%
RC205	3491	3491	0%	3508	+0.4%
RC206	3541	3476	−1.8%	4092	+15.6%
RC207	3751	3571	−4.78%	4438	+18.3%
RC208	3052	3052	0%	3099	+1.5%

Table 11
Results to analyze the impact of reducing the distance from the shippers to the empty depot. The "Reduced distances from the empty depot" heading refers to the transformed instances.

	Basic real-world instances - B&P		Reduced distances from the empty depot - B&P		
	Obj.	visits empty depot	Obj.	visits empty depot	Obj. Improvement
Inst1	1559	10	1385	8	−11.1%
Inst2	1775	15	1712	18	−3.5%
Inst3	1230	3	1164	4	−5.3%
Inst4	1131	3	1001	5	−11.4%
Inst5	1670	5	1603	6	−4.1%

(see right-hand side of the table). This is due to the necessity to visit more times the empty depot in order to satisfy the demand. Noticeable is the result from instance 3, from three to nine visits.

In order to provide more solid insights on the impact of empty container re-usage, we also perform this experiment on the 50-nodes Solomon's instances. Results are reported in Table 8. The impact is quite substantial and is visible both in the gap between the two solutions and the number of visits to the empty depot.

5.3.3. Evaluating the impact of the initial empty stock

In this experimentation, we show the impact of the initial number of empty containers at the inland terminal on the planning.

In line with previous indications, Table 9 shows an expected increase when initial stock is not available. At the same time, an increase in the number of empty containers is irrelevant in instances 3, 4, and 5, because the number of available containers (respectively 4, 3, and 5) is enough to properly satisfy the demand at minimum cost. Further analysis of the solution indicates that in a few cases, the empty depot was still used, even when containers were available at the inland terminal, in case the truck was close to its proximity. With the proposed model, managers may conduct a cost-benefit analysis and decide how much to penalize the visit to the depot by increasing or decreasing the cost of the by-pass arcs through the empty depot (See experiments in Section 5.3.4).

Also for this test, we perform experiments on the 50 nodes Solomon's instances to provide a better indication of the impact of the initial empty stock. Results are reported in Table 10. Increasing the stock provides some improvements; however, the results indicate that the considered stock in the regular instances is appropriate in most cases. On the other hand, the cost increase when initial stock is not present is quite substantial. For some instances, the deterioration of the solution is minimal, and this can be explained by the presence of more shippers releasing empty containers in the network, thereby compensating for their scarcity at the inland terminal. Compared to the results of Table 8 the impact is slightly less pronounced, but yet very relevant from a cost perspective.

5.3.4. Impact of the distance of the empty depot from the inland terminal

In this experiment, we evaluate the impact of the distance of the empty depot from the inland terminal and the shippers. In particular, we decrease the distances from the shippers by 75%. The results in Table 11 show an increase in the number of visits to the empty depot and a reduction in the total cost. However, the increase in visits is not substantial, meaning that the solutions keep using street-turning or the available stock at the inland terminal to satisfy the demand. This experiment also shows the flexibility of the model in considering a penalty for multiple visits, without cumbersome extra costs in the objective function.

5.3.5. Discussion and managerial insights

From our analysis, we can draw managerial insights into the planning practices of the terminal and the inland container transport system.

With regard to planning practices, an analysis of the data set showed that just a few containers shared the same trip reference number, indicating how the planners prefer to keep trips simple without performing street-turns. Also, we could notice a lack of synchronization between the transport operator and the shippers, who required multiple visits by different trucks, limiting the pos-

sibility of consolidation. The planners pointed out the complexity when composing trips and the considerable effort required when synchronizing the trips with the shippers' requirements. This suggests how IT investments in optimization software are more and more relevant as the requests of shippers become more heterogeneous, and both import and export flows are present in the region, increasing complexity in the decision-making.

Second, our experiments in Section 5.3.2 showed a substantial increase in covered mileage in case of fewer opportunities for empty containers' re-usage. The transport operator of our case-study highlighted how empty containers are typically bound to so-called "detention" time windows (Fazi & Roodbergen, 2018), requiring the transport operators and the shippers to return them to empty depots or seaports without the possibility of re-utilization. This is also replicated in the experiments when fewer $\Diamond^{FE}$ customers are present. From the proposed experiments, it is visible that more visits to the empty depot are required. In line with the study of Fazi & Roodbergen (2018), detention cost structures are quite detrimental to the overall efficiency of the supply chain. We suggest that in case of re-usage, shipping lines (typically the owners of the containers) should grant an extension of the due date or a renewal of the detention period assigned to the new shipper using the container. However, we believe that more could be done in this respect, in practice, to promote empty container re-usage. The industry still lacks visibility and advanced information systems that can match supply and demand advantageously and in this case, disclose information on available empty containers to be re-used. Booking platforms in line with the concept of Mobility-as-a-Service applied to freight may be a solution to favour this process of re-utilization of empty containers (Le Pira, Tavasszy, de Almeida Correia, Ignaccolo, & Inturri, 2021).

Next, the presence of empty containers at the inland terminal provides the required flexibility to meet due dates and avoid empty trips. In many cases, the lack of empty containers forces transport operators to negotiate with the shippers to extend the time windows when containers can be delivered. This is detrimental to both the service level and the required effort in seeking a container in the network and synchronising the requests. This is especially relevant when considering a Merchant Haulage type of transportation (Nazemzadeh & Vanelslander, 2015), namely when transportation is carried out by independent transporter and not shipping lines. Despite added flexibility for the shippers in such a case, costs may become extremely high in case of setbacks in the system, especially if related to detention and empty container costs.

The depot for empties has a relevant role in container systems, and these depots are typically owned by shipping lines. The company of our case study explained that certain deals could be made with the shipping lines to have direct access to empty containers. This access can depend also on whether there is a contract in place between the shipping line and the shipper, the consignee of the container. In this respect, our model, by simply tweaking distances, is able to characterize the access for a shipper to these empty containers. Also, the reuse of containers may not always be possible due to different contracts established by the shippers. This can be accounted for in the model by tweaking the by-pass arcs defined in Section 3. From a practical point of view, improved access requires coordination between shipping lines, which should consider the possibility of sharing resources outside established alliances or contracts.

Finally, the proposed model and solution framework are quite adaptable to a large variety of cases of inland movements. Even when shippers may require more visits at different moments of the day, this can be easily tackled by the proposed solution. Besides the support for decision-making, this comprehensive framework can be beneficial to increase margins for transport operators, whose sector has been recently more and more hit by recent economic and social crises (Mavi, Mavi, Olaru, Biermann, & Chi, 2022).

6. Conclusions

In this paper, we studied a relevant drayage problem for an inland terminal transportation system. We considered a fleet of trucks that depart from the inland terminal and serve a set of shippers (customers), requiring pickup or delivery of full or empty containers. Full containers may be delivered or also picked at the reference seaport. The demand for empty containers may be satisfied by the stock at the inland terminal or by reusing empties released by the shippers. If necessary, an empty depot is also available to fulfil the demand. Multiple trips can be performed during the day, and the goal is to minimize the distance travelled. The peculiarity of the problem is the synchronization of trucks' trips to avoid exceeding the stock of empty containers at the inland terminal.

Our literature review showed a gap. In particular, the multi-trip component and the synchronization of trucks' trips have never been tackled together in both the drayage-related literature and general one related to VRP. A few models are available for the inventory of empty containers in drayage operations, but the proposed constraints are non-linear. Also, to the best of our knowledge, an exact non-discretized approach for a limited resource setting that needs synchronization is missing in this setting. In this study, we proposed a MILP formulation and developed an exact method based on a row-and-column generation framework. Two Branch-and-Price algorithms were developed. The first algorithm creates multi-trip routes in the pricing step. The second generates single trips combined in the restricted master problem. Also, a set of pricing algorithms have been developed, and effective strategies were proposed. Numerical experiments have been conducted on adapted Solomon's instances, showing excellent performances of our algorithms against CPLEX. The B&P algorithm where column-and-row generation generates single trip routes in its pricing problem, in line with previous results in the literature, outperforms the other B&P approach. In general, the method proved to be very effective even for larger instances. A second set of experiments on instances inspired by a real-world case in the Dutch hinterland, showed great potential for cost reduction when combining multiple requests in single trips. Moreover, this example highlighted that: current shipping lines' practices to control containers are detrimental to the re-usage of the containers; better synchronization is required between shippers and transport operators, along with more visibility of the containers available. The latter point could be tackled by sharing information in Mobility-as-a-service kind of platforms and by allowing direct booking of transport demand.

For future research, the development of metaheuristics is recommended to tackle large instances. The proposed algorithms can be used to benchmark their performances. Modelling extensions may include the allocation of drivers and constraints related to trip duration due to legal requirements. From an operational point of view, two drivers may be allocated to one truck in case of lengthy trips to limit breaks. This should provide an interesting trade-off in case of limited resources. Finally, in drayage operations, the chassis is a resource that can be decoupled from the tractor and could be taken into account within the scheduling.

Acknowledgments

We want to thank the Center for Information Technology of the University of Groningen for their support and for providing access to the Peregrine high performance computing cluster.

References

Benantar, A., Abourraja, M. N., Boukachour, J., Boudebous, D., & Duvallet, C. (2020). On the integration of container availability constraints into daily drayage operations arising in France: Modelling and optimization. *Transportation Research Part E: Logistics and Transportation Review, 140*, 101969.

Bombelli, A., & Fazi, S. (2022). The ground handler dock capacitated pickup and delivery problem with time windows: A collaborative framework for air cargo operations. *Transportation Research Part E: Logistics and Transportation Review, 159*, 102603.

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

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

Bredström, D., & Rönnqvist, M. (2008). Combined vehicle routing and scheduling with temporal precedence and synchronization constraints. *European Journal of Operational Research, 191*(1), 19–31.

Bruglieri, M., Mancini, S., Peruzzini, R., & Pisacane, O. (2021). The multi-period multi-trip container drayage problem with release and due dates. *Computers & Operations Research, 125*, 105102.

Caris, A., & Janssens, G. K. (2009). A local search heuristic for the pre-and end-haulage of intermodal container terminals. *Computers & Operations Research, 36*(10), 2763–2772.

Cattaruzza, D., Absi, N., & Feillet, D. (2016). The multi-trip vehicle routing problem with time windows and release dates. *Transportation Science, 50*(2), 676–693.

Chen, R., Meng, Q., & Jia, P. (2022). Container port drayage operations and management: Past and future. *Transportation Research Part E: Logistics and Transportation Review, 159*, 102633.

Cui, H., Chen, S., Chen, R., & Meng, Q. (2022). A two-stage hybrid heuristic solution for the container drayage problem with trailer reposition. *European Journal of Operational Research, 299*(2), 468–482.

de Ricqlès, J. (2019). Containerized sea freight: Is it time to switch from TEU to FEU? https://market-insights.upply.com/en/containerized-sea-freight-is-it-time-to-switch-from-teu-to-feu (accessed January, 2022).

Desaulniers, G., Lessard, F., & Hadjar, A. (2008). Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. *Transportation Science, 42*(3), 387–404.

Drexl, M. (2012). Synchronization in vehicle routing–A survey of VRPs with multiple synchronization constraints. *Transportation Science, 46*(3), 297–316.

Ebben, M. J. R., van der Heijden, M. C., & van Harten, A. (2005). Dynamic transport scheduling under multiple resource constraints. *European Journal of Operational Research, 167*(2), 320–335.

Fazi, S., & Roodbergen, K. J. (2018). Effects of demurrage and detention regimes on dry-port-based inland container transport. *Transportation Research Part C: Emerging Technologies, 89*, 1–18.

Feillet, D. (2010). A tutorial on column generation and branch-and-price for vehicle routing problems. *4or, 8*(4), 407–424.

Fink, M., Desaulniers, G., Frey, M., Kiermaier, F., Kolisch, R., & Soumis, F. (2019). Column generation for vehicle routing problems with multiple synchronization constraints. *European Journal of Operational Research, 272*(2), 699–711.

Fransoo, J. C., & Lee, C.-Y. (2013). The critical role of ocean container transport in global supply chain performance. *Production and Operations Management, 22*(2), 253–268.

Grangier, P., Gendreau, M., Lehuédé, F., & Rousseau, L.-M. (2021). The vehicle routing problem with cross-docking and resource constraints. *Journal of Heuristics, 27*(1), 31–61.

Guerriero, F., Di Puglia Pugliese, L., & Macrina, G. (2019). A rollout algorithm for the resource constrained elementary shortest path problem. *Optimization Methods and Software, 34*(5), 1056–1074.

Hempsch, C., & Irnich, S. (2008). Vehicle routing problems with inter-tour resource constraints. In *The vehicle routing problem: Latest advances and new challenges* (pp. 421–444). Springer.

Hernandez, F., Feillet, D., Giroudeau, R., & Naud, O. (2016). Branch-and-price algorithms for the solution of the multi-trip vehicle routing problem with time windows. *European Journal of Operational Research, 249*(2), 551–559.

Ileri, Y., Bazaraa, M., Gifford, T., Nemhauser, G., Sokol, J., & Wikum, E. (2006). An optimization approach for planning daily drayage operations. *Central European Journal of Operations Research, 14*(2), 141–156.

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

Imai, A., Sasaki, K., Nishimura, E., & Papadimitriou, S. (2006). Multi-objective simultaneous stowage and load planning for a container ship with container rehandle in yard stacks. *European Journal of Operational Research, 171*(2), 373–389.

Irnich, S., & Desaulniers, G. (2005). Shortest path problems with resource constraints. In *Column generation* (pp. 33–65). Springer.

Kiani Mavi, R., Kiani Mavi, N., Olaru, D., Biermann, S., & Chi, S. (2022). Innovations in freight transport: A systematic literature evaluation and COVID implications. *The International Journal of Logistics Management, 33*(4), 1157–1195.

Le Pira, M., Tavasszy, L. A., de Almeida Correia, G. H., Ignaccolo, M., & Inturri, G. (2021). Opportunities for integration between mobility as a service (MaaS) and freight transport: A conceptual model. *Sustainable Cities and Society, 74*, 103212.

Li, S., & Jia, S. (2019). The seaport traffic scheduling problem: Formulations and a column-row generation algorithm. *Transportation Research Part B: Methodological, 128*, 158–184.

Liu, R., Tao, Y., & Xie, X. (2019). An adaptive large neighborhood search heuristic for the vehicle routing problem with time windows and synchronized visits. *Computers & Operations Research, 101*, 250–262.

Lozano, L., Duque, D., & Medaglia, A. L. (2016). An exact algorithm for the elementary shortest path problem with resource constraints. *Transportation Science, 50*(1), 348–357.

Maher, S. J. (2016). Solving the integrated airline recovery problem using column-and-row generation. *Transportation Science, 50*(1), 216–239.

Mingozzi, A., Roberti, R., & Toth, P. (2013). An exact algorithm for the multitrip vehicle routing problem. *INFORMS Journal on Computing, 25*(2), 193–207.

Nasir, J. A., & Kuo, Y.-H. (2020). A decision support framework for home health care transportation with simultaneous multi-vehicle routing and staff scheduling synchronization. *Decision Support Systems, 138*, 113361.

Nazemzadeh, M., & Vanelslander, T. (2015). The container transport system: Selection criteria and business attractiveness for north-european ports. *Maritime Economics & Logistics, 17*(2), 221–245.

Paraskevopoulos, D. C., Laporte, G., Repoussis, P. P., & Tarantilis, C. D. (2017). Resource constrained routing and scheduling: Review and research prospects. *European Journal of Operational Research, 263*(3), 737–754.

Reinhardt, L. B., Pisinger, D., Spoorendonk, S., & Sigurd, M. M. (2016). Optimization of the drayage problem using exact methods. *INFOR: Information Systems and Operational Research, 54*(1), 33–51.

Rodrigue, J.-P., Debrie, J., Fremont, A., & Gouvernal, E. (2010). Functions and actors of inland ports: European and north american dynamics. *Journal of Transport Geography, 18*(4), 519–529.

Sadykov, R., & Vanderbeck, F. (2013). Column generation for extended formulations. *EURO Journal on Computational Optimization, 1*(1-2), 81–115.

Şen, A., & Bülbül, K. (2008). A survey on multi trip vehicle routing problem. In *VI. International Logistics & Supply Chain Congress 2008, Istanbul*.

Taillard, E. D., Laporte, G., & Gendreau, M. (1996). Vehicle routeing with multiple use of vehicles. *Journal of the Operational research society, 47*(8), 1065–1070.

Toygar, A., Yildirim, U., & Gani Mustafa, I. (2022). Investigation of empty container shortage based on SWARA-ARAS methods in the COVID-19 era. *European Transport Research Review, 14*(1), 1–17.

Zhang, R., Huang, C., & Wang, J. (2020). A novel mathematical model and a large neighborhood search algorithm for container drayage operations with multi-resource constraints. *Computers & Industrial Engineering, 139*, 106143.

Zhang, R., Yun, W. Y., & Kopfer, H. (2010). Heuristic-based truck scheduling for inland container transportation. *OR spectrum, 32*(3), 787–808.

Zhang, R., Yun, W. Y., & Moon, I. (2009). A reactive tabu search algorithm for the multi-depot container truck transportation problem. *Transportation Research Part E: Logistics and Transportation Review, 45*(6), 904–914.

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